SESSION

CLOUD COMPUTING AND RELATED ISSUES

Chair(s)

TBA

Accounting Models for Cloud Computing: A Systematic Mapping Study

Francisco Airton Pereira da Silva[1], Paulo Anselmo da Mota Silveira Neto[1],
Vinicius Cardoso Garcia[1], Rodrigo Elia Assad[1], Fernando Antonio Mota Trinta[2]
[1]Federal University of Pernambuco (UFPE) – Recife, Pernambuco, Brazil
[2]Federal University of Ceará (UFC) – Fortaleza, Ceará, Brazil
{faps,pamsn,vcg,rea}@cin.ufpe.br {fernando.trinta}@virtual.ufc.br

***Abstract*—Cloud services change the economics of computing by enabling users to pay only for the capacity that they actually use. In this context, cloud providers have their own accounting models including their billing mechanisms and pricing schemes to achieve this efficient pay-as-you-go model. Thus it is important to study this heterogeneity aiming to map out the existing accounting models to become possible new proposals or future standardizations. Therefore, this paper focuses on mapping accounting models for cloud computing, where a mapping study process was undertaken, and a total of 23 primary studies were considered, which evidenced 5 accounting models, 23 different pricing scheme types and 4 primary studies related to SLA (Service-Level Agreement) composition. Although the significant number of studies found address grid computing it was possible to identify one accounting model which was very complete from different points of view for cloud environments.**

***Index Terms*—Cloud Computing; mapping study; pricing scheme; accounting model; Service Level Agreement.**

I. Introduction

Cloud computing has become an established paradigm for running services on external infrastructure, where virtually unlimited capacity can be dynamically allocated. However this unlimited aspect in some cases can become expensive, and research projects have tried to mitigate it through the development of new architectures, exploring different accounting models [1] [2] [3] [4].

Accounting in cloud computing is a recent discipline, hence there have few attempts to find a model which considers all the accounting requirements, and none work has tried to address a mapping of the existing accounting models that could identify research gaps and encourage future proposals.

In this context, this paper introduces a mapping study performed between July and December, 2011, addressing accounting models for Cloud Computing environments and other aspects related also to Grid Computing.

We had to encompass the grid computing research field, mainly due to three considerations. The first point is the correlated aspects between cloud and grid computing, the second point is the older grid origin with probable relevant contributions and as final reason, the existing mature accounting models under this research area.

In [32] the authors perform a comparison between the six most known accounting systems in grid computing, evidencing the advantages and disadvantages of them whereas allowing to realise what aspects they have in common.

First, they use a proper taxonomy to describe their functions which make part of an accounting process (a set of operations that manages the data regarding the use of the resources [5]).

Next, they present a measurement unit mechanism to apply under the resource consumption and accordingly charge for it, called pricing scheme [6].

Finally, all of them worry about QoS Requirements and explores how to monitor this Quality of Service. In some cases establishing Service Level Agreements (SLA).

Based on aforementioned items and previous literature investigation, four research questions were derived to guide this mapping study, as follows:

- RQ1: Is there any taxonomy for concepts related to accounting process in cloud computing?
- RQ2: What are the existing accounting models for cloud computing?
- RQ3: What are the existing pricing schemes for cloud or grid computing?
- RQ4: What are the aspects taken into account to compose a SLA in cloud/grid computing scenario?

The remainder of the paper is structured as follows: Section II introduces the related work; Section III presents the systematic mapping study process; Section IV describes the main findings of the study; Section V presents the analysis of the results, studies classification and mapping; Section VI introduces some threats to validity. Finally, Section VII presents the conclusions and future research.

II. Related Work

Basically our research started motivated by the evolution in federated cloud infrastructures field, which two works stands out (RESERVOIR and JiT Clouds).

RESERVOIR Project [2] presents an architecture (including an advanced accounting model) that allows providers of cloud infrastructures to dynamically partner with each other to create a virtually infinite pool of resources.

JiT Clouds Project [7] also allows providers of cloud infrastructures to dynamically partner with each other, but with the advantage where providers does not need keep dedicated

resources to meet the service providers demands, however does not have an accounting model.

In [8] the authors present a comparative review of grid and cloud computing pricing models. Unlike our proposal, this paper is not a systematic study and related only with our RQ3.

III. Systematic Mapping Study Process

A Mapping Study is a systematic process that provides an overview and summarizes published paper results of a particular research area, by answering questions and categorizing the studies. As main benefit, it can be used to identify gaps in the existing research that will lead to topics for further investigation [9].

Therefore, a Systematic Mapping Study was used in this research to "map out" the accounting models for cloud computing, performing five steps (Questions Definition, Search, Screening, Keywording and Extraction) [9].

A. Conduct Search

The strategy used to construct the search terms, follows the same approach used in [10], since it is systematized in essence and defines steps to derive the search strings from the questions and the viewpoints of experts in the area and relevant papers. The strategy steps are described as follows:

- Derive major terms from the questions by identifying the population, intervention, outcomes and study design;
- Identify, by inquiries with experts in the field, alternative spellings and synonyms for major terms; and
- Check the keywords in the relevant papers.

The complete list of search strings and their combination are presented in Table I.

TABLE I
Search String

SLA OR "Service Level Agreement" OR billing OR pricing OR payment OR accounting AND "cloud computing" OR "grid computing" OR "Infrastructure as a Service" OR "Platform as a Service" OR "Software as a Service"

Firstly an *automatic search* was conducted in different search engines (IEEEXplore, ACM Digital Library, Scopus and ScienceDirect digital databases). It is important to mentioned that all search strings were calibrated regarding to each search engine. Next, a *manual search* was performed by visiting some important conference proceedings. As a results from the application of both search strategies 580 studies were collected.

At this point, the studies were excluded according to the exclusion criteria:

- Studies did not address or just mentioned accounting models/processes, pricing schemes, SLA composition on cloud/grid computing;
- Studies only available as abstracts or presentations; and
- Duplicate studies. When a study has been published in more than one publication, the most complete version will be considered.

B. Screening of Papers

Firstly, the exclusion criteria were applied on the title and abstract of the identified studies, resulting in 98 studies being selected. The large number of duplicated studies contributed to this large difference. Next, a second filter was applied, analysing the introduction and conclusion, which resulted in 23 studies ([1], [2], [3], [4], [5], [6], [8], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25] and [26]).

C. Keywording

A classification scheme was built which analysed the abstract, titles and keywords of the selected primary studies to identify different facets. This way, three different facets were used. They are described following:

- **Contribution Type**: Method, Process, Technique, Model and Framework [27];
- **Accounting Model Features**: Pricing, Metering, Mediation, Accounting, Roaming, Billing, Charging, Financial Clearing, Cloud Federation, Just in Time Clouds, User Interface, Security Support, SLA Support and Variable Payment Models;
- **Research Type**: Validation Research, Evaluation Research, Solution Proposal, Philosophical Papers, Opinion Papers, and Experience Papers [28] (see definitions in Table II).

TABLE II
Research Type Facet [28]

Class	Description
Validation Research	Techniques investigated are novel and have not yet been implemented in practice. Techniques used are for example experiments, i.e., work done in the lab.
Evaluation Research	Techniques are implemented in practice and an evaluation of the technique is conducted. That means, it is shown how the technique is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation). This also includes identification of problems in industry.
Solution Proposal	A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution is shown by a small example or a good line of argumentation.
Philosophical Papers	These papers sketch a new way of looking at existing things by structuring the field inform of a taxonomy or conceptual framework.
Opinion Papers	These papers express the personal opinion of somebody whether a certain technique is good or bad, or how things should been done. They do not rely on related work and research methodologies.
Experience Papers	Experience papers explain what and how something has been done in practice. It has to be the personal experience of the author.

D. Data Extraction

A data extraction form was designed in order to gather the required information to address the objectives of this study, classifying and answering the research questions. The full paper was read and the following information was extracted

from each study: the research categorization (Contribution Type, Accounting Model Features and Research Type), in addition the information required to answer some of the research questions.

IV. Results

In this section, each topic presents the findings regarding to a specific research question, highlighting the evidences gathered from the data extraction process.

A. RQ1 - Is there any taxonomy for concepts related to accounting process in cloud computing?

In our research only one primary study effectively answered this question. The study [5] presents a taxonomy of full accounting process and its functions from the resource usage to the financial clearing. It is not applied only to cloud computing, but other areas related to Services on the Internet (see Table III).

TABLE III
Taxonomy of Accounting Process [5]

Concept	Function
Pricing	Function of giving a price to a certain resource usage.
Metering	Collects raw information regarding the resource usage of a certain service by a consumer and its usage.
Mediation	Is intended to do a first treatment of raw technical data by transforming these metering records into a data format that can be used for storing and further processing.
Accounting	Has the function of filtering and treat more accurately the records passed by mediation function.
Roaming	Allows using more than one provider while maintaining a formal, customer-vendor relationship just with one.
Billing	Also called of invoicing, is the process of transforming charge records into the final bill, summarizing the charge records of a certain time period and indicating the amount of monetary units to be paid by the customer.
Charging	Is the process of calculating the cost of a resource usage, the function that translates technical values into monetary units by applying a pricing function to the session records.
Financial Clearing	Includes activities from a commitment for a transaction to its settlement. In the case of resource accounting, this function implies the payment of a bill.

Although it was found only this taxonomy formally defined, other terms are widely used with the same meanings. For example, monitoring has the same sense of metering. According to [25] the metrics generated by the monitoring function can be used both for accounting purposes as for performance analysis. In other study [5], monitoring is a sub-function of metering that collects the information of a resource usage as raw data and provides usage metrics to the metering function.

B. RQ2 - What are the existing accounting models for cloud computing?

When performing the analysis, were found five primary studies ([1], [2], [3], [4] and [25]) that proposed some kind of accounting model, summarized following:

a) Flexible Accounting Model [1] - This paper proposes a flexible accounting model suitable to any service of cloud computing. This model is based on the accounting process of the Internet and it can fit any pricing scheme using jBilling accounting platform and mainly through the use of IPDR (Internet Protocol Detail Record).

b) A Model for Federated Clouds [2] - This primary study presents a solution for an accounting and billing architecture for use in federated cloud environments like the RESERVOIR project (funded by European Union). The model is organized in layers(Accounting, Billing and Business Layer)

c) ABS for SOA [3] - This primary study presents a framework wherein authentication of the clients and billing of services used by client is carried out. So this paper threats the security as an essential requirement in billing services. like generates instances of virtual machines for a particular time period ordered by user (time-based pricing scheme) in safe mode.

d) THEMIS [4] - This model proposed a mutually (provider and user) verifiable billing system called THEMIS to Cloud Computing scenario in which has as main requirements the transparency, security and low latency in billing transactions. Thus, the system introduces the concept of a *Cloud Notary Authority* to supervise billing transactions, using a level of security that is identical to that of a Public Key Infrastructure (PKI), combating the malicious behaviour of users and providers.

e) Cloud Supply Chain [25] - This model proposes the *Cloud Supply Chain* concept, which represents a network of interconnected businesses in the cloud computing area involved in the end-to-end provision of product and aggregated service packages required by end cloud service customers. This includes the actual provisioning of infrastructure services and the Information Model supporting monitoring, accounting and billing processes.

C. RQ3 - What are the existing pricing schemes for cloud or grid computing?

Table IV summarizes all the pricing schemes found with their respective concepts and in which study they were discussed.

There is a lot of work that mention some type of pricing scheme. However it is used different terms to the same pricing scheme meaning. In [8], [15] and [16] the authors refers to pricing schemes as *pricing models* and specially in [8] the pricing models are grouped in a general way called *economic models*. For example, the economic model *Commodity Market* (price defined based on amount of resource that users used) has as pricing models: *Usage Duration* and *Flat Fee*.

D. RQ4 - What are the aspects taken into account to compose a SLA in cloud/grid computing scenario?

In order to cloud providers supply clients with services that meet their quality constraints, they both need to negotiate the clients requirements and the provider's infrastructure capabilities. It is known as Service Level Agreement (SLA) . However, this is not an easy task, according to [29] there are many difficulties to formalize a SLA, such as lack of flexibility and precision. This way, to compound a SLA it is important to know which aspects have to be taken into account.

TABLE IV
PRICING SCHEMES

Pricing Scheme	Definition	Studies
Time-based	Pricing based on how long a service is used.	[24], [1], [6]
Paris-Metro pricing	Used for shared resources. Resources are split by the amount of users per split.	[1]
Priority pricing	Services are labelled and priced according to their priority.	[1]
Flat-rate	A fixed tariff for a specified amount of time.	[24], [1], [6]
Edge pricing	Calculation is done based on the distance between the service and the user.	[1]
Responsive pricing	Charging is activated only on service congestion.	[1]
Effective bandwidth pricing	Charging is based on an expected usage function.	[1]
Proportional fairness pricing	It is according to the user's willingness to pay, in other words, It is based on the real value of product or service.	[16], [1]
Cumulus pricing	Based on flat pricing and dynamically priced by using a credit point system.	[1]
Session-oriented	Based on the use given to the session.	[1]
One-off charge per service	One charge per service session.	[1]
Usage-based	Pricing based on the general use of the service for a period of time, e.g. a moth.	[6], [15], [1]
Content-based	Pricing based on the accessed content.	[1]
QoS-based	Pricing depends on the hired quality of service.	[22], [1]
Location-based	Pricing based on the access point of the user.	[1]
Service type	Pricing based on the usage of the service.	[1]
Volume-based	Pricing based on the volume of a metric (e.g. downloaded bytes).	[22], [1]
Differentiation on time-of-day	Pricing based on the hour when the service is used.	[1]
Progressive Co-design	Seller and buyer try to convene on a pricing plan. The seller announces a fixed price pair (p1, p2), where p1 ¡ p2. Subsequently, the buyer commits a consumption level quality related to each price announced and if agreed so he can buy additional units progressively if needed.	[6]
Competitor-Oriented (CO) Pricing	At first, the vendor agent needs to select the competitor to compete with. Then, the vendor simply decreases the price just below of the rival's price. This algorithm requires perfect information of the rival's price.	[20], [22], [16]
Cost-based	Following the approach of cost-based pricing, the price level is established using cost accounting. According to it price determination based on costs can make good sense for SaaS.	[16]
Supply and Demand based	In general way the unit price will vary until it settles at a point where the quantity demanded by consumers (at current price) will equal the quantity supplied by providers (at current price), resulting in an economic equilibrium of price and quantity.	[18], [23], [24], [26], [19], [20]
Real-Time Pricing (RTP)	Is a pricing model that dynamically changes its rate reacting to the classical supply and demand rule, but with the difference that there is only one supplier. Amazon Web Services (AWS) offer a simplified form of this pricing model called Spot Instances.	[26]
Derivative Follower Model	It's a kind of supply and demand based model simply adjusts prices by incrementally increasing or decreasing them until the observed profitability level falls, then the direction of price adjustment is reversed, thus seeking a local maximum of profitability.	[19], [20]
Hybrid Pricing Model	This model allows a third entity called Price Authority dynamically adjust prices within static limits to balance the workload on the basis of the queue wait times of jobs in grid environments.	[19]
Auction based	Services are priced in an auction and usually carried out by a third party, called the market maker, which collects the bids, selects the winners and computes the payments.	[17], [18], [1], [24]
English Auction	All bidders are free to increase their bids exceeding other offers. When none of the bidders are willing to raise the price anymore, the auction ends, and the highest bidder wins the item at the price of his bid.	[24]
First-Price Sealed-Bid Auction	Each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of his bid.	[24]
Vickrey	Each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of the second highest bidder.	[24]
Dutch Auction	The auctioneer starts with a high bid/price and continuously lowers the price until one of the bidders takes the item at the current price. It is similar to a first-price sealed-bid auction because in both cases the bid matters only if it is the highest, and no relevant information is revealed during the auction process.	[24]
Double Auction	In the double auction model, buy orders (bids) and sell orders (asks) may be submitted at any time during the trading period. If at any time there are open bids and asks that match or are compatible in terms of price and requirements (e.g., quantity of goods or shares), a trade is executed immediately.	[24]

When performing the analysis, few studies explicitly stated the formalization of SLA in Cloud/Grid Computing scenario. However 4 primary studies ([11], [12], [14] and [21]) are complementary. They are summarized following.

a) In [11] is introduced a framework that enables dynamic specification and verification of SLAs on the Cloud. Its main contribution to our research is an format of SLA-Description based on XML specification which defines the main Quality of Services (QoS) along with their threshold values agreed up on selection of cloud services. It also defines the period of service provision, the cost of using the service, and the possible actions that should be taken if QoS provision is frequently violated.

b) In [12] is presented a framework which the SLA pa-

rameters are specified by metrics. These metrics define how cloud service parameters can be measured and specify values of measurable parameters. In addition to specific metrics this study also propose general metrics that can be defined for SLA with any or all types of cloud users.

c) In [14] the authors addressed the use of Cloud Computing for web hosting providers by creating a Cloud Hosting Provider (CHP). They designed an SLA-aware web servers management system in order to address the resources outsourcing mechanism on the provider's part, defining important economic variables to this kind of technology.

d) In [21] is proposed an unambiguous and flexible language for formalizing SLAs and an architecture for specifying and monitoring SLA's on grid computing scenario. It references a typical SLA formulated by Morris et al. [29] that includes the components: *Purpose*, *Parties*, *Validity Period*, *Scope*, *Restrictions*, *Service-Level Objectives*, *Service-Level Indicators*, *Penalties*, *Optional Services*, *Exclusions* and *Administration*.

V. Analysis of the Results and Mapping of Studies

By analysing the results, it can enable us to present the number of studies tabulated in each category defined in this study. Thus, it is possible to identify what have been emphasized in past research and determine gaps and opportunities for future research [9].

A. Research Type Classification

TABLE V
Research Type Classification

Research Type	Studies	Quantity
Validation Research	[14], [17], [18], [8], [22],[23], [25]	7 (30,4%)
Evaluation Research	[4], [20], [1], [24], [26], [25]	6 (26%)
Solution Proposal	[11], [12], [13], [2], [15], [16], [3], [4], [19], [21], [22], [5]	12 (52,1%)
Philosophical Papers	[6]	1 (4,3%)
Opinion Papers	-	0 (0%)
Experience Papers	-	0 (0%)

Initially, let us analyse the studies distribution regarding to the research type classification (Table V).

It was notorious the "Opinion" and "Experience" papers inexistence, while a number of "Validation", "Evaluation" and mainly "Solution Proposal" was found. Perhaps the rationale was the contribution level desired by researches proposing evaluable solutions to have more scientific relevance.

However, another more important point was observed related to "Evaluation Research", it is notable the small quantity of studies that matches this facet indicating insufficient experimentation in industry.

Certainly there is progress in this direction, but the accelerated growth in the cloud providers number (reported by [30]) influences the degree of competitiveness, causing the non-disclosure of their proposals in the scientific community. This fact encourages us to perform another research analysing cloud provider's accounting models in practice and comparing them.

B. Contribution Type Classification

Table VI shows the contribution type classification scheme, which we can observe the most of studies propose concrete "Models" or "Frameworks" instead of address activities related to accounting functions. This way, few "Processes", "Techniques" and none "Method" was registered. One possible explanation may be the observation made earlier, regarding the lack of practical results disclosed by the industry. In this case, we can conclude that even small-scale, companies publish "what they did" (models and frameworks) but hide the "how they did" (processes, techniques and methods).

TABLE VI
Contribution Type Classification

Contribution Type	Studies	Quantity
Method	-	0 (0%)
Process	[11], [4], [21], [5], [25]	5 (21,7%)
Technique	[14]	1 (4,3%)
Model	[13], [6], [2], [15], [16], [19], [20], [1], [21], [8], [22], [23], [24], [25]	14 (60,8%)
Framework	[11], [12], [6], [17], [18], [3]	6 (26%)

C. Research Types X Research Questions

There were an effort in analysing the relationship between the research questions and the research type, using a bubble plot to represent the interconnected frequencies (Figure 1).

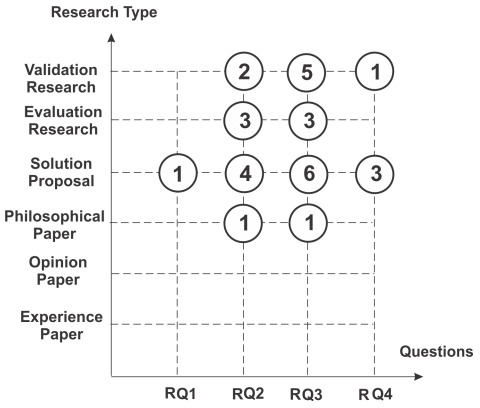


Fig. 1. Research type x Research questions

By analysing the chart upwards, none question was answered by papers that addressed personal opinion or experience, on the other hand one paper gave a big contribution, answering RQ2 and RQ3, classified as philosophical paper, discussing a general pricing scheme that can be applied to define variations for any computational element [12].

Most of the information (about 4 studies) related to accounting models comes from papers classified as "Solution Papers". Since this category includes 12 primary studies, it is clear that there has been few research effort directed to the issue of mitigating mechanisms aiming architecture improvements.

Related to RQ3, although the majority of studies was classified as "Solution Proposal" and "Validation Research",

the papers doesn't discusses how the pricing schemes could be applied in a detailed way, unlike give just short concepts. So, as our research aims to give a general overview mapping the pricing schemes, future researches can focus on explain how the pricing schemes can works in practice.

D. Accounting Models Analysis

The accounting models collected by this research were categorized according to their features (see Table VII).

TABLE VII
ACCOUNTING MODELS ANALYSIS

Studies	Features	Pricing	Metering	Mediation	Accounting	Roaming	Billing	Charging	Finantial Clearing	Cloud Federation	Just in Time Clouds	User Interface	Security Support	SLA Support	Variable Payment Models
[1]		X	X	X	X	X	X	X	X			X	X	X	
[2]							X			X		X	X		X
[3]							X							X	
[4]							X						X		
[25]		X			X		X							X	X

Firstly, we used the taxonomy proposed by [5], aiming to check what functions the proposed models used. Thus, the terms *pricing*, *accounting* and *billing* appeared in more than one paper and with the same meanings, which this homogeneity indicates a certain taxonomy validity. Related to *accounting*, two information stands out:

- In [1] the authors disambiguated the expressions *accounting process* and *accounting function*. Whereas *accounting process* refer to a meta-concept that includes all the taxonomy functions, *accounting function* is related to recording and summarizing technical data in terms of money, transactions and events;
- In [25] the *accounting* and *billing* functions are grouped as integrated sub processes forming a type of macro-process.

Lastly, it is important to highlight that the term *billing* was cited by all primary studies. We attribute this result to the influence of other areas such as telephony that has used largely this term before cloud computing became a research trend.

Other features were derived from the most relevant aspects found in primary studies. *Cloud Federation* was the first feature. In this case it was observed a research gap in which only one accounting model [2] were directed to federated cloud infrastructures, needing to stress that this paper and [25] belong to the same research group (the RESERVOIR project [31]), showing as pioneer researchers in the area.

The feature *Just in Time Clouds* is a recent concept in which providers only allocate resources when they are demanded and only for the duration they are needed by their clients [7]. To explore this mechanism showed promise, because none accounting model addressed this feature. Something previously expected, due to be a recent issue.

The *User Interface Support* was analysed, noticing that some proposed models own a user interface that gives the access control to managing accounting mechanisms on the systems, but not all worried with this feature, only 40% of them had a final user or admin user interface support.

In *Security Support*, just 60% of studies at least cited some security mechanism like user authentication or transaction authorization. When analysed *SLA Support*, it was verified if the studies had SLA monitoring or the customer would choose their service quality desired, noticing that, as such *Security Support*, 60% fit this requirement. Therefore, *SLA* and *Security Support* have been showed as relevant topics of interest in accounting model field for cloud computing.

As last feature, it was investigated if the models were prepared to support different payment models (*Variable Payment Models*) such as Pre-Paid, Pos-Paid or Hybrid. These models are in no way unique to clouds and on the contrary they are well known to customers after being used for years in other utility markets, most notably the mobile phone industry [25]. Hence some accounting models (40%) are ready, for example, to work with resource consumption based on previous purchased credits (Pre-Paid).

It has to be mentioned that initially it was thought to include the term *monitoring*, however was preferred to use the term *SLA Support* instead, due its less ambiguous concept. According to [25], SLA and monitoring are strictly related each other, because the metric concept (from a monitoring point of view) is very semantically close to the "Key Performance Indicators" concept (from a SLA point of view).

Concluding, observing the fourteen features, one paper had a greater coverage. The primary study [1] proposed a flexible accounting model which can fit any service of cloud computing that encompassed almost all features taken into account by our classification. Therefore this paper can be used as a starting point for future accounting models propositions.

VI. THREATS TO VALIDITY

There are some threats to the validity of our study, which we briefly describe below.

- **Research Questions**: The research questions we defined cannot provide complete coverage of the accounting field related cloud and mainly grid computing, however, we had several discussions to validate the questions.
- **Publication Bias**: We cannot guarantee that all relevant studies were selected. We mitigated this threat as much as possible, by following references in the relevant studies.
- **Data Extraction**: The studies were classified based on our judgement, however, some studies could have been classified incorrectly. To mitigate this threat, the classification was performed by more than one researcher.

VII. CONCLUSION AND FUTURE WORK

We have introduced the results of a systematic mapping study about accounting models for cloud computing investigating scientific literature. In the end, starting from 580 papers, 23 filtered studies answered the research questions.

As major contribution, this paper provides an overview of the area and specific findings related to *i)* taxonomy for accounting process, *ii)* accounting models, *iii)* pricing schemes and *iv)* SLA composition.

i) The terms *pricing*, *accounting* and *billing* are the most used terms. Among these, the term *billing* surely is the main term in the area. This result is influenced by other fields such as telephony that has used largely this word before cloud computing became a research trend.

ii) In general there are few studies related to accounting models for cloud computing, mainly in industry environment. Besides there is a need for new proposals in federated cloud infrastructures whereas the topics related to SLA and Security have gained considerable attention.

iii) Despite the large amount of existing pricing scheme types, there is a need in expose how they could be applied in a detailed way, unlike give just short concepts.

iv) Related to SLA composition there are studies that propose possible general items to compose the contract (e.g. *Scope*, *Penalties*, *Restrictions*), others propose specific metrics to monitor the services quality and others presents mechanisms based on XML to specify metrics. Thus studying these results it is possible to develop new solutions combining ideas.

Future work will focus on analyse more accurately these mapping study results in order to match mainly the SLA composition ideas with accounting processes/models found to develop a more advanced accounting model. Also we intend to study the use on real market of the pricing schemes identified.

ACKNOWLEDGEMENTS

This research was sponsored by the Program Center for the Research and Development on Digital Technologies and Communication (CTIC) of the Brazilian Minister of Science and Technology, grant 68/2012. In addition, this work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[1]), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08

REFERENCES

[1] I. R. Agundez *et al.*, "A flexible accounting model for cloud computing," in *Proc. Global Conference (SRII 11). IEEE Computer Society*, Washington, DC, USA, Jul. 2011, pp. 277–284.

[2] E. Elmroth *et al.*, "Accounting and billing for federated cloud infrastructures," in *Proc. Int. Conference on Grid and Cooperative Computing (GCC 09)*, Aug. 2009, pp. 27–29.

[3] T. Pandey *et al.*, "Authentication and billing framework for service oriented architecture," in *Proc. Int. Conference on Systems, (ICONS 09)*, Mar. 2009, pp. 91–95.

[4] K. W. Park *et al.*, "Themis: Towards mutually verifiable billing transactions in the cloud computing environment," in *Proc.Int. Conference on Cloud Computing (CLOUD 10)*, Jul. 2010, pp. 139–147.

[5] I. Agundez *et al.*, "A taxonomy of the future internet accounting process," in *Int. Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 10)*, Jun. 2010, pp. 111–117.

[6] A. Caracas and J. Altmann, "A pricing information service for grid computing," in *Proc. Int. Conference on Middleware (MGC '07)*, New York, NY, USA, 2007, pp. 1–6.

[7] R. Costa *et al.* (2010) Just in time clouds: Enabling highly-elastic public clouds over low scale amortized resources. [Online]. Available: http://www.lsd.ufcg.edu.br/index.php/documentacao-menu#tr

[1] www.ines.org.br

[8] P. Samimi and A. Patel, "Review of pricing models for grid and cloud computing," in *Proc. IEEE Symposium on Computers and Informatics (ISCI 11)*, Mar. 2011, pp. 634–639.

[9] Petersen *et al.*, "Systematic mapping studies in software engineering," *12th Int. Conference on Evaluation and Assessment in Software Engineering*, vol. 17, no. 1, pp. 1–10, 2007.

[10] B. A. Kitchenham *et al.*, "Cross versus within-company cost estimation studies: A systematic review," in *Proc. IEEE Transactions on Software Engineering*, May 2007, pp. 316–329.

[11] A. A. Falasi and M. A. Serhani, "A framework for sla-based cloud services verification and composition," in *Proc. Int. Conference on Innovations in Information Technology (IIT)*, Apr. 2011, pp. 287–292.

[12] M. Alhamad *et al.*, "Conceptual sla framework for cloud computing," in *Proc. International Conference on Digital Ecosystems and Technologies (DEST)*, Apr. 2010, pp. 606–610.

[13] M. Buthelezi *et al.*, "Accounting, pricing and charging service models for a guiset grid-based service provisioning environment," in *Proc. CSREA EEE*, 2008, pp. 350–355.

[14] J. O. Fito *et al.*, "Sla-driven elastic cloud hosting provider," in *Proc. 18th Euromicro Int. Conference on Parallel, Distributed and Network-Based Processing (PDP 10)*, Feb. 2010, pp. 111–118.

[15] B. Jai, "The economy of parallel and distributed computing in the cloud," in *Proc. Int. Symposium on VLSI Design, Automation and Test (VLSI-DAT 2011)*, Apr. 2011, pp. 25–28.

[16] Lehmann *et al.*, "Pricing strategies of software vendors," *Business and Information Systems Engineering*, pp. 452–462, 2009.

[17] M. Mihailescu and Y. M. Teo, "On economic and computational-efficient resource pricing in large distributed systems," in *Proc. Int. Conference on Cluster, Cloud and Grid Computing (CCGRID 10)*, Washington, DC, USA, 2010, pp. 838–843.

[18] M. Mihailescu and Y. Teo, "Dynamic resource pricing on federated clouds," in *Int. Conference on Cluster, Cloud and Grid Computing (CCGrid)*, May 2010, pp. 513–517.

[19] R. M. Piro *et al.*, "An economy-based accounting infrastructure for the datagrid," in *Proc. Int. Workshop on Grid Computing(GRID 03)*, Nov. 2003, pp. 202– 204.

[20] Rohitratana *et al.*, "Agent-based simulations of the software market under different pricing schemes for software-as-a-service and perpetual software," in *Economics of Grids, Clouds, Systems, and Services*, ser. Lecture Notes in Computer Science, Altmann *et al.*, Eds. Springer Berlin / Heidelberg, 2010, vol. 6296, pp. 62–77.

[21] A. Sahai *et al.*, "Specifying and monitoring guarantees in commercial grids through sla," in *Proc. Int. Symposium on Cluster Computing and the Grid (CCGrid 2003)*, May 2003, pp. 292–299.

[22] J. Song *et al.*, "Competitive pricing model for resource scheduling in grid computing," in *Int. Conference on Semantics, Knowledge and Grid*, Oct. 2007, pp. 406–409.

[23] J. Yu *et al.*, "A service-oriented accounting architecture on the grid," in *Proc. Int. Conference Parallel and Distributed Computing: Applications and Technologies (PDCAT 04)*, Dec. 2004, pp. 310–313.

[24] R. Buyya *et al.*, "Economic models for resource management and scheduling in grid computing," in *Concurrency and Computation Practice and Experience 14, Wiley Press*, 2002, pp. 1507–1542.

[25] M. Lindner *et al.*, "The cloud supply chain : A framework for information, monitoring, accounting and billing," in *2nd International ICST Conference on Cloud Computing (CloudComp 2010)*, 2011.

[26] W. Sewook, "Debunking real-time pricing in cloud computing," in *Proc. Int. Symposium on Cluster, Cloud and Grid Computing (CCGRID '11)*, May 2011, pp. 585–590.

[27] I. Freitas *et al.*, "Agile software product lines: a systematic mapping study," *Softw. Pract. Exper.*, vol. 41, pp. 899–920, July 2011.

[28] Wieringa *et al.*, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requir. Eng.*, vol. 11, pp. 102–107, December 2005.

[29] R. Jain *et al.*, "Congestion avoidance in computer networks with a connectionless network layer," Digital Equipment Corporation, MA, Tech. Rep. DEC-TR-506, Aug. 1987.

[30] G. Jeremy. (2010, Oct.) The top 250 players in the cloud computing ecosystem. [Online]. Available: http://cloudcomputing.syscon.com/node/1386896

[31] RESERVOIR. (2012) Resources and services virtualization without barriers. [Online]. Available: http://62.149.240.97/

[32] M. Gohner *et al.* (2006) Accounting-ansatze im bereich des grid-computing. [Online]. Available: http://tinyurl.com/87goucy

A SECURE CLOUD-ENABLED WIRELESS SENSOR NETWORK PLATFORM

Felix Njeh and Dr. Bo Yang
Department of Computer Science
Bowie State University, Bowie, Maryland 20715

Abstract *- Nowadays, Wireless Sensor Networks (WSN) have become a key enabling technology and consequently demands a secure, distributed and globally available network of sensors. Various technologies have evolved and are used to facilitate the deployment and use of WSNs. The rapid development in Micro-Electro-Mechanical Systems (MEMS) technology has facilitated the development of smart and highly capable sensors to solve a multitude of real world problems.*

Keywords: *Wireless Sensor Networks, Cloud Computing, SOA*

1 Introduction

The emergence of the Internet, for example has enabled the deployment of sensor network applications accessible through the World Wide Web. Cloud Computing has become another technology of choice for many due to the numerous benefits it brings to the Information Technology industry. Grid Computing uses the concept of parallel processing to introduce a platform on which a computationally intensive problem could be solved by harnessing the unused compute power of many computer resources distributed globally. The Internet of Things is another paradigm that extends the capabilities and use of WSNs.

With these concepts, we propose a Service-Oriented Architecture (SOA) in which WSNs distributed world-wide can be interconnected into a secure global network of sensors. This secured and unified platform will provide the end-user with a large choice of virtual configurations for sensing, monitoring and analytics capabilities. An experimental testbed has been setup and a feasibility study of the model will be conducted and the results analyzed.

2 Background

2.1. Wireless Sensor Networks

A Wireless Sensor Network (WSN) consists of a large number of low-cost, low-power, multifunctional and resource-constrained sensor nodes with each sensor node consisting of sensing, data processing, and communicating components; these nodes can operate unattended for long durations. Sensor nodes perform measurements of some physical phenomena, collect and process data, communicate with other peers or a central information processing unit, the sink. These nodes are capable of sensing various phenomena, such as Pressure, Temperature, Humidity, Position, Velocity, Acceleration, Force, Vibration, Proximity, Motion, Biochemical agents, and more.

There are several characteristics that influence the design and use of WSNs. Some of such considerations include: robustness, fault tolerance, self-configuration, energy efficiency and lifetime maximization. Standards have been developed to remediate some of the issues in Sensor Networks. A good example is the ZigBee/IEEE 802.15.4 standard.

2.2. ZigBee/IEEE 802.15.4

ZigBee is a specification for a reliable, low-cost, low-power consumption, self-organizing, ad-hoc, mesh networking standard. It is based on the IEEE 802.15.4 standard for Low-Rate Wireless Personal Area Networks. ZigBee operates in unlicensed bands - 2.4 GHz Global Band at 250kbps, 868 MHz European band at 20kbps and 915 MHz North American band at 40kbps. ZigBee was initiated when it became clear that Wi-Fi and Bluetooth technologies were going to be unsuitable for many wireless applications. The standard provides low-cost, long battery life, secure wireless networking for tracking, control and monitoring. Compared to other wireless standards, ZigBee connects the widest variety of devices. With these capabilities, ZigBee can be used for military applications, industrial control, embedded sensing, medical data collection, smoke and intruder warning, building automation, home automation, and more.

Market Name	ZigBee®	---	Wi-Fi™	Bluetooth™
Standard	802.15.4	GSM/GPRS CDMA/1xRTT	802.11b	802.15.1
Application Focus	Monitoring & Control	Wide Area Voice & Data	Web, Email, Video	Cable Replacement
System Resources	4KB - 32KB	16MB+	1MB+	250KB+
Battery Life (days)	100 - 1,000+	1-7	.5 - 5	1 - 7
Network Size	Unlimited (2^{64})	1	32	7
Maximum Data Rate (KB/s)	20 - 250	64 - 128+	11,000+	720
Transmission Range (meters)	1 - 100+	1,000+	1 - 100	1 - 10+
Success Metrics	Reliability, Power, Cost	Reach, Quality	Speed, Flexibility	Cost, Convenience

Table 1.0: Comparison of ZigBee and other wireless standards
Source: http://www.zigbee.org/About/FAQ.aspx

The IEEE 802.15.4 standard provides specifications for point-to-point or point-to-multipoint networks. The nodes in the network are either full function devices or reduced function devices.

WSNs have been studied for a good length of time resulting in improved sensors or sensor-enabled systems. These WSNs have evolved and have become quite popular that they spand various applications and industries. WSNs have been used in health, commercial, military applications including battle-field surveillance and enemy tracking, home automation and security, habitat and environmental monitoring. This illustrates the significance of WSNs in these industries.

2.3. Cloud Computing

Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models [2]. Figure 1 below illustrates the service and deployment models.

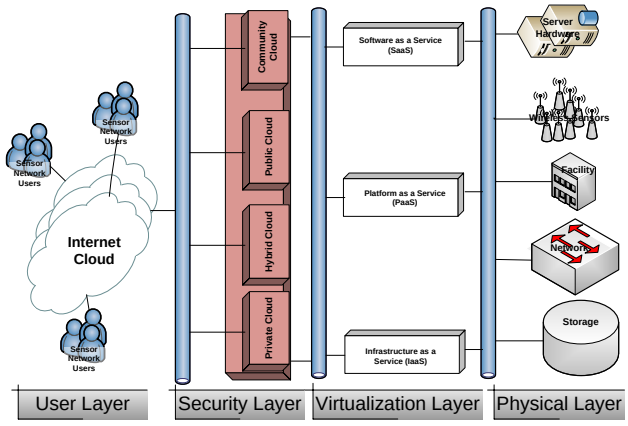


Figure 1: The integrated Cloud WSN Platform

Cloud Computing consists of three service models, Software as a Service (SaaS), Platform as a service (PaaS), and Infrastructure as a Service (IaaS); and four deployment models, Private cloud, Hybrid cloud, Public cloud and Community cloud.

Service Models

The three service models include Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

2.3.1. Software as a Service (SaaS)

This is a software delivery model in which the provider gives customers on-demand access to the applications hosted in a cloud infrastructure. The infrastructure is managed by the provider while the consumer has only limited user-specific application configuration settings. SaaS is increasingly becoming a common delivery model for most business applications. The consumer usually pays a subscription fee instead of a licensing fee.

2.3.2. Platform as a service (PaaS)

This service delivery model allows the customer to rent the cloud infrastructure (virtualized servers and associated services) to run consumer-created or acquired applications or to develop and test new ones. The infrastructure is managed and controlled by the provider; the consumer has some control over the deployed applications and possibly application hosting environment configurations.

2.3.3. Infrastructure as a Service (IaaS)

IaaS is a delivery capability in which the consumer provisions processing, storage, networks, and other fundamental computing resources. The consumer can deploy and run arbitrary software (operating systems and applications) but does not manage or control the underlying cloud infrastructure.

Deployment Models

The deployment models include Private cloud, Hybrid cloud, Public cloud and Community cloud.

2.3.4. Private Cloud

With this model, the internal or corporate cloud infrastructure (systems and services) is operated solely for an organization. This gives the organization better management and control over their data and systems. It is also considered a proprietary network or a data center that supplies hosted services to a limited number of people.

2.3.5. Hybrid Cloud

A Hybrid Cloud is made up of at least one private cloud and at least one public cloud. An example is when a vendor has a private cloud and forms a partnership with a public cloud provider, or a public cloud provider forms a partnership with a vendor that provides private cloud platforms. In other instances, the organization owns and manages some of the cloud resources internally while others are made available

externally. A hybrid cloud provides the consumer the best of both worlds.

2.3.6. Public Cloud

A public cloud is a cloud model in which the cloud provider makes the cloud infrastructure available to the general public; and is owned by the cloud provider. This model is also considered as external cloud. It has several advantages to include: lower cost of deployment, scalability and efficient use of resources (since you only pay for what you use).

2.3.7. Community Cloud

A Community Cloud allows the cloud infrastructure to be shared by several organizations and supports a specific community that has shared concerns. This model can be managed by the organizations involved or a third party, and may exist on premise or off premise.

3 Related Works

In (Briefings, 2009), the authors evaluated the potentials of the integration of Wireless Sensor Networks and Cloud Computing and drew the conclusion that this marriage is not only possible but makes it more feasible to collect, analyze and share sensor data. A content-based publish/subscribe platform is proposed; where the WSNs publish the data collected while the subscribers consume the data. Services are delivered to consumers continuously, periodically, event-based or query-based.

(Hassan & Korea, 2009) propose "a content-based pub-sub model which simplifies the integration of sensor network with cloud based community centric applications". In this study, the provider also publishes data and consumers access the data and applications through a cloud infrastructure on-demand from anywhere. A Pub/Sub broker monitors, processes and delivers events to registered users through SaaS applications. An event matching algorithm matches subscribed users to events of interest. A simulation was done to test this algorithm but with no use of any real-world data. The literature does not clearly state how data transitions through the different layers or stacks of the CC model.

WSNs generate huge amounts of data. (Bose & Liu, n.d.) believe that sensor data will continue to increase exponentially, with the side effect that traditional platforms cannot sustain this increase. They suggest that CC is a viable answer to this problem. (Kurschl & Beer, 2009) also identify the massive amounts of data generated by WSNs as one of the key motivations to amalgamate sensor networks with CC, and agree that with other studies that the resulting platform will provide for interoperability with other vendors' sensors, scalability of system resources (storage, compute, network, etc.), accessibility to sensor data from any location worldwide. A model is proposed that is based on pipes and filters; where the filters process and transform input data while pipes provide an interconnection mechanism for filters. Base services identified include: Sensor Data Management, Runtime for Filter Chains, Filter Chain and Filter Management, Visualization, and Notification Service. A prototype Energy Monitoring shows a Zigbee based WSN for gathering energy consumption data and dispatching into the Cloud.

(Benson, Dowsley, & Shacham, 2011) examines the issue of geolocation of data in the Cloud and propose a method of efficiently retrieving data stored as multiple copies in geographically disparate datacenter locations. For critical sensor applications, it is important to know the location of data in the Cloud to ensure accessibility and security. It is important to route application requests to the nearest data center in order to minimize response time.

In (Hauswirth & Decker, 2007), the authors discuss the unification of the real and the virtual worlds using sensor technologies and the Semantic Web; with applications in monitoring, manufacturing, health, tracking and planning. (Melchor, n.d.) proposes a toolkit for sensor-cloud integration. Both approaches fail to identify or leverage the features of CC as enabling characteristics.

4 Motivation

Sensor networks are found in disparate locations all around the world supporting a myriad of applications. For example, in military applications we find robots carrying sensors in remote tactical environments to monitor or track the enemies.

With the recent advancements in cloud computing, we realized the importance of defining a standard architecture that unifies these disparate wireless sensor networks. This will provide for global connectivity and accessibility to these sensor networks. Figure 1 below illustrates sensor networks in disparate locations around the world. In Figure 3, we illustrate the experimental network setup showing a WSN connecting to the iDigi Cloud from which web client applications provide sensor data or information to approved and authenticated users.

5 The Integrated Cloud WSN Platform

Figure 1 shows the integration between the Cloud and Wireless Sensor Networks. WSNs are often omitted as part of the Cloud infrastructure. In this paper we strive to fit WSNs into the Cloud architecture and conduct research to find out how well the WSNs fit with other famous IaaS components. In order to perform these findings, we look at performance metrics of worldwide Cloud providers to gauge if the existing

platform gives room for the introduction of other infrastructure components.

Our goal is to leverage the beneficial features of Cloud Computing which include on-demand self-service, broad network access, resource pooling (location independence), rapid elasticity, measured service, massive scale, homogeneity, virtualization, resilient computing, geographic distribution, service orientation and advanced security technologies.

5.1. Cloud Performance Metrics

Performance and availability are key metrics when considering any Cloud application. CloudSleuth[1] provides us with the tools to measure these metrics. From the available data we see that Cloud providers in the United States and Europe provide reasonable response time of less than 3 seconds (see Figure 2a below).

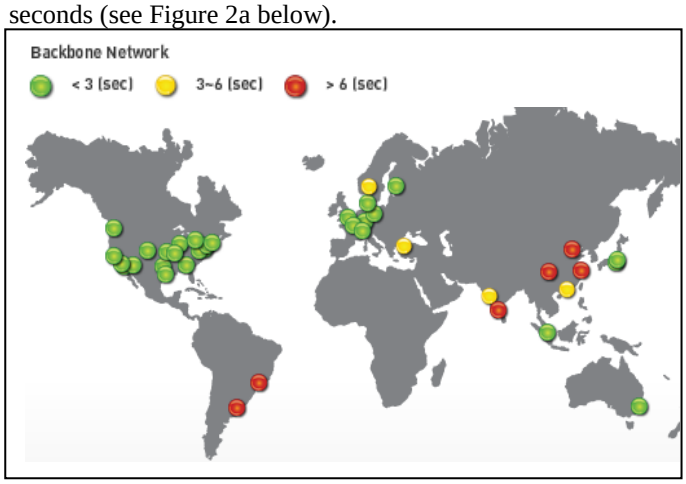


Figure 2a: Performance of worldwide Cloud Providers
Source: https://cloudsleuth.net

Figure 2b: Performance of worldwide Cloud Providers
Source: https://cloudsleuth.net

In terms of availability, most of the major cloud service providers worldwide were available almost 100% of the time, this is quite promising and gives us confidence that our integrated platform will survive in real applications.

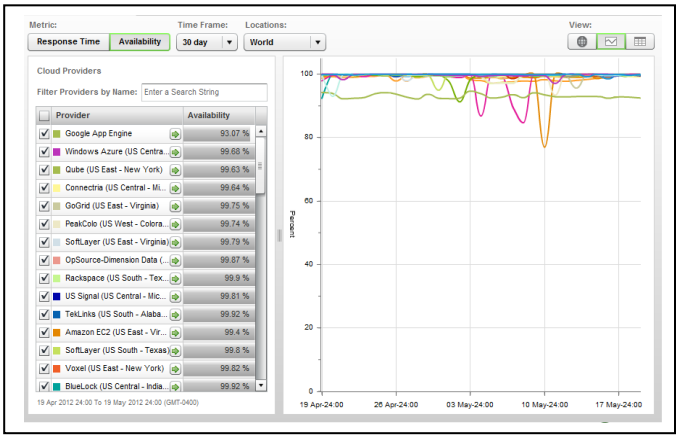


Figure 2c: Performance of worldwide Cloud Providers
Source: https://cloudsleuth.net

6 Experimental Setup

In our experiment, we setup a WSN that monitors light, temperature, power draw, and control. The WSN connects to a Digi[2] gateway which sends all traffic into the iDigi cloud through an access gateway.

The setup uses the iDigi[3] Gateway Development platform which consists of the Digi ConnectPort® X4 (ZigBee Ethernet gateway), the XBee Smart Plug™, the XBee Sensor and the ESP Integrated Development Environment (IDE) for iDigi Dia/Python development. The setup provides tools to setup a ZigBee network, design, test and upload applications, make web service calls and provide connectivity to the Internet. Web applications can be designed to access real-time sensor data. The experimental testbed is illustrated below.

Figure 3: iDigi Gateway Development kit
Source: http://www.digi.com/

[1] *https://cloudsleuth.net*

[2] Digi International Inc. develops networking products and solutions.

[3] http://www.digi.com/

This platform is called the iDigi Device Cloud. It facilitates the creation and deployment of device applications. This iDigi Cloud platform provides desirable features such as performance, reliability, scalability, security, seamless device and application integration.

6.1. Data Collection and Visualization

In our prototype, the wireless sensors collect data which is retrieved and transmitted wirelessly through the XBee network. The portable battery-powered sensors can be dropped into an environment of interest for data collection and communication.

The information collected is displayed as a dashboard, graphs or charts. Data or information collected is displayed through a web interface or on a smart device.

- **Light** - the XBee Smart Plug Ambient light sensor measures indoor light intensity.
- **Temperature** - the XBee Sensor is a battery-powered sensor that measures temperature and light.
- **Power draw** - the XBee Smart Plug detects current draw from the AC socket (standard AC 110V, 3-prong).
- **Control** – The XBee Smart Plug provides power control to the user outlet. It is configured through the XBee module's digital I/O channel, D4 which can be set to high or low to turn power on or off.

Our experimental setup was limited to one sensor network. For that reason we extrapolate the capability of the experiment by looking at the behavior of the worldwide cloud content delivery network (CDN). Figures 2a Figure 2b and Figure 2c above show availability and response time for worldwide cloud providers.

7 Conclusions

In this paper we propose a new type of platform which integrates and leverages the features of two key technologies, Cloud Computing and Wireless Sensor Network. This unified platform leverages the key benefits of two core technologies to provide a secure platform through which sensor data can be processed and assimilated. We ran Cloud performance tests for availability and response time to weigh Cloud performance through worldwide Cloud providers' backbones. The performance results indicate the potentials of Cloud Computing and further give us confidence in our endeavor to merge the two technologies together to provide a secure worldwide cloud-enabled WSN services.

In future works, we will address the lack of widely accepted open standards and interoperability between sensor-cloud platforms. Data analytics and security also require more research to highlight issues affecting the platform.

8 References

[1] Peter Mell, Tim Grance. "Effectively and Securely Using the Cloud Computing Paradigm", NIST, Information Technology Laboratory.
[2] NIST Cloud Computing Standards Roadmap. "NIST CCSRWG – 070 Eleventh Working Draft". May 2, 2011
[3] iDigi® Gateway Development Kit. " Development Kit for ZigBee Gateways and iDigi Platform"
[4] Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal: "Wireless sensor network survey. Computer Networks" 52(12): 2292-2330 (2008)
[5] The ZigBee Alliance, http://www.zigbee.org/About/AboutTechnology/Standards.aspx, 2011 ZigBee Alliance
[6] Theoretical and practical aspects of military wireless sensor networks"
[7] Madoka Yuriyama, Takayuki Kushida, "Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing", IBM Research - Tokyo, March 17, 2010
[8] Peter Mell and Tim Grance. "The NIST Definition of Cloud Computing". 10-7-09
[9] Duane Nickul et al. "Service Oriented Architecture (SOA) and Specialized Messaging Patterns"
[10] Peter Mell, Tim Grance. "The NIST Definition of Cloud Computing. NIST Special Publication 800-145
[11] Mark Gaynor, Steve Moulton et al. "Integrating Wireless Sensor Networks with the Grid", AUGUST 2004
[12] Cuno Pfister. "Getting Started with the Internet of Things". 2011. Published by O'Reilly Media, Inc.
[13] Aditya Goel and Ajaii Sharma. "Performance Analysis of Mobile Ad-hoc Network Using AODV Protocol". International Journal of Computer Science and Security (IJCSS)
[14] Briefings, S. (2009). Can We Plug Wireless Sensor Network to Cloud? cloud computing pinnacle of IT Infrastructure democratization, 7(7), 33. Retrieved from http://www.infosys.com/infosys-labs/publications/documents/cloud-computing.pdf#page=35
[15] Hassan, M. M., & Korea, S. (2009). A Framework of Sensor - Cloud Integration Opportunities and Challenges, 618-626.
[16] Kurschl, W., & Beer, W. (2009). Combining cloud computing and wireless sensor networks. Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services - iiWAS '09, 512. New York, New York, USA: ACM Press. doi:10.1145/1806338.1806435
[17] Bose, B. S., & Liu, R. (n.d.). Cloud Computing Complements Wireless Sensor Networks to, 10-11.
[18] Kang, Y., Zhou, Y., Zheng, Z., & Lyu, M. R. (2011). A User Experience-Based Cloud Service Redeployment Mechanism. 2011 IEEE 4th International Conference on Cloud Computing, 227-234. Ieee. doi:10.1109/CLOUD.2011.20

[19] Benson, K., Dowsley, R., & Shacham, H. (2011). Do you know where your cloud files are? Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11, 73. New York, New York, USA: ACM Press. doi:10.1145/2046660.2046677

[20] Hauswirth, M., & Decker, S. (2007). Semantic Reality – Connecting the Real and the Virtual World Position Paper, 1-4.

A Novel Heuristics based Energy Aware Resource Allocation and Job Prioritization in HPC Clouds

Thamarai Selvi Somasundaram[1], Kannan Govindarajan[1], T.D. Rohini[1], K. Kavithaa[1], R. Preethi[1]
[1]Department of Computer Technology, Anna University, Chennai, Tamil Nadu, India
Email: stselvi@annauniv.edu, kannan.gridlab@gmail.com, rohinitd@gmail.com, kavithaa.aswi@gmail.com, preethi.r@gmail.com
Website: www.annauniv.edu/care

Abstract - *Cloud Computing provides the computational, storage, network and database resources to the consumers in a pay-as-per usage mode. In recent years, the data centers play a major role in hosting the cloud applications in the cloud infrastructure. The data centers are consuming huge electrical power and emits large amount of carbon footprint. It is essential to incorporate the Energy Efficient Resource Management (EERM) mechanism to control the electric power consumption and reduce the carbon footprint emission. EERA comprises of matching the user application requests with available cloud resources and allocating the user application requests to the matched cloud resources in an efficient manner. This paper mainly focused on proposing a novel heuristics based Energy Aware Resource Allocation (EARA) mechanism to allocate the user applications to the cloud resources that consumes minimal energy and incorporating the prioritization mechanism based on the deadline. It is simulated using the CloudSim toolkit and by generating High Performance Computing (HPC) type of application requests with the generated Eucalyptus based private Cloud environment. The results prove the effectiveness of the proposed mechanism in the cloud infrastructure by maximizing the number of users completed their applications within deadline and minimize the energy consumption in the cloud resources.*

Keywords: Cloud Computing; Resource Management; Energy Efficiency; Eucalyptus; Heuristics.

1. Introduction

Cloud Computing [1] provides ondemand computing in terms of application, platform and infrastructure in a pay as per usage mode. The Cloud service models are categorized into three major types based on the applications, platform and infrastructure namely SaaS, PaaS and IaaS. The IaaS service delivery model is plays a major role in hosting the PaaS or SaaS in the data centers. The four major players of the Cloud are (i) Cloud Users (CUs) (ii) Cloud Service Providers (CSPs) (iii) Cloud Applications (CAs) and (iv) Cloud Service Brokers (CSPs). The CUs are submitting the jobs with software, hardware and QoS parameters. The requirements are varied in terms of hardware (Processor Speed, RAM Memory, Bandwidth and etc.), software (Java 1.6, apache tomcat-5.0.27, MPICH-1.2.7, Charm++ 3.x and etc.) and QoS (deadline, throughput and etc.). The CSPs are managing the huge datacenters for the purpose of computation, storage and etc. CSPs are managing the physical resources to host the Cloud applications in the virtual resources. The CRPs have to consider the user required parameters when they are selecting the resources to run the user applications. In this scenario, CRP's are facing the problem in the selection of resources to run the application. The CAs may be of different types such as web sites, web applications, high-performance computing applications and etc. In recent years, the huge datacenters are popular for hosting the CAs. The CSBs acts as the mediator between the CUs, CSPs and user's CA, so it is essential to incorporate the efficient Resource Management (RM) technique. RM is the challenging task due to the dynamic nature of Cloud Computing environment and ondemand user requirement. It mainly consists of five major functionalities are shown in Figure 1 and they are (i) Matchmaking the user job requests with available cloud resources (Resource Discovery) (ii) Allocating the user job requests with available cloud resources in an efficient manner (Resource Selection) (iii) Provisioning of virtual resources in the selected resources (Resource Provisioning) (iv) Running the user jobs in the created virtual resources (Running Application) and (v) Monitoring the running applications (Monitoring Applications).

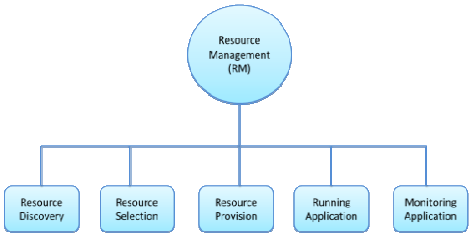


Figure 1: Functionalities of Resource Management (RM)

Nowadays scientific applications are becoming complex and it is composed of various application components and it requires heterogeneous set of resources. The High Performance Computing Clouds (HPCCs) or Science Cloud provides a great platform for researchers to test their ideas using simulation process. The scientific applications requires large amount of computational steps and customized execution environment and also it processes and generates

huge amount of data. Cloud Computing provide benefits to the scientific applications using the concept of resource provisioning through virtualization technology and it provides different operating systems with different software configurations.However, it is very difficult to incorporate the efficient RM mechanism in every cloud provider site and also it is very tedious for the cloud user's to search the suitable cloud resources that are geographically distributed in nature to run their applications. These drawbacks can be achieved by integrating the efficient RM mechanism in CSB to efficiently manage the user requests and Cloud resources. In recent days, the data centers are consuming more amount of electrical energy and emitting large amount of carbon foot prints. The high energy consumption increases the running cost of the data centers. So, it is essential to decrease the high energy consumption of the data centers that will maximize the revenue of the resource providers, reduce the carbon emission and running cost of the data centers. To achieve the above objectives, we have proposed the energy efficient resource management mechanism that is mainly aimed to improve the maximum number of users completed their jobs within deadline and minimize the consumption of energy in the datacenters. These two factors influence the revenue of the cloud resource providers in an impressive manner. The maximum number of users completed within the deadline is achieved by giving more priority to the jobs nearer to the deadline. The resource selection process is carried out by employing the optimization algorithm of Particle Swarm Optimization [2]. The proposed approach selects the resources that consume less energy. In addition to that, it accommodates or allocates the maximum number of user requests in the datacenters that will increase the revenue of the service providers and increase the utilization of the resources. In brief the contributions of the research work are summarized below.

a. To design and develop the matchmaking algorithm for matching the HPC user requests with available cloud resources. (A)
b. To design and develop a Particle Swarm Optimization based Energy Aware Resource Allocation (PSOEAR) mechanism for allocating the user requests to the Cloud resources in a near optimal manner. (B)
c. Integration of (A) and (B) with Cloud Service Broker (CSB) for matchmaking, allocating and provisioning for the HPC user requests. (C)
d. The proposed work is simulated and the results have been analyzed in the simulation based cloud environment. (D)

The rest of the paper is organized as follows: Section 2 presents the high-level architecture of proposed framework; Section 3 presents the proposed system model and its description. Section 4 describes the simulation results and its inferences observed from the simulation. Section 5 describes the related works closely related to our proposed work. Section 6 concludes the proposed work and explores the feasibility of future work.

2. Proposed High-Level Architecture

The proposed high-level architecture for cloud resource management framework with energy aware allocation is shown in Figure 2. It consists of the five major components and the functionalities of each component are described in detail.

2.1 Request Handler & Match Maker

The user submits the job requirements as an XML file or through Graphical User Interface (GUI). The parser in the request handler parses the job requirements and the parsed information is updated in the User Job Request Pool (UJRP). Once the job requests are parsed it invokes the Match Maker to match the user job requirements with the available cloud resources. The matchmaker component filters the potential resources that are capable of creating virtual resources and run the job. Finally, it generates the matched resource list and the generated list is sent to the Particle Swarm Optimization based Energy Aware Resource Allocator (PSOEARA). The matchmaking algorithm for HPC job request is shown below.

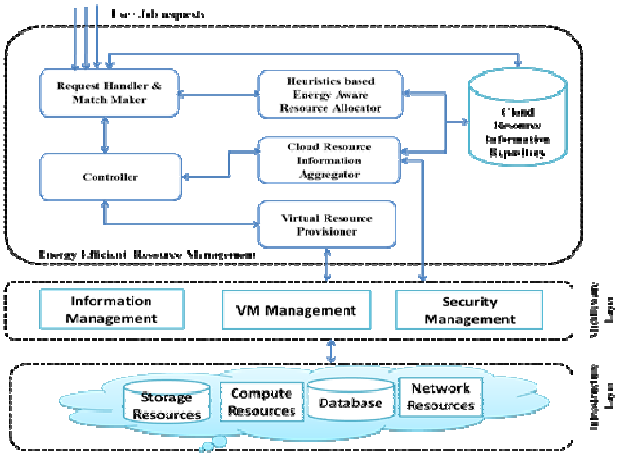


Figure 2: Cloud Resource Management Framework for Energy Aware Resource Allocation

Algorithm 1 Matchmaking Algorithm

Input : Fetch the job requests with hardware, software and QoS requirements.
Output: Matchmaking the job requests with available cloud resources and generate the matched resource list.
Step 1 : Submit the job requests with requirements, parse the requirements and store it in the Broker Queue (BQ).
Step 2 : Match the job requirements with available cloud resources and generate the resource list that are capable of creating the virtual instances and run the job.

Step 3 : For (I= 1 to N 'Job Request')
{
For (J = 1 to M 'Datacenters')
{
For (K=1 to O 'Hosts')
{
Match the job requirements with available cloud resources.
Generate the matched host list that is capable of satisfying the user job requests.
}
Generate Matched Datacenter List that has capable hosts to create the virtual instances and run the jobs.
}
}
Step 4 : End

2.2 Particle Swarm Optimization based Energy Aware Resource Allocator (PSOEARA)

PSOEARA is implemented with Particle Swarm Optimization (PSO) and Energy Aware Resource Allocation algorithm. PSO is a population based stochastic optimization technique. It is initialized with a group of random particles or solutions. Each particle is updated by the two best values known as pbest (the personal best) and gbest (the global best) in every iteration. Pbest represents the best solution achieved by one particle and gbest represents the best value obtained by any particle in the population. PSOEAR mainly consists of three fold processes such as (1) Initial job assignment to the matched resource list (2) Calculation of Expected Completion Time (ECT) and Energy Consumption (EC) of the job (3) Final job assignment to the selected cloud resource.

2.2.1 Initial job assignment to the matched resource list - PSOEARA takes the batch of jobs as input and each job is randomly allocated to the matched resource list. In each assignment the ECT and the EC is computed for each job.

2.2.2 Calculation of ECT and EC of the jobs - The ECT is computed using the Equation (1).

$$ECT_{ij} = ST_{ij} + BT_{ij} + EET_{ij} \quad - (1)$$

$$EETij = \text{Job Length / MIPS of VM} \quad - (2)$$

$$\text{Job Length} = \text{Million Instructions (MI) } /60 \quad - (3)$$

$$\text{MIPS of VM} = \text{Job Length/Deadline} + x \quad - (4)$$

Where ECT_{ij} represents the expected completion time of job i on resource j, ST_{ij} represent the start time of job i on resource j, BT_{ij} represents the boot time of the virtual instances for the ith job on resource i, EETij represents the estimated execution time of job i on resource j. The EC is computed using the Equation (5) and it is given below.

$$EC_{ij}\,(\text{in watts}) = \sum_{i=1,j=1}^{N,M} RMIP_i / AMIP_j * 100 \quad - (5)$$

Where EC_{ij} represents the energy consumption of the particular job, RMIPS represents the Requested Millions of Instructions per Second for job i and AMIPS represents the Available Millions of Instructions Per Second in resource j. Where i= 1 To N represents number of VMs, j=1 To M represents the number of hosts.

2.2.3 Final job assignment to the selected cloud resource - In the final assignment process the jobs are assigned to the cloud resource which completes the job within the deadline and consumes less energy.

Algorithm 2 PSOEARA Algorithm

Input: Fetch the job requests with matched host list and datacenter list.
Output: Optimal selection of cloud resources that consumes less energy and completes the job requests within deadline.

Step 1 Get the 'N' number of job requests with matched datacenter list and the host list.
Step 4 For (I= 1 to N 'Job Request')
{
For (J=1 to M 'Virtual Machines' of each job)
{
For (K = 1 to O 'Matched Datacenter List')
{
GetMatchedHostList ( );
For (L = 1 to P 'MatchedHostList')
{
Compute the Expected Completion Time (ETC) using the Equation (1);
Compute the Present Energy Consumption (PEC) using the Equation (5);
}
If (K==0) {
Pbest = PEC;
Chosen DC= DC (0)
}
Else
{
If (PEC < Pbest && ECT < Deadline) //Compare the energy consumption difference
{
Gbest = PEC;
Chosen DC = DC (K);
}
Else If
{
Gbest = Pbest;
}
}
}
}}

2.3 Cloud Resource Information Aggregator

This work is extension of our previous work Cloud Monitoring and Discovery Service (CMDS) [3]. CMDS will aggregate the cloud resource information such as processor, memory and network using the external information providers Ganglia, NWS and our own user-defined script. CMDS is extended to aggregate the energy and load information from the cloud resources. The collected information is updated in the Cloud Resource Information Repository (CRIP).

2.4 Virtual Machine Provisioner

It is mainly responsible for interacting with Cloud middleware to provision the virtual machine instances. It fetches the virtual machine request with the parameters of type of instances, ram capacity and number of instances to be created to run the job.

2.5 Virtual Machine Energy Monitor

It is running in the cloud resources and it collects the energy consumed by the virtual machine instances. The collected information is updated to the VM Energy Aggregator.

3. Implementation Details

In this paper we have simulated and compared the proposed PSO based energy aware resource allocation with DVFS and Round Robin. The simulation is carried out using the CloudSim [4] toolkit. The CloudSim source code is analyzed and incorporated with the major modifications in the classes DatacenterBroker.java, Host.java, Cloudlet.java and newly added PSOEARAllocationPolicy.java. The available resources in the cloud environment are represented as 'A_{CR}'. Each cloud resource has 'm_h' number of hosts and every host is capable of hosting/creating 'n_v' number of virtual machine instances. The proposed system is accessed by 'm_u' number of users the users are arrived at a regular interval of 'I' in a Poisson distribution manner. Each user job request will require 'N' number of nodes, 'M' amount of RAM memory, 'P' amount of processor speed. The sample HPC job request is shown in Table 1 and it is generated by doing the modifications in the Cloudlet.java class. We have generated the job requests for three types of HPC applications such as NAMD [5], Clustal [6] and FASTA [7]. The Host.java class is modified and the generated Eucalyptus based private cloud resources is shown in Table 2.

Table 1: Simulated HPC User Job Requests

User Name	Job Type	Number Of Nodes	Processor Speed (MHZ)	RAM Memory (MB)	Disk Memory (GB)
stselvi	NAMD	5	2200	512	10
preethi	CLUSTAL	10	2000	1024	20
Rohini	NAMD	5	2200	1024	10
kavitha	FASTA	5	2000	512	20

Table 2: Simulated Eucalyptus based Private Cloud Resources

Resource Name	Number of Hosts	Processor Speed (MHZ)	Harddisk Memory (GB)	RAM Memory (MB)	Types of VM Instances	Number of Instances can be created
cloudserver1.care.mit.in	10	2800	120	2048	(128) m1 small, (256) c1 medium (512) c1 large	6 4 2
cloudserver2.care.mit.in	4	3000	120	2048	(128) m1 small, (256) c1 medium (512) c1 large	6 4 2
cloudserver3.care.mit.in	4	3000	120	2048	(128) m1 small, (256) c1 medium (512) c1 large	6 4 2

4. Simulation Details and its Inferences

The experimentation is carried out by generating a Cloud Service Broker (CSB) with multiple Cloud Service Providers (CSPs). We have considered 5 CSPs each CSP maintains one datacenter. Each datacenter is generated with 1000, 2000, 3000, 1000, 2000 cloud hosts respectively. The cloud hosts has different capabilities in terms of number of processors, processor speed, ram speed, hard disk memory, bandwidth, latency, type of hypervisor and etc. The job request is generated randomly using the random access model that generates the job requests as Cloudlets in the range of 1000 to 10000 in the random fashion. The job parameters such as length of job (J_A), job arrival rate (A_A) and number of Job requests (N_J) also generated. The job requests are mapped with available cloud resources for creating virtual instances and running the applications. The experimental setup is shown in Figure 3. The simulation has been carried out for type of use cases (i) Use Case 1 - Resource Allocation within datacenter (ii) Use Case 2 - Resource Allocation across datacenters. The performance measures such as number of users completed within deadline, energy consumption of the datacenters are represented figuratively.

(i) Use Case 1 - Resource Allocation within datacenter – In this use case, the resource allocation policy finds out the suitable resources for every job requests that consumes less energy within the single datacenter. If the job requests could not able to satisfy within single datacenter the resource allocation policy sends the message to the broker COULD NOT be ABLE TO CREATE required VIRTUAL MACHINE WITHIN SINGLE DATACENTER". The broker invokes the resource co-allocation policy to satisfy the job request could not be processed in the single data center.

(i) Use Case 1 - Resource Allocation across datacenters – In this use case, the resource allocation policy finds out the suitable resources for every job requests that consumes less energy across the datacenters. If the job requests could not

able to satisfy across the datacenters the resource allocation policy sends the message to the broker "COULD NOT be ABLE TO CREATE ENOUGH VIRTUAL MACHINES ACROSS THE DATACENTERS". The broker rejects the request and notifies to the user.

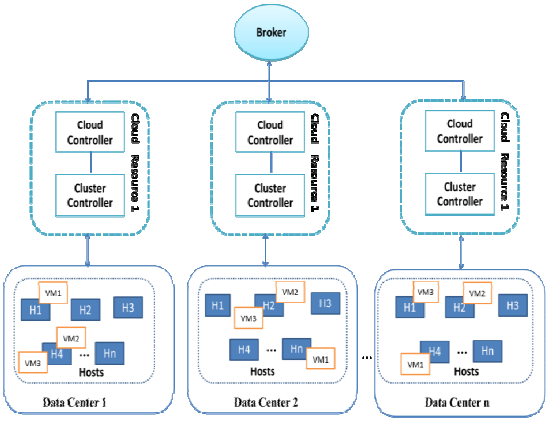


Figure 3: Experimental Setup

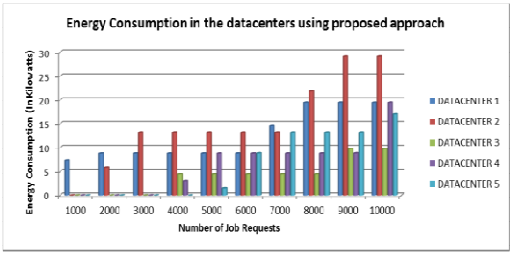


Figure 4: Energy Consumption using PSOEARA

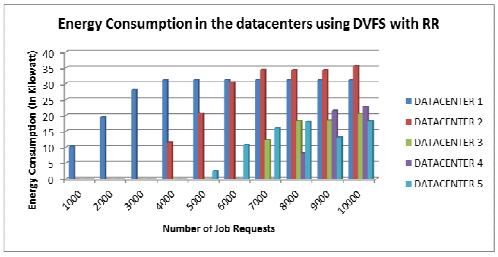


Figure 5: Energy Consumption using DVFS with RR

The job requests are generated in the order 1000 to 10000 cloudlets and the job rejection rate of PSOEARA is compared DVFS with RR. The proposed mechanism has the job rejection rate with an average of 10% and the RR has the job rejection rate of 35%.

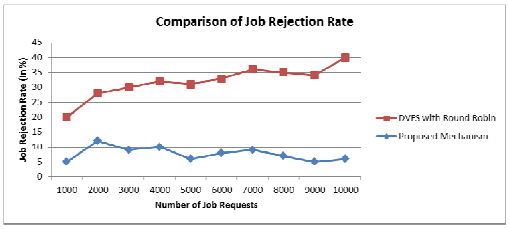


Figure 6: Comparison of Job Rejection Rate PSOEARA versus DVFS with RR

5. Related Works

Thamarai Selvi et. al [8] has proposed and implemented a Java based architectural framework to schedule and support the virtual resource management in the Grid environment. It handles the various scheduling scenarios of Physical, Coalloc, Virtual Cluster and etc. Eucalyptus [9] is the open source cloud middleware and it consists of cloud controller, cluster controller, node controller and storage controller. These components are arranged in a hierarchical fashion and eucalyptus has incorporated with Greedy, Round Robin, and Power Save scheduling algorithm. These three scheduling algorithms do not select the resource in a near optimal manner and also it does not have considered the priority. OpenNebula (2005) [10] is an open source cloud middleware that creates virtual machines in a physical cluster and its main focus is virtual resource management in the infrastructure. It has incorporated with rank based scheduling approach and does not consider the energy efficiency and deadline parameters and it is mainly working in the host level. Das et. al. [11] has built the commercialized computing system called Unity; the main aspects are application environment centric, computation of optimal configuration of resources in the datacenters, absence of the cost of components during the problem formulation.

Biao Song [12] has discussed the heuristic based task selection and allocation framework in cloud environment. They have classified the resource allocation problem into two things such as heavy workload and light workload. In the heavy workload scenario they have consider the Quality of Service (QoS) is their major focus and in the light workload scenario the resource utilization is their main focus. They maintained the threshold value based on that value they will allocate the tasks to the resources with an objective of increasing the resource utilization. But they have not discusses anything about the energy efficiency and deadline of the job requests. Hai Zhong et. al [13] proposed the optimized resource scheduling for open-source cloud systems using the Improved Genetic Algorithm (IGA). They have derived the fitness algorithm using the dividend policy mechanism. They have compared their proposed approach with First Fit and RR. They claimed that their proposed algorithm increases the utilization of the cloud resources and saves much energy. The major difference of our work from their work is they have not

discussed anything HPC job requests, energy efficiency in detail and deadline of the job requests.

6. Conclusion and Future Work

The datacenters are consuming huge amount of electric power and emits large amount of carbon footprints that pollutes the environment. This paper mainly aimed to provide an efficient resource management mechanism in the cloud service broker. It handles the user job requests as HPC applications based on the user required parameters it selects the cloud resources in a near optimal manner using the heuristics based energy aware resource allocation mechanism. The proposed work minimizes the consumption of power and maximizes the revenue of the CRP's. The main contributions of the proposed work are summarized as follows: ability of handling the HPC job requests in Cloud Service Broker, matchmaking the user job requests and allocating the user job requests to the available cloud resources that consumes less energy in an optimal manner and completes the job within deadline. It increases the maximum number of jobs completed within the deadline and minimize the consumption of energy in the datacenters. These two factors influence to maximize the revenue of the cloud resource providers. The proposed work is simulated using the CloudSim toolkit and compared with the most well-known algorithm DVFS using Round Robin. The results are evident that proposed work minimizes the consumption of energy in the datacenters and maximizes the number of users completed within the deadline.

As a future work the proposed work to be tested in the Eucalyptus based real private cloud environment for HPC applications. And also, it can be extended for decentralized mode incorporated with load balancing mechanism that will enhance the scalability and utilization of cloud resources further.

ACKNOWLEDGMENT

The authors sincerely thank the Ministry of communication and Information Technology, Government of India, for financially supporting the Centre for Advanced Computing Research and Education of Anna University Chennai, India in this project

REFERENCES

[1] NIST, National Institute of Standards and Technology (2011), http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf.

[2] Particle Swarm Optimization (PSO), http://www.swarmintelligence.org/.

[3] Thamarai Selvi Somasundaram, Kannan Govindarajan, "Cloud Monitoring and Discovery Service (CMDS) for IaaS resources", has been accepted in ICoAC 2011.

[4] CLOUDSIM, http://www.cloudbus.org/cloudsim/.

[5] NAMD, http://www.ks.uiuc.edu/Research/namd/.

[6] CLUSTAL, http://www.clustal.org/.

[7] FASTA, http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml.

[8] Thamarai Selvi Somasundaram, Balachandar R. Amarnath, R. Kumar, P. Balakrishnan, K. Rajendar, R.Rajiv, G. Kannan, G. Rajesh Britto, E. Mahendran & B. Madusudhanan, "CARE Resource Broker: A framework for scheduling and supporting virtual resource management", Journal: Future Generation Computer System, Volume 26, Issue 3, March 2010, Pages 337-347, doi:10.1016/j.physletb.2003.10.071.

[9] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov ,"The Eucalyptus Open source Cloud-computing System", Proceedings of Cloud Computing and Its Applications, October 2008.

[10] OpenNebula (2011): The Open Source Toolkit for Cloud Computing. http://opennebula.org/start.

[11] R. Das, J. Kephart, I. Whalley and P. Vyas, "Towards Commercialization of Utility-based Resource Allocateion," in ICAC '06:IEEE International Conference on Autonomic Computing, 2006, pp.287-290.

[12] Biao Song, Mohammad Mehedi Hassan, Eui-nam Huh: A Novel Heuristic-Based Task Selection and Allocation Framework in Dynamic Collaborative Cloud Service Platform. CloudCom 2010: 360-367.

[13] Hai Zhong, Kun Tao, Xuejie Zhang, "An approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems," The Fifth Annual ChinaGrid Conference, 2010. DOI 10.1109/ChinaGrid.2010.37.

SESSION

WORKFLOW + SCHEDULING

Chair(s)

TBA

TSM-SIM: An Evaluation Tool for Grid Workflow Scheduling

Mohamed Amine Belkoura[1]**and Noé Lopez-Benitez**[1]
[1]Department of Computer Science, Texas Tech University, Lubbock, Texas, USA

Abstract— *Implementing efficient workflow job scheduling is a key challenge in grid computing environments. Simulation is an efficient mechanism to evaluate scheduling algorithms and their applicability to various classes of grid applications. Several grid simulation tools exist, and provide a framework for studying grid job execution in conjunction with different scheduling algorithms. However, these simulators are tailored only to independent grid jobs, with limited support for complex grid workflows submission and scheduling.*

This paper presents TSM-SIM, a two-stage metascheduler simulator for grid workflow applications. It supports dynamic grid resource and job simulation, and provides a submission interface for workflow grid applications as a single unit, rather than as a set of grid jobs. We detail the overall architecture TSM-SIM as well as example of its scheduling algorithms. We demonstrate how it can be used to collect performance scheduling data of complex grid workflow benchmarks.

Keywords: metascheduling, task management, middleware, grid, simulator, architecture

1. Introduction

Grid workflows are orchestrated by grid meta-schedulers, matching grid application jobs with available resources, in order to achieve an optimal execution [1]. To satisfy such requirement for complex grid flows, a Two-Stage Metascheduler (TSM) decouples logical task metascheduling from physical task/node matchmaking, while achieving an improved overall performance [2]. In order to validate the effectiveness of this scheduling architecture, only a comprehensive and rigorous testing process can produce accurate and meaningful results. Given its inherent complex and dynamic nature, computing grids are hard to evaluate. Setting up grid testbeds that are both realistic and adequately sized is an expensive and time consuming process, and therefore represents a barrier to meta-scheduler algorithm evaluation. We propose the use of grid simulators to measure the efficiency of metascheduling algorithms in a diverse and comprehensive set of scenarios. To evaluate the efficiency of two-stage metascheduling, it is important to run a number of tests, varying different parameters and platform scenarios, with the goal of producing statistically significant quantitative results. However, real-world grid platforms are hard to setup, labor-intensive, and are generally constrained by the available hardware and software infrastructure. To preserve the security and consistence of valuable grid resources, grid administrators tend not to allow users to modify some grid parameters, such us participating nodes, network connections and bandwidth, and some lower level grid middleware and operating system configuration. For all these reasons, a simpler and reproducible approach to evaluate grid application scheduling requires the use of simulators.

2. Background and Related Work

Several solutions were proposed in the realm of grid application scheduling simulation. Bricks simulator [8] is a JAVA simulation framework used to evaluate the performance applications and scheduling algorithms in Grid environments. It consists of a discrete event simulator, a simulated grid computing and data environment, as well as network components. It allows the analysis and comparison of various scheduling algorithms on simulated grid settings, taking into consideration network components effect on the overall performance. SimGrid [6] is another widely used toolkit for the simulation of parallel and grid application scheduling. It supports an out-of-the-box creation of time-shared grid and cluster resources. It also supports varying resource loads statically and dynamically. It also provides an extensibility programming layer for adding or customizing grid jobs and resources creation based on various parameters. Its programming interface provides several mechanisms to implement resource scheduling policies. GridSim [3] is also a popular simulation framework for grid and parallel applications. It supports different resource schedulers, including time-shared and space-shared resources. It contains a network simulation component, used for simulating network topologies, links and switches. it also allows incorporating resource failure into the grid application simulation. OptorSim [9] is a java based grid simulator focusing on data grids. It can simulate grid resources of different storage or computing elements, and allows the testing of data replication strategies. Its scheduling simulation is achieved through a resource broker, which implement scheduling schemes. It treats sites computing or data facility as network nodes and routers. For data replication, it features a replica manager and optimizer that handles advanced data manipulation and management.

While these simulators provide mechanisms for a flexible grid application modeling, they do not support the submission of the whole grid application workflow as an input, with all its data and sequence job dependencies, but only support

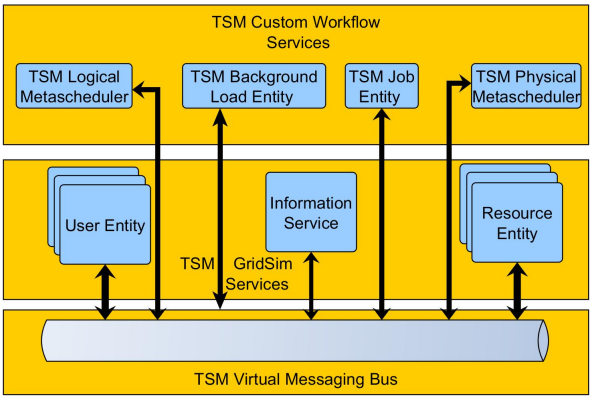


Fig. 1: TSM Simulator System View.

individual jobs submission. The proposed TSM simulator in this paper will extend some of GridSim components, and provide a two stage logical/physical scheduling module based on the TSM architecture.

3. Simulator System Overview

TSM-SIM allows a comprehensive study of the dynamic interaction of multiple grid components, including grid users, resources, networks and various scheduling algorithms. It provides a virtual grid infrastructure that enables grid workflow application experimentation with dynamic meta-scheduling algorithms, supporting controllable, repeatable, and observable experiments. From a system view, TSM-SIM is composed of three main components: TSM Virtual Messaging Bus, TSM GridSim Services, and TSM Custom Workflow Services, as shown in figure 1. It uses an inter-process discrete event based system for communication. Each layer exposes functions for reuse with other services. The following section provides a detailed description of each layer components.

At the core of the TSM simulator is a virtual messaging bus implemented using Simjava framework, inherited from GridSim [5]. Simjava is a inter thread messaging framework that allows sending tagged event from one entity to another within the same java process. Simjava entities are connected to each other using ports and can inter communicate by sending and receiving tagged event objects. A separate thread controls the lifecycle of the entity threads, by synchronizes their execution

4. Simulation Algorithms

In this section, TSM logical and physical metaschedulers algorithms are presented. For the logical metascheduler, details of the Execution Set Algorithm (ESA), the Delayed Execution Set Algorithm (DESA), and the Block Notification Execution Set Algorithm (BNESA) are given. For the physical metascheduler, the ClassAdd Matchmaking Algorithm (CMA), and the Workflow Weight Algorithm (WWA) are outlined.

4.1 Logical Metascheduler Algorithms

TSM scheduling approach consists of decomposing a grid workflow into a set of individual jobs that can be executed in parallel. Any logical algorithm should compose these tasks either dynamically or statically, taking into consideration only job dependencies. The rest of this section presents three logical metascheduler algorithms: the Execution Set Algorithm (ESA), the Delayed Execution Set Algorithm (DESA), and the Block Notification Execution Set Algorithm (BNESA).

4.1.1 Execution Set Algorithm

The Execution Set Algorithm (ESA) accepts as an input the grid application flow in a digraph format (directed graph). It processes the flow composing tasks (graph nodes), data and control flow (edges), and produces a set of task pools, called Execution Sets (E). These sets are submitted to the physical metascheduler in order. None of the task of certain set can be submitted to a physical scheduler unless all the tasks of the preceding set have been submitted. However, the composing tasks of each pool can be submitted in any order. The execution set is updated after execution success notification.

The ESA logic is described by the following algorithm.

ESA Algorithm

$P \leftarrow$Initial flow graph of N nodes.
$E \leftarrow$Current execution set.
$S \leftarrow$Set of nodes submitted, but not executed yet.
$G \leftarrow$Graph of N nodes; G=$\{n_i; i \in [1; N]\}$.
while E $\neq \oslash$ **do**
  Submit S elements for execution.
  if receipt of sucessful execution of n_j **then**
    $E \leftarrow$E - $\{n_j\}$.
    $S \leftarrow$E
    Remove mode n_j and its edges from P
    Update E with additional free nodes of P

4.1.2 Delayed Execution Set Algorithm

The Delayed Execution Set Algorithm (DESA) is a variant of ESA that introduces a delay between the receipt of the first notification, and the submission of the next execution set. A delay allows collecting more individual grid jobs into individual execution sets. This reduces the number of submissions and execution notifications, but generates bigger execution sets. DESA is analyzed with various delay times, in order to study its impact on grid utilization, and total grid application execution time.

4.1.3 Block Notification Execution Set Algorithm

The Block Notification Execution Set Algorithm (BNESA) is a second variant of ESA, where the next algorithm execution set will not be updated immediately after the first successful job notification. Instead the algorithm waits for k numbers of notifications, where k is directly correlated with the size of the execution set E. As for DESA, a delay will allow for potential addition execution notifications, therefore a bigger execution set. This variant is studied with various values of k, in order to analyze its impact on grid utilization and total grid application execution time.

4.2 Physical Metascheduler Algorithms

Different grid job/grid resource algorithms are used in grid environments. They can be classified into three main categories: time-shared, space-shared and backfill algorithms. Time-shared grid scheduling algorithm allocates grid resources in a round robin scheme, and exclusively allocated a grid resource to a grid job until it is completed. Space shared grid scheduling algorithm allocates grid resources in a First Come First Serve (FCFS) and executes more than one processing element (PE) to a grid job. Backfill algorithms attempts to reorder jobs queued to be executed, by moving small jobs ahead of big ones in the scheduling queue. The job prioritization is done to fill in holes in the schedule, without delaying the first job in the queue. Two variants of this class of algorithms exist. The first is called aggressive backfilling, where short jobs will automatically priority over long jobs. The second is called conservative backfilling, where the acceleration of short jobs happens only of such reorder does not delay any job in the schedule queues.

4.2.1 Condor ClassAd Algorithm

TSM physical metascheduler implements *Condor ClassAd Algorithm* (CCA), the standard matchmaking alogorithm used by Condor [10] . Each executing node in the grid advertises its resource ClassAd, while each workflow grid job is defined by its processing requirement within its job ClassAd. For each execution set received from the logical metascheduler, a single round of matchmaking algorithm is made, based on the constraints defined by both the resource and job ClassAds. If a requirement of a job is not met during the matchmaking process, it is delayed until the current execution set is refreshed after the next logical execution set is received. However, if a grid resource is a positive match for a grid job, such matching is selected, and no other possible matchmaking combination is evaluated. In our implementation of CCA, only the requirement part of the ClassAd is considered. The optional rank attributes are not used in our matchmaking process.

4.2.2 Workflow Weight Algorithm

The second TSM physical metascheduler prototyped is called *Workflow Weight Algorithm* (WWA). It is a time shared, first-come-first-served class algorithm that captures the instant load of each grid resource using an auction style election process. Each grid resource ($R_j 1 < i < m$, m is the number of resources) is characterized by its number of machines Rm_i ($Rm_i = 1$ for a non-cluster resource), the number of its processing units Rpu_i, its processing power Rpp_i in Million Instructions Per Seconds (MIPS), its available memory $Rmem_i$, and its network connection speed Rn_i. Each grid job/task ($T_j 1 < j < n$, n is the number of tasks) has a strict number of processor units Tpu_j and memory requirement that need to be satisfied at a single grid resource, in order to be considered in the match-making. Each grid job will advertise its computing power need Tpp_j, its memory requirement $Tmem_j$, its total data input size Tin_j, its total output size $Tout_j$, its height in the workflow tree Thi_j, and its offspring count $Toff_j$.

The algorithms works as follow: In the physical metascheduler, a discrete scheduling interval $\nabla\tau$ will be defined (for example a 10 second interval). At each interval beginning, the metascheduler calculates a scalar value called grid task weight *Tweight*$_j$. This value provides a quantifying value of all the computing characteristics of a grid job/task, and is defined as follows:

$$Tweight_j = C_T \times Tpu_j \times Tpp_j \times Tmem_j \times Tin_j \times Tout_j \times Thi_j \times Toff_j, \;\; 1 < j < n \tag{1}$$

where C_T is a constant at each scheduling iteration. Simultaneously, a similar weight, called the Resource Weight *Rweight*$_i$, is calculated. The logical scheduler will request from each grid resource site its dynamic computing data. Only grid resources that are free submit their data, indicating that they are willing to participate in the current scheduling round. The metascheduler will then calculate the Resource Weight *Rweight*$_i$ defined as follows:

$$Rweight_i = C_R \times Rmi_i \times Rpu_i \times Rpp_i \times Rmem_i \times Rn_i \quad 1 < i < m \tag{2}$$

The next step of the algorithm is to sort all the values of Tweight and Rweight in a descending order. A height Rweight value indicates a fast grid resource, while a high Tweight value indicates a demanding grid job/task. The algorithm assigns the grid job/task of highest Tweight value to the grid resource of the highest Rweight value, with the condition that equation 3 satisfied:

$$Tpu_j \leq Rpu_i \text{ and } Tmem_j \leq Rmem_i \quad (3)$$
$$1 < j < n \; and \; 1 < i < m$$

Note that the number of grid tasks "n" is generally different than the number of grid resources "m" (being equal is only one special case). In case the case of n<m, only the available fast grid resources of the grid are being used. In the case of n>m, a resource starvation is happening, and only a portion of the execution set is actually assigned a grid resource. Grid tasks that are not scheduled will be part of the next scheduling round.

5. Grid Benchmarks

NAS Grid Benchmarks (NGB) were used to test the TSM simulator. NGB is a benchmark suite designed by NASA, based on the NAS Parallel Benchmarks (NPB) [11]. The suite contains different classes of workflow application, and thus helps measuring the capability of a grid infrastructure to execute distributed, communicating processes while testing its functionality and efficiency. NGB benchmarks are defined as data flow graphs, with nodes and arcs representing computations and communications respectively. NGB benchmarks are used to measure each node execution time, as well as the data transfer capabilities of the communication network, particularly latency and bandwidth. An instance of NGB benchmark grid flow is a collection of six types of computing programs. They are called Block Traditional solver (BT), Scalar Pentadiagonal solver (SP), Lower-Upper symmetric solver (LU), Multigrid solver (MG), fast Fourier Transform solver (FT), and Mesh Filter solver (MF). Each instance of these programs is characterized by a class, which describes the size of its input data. These programs different classes are S, W, A, B, C. In our experiments, we considered only S, A, B classes. Every benchmark program code (BT, SP, LU, MG, or FT) is specified by class (mesh size), number of iterations, source(s) of the input data, and consumer(s) of solution values. The DFG consists of nodes connected by directed arcs. It is constructed such that there is a directed path from any node to the sink node of the graph. All of NPB's mesh based problems are defined on the three-dimensional unit cube. However, even within the same problem class (S, W, A, B, or C) there are different mesh sizes for the different benchmark codes.

Table 1 gives the problem size and the memory requirement for every computing program used in the NGB benchmark used in [12].

5.1 Class of NGB Benchmarks

NGB benchmarks consist of four families of problems: Embarrassingly Distributed (ED), Helical Chain (HC), Visualization Pipeline (VP), and Mixed Bag (MB). These benchmarks are described in the rest of this section.

Program	Class	Problem Size	Memory requirement (MW)
SP	S	12^3	0.2
	A	64^3	6
	B	102^3	22
BT	S	12^3	0.3
	A	64^3	24
	B	102^3	96
LU	S	12^3	0.3
	A	64^3	30
	B	102^3	122
MG	S	32^3	0.1
	A	256^3	57
	B	256^3	59
FT	S	64^3	2
	A	$256^2 \times 128$	59
	B	$256^2 \times 512$	162

Table 1: NGB programs size and memory requirement

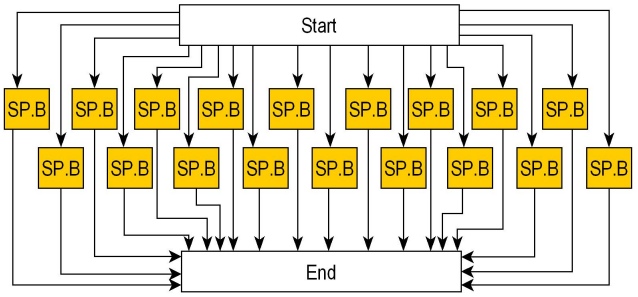


Fig. 2: ED, class B (18x1) grid flow.

5.1.1 Embarrassingly Distributed

The Embarrassingly Distributed (ED) benchmark represents a class of grid applications called parameters studies, where the same basic program is executed multiple times, and each time with a different input data. This class of benchmark models applications that can be obviously divided into a number of independent tasks. The application tasks are executed independently, with different inputs. Figure 2 shows the b-class ED benchmark grid workflow used in our simulations.

5.1.2 Helical Chain

Helical Chain (HC) benchmark models grid application with long chains of repeating programs, such as a set of flow computations executed in order. It consists of a sequence of jobs that model long running simulations that can be divided into different tasks. Each job in the sequence uses the computed solution of its predecessor to initialize. Figure 3 shows an example of a b-class HC benchmark grid workflow.

5.1.3 Visualization Pipeline

Visualization Pipeline (VP) benchmark models grid workflow application composed of multiple chains of compound processes. It represents a chain of grid jobs, but with limited parallelism. It models grid applications where the last itera-

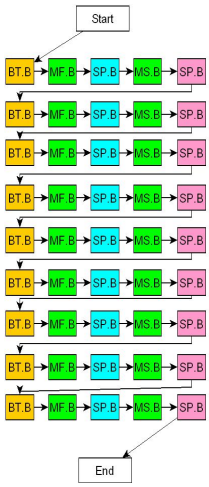


Fig. 3: HC, class B (5x9) grid flow.

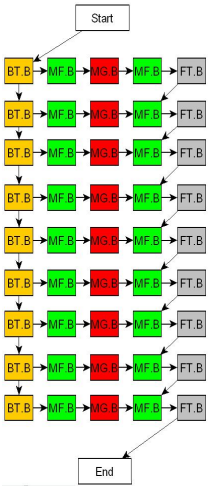


Fig. 4: VP, class B (5x9) grid flow.

tion step is a visualization/analysis task. Figure 4 illustrate an example of a b-class VP benchmark grid workflow.

5.1.4 Mixed Bags

Mixed Bag (MB) benchmark models grid applications composed of post-processing, computation and visualization computing tasks, but with inter asymmetric communication. It also features different tasks that require both different data and computing power. It introduces double and triple dependencies, where some jobs have two or three parent tasks. It constitutes the most complex benchmark in the NGB suite, and thus making it hard for any scheduler to schedule its tasks efficiently. Figure 5 shows an example of a b-class MB benchmark grid workflow.

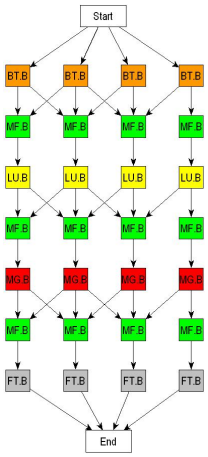


Fig. 5: MB, class B (5x9) grid flow.

Cluster Name	Cluster Name	Node Name	Memory (GB)	Computing Power (MFLOPS)	Network Speed (GB/S)
Main TTU Campus	Cluster 1	Compute1-1	12	9320	10
		Compute1-2	12	9320	10
	Cluster 2	Compute3-1	4	6400	10
		Compute3-2	4	6400	10
	Cluster 3	Compute8-1	64	10400	10
	Cluster 4	Compute6-9	4	6400	10
		Compute6-10	4	6400	10
	Cluster 5	Compute6-1	12	9320	10
		Compute6-2	12	9320	10
TTU Reese Campus	Cluster 6	Compute10-1	4	12000	10
		Compute10-2	4	12000	10
	Cluster 7	Compute11-17	4	9320	10
		Compute11-18	4	9320	10

Table 2: Resource Properties of the Grid Testbed

6. Simulation Environment Setup

In our experiments, we simulated a subset of Texas Tech Hrothgar and Antaeus clusters [13] as part of a computing grid. The grid modeled in our simulations contains 13 resources, spread both among the main and satellite Texas Tech campuses. We modeled each of its grid resources with a total number of processing elements (PEs) characterized by their MIPS rating (Million Instructions per Second) and their internal memory capacity. We also model the network connecting all the grid computing elements, by specifying, the network layout, the number of routers, and the network link properties such as bandwidth in bits/second and Maximum Transmission Unit (MTU) in bits.

Table 2 shows the grid test bed properties simulated, while figure 6 details the network topology of the simulated grid environment.

6.1 Experimental Methodology

A set of each NGB benchmark grid application is generated, and submitted to the TSM simulator, under the different benchmark type (ED, HC, VP and MB), class (S, A, and

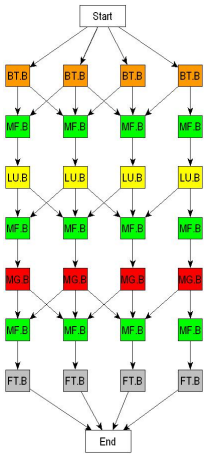


Fig. 6: Network Topology of the Grid Test Bed.

Load Pattern	normal distribution size(MB)		Poisson distribution inter-arrival time (s)	
	Min	Max	Min	Max
Medium	2,5	5	10	50
High	10	20	10	50

Table 3: Background Traffic Generator Pattern Parameters

B), background load (none, medium, high). Both Execution Set Algorithm (ESA) and Delayed Execution Set Algorithm (DESA) for the logical metascheduler and Workflow Weight Algorithm (WWA) for the physical metascheduler were implemented. A background traffic generator is used to simulate a non-exclusive access schema to a grid. The background traffic generator was configured with two different load patterns: a medium and a high load pattern, which are generated based on the size of background data, the job size, and the inter-arrival time. The background load data size follows a normal distribution of a minimum size of 2.5 KB and a maximum of 5 KB. Its arrival time follows a Poisson distribution, with an inter arrival times varying from 10 to 50 seconds. The traffic generator is bound to each resource, so the background traffic and load hits all resources and its network route, starting from the TSM simulator. Table 3 shows the values for each background traffic generator pattern parameters.

6.1.1 Performance Metrics of TSM Algorithms

Various metrics were defined and captured during each simulation execution. The overall performance of a grid workflow application can be measured by the time it takes to finish its total execution, which starts with the time it is submitted to the TSM metascheduler and finishes when the last composing job executes successfully, and its output is received by the TSM metascheduler. We refer to this time as Grid Application Execution Time (T_{GAET}). This execution time takes into consideration the execution of the grid application composing tasks, as well as the network time consumed to transfer input and output files needed by each composing task. Because a grid is a parallel execution environment, T_{GAET} is not the sum of each task execution time, and each file transfer time. As more than a task can execute at the same time, several execution and transfer tasks overlap. On the other hand, contention over grid resources and network connection introduces additional delays counted toward T_{GAET}. In addition, the use of the TSM algorithm introduces another meta-scheduling computing time. As a result, the total workflow execution time consists mainly of three components: a task execution component T_{EX}, a data transfer time component T_{DT}, a meta-scheduling component T_{TSM}, and an idle time component T_{IDLE}, spent either waiting for resources to be available, or when a job is queued at the local grid machine scheduler. Therefore, T_{GAET} calculation formula is obtained as follows.

$$T_{GAET} = T_{EX} + T_{DT} + T_{TSM} + T_{IDLE} \quad (4)$$

T_{EX} is the time spent running the task program on a grid resource, and does not count network time. T_{DT} is the total time use to transfer data in and out of grid resources. T_{TSM} is the total meta-scheduling time taken by both the logical and physical metascheduler to allocate resources to grid jobs will depend on how many times it execute the matchmaking algorithm. T_{IDLE} is the time slot not used for the three main active times is considered idle or unused. We also define the Total Grid Time T_{TGT}, which constitute the total time per grid resource that was spent executing the grid application. It is obtained as follows:

$$T_{TGT} = T_{GAET} \times N_r \quad (5)$$

Where N_r is the number of available grid resources. In our experimental simulated grid environment, $N_r = 13$.

7. Experiment Results

The purpose of these experiments is to study the effect of varying both the background load and the scheduler variant on the performance of the scheduling policies, and show how grid workflow applications benefit from the two-stage scheduling in real workflow situations. It also showcases the value of TSM-SIM producing experimental results for various grid scheduling and load conditions. We first present an analysis of background load on grid workflow scheduling, where we test the combination of ESA/WWA algorithms. Second, we analyze the impact of delayed submission, using the DESA/WWA algorithm combination.

7.1 Background Load Effect

We first consider the effect of background load on scheduling different class of workflow grid applications. The effect of background load effect on the total time is shown in figures 7 to 10. As the load increases, the total time increases, especially for the pure parallel flow (ED), and the

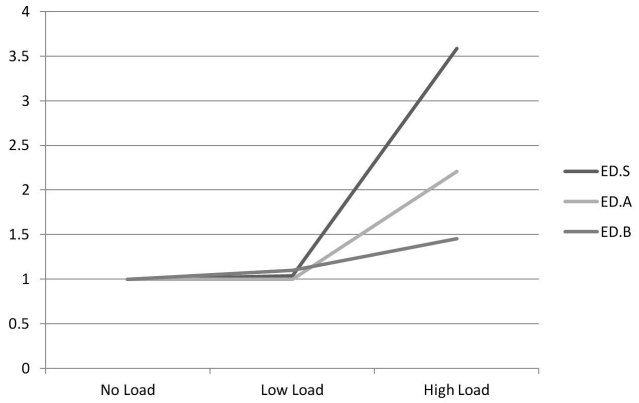


Fig. 7: Load Effect on ED Class Workflow.

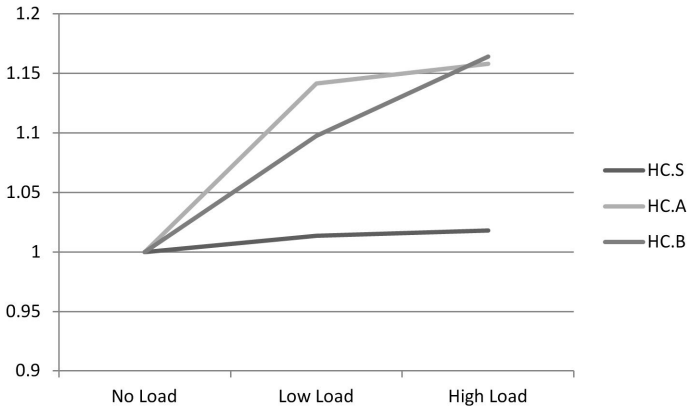


Fig. 8: Load Effect on HC Class Workflow.

pure sequential flow (HC). For the case of (ED), only high background load effect the response time, while it only takes some low background load to slow the workflow execution in case of HC. Also, the effect of high load on the most complex (ED) class workflow is more than 3 times the effect of the type of load on HC class workflow. Note also that slowdown due high load on HC class workflows is more influenced by the size of the data, than by the complexity of the workflow.

For a more general type workflow, such as class VP and MB, the background load have less effect than the case of ED and HC, with a maximum of 1.1 ratios for VP, and of 1.5 for MB. The VP figure (figure 9) shows an insignificant load effect on the total workflow execution time, with no more that 0.1 increase ratio. This means a close to optimal experimental utilization of grid resources. However, the higher level of parallelism in a workflow (case of MB), the more significant is the effect of background load, especially in when it is high. In fact, for the case of MB benchmark, which is the most complex benchmark, the overall slowdown aproaches 50% under sustainable grid load, especially in the case of the MB.B class benchmark.

As a conclusion, we can state that experimental tests using TSM-SIM show that the effect of background grid loads have a higher impact of grid workflows with high level of parallelism (ED and MB). This can be correlated to the average execution set size. In fact, a high level of parallelism in a grid workflow causes the TSM logical metascheduler to generate bigger execution sets. The jobs composing these execution sets are more penalized by the background load, because they also compete with each other for fast resources.

This peer competition effect, while it can also effect non-related grid jobs, significantly impacts grid workflow applications more than isolated grid jobs. The background load effect in case of grid workflows is higher, because its impact on scheduling and execution is compounded.

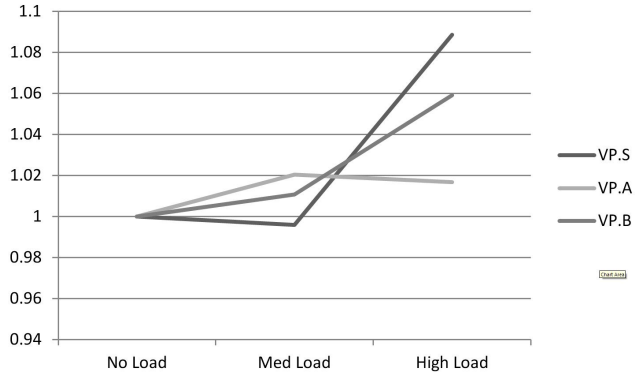


Fig. 9: Load Effect on VP Class Workflow.

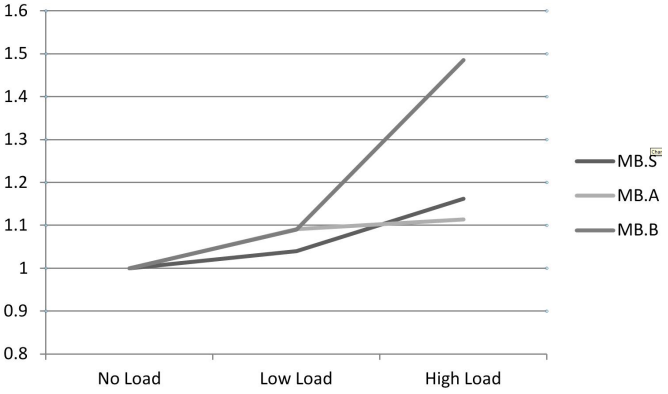


Fig. 10: Load Effect on MB Class Workflow.

7.2 Submission Delay Effect

In this experiment, we test the difference between the ESA/WWA and DESA/WWA, and study the effect of the introduction of a delay in submitting execution sets by the logical meta-scheduler. In a real grid environment, the motivation of such delay is to allow other grid resources to become available, so that a better choice of grid resources is possible. The intention is to wait for potential powerful grid resources to join the pool of available grid resources, which can be beneficial for the overall grid execution. We want to measure if the time wasted waiting for such resources can be easily made up by using a powerful grid resources. The goal of this simulation run is to experimentally study when such strategy is beneficial for grid workflow scheduling, and identify application and environment properties that impact this scheduling strategy. We simulate the submission delay variant (TSM-SDV) in TSM-SIM, by keeping a constant delay of 10 seconds, while varying the background load on the grid infrastructure (grid network and resources). We run 10 simulation of each kind, and measured the average simulation time. We tested this scheduling variant for each grid NGB benchmark (with the S, A, and B complexity classes).

Figures 11 to 14 show the summary of these simulation runs. The Y axis shows the improvement rate $\text{RATE}_{TSM-SDV}$ that DESA contributes to the total workflow execution time compared to ESA. $\text{RATE}_{TSM-SDV}$ is calculated using the following equation:

$$RATE_{TSM-SDV} = Tnd_{GAET}/Tnd_{GAET} \qquad (6)$$

with Tnd_{GAET} is the experiment total workflow execution time in case of no submission delay, and Tnd_{GAET} the same time with submission delay.

The common observation is that submission-delay negatively impacts the grid workflow execution time. In most of the cases, a 50% performance hit is observed. (ED) benchmarks experience the worst performance. The impact can be as much as 500% (rate of 0.2) for the simple class S under no background load. The impact is less visible in case of high than low background load. We can explain this by the 100% parallelism of ED applications. Delaying the submission of the execution set, which contains most of the grid workflow jobs in case of ED benchmarks, gives the opportunity to background load jobs to use fast grid resources. Thus, the penalty of any delay is greater than any benefit that might be achieved. (HC) benchmarks suffer similar negative impact. The performance hit, however, is about constant, varying from a 0.35 to 0.6 factor depending on the load. (MB) benchmarks, the most complex type amount tested benchmarks, do record a similar performance hit in the range of 0.25 to 0.75. The only difference with the impact is amplified by heavy load in a significant proportion. The response time for (VP) type benchmarks seems to be different from other benchmarks. While the behavior seems to be similar than other benchmarks in case of no or little background load, the total grid application performance seems to be un-affected by the submission delay variant. In fact, in case of no load, a slight 10% improvement is recorded, making it the only case where the submission delay benefits the overall grid workflow performance.

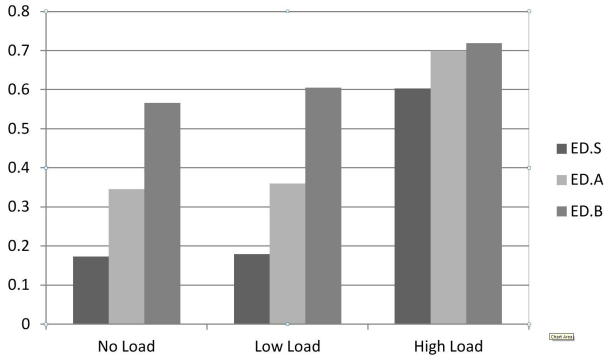


Fig. 11: Submission Delay Effect on ED Class Workflow.

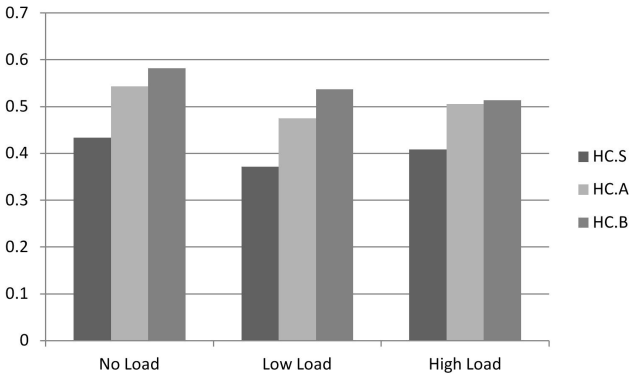


Fig. 12: Submission Delay Effect on HC Class Workflow.

8. Conclusions and Future Work

This paper outlined the details of TSM-SIM, a two-stage grid metascheduling simulator aimed at grid workflow applications. It is primarily intended to test the TSM architecture on a simulated environment, by building on existing GridSim services to configure two-stage scheduling services. We have demonstrated how it can be used to evaluate grid workflow scheduling algorithms using NAS Grid benchmarks.

For future work, TSM-SIM will be used to analyze the performance of other logical and physical scheduling algorithms, using grid workflows. We also intend to build extensibility modules to support high level grid schedulers such as GridWay [14].

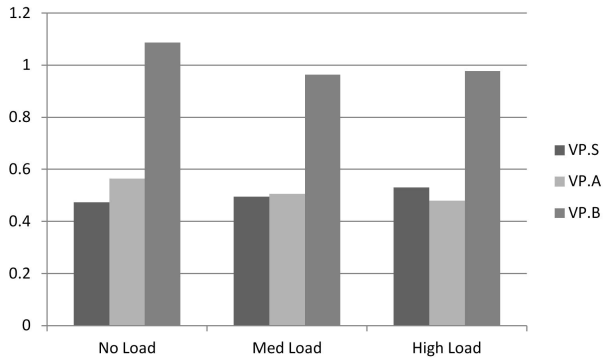


Fig. 13: Submission Delay Effect on VP Class Workflow.

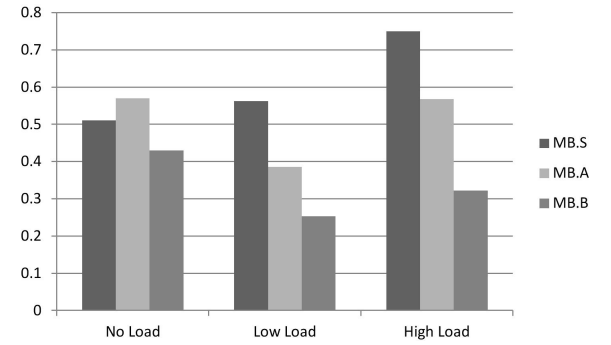


Fig. 14: Submission Delay Effect on MB Class Workflow.

References

[1] M. Amine Belkoura and N. Lopez Benitez, *Two-Stage Metascheduling for Computational Grids*, World Congress in Computer Science, Computer Engineering, and Applied Computing, 2009.

[2] M. Amine Belkoura and N. Lopez Benitez, *TSM-SIM: A Two-Stage Grid Metascheduler Simulator*, International Journal of Grid Computing Applications (IJGCA), 2(4), 11 - 26, 2012.

[3] R. Buyya and M. Murshed, *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, Journal of Concurrency and Computation: Practice and Experience (CCPE), 2002.

[4] Anthony Sulistio, Uros Cibej, Srikumar Venugopal, Borut Robic and Rajkumar Buyya, *A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim* , Concurrency and Computation: Practice and Experience (CCPE), Online ISSN: 1532-0634, Printed ISSN: 1532-0626, 20(13): 1591-1609, Wiley Press, New York, USA, 2008.

[5] Agustin Caminero, Anthony Sulistio, Blanca Caminero, Carmen Carrion and Rajkumar Buyya,*Extending GridSim with an Architecture for Failure Detection* , Proc. of the 13th International Conference on Parallel and Distributed Systems (ICPADS 2007), Dec. 5-7, 2007, Hsinchu, Taiwan.

[6] S. De Munck, K. Vanmechelen and J. Broeckhove, *Improving The Scalability of SimGrid Using Dynamic Routing*,Proceedings of ICCS, 2009.

[7] Fred Howell and Ross McNab, *Simjava: A Discrete Event Simulation Package for Java with Applications in Computer Systems Modelling*, in Proc. of First International Conference on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation, 1998.

[8] A. Takefusa, K. Aida, S. Matsuoka, *Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms*, Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC8), 1999.

[9] William H. Bell and David G. Cameron and Luigi Capozza and A. Paul Millar and Kurt Stockinger and Floriano Zini, *OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies*, International Journal of High Performance Computing Applications, 2003.

[10] M. Litzkow and M. Livny, *Experience with the Condor Distributed Batch System*, Proceedings. of IEEEWorkshop on Experimental Distributed Systems , 1990.

[11] Michael A. Frumkin and Rob F. Van der Wijngaart, *NAS Grid Benchmarks: A Tool for Grid Space Exploration*, HPDC, 2001.

[12] Rob F. Van Der Wijngaart and Michael Frumkin, *NAS Grid Benchmarks Version 1.0*, 2002.

[13] Anonymous, *HPCC OSG Cluster Grid*, available at http://antaeus.hpcc.ttu.edu/wordpress/.

[14] Huedo, Eduardo and Montero, Ruben S. and Llorente, Ignacio M. *A Framework for Adaptive Execution in Grids*, SoftwareÛPractice Experience, Volume 34 Issue 7, 2004.

Multi-objective Heuristic for Workflow Scheduling on Grids

Vahid Khajehvand[1], **Hossein Pedram**[2], **and Mostafa Zandieh**[3]
[1]Department of Computer Engineering and Information Technology, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[2]Department of Computer Engineering and Information Technology, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
[3]Department of Industrial Management, Shahid Beheshti University, G.C., Tehran, Iran

Abstract—*The Utility Grids develop a cyber-infrastructure for using services transparently in a distributed environment. The parameters of the Quality of Service such as the allocation-cost and turnaround time, needs to be taken care of for scheduling a workflow application in the Utility Grids. These target parameters are sometimes likely to be in conflict. In this paper, a multi-objective cost-based model along with a heuristic algorithm is presented for scheduling a workflow application in order to optimize the multi-objective allocation-cost and makespan in a very low runtime. The results of the wide-spread simulation indicate that the proposed algorithm is effective against an increase in the application size. The proposed algorithm effectively outperforms the current algorithms in terms of the allocation-cost, makespan and runtime scalability.*

Keyword*:* Utility Grids; Application Scheduling; Multi-objective Optimization.

1 Introduction

Grid computing is capable of controlling a wide variety of heterogeneous distributed resources to execute computation and data intensive applications. Grid computing has recently been oriented towards pay-as-you-go models. In these models, the resource providers receive fees from the users for presenting computing and data services. Shared distributed infrastructures come up with the grid environment software and hardware resources, in order to conduct large-scale computations. These infrastructures turned out to be efficient for executing applications in sciences such as astronomy [1], high energy physics [2] and others.

The challenge faced by the scientists in these fields is how to use cyber-infrastructure for transferring knowledge from the scientific environments to the distributed computing environments. The workflow is the most common approach to describe an application in a high level form regardless of the distributed computing environment. A workflow is represented in a "Direct Acyclic Graph" (DAG) with nodes and edges representing the tasks and data dependencies between the tasks, respectively. Once an application is transformed into the workflow structure, a workflow management system will be ready to control and manage the execution of workflow on the distributed infrastructure. In these environments, indeed, access to the shared computational resources is carried out through the queue-based Local Resource Management (LRM) system.

The grid computing is an interactive environment in which at one end, the users are expecting to receive services for their applications, whereas the resource providers are ready to offer services to the users at the other. The resource providers advertise the available resources set to be planned by the users, brokers and the application-level schedulers who receive fees upon providing services. An environment characterized with the above-mentioned users and service providers is known as Utility Grids. A competition develops among users caused by the resources-pricing policies so that users begin being involved in a competition with one another only to gain a resource with an affordable cost and an efficient processing capability. Similarly, resource providers are driven into a competition with one another to sell their idle resources to the users in order to gain more profits as well as enhance the resource utilization.

The scheduling problem becomes highly complicated and NP-complete [3] in such an environment due to the different resource consumers and providers so that each side pursues its own profits. It is worth noting that the resource consumers and providers are acting independently with conflicting aims. The resource consumers seek the minimum time (makespan) and allocation-cost for scheduling application, whereas the resource providers seek the resource utilization gains. Thus, the main challenge confronted by the users in this environment, will be scheduling an application on the heterogeneous resources in which the users have no explicit control so that both time and allocation-cost can be minimized.

The present paper deals with developing a Workflow Planning Cost-based (WPC) model in order to effectively schedule an application in the Utility Grids so that the application makespan and allocation-cost can be minimized. In fact, the WPC model allows the users to make a trade-off between an application makespan and

allocation-cost. Next, a First-fit Cost-Time Trade-off (FCTT) heuristic algorithm is employed to solve the WPC model. The FCTT is a heuristic algorithm that schedules an application in a form that both the makespan and the allocation-cost can be optimized due to the trade-off factor. The trade-off factor shows the preference of the allocation-cost optimization to the turnaround-time. Finally, to study and evaluate the efficiency of the proposed algorithm on the proposed model, a handful of experiments have been conducted and simulated. The simulation results show that the FCTT algorithm is effective due to an increase in workflow size. The main contributions of the present paper are as follows:

- Developing a WPC model based on provisioning the resources for scheduling a workflow, so that the application makespan and allocation-cost can be minimized.
- Developing a multi-objective FCTT heuristic algorithm based on the WPC model with an effective performance due to an increase in the workflow size.

The rest of this paper is organized as follows: Section 2 discusses the related works. Section 3 introduces an application scheduling problem and execution environment. A proposed detailed model and heuristic algorithm is described in section 4. Section 5 involves a simulation setup and its relevant experiments in order to evaluate the efficiency of the proposed algorithm. In section 6, the results have been analyzed. Finally, section 7 ends with a conclusion.

2 Related works

As shared multiprocessing systems advance, the issue of the application scheduling has been the main concern. To tackle the problem, the providers seek to maximize the utilization of the resources whereas the users seek to minimize turnaround time of the application. There is a comprehensive introduction on the job scheduling strategies [4, 5]. Moreover, in [6], the computational models are surveyed for Grid scheduling problems and their resolutions using the heuristic and meta-heuristic approaches.

In the queue-based systems, the users submit the tasks to the resource queues, whereas the resource allocation will subsequently be conducted due to the strategy of LRM system. In such systems neither has the user explicit control on the allocating resources to the tasks nor can the user optimize the performance. This delivered quality of service to the users is known as the best effort QoS.

The alternative approach is one of the planning-based systems [7]. In these systems, according to agreements the start time of the task can be established in advance instead of the task waits in queue in order to get access to the resource. The above-mentioned agreements are based on an abstract description, so-called "slot" so that the slots are specified by the start time, the number of available processors, the cost and the duration parameters. In this paper, the planning-based system is exploited as the resource management strategy.

In [8], a heuristic algorithm is presented for scheduling many parallel applications on the Utility Grids so that it can manage and optimize the cost-to-time trade-off. This approach is close to the studies conducted for this paper and its main difference from that of the proposed approach lies scheduling the parallel applications, whereas the approach adopted by present paper is based on scheduling the workflow application. Due to the data dependencies among tasks, scheduling the workflow application becomes more complex than scheduling the parallel application.

The main objective of the conventional workflow scheduling is the minimization of the time. A large number of the workflow-based scheduling algorithms rest on the list-scheduling technique. Due to this technique, a rank is typically assigned to each application task, the tasks are, subsequently, sorted and scheduled in a descending order of the corresponding rank. The Heterogeneous Earliest Finish Time (HEFT) algorithm [9] is one of the most common list-based workflow scheduling algorithms. To obtain the list-scheduling, the HEFT takes the task runtime and the data transfer between the tasks and the heterogeneity of the resources into account. The HEFT schedules the workflow application with a high performance in the heterogeneous environment [9, 10].

There is a handful of the different studies conducted on the cost optimization of the workflow scheduling close to the current paper's study. In [11], a genetic algorithm is proposed to find an optimized mapping of the tasks to the resources which minimizes both financial cost and makespan. This approach is developed in [12, 13] which presents the cost-based model in which the resource providers advertise the available resource slots to the users. A multi-objective genetic algorithm is presented which is capable of provisioning a subset of the resource slots to minimize the application makespan under the minimum resource allocation-cost. The main difference between these cost minimization algorithms and the present paper's algorithm lies in the fact that these minimization algorithms rely on a cluster with all processors which are homogeneous. Thus, in [12, 13], the entire resources possess identical CPU ratings and cost processing whereas in the proposed model, all resources are constituted of the heterogeneous clusters with different processing cost and CPU ratings in the real-world Utility Grids environments. Hence, removing this resource homogeneity complicates the identification of an appropriate resource selection.

Since the above-mentioned cost optimization algorithms [12, 13] are genetic-based ones, the runtime

takes a longer time. In case, the slots' characteristics undergo a change during scheduling, the slots' characteristics are to be updated and a rescheduled resulting in a far longer runtime. Hence, these approaches do not serve the purpose in the dynamic environments such as the Grids.

3 Application scheduling problem

Workflow execution planning is carried out prior to the workflow execution. It intends to examine users' execution requirements and to generate suitable execution schedules. The formulation of the optimization problem of the workflow execution and execution environment is presented for the workflow planning problem beneath.

The agreement-based resource management allows an application-level scheduler to attain the resources in the desired time. The workflow management system, therefore, ensures access to the desired resources within the agreed time and cost. In the most resources, an abstract-agreed structure is reached between the provider and consumer in terms of available time slots. In clusters, for instance a slot indicates the availability of a number of the related processors, start time, duration and cost. Once a slot is obtained, it can later on be used without an extra interaction between the provider and consumer. For example, a slot on a cluster is likely to be used to execute a workflow consisting of a number of tasks.

A workflow-application is represented in a DAG. A DAG is defined as $G = (V, E)$, where V is a set of nodes, each node representing a task, and E is a set of links, each link representing the execution precedence order between two tasks. For example, a link $(i, j) \in E$ represents the precedence constraint that task v_i needs to be completed before task v_j starts. The data is a $V*V$ matrix of the communication data, where d_{ij} is the amount of the data required to be transmitted from the task v_i to the task v_j. As a workflow may consist of sub-workflows with multiple entries and exits so the first thing to be done is to add two pseudo-tasks, a top task and a bottom task, with zero execution time indicated by 0 and $n + 1$, respectively. The top task spawns all actual entry tasks of the workflow to be linked to a single node, while the bottom task joins all actual exit tasks to a single node.

The user submits the application characteristics to the application-level scheduler only to be executed on the grid environment. The user expects to have his application executed with the minimal time and allocation-cost. Certainly, the users exploit trade-off factor in order to show a preference for cost to time. In cases where this factor is not specified by the users, the default trade-off factor is considered as equal.

In fact, the application-level scheduler acts as a mediator between the resource providers and users. Due to the reports of the available slots obtained from the resource providers, the application-level scheduler plans the application. The entire slots exploited in planning the application, will be submitted to LRM in order to provision the resources. Each computational resource is equipped with a number of the processors, the memory and the network interfaces which reveals an independent processing unit. The entire resources are fully-connected while being capable of executing all application-tasks. All of the computational resources can act as a service-provider (site) for time-slots.

The application-tasks will be non-preemptively executed, so that one or a multiple of computational resources are exclusively applied to in order to be executed in due time. We suppose that the application-task performance models are clear on each resource. The execution time of a certain task, therefore, may be obtained from a certain resource due to application performance models. Also, the execution of a single task consists of three phases: (a) the input data retrieval from the resource executing the immediate predecessors of the task (b) the task execution and (c) the output data communication from the current resources to the resources presumed to execute successors of the task.

To transfer the data between the application-tasks, three data-management strategies have been proposed by Deelman et al. [14] known as the regular, dynamic cleanup and the remote I/O (on demand). In this paper, the remote I/O (on demand) strategy has been used, so that the output data are submitted to the resource that is seeking to execute immediate the successor-tasks from the immediate predecessor-tasks using the existing high-speed network among the resources. As the application tasks are assumed to be rigid, eventually, processors in need are simultaneously and exclusively handed over to desired task throughout the execution time.

4 Proposed model and heuristic algorithm

In general, the users are in need of two QoS: the deadline and budget of their applications on the pay-per-use services [15]. The users normally tend to run their applications in as the minimum time and cost as possible. Thus, a trade-off factor indicating the significance of the cost to time will be used. In this section, the issue of application scheduling will be stated and the WPC model will be presented and then solved in order to optimize the application cost-time trade-off. Finally, a heuristic algorithm will be developed to conduct the application scheduling with the aim of optimizing the cost and time.

4.1 The proposed multi-objective cost-based model

The execution model consists of a set of heterogeneous consumers and resource providers where the consumers seek to schedule their workflow applications with the minimum cost and time. In this model, R is a set of available heterogeneous resources and V is a set of the tasks of the workflow application. Each resource consists of a set of slots for executing the task v_i.

Services have different processing capabilities which are delivered with different prices. The time(v_{ij}) is the normalized completion time of v_i on the resource r_j and the cost(v_{ij}) is the normalized allocation-cost of v_i on the resource r_j. The normalization matters since it is not clear what value ranges the allocation-cost and finish time will take in a given solution. The scheduling optimization problem seeks to generate solution S, which maps every task v_i to a suitable resource r_j to achieve the multi-objective cost-based metric defined by

$$S_{ij} = \alpha \times time(v_{ij}) + (1-\alpha) \times \cos t(v_{ij}), \ \ \forall v_i \in V, \forall r_j \in R \qquad (1)$$

where α is a trade-off factor that indicates a preference of the allocation-cost to the execution time of the workflow scheduling. Thus, the objective function of the application scheduling problem is obtained by the minimization of the sum of the multi-objective cost-based metrics for the whole application-tasks reached by

$$\min\Big(\sum_{\forall v_i \in V} \min_{\forall r_j \in R} S_{ij}\Big). \qquad (2)$$

The application scheduling problem involves mapping each task v_i to the suitable slot of the resource r_j, so that the application makespan and allocation-cost can be minimized. Upon the completion of the whole application tasks, makespan and allocation-cost will be computed. In the following section, a heuristic algorithm is presented to solve the WPC model as a whole.

4.2 The proposed heuristic algorithm

The FCTT is an algorithm which selects the most appropriate slots for each task, which are ready to be executed. There is a handful of choices for each task, among which the choice capable of minimizing the multi-objective cost metric of (1) will be selected as the best solution. According to the best solution, the Earliest Start Time (*EST)* needs to be computed to execute immediate successor tasks and this procedure will be carried on so long as the execution of the whole application tasks will be finished.

The FCTT algorithm pseudo-code is presented in algorithm 1 which operates according to the WPC model. The algorithm obtains the available slot lists to all resources and the unscheduled tasks as an input parameter (lines 1, 2). Moreover, the *EST* is initialized with simulation current time (line 3). The application-level scheduler carries out the planning of each application task due to available slots list characteristics with an eye on the multi-objective cost metric presented in (1), (lines 4 to 14). Initially, a list of unplanned tasks which are eligible to be executed is selected (line 5). Next, the eligible tasks are defined as the ones whose parents' tasks execution is completed, though the very same tasks have not been executed yet. The available slots list of each resource is obtained by line 7. In line 8, the *EST* of the task *T* on all the resources is computed. Eventually, the Earliest Finish Time (*EFT)* of the task *T* is computed, (line 9).

The *EST* is computed on the basis of the completion-time of the latest parents tasks *T*. Next, the best slot capable of executing the task is selected for each task *T* on each resource. In cases, the selected resource does not match with the resource which executes the parents' tasks, the data-transfer time needs to be added to the *EST*.

Once the best slot to execute task *T* is obtained on each resource, the resource which minimizes the multi-objective cost metric in (1) will be selected as the best resource (line 10). Now, it comes to allocating the task *T* to a selected resource (line 11) as well as updating the slots list of the selected resource (line 12). This procedure needs to be continued as long as there still exists an eligible task (lines 4 to 14). Finally, when the entire application tasks are planned, the time and allocation-cost need to be computed. At the end of the completion of the whole application tasks, the slots assigned to the application tasks will be released.

5 Simulation setup

To conduct an experimental evaluation of the efficiency of algorithm 1, the GridSim [16] is used to simulate the application-level scheduler in the Utility Grids environment. The Grids environment which is modeled in this simulation consists of ten sites belonging to a subset of the European Data Grid (EDG) spread across five countries which are interconnected via a high-speed network [8, 17]. The workload simulated on these sites follows the workload model generated by Lublin [18]. The main purpose of the use of this model is to create a realistic simulation environment where the tasks compete with one another.

The Lublin workload model determines the arrival-time, the number of required processors and the estimated runtime parameters. This model is derived from the trace of the existing model to do rigid tasks. Table 1 shows the workload parameters values applied to in the Lublin model. Table 2 shows resource configuration on the Grids test-bed in order to simulate the distributed system as well as the cost of using a processor, a CPU rating, the number of CPUs and the site-location of each resource.

This resource configuration is used in order to show the heterogeneity of the execution environment. The entire

resources are simulated using the advance reservation policy and the conservative backfilling policy in order to improve response time. In general, in the real-world, the resource pricing is controlled by different economic factors, thus, the time and allocation-cost minimization is likely to conflict with one another.

To conduct experiments, a parameterized graph generator is used to create a synthetic workflow application [9]. The application characteristics contain n=100 tasks with an average execution time of 1000 *s* [13]. The workflow on the average consists of $\sqrt{n}$ levels (the workflow graph depths) and $\sqrt{n}$ tasks at each level. Each task on the average needs 25 CPUs for executing. The mean value of the data transfer among the tasks is 1000 *Gb*. The mean bandwidth value among resources is 10 *Gb/s* with a mean latency time of 150 *s*.

Algorithm 1: The pseudo-code for the FCTT algorithm

Input: An application characteristics with an instruction length for each task and the required CPUs
The resource characteristics and the available slots to each resource

Output: The workflow scheduling

```
Get the list of the available time slots for all resources
UnScheduledTask = get the list of the tasks which have not been scheduled yet.
Assign the simulation current time to the Earliest Start Time(EST).
While UnScheduledTask is not empty do
    EligibleTasks = select all tasks which executions of their parents have been completed.
    for each T in the EligibleTasks do
        Acquire the available slots of each resource.
        Compute the EST of the task T on each resource.
        Compute the Earliest Finish Time (EFT) of the task T.
        Find a time slot (TS) which is feasible for the task T while minimizing the multi-objective cost-based metrics defined by (1).
        Allocate the TS on the resource r to the task T
        Update the list of available slots to the resource r
    end for
end while
Compute the makespan and allocation-cost of the application.
```

Table 1: Lublin workload model parameter values.

Workload parameter	Value
JobType	Batch JOBS
Maximum number of CPUs required by a job(p)	1000
uHi	Log_2(p)
uMed	uHi-2.5
Other parameters	As created by Lublin model

Table 2: Simulated EDG testbed resources.

Resource name (Location)	Number of CPUs	Single CPU rating(MIPS)	Processing cost(G$)
RAL(UK)	20	1140	0.0061
Imperial College(UK)	26	1330	0.1799
NorduGrid(Norway)	265	1176	0.0627
NIKHEF(Netherlands)	54	1166	0.0353
Lyon(France)	60	1320	0.1424
Milano(Italy)	135	1000	0.0024
Torina(Italy)	200	1330	1.856
Catania(Italy)	252	1200	0.1267
Padova(Italy)	65	1000	0.0032
Bologna(Italy)	100	1140	0.0069

At this stage, the scheduling algorithm which uses the best-effort QoS for scheduling, is simulated and tagged as the BE. As the number of the resources is *m* and the resources are heterogeneous in terms of CPU rating and allocating-cost, a heuristic algorithm needs to be taken into account to select a suitable resource in the best-effort QoS. In BE, the exploited heuristic method selects a resource with the minimum number of tasks in the waiting and running queues. The majority of the resource management systems make it possible for users to obtain the number of the tasks in the waiting and running queues [13].

An application scheduling algorithm using cost model is presented by Singh et al. [12, 13]. Their algorithm has provisioned a set of the slots to optimize performance under the minimum allocation-cost in order to execute application on the provisioned slots. This cost-modeled algorithm makes a trade-off between scheduling and allocation-cost based on trade-off factor. After that, the scheduling takes place using a multi-objective genetic algorithm [19], as well as simulating the algorithm. It is tagged as the MOGA for brevity [12, 13].

The FCTT, the MOGA and the BE algorithms are simulated and their performance is evaluated through conducting a number of experiments. Finally, the results from the algorithms are compared with one another. In the next section, simulation results which are compared will be thoroughly analyzed.

6 Analysis of results

In this section, the application performance results are compared and analyzed with criteria such as the makespan, allocation-cost and runtime of the proposed FCTT algorithm along with the MOGA and the BE algorithms [12, 13]. Also, it will be shown how the proposed heuristic schedules the application through optimizing the makespan and the allocation-cost in the minimum runtime. According to the presented characteristics in the section 3, a synthetic workflow application is generated considering *"trade-off factor=0.5"*. The rest of the simulation parameters is compatible with the setups in the section 5. The Y-axis is drawn in logarithmic scale to make the experiments results discernable.

A few experiments have been conducted to determine the impact of the workflow size on the allocation-cost, makespan and runtime in terms of the number of the application tasks. It is followed by an analysis of the comparison between the FCTT, MOGA and BE algorithms. The experiments were conducted with the application tasks' sizes of 25, 50, 100, 200, 300 and 500 in order to study the impacts on the allocation-cost, makespan and runtime in the application scheduling due to the increasing number of the application tasks.

Figs. 1 and 2 show the impact of the workflow size on the allocation-cost and makespan in the application

scheduling, respectively. As Figs. 1 and 2 indicate, the allocation cost and makespan of the proposed algorithm which the average of all its instances are around 37% and 1% less than the MOGA algorithm, respectively, and one order of magnitude less than the BE algorithm. The low cost and makespan in proposed algorithm is explained by the fact that it selects a slot with the earliest start-time to run the eligible task from the whole existing slots according to the multi-objective cost metric of (1). However, the MOGA algorithm randomly selects a subset of the slots for scheduling the whole tasks. Due to the existing data dependency among tasks, if the execution of an eligible task is postponed, it will result in lengthening the makespan. In the BE algorithm, as long as the executions of the parent tasks are not completed, child-tasks will not be submitted. As the workflow graph depth is $\sqrt{n}$, the higher the number of the tasks *n* is, the deeper the workflow will be. Eventually, an increase in the workflow graph depth leads to an increase in the number of the times a task needs to wait, causing an increase in the makespan.

Fig. 3 reveals the FCTT, MOGA and BE algorithms' runtime relative to an increase in the number of the application tasks. As the figure shows, the proposed algorithm in all instances is almost three orders of magnitude less than the MOGA and the BE algorithms. The low time-complexity of the proposed algorithm is explained by the fact that it seeks the best slot for a single task just once, while the MOGA algorithm is implemented based on the genetic algorithm. One of the disadvantages of the genetic algorithms is length of their runtime. Moreover, in order to seek a subset of proper slots, the MOGA algorithm needs to repetitively plan the whole chromosomes of each generation of the population so that the best solution of each generation can be selected. The whole process involves a very high time-complexity. Therefore, the higher the number of the application tasks is, the longer the runtime of the algorithm will be. Due to Fig. 3, if the number of the tasks increases from 300 tasks to 500 tasks in the MOGA algorithm, its runtime will increase around one order of magnitude. The BE algorithm employs the best-effort service while neglecting the cost metric optimization. After the executions of all the parent tasks of a single task are completed the execution of the desired task will start which results in a longer runtime.

According to Fig. 3, due to an increase in the application tasks even when it is running 500 tasks the FCTT algorithm requires much lower runtime. The runtime required by the FCTT algorithm is around 0.7 second for 500 tasks to be executed, whereas in the MOGA algorithm, the application runtime takes almost one hour and twenty minutes. As a result, the FCTT algorithm is scalable caused by an increase in the application tasks as well as capable of scheduling huge applications with the lowest runtime in the heterogeneous environment.

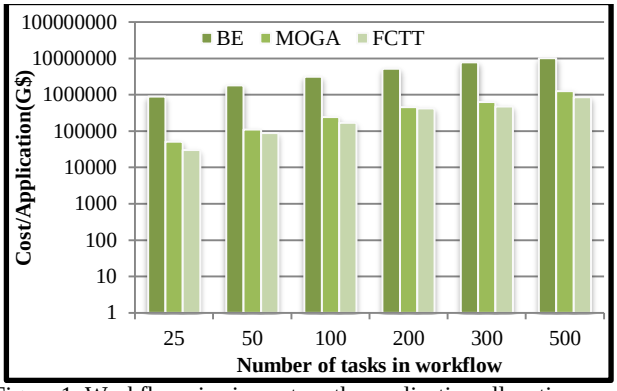


Figure 1. Workflow size impact on the application allocation-cost.

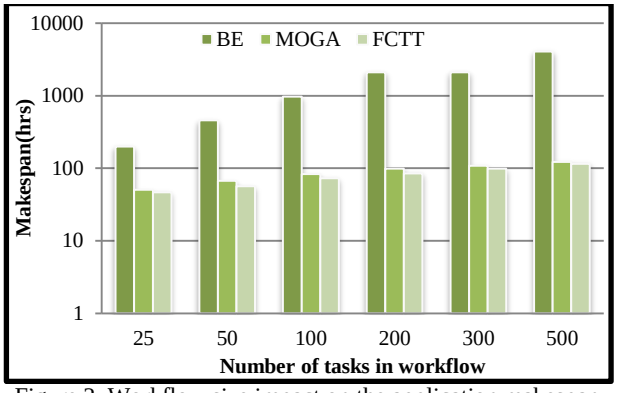


Figure 2. Workflow size impact on the application makespan.

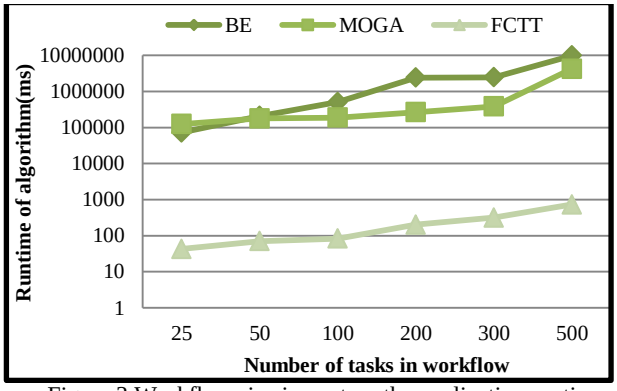


Figure 3.Workflow size impact on the application runtime.

7 Conclusion

The present paper deals with designing, implementing and evaluating the FCTT heuristic algorithm in order to schedule a workflow application. The paper seeks to optimize the multi-objective cost-time based on the proposed WPC model. To develop a real distributed environment, the resources workload is simulated based on the Lublin model. Due to many experiments conducted on a generated syntactic workflow, it was shown that the FCTT heuristic algorithm is far more effective than the existing algorithms in terms of the cost-time optimization and scalability for scheduling the workflow application.

Also, in this paper, a few experiments have been conducted to determine the impact of the workflow size on the allocation-cost, makespan and runtime in terms of the number of the application tasks. Next, it is followed by an analysis of a comparison between the FCTT, MOGA and BE algorithms. As a result, it was shown the FCTT algorithm is scalable due to an increase in the application tasks as well as capable of scheduling huge applications with the lowest runtime in the heterogeneous environment.

8 References

[1] D. S. Katz, J. C. Jacob, G. B. Berriman, J. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M. H. Su, and T. A. Prince, "A comparison of two methods for building astronomical image mosaics on a grid," in *proceedings of the 34th International Conference on Parallel Processing Workshops (ICPP 2005 Workshops)*, Oslo, Norway, 2005.

[2] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, and S. Koranda, "GriPhyN and LIGO, building a virtual data grid for gravitational wave scientists," in *11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, Edinburgh, Scotland, UK, 2002.

[3] J. D. Ullman, "NP-complete scheduling problems," *Journal of Computer and System Sciences*, vol. 10, pp. 384-393, 1975.

[4] D. Feitelson and L. Rudolph, "Parallel job scheduling: Issues and approaches," in *1st Workshop on Job Scheduling Strategies for Parallel Processing*, Santa Barbara, CA, 1995, pp. 1-18.

[5] D. Feitelson, L. Rudolph, U. Schwiegelshohn, K. Sevcik, and P. Wong, "Theory and practice in parallel job scheduling," in *3rd Workshop on Job Scheduling Strategies for Parallel Processing*, Geneva, Switzerland, 1997, pp. 1-34.

[6] F. Xhafa and A. Abraham, "Computational models and heuristic methods for Grid scheduling problems," *Future Generation Computer Systems*, vol. 26, pp. 608-621, 2010.

[7] M. Hovestadt, O. Kao, A. Keller, and A. Streit, "Scheduling in HPC resource management systems: Queuing vs. planning," in *9th Workshop on Job Scheduling Strategies for Parallel Processing*, Seattle, WA, 2003, pp. 1-20.

[8] S. K. Garg, R. Buyya, and H. J. Siegel, "Time and cost trade-off management for scheduling parallel applications on Utility Grids," *Future Generation Computer Systems*, vol. 26, pp. 1344-1355, 2010.

[9] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 260-274, 2002.

[10] M. Wieczorek, R. Prodan, and T. Fahringer, "Scheduling of scientific workflows in the ASKALON grid environment," *ACM SIGMOD Record*, vol. 34, pp. 56-62, 2005.

[11] G. Singh, C. Kesselman, and E. Deelman, "Application-level resource provisioning on the grid," in *E-SCIENCE '06 Proceedings of the Second IEEE International Conference on e-Science and Grid Computing* Amsterdam, The Netherlands, 2006, pp. 83-83.

[12] G. Singh, C. Kesselman, and E. Deelman, "A provisioning model and its comparison with best-effort for performance-cost optimization in grids," in *Proceedings of the 16th international symposium on High performance distributed computing*, Monterey, CA, USA, 2007, pp. 117-126.

[13] G. Singh, C. Kesselman, and E. Deelman, "An end-to-end framework for provisioning-based resource and application management," *Systems Journal, IEEE*, vol. 3, pp. 25-48, 2009.

[14] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, NJ, USA, 2008, pp. 1-12.

[15] J. Yu, R. Buyya, and C. K. Tham, "Cost-based scheduling of scientific workflow application on utility grids," in *First International Conference on e-Science and Grid Technologies (e-Science'05)*, Melbourne, Australia, 2005, pp. 140-147.

[16] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, pp. 1175-1220, 2002.

[17] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data management in an international data grid project," in *Grid Computing - GRID 2000: First IEEE/ACM International Workshop*, Bangalore, India, 2000, pp. 333-361.

[18] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: modeling the characteristics of rigid jobs," *Journal of Parallel and Distributed Computing*, vol. 63, pp. 1105-1122, 2003.

[19] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proceedings of the 5th International Conference on Genetic Algorithms*, Urbana-Champaign, IL, USA, 1993, pp. 416–423.

Adaptive Divisible Load Scheduling in Computational Grids

Luis de la Torre[1] Héctor de la Torre[2]
[1] Science and Technology School, Universidad Metropolitana, San Juan, PR
[2]School of Engineering, Turabo University, Gurabo, PR
Ana G. Mendez University System.
del1@suagm.edu, hectorfabio50@hotmail.com

Abstract

Several problems in science and engineering admit computational solutions that are implementable over Grid computing platforms. One problem, frequently faced by implementers is how to divide and distribute the workload into chunks among the Grid workers, the so-called load scheduling problem. Most commonly researches have studied this phenomena departing from a statics approach. This assumption is not fully functional in Grid environment where the resources are non-dedicated. This research proposed a methodology to integrate a statistic heterogeneous platform scheduling with a dynamic resources prediction to distribute the workload depending on future available resources. The SCOW algorithm is integrated to a tendency-based method, a mechanism to predict CPU utilization. These implementations can retro aliment the statistic scheduling algorithm to produce accuracy estimation to the Grid resources changes.

Kewords: Scheduling, grid computing, divisible load, divisible task, makespan, Throughput

1. Introduction

Divisible workload consists of workloads that can be partitioned into arbitrary tasks or chunks. These chunks in many cases are a core task that is repeated a number of times over different data. In single-program multiple-data (SPMD) style, these tasks are implemented as nested sequences of do-loops around the core task. Usually, a master process scheduler the chunks across all participating workers (round of data installments), so the execution time of the entire load (makespan) is minimum. In this research, is assumed that the distribution process use the network connection in a sequential fashion [3]. Each data installment is followed by a receive and a compute operation, both performed by the receiving worker. The p workers can compute and receive the next tasks concurrently. In SPMD implementations, rounds are controlled by an external do-loop, which imposes the periodic character of the job's execution. The main parameters in a SPMD implementation are thus, the number of rounds, denoted below by $\boldsymbol{m}$, the number of workers involved in the concurrent computations, denoted below by $\boldsymbol{q}$ and the chuck sizes $\boldsymbol{x}_i$ to the worker i, $1 \leq i \leq p$.
Two approaches dominate among the methodologies developed for scheduling of master-workers load. These methodologies are: steady state scheduling (SSS)[6, 7], and divisible load theory (DLT) [2, 3, 4.]. Both methodologies assume dedicated resources. These assumptions make the algorithm poor in real time environment such as Grid computing platforms with non-dedicated workers.

SCOW is a periodic user-level scheduler that tunes some selected parameters in a single-program multiple-data implementation of a master-worker parallel solution. SCOW minimizes the job make-span under either maximal production per period, or perfect worker utilization. This paper presents the theoretical foundations of SCOW to maximal production per period improving with the mixed tendency based strategy for predicting the CPU utilization of workers.

UMR [12] is a DLT multi-round algorithm for scheduling divisible loads on parallel computing systems. For homogeneous systems, the method uses uniform rounds meaning that in each round, each worker receives the same amount of work. There are two versions for the UMR idea. In the original, called UMR, the uniform amount of work is increased with each round. In a revised version, called UMR2, the uniform amount is increased or decreased depending on a parameter θ. If $\theta > 1$ the amount is increased, and decreased if $\theta < 1$. The UMR scheduler maintains perfect worker utilization throughout the execution, but if $\theta < 1$ there is no perfect worker utilization for URM2. Both methods model the system with a set of affine equations expressing execution times in terms of load. These equations are similar in spirit to the one used for SCOW. The model gives the amount of work for the first round and, through a recursive formula, the increments or decrements per round. Authors in [13] improve the UMR by predict the workers CPU utilization with a mixed tendency based strategy.

This paper is organized as follows: Section 2 discusses the model. Section 3 describes the Theoretical Foundations of SCOW. Section 4 shows the Makespan minimization problem to develop a multiround divisible load scheduling algorithm for affine cost models. In section 5, the local tasks CPU utilization in a CPU prediction strategy is incorporated. SCOWS was evaluated with extensive simulation in Section 6 and the executions is discuss later in on section 7.
Finally, Section 8 concludes the paper and discusses future directions.

2. Model

As stated in 1, is assumed a total number $\boldsymbol{X}$ of core tasks. These core tasks can be agglomerated to produce different chunks sizes (portion of job). These jobs are independent in the sense that neither ordering between them, nor synchronization among them is necessary.

2.1. Notation

As illustrated in Figure 1, the STARAFFINE network consists of p +1 processor, P= {P_0,P_1,P_2, ..., Pp}. The master processor is denoted P_0 while the p workers are labeled as P_i, $1 \leq i \leq p$. There are p communication links l_i from the master P_0 to each one of the workers Pi. Let x_i be the number of units of core tasks sent to worker P_i. l_i (x) measures the time units that takes for a load x to be moved from the master to the i^{th} worker in affine mapping model. Each worker i performs two operations, as well. These operations are message reception and the actual execution of the job, referred as computation. The worker i spends $w_i(x)$ time units in executing x core tasks, $w_i(x)$ is supposed to be an affine mapping.

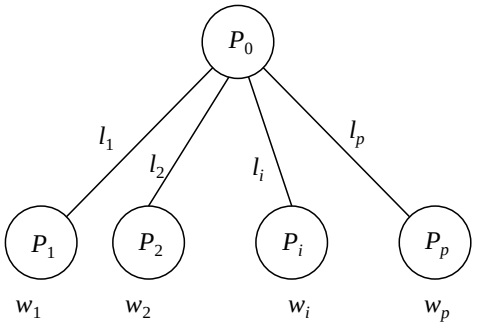


Figure1. Heterogeneous Star Graph

2.2. Architectural Model

As mentioned before, within a round, the master performs a sequence of data sends operations. Each data retrieval and send is followed by the data transmissions (l) over the network. Receive and compute (w) operations are performed by the workers upon the arrival of the data package. As a result, two major concurrent time segments are distinguished within a round: L, time spent by all network link in transmitting a round of data chunks; W maximum time spent by all workers in completing the reception of the data and execution of the corresponding data chunk. This research assumes the *full overlap, single-port model*. In this model, the master uses the network connection in a sequential fashion and the workers can perform the computation concurrently with data reception. One of the assumptions in this research is that the workers are non-dedicated processors. In Grid environmental the CPU power is distributed between local task and the Grid users.

2.3. Affine mapping

This subsection is a brief discussion of the affine maps in which the mathematical model is based. The execution times to each operations of data communication, and tasks execution vary as an affine mapping on the number of agglomerated core tasks x. This is,

$$l_i(x) = l_i \cdot x + L_i \quad (1)$$
$$w_i(x) = w_i \cdot x + W_i \quad (2)$$

for $1 \leq i \leq p$, where L_i is the initial cost of establishing a connection between the master P_0 and worker i, l_i is the send time associate to the data of a single core task; W_i is the overhead (startup time) of the computation in processor i and w_i corresponds to the execution time of a single core task.

3. Theoretical Foundations of SCOW

SCOW is designed as a periodic user-level scheduler for allocating agglomerated core tasks on parallel heterogeneous computing systems. This means that the mathematical framework behind SCOW is designed to return optimal constant values of the three parameter describe above $\boldsymbol{m}$, $\boldsymbol{q}$, and $\boldsymbol{x}_i$ to a master-worker SPMD implementation; under some specific constraints. Indeed, SCOW minimizes the make-span of the job under either maximal production per round or perfect system utilization [1]. In this research, the maximal production per period SCOW ability is selected, refers to a distribution of agglomerated core tasks across the workers that maximizes the total number of tasks completed in a round.

3.1. Maximal periodic production

In this section a brief description of scheduler theory is presented. The maximal production problem (MP) is a problem that imposes a restriction in the period to find the best approximation to the maximum number of task performed.

3.2. Problem

Suppose

$$m\sum_{i=1}^{p} x_i = X \tag{3}$$

where m is a number of round of data installments. The (MP) problem is stated as follows: Given a time period T, find a subset of $q+1$ workers such as

Maximize $$\sum_{i=1}^{q+1} x_i \tag{4}$$

Subject to $$L = \sum_{i=1}^{q+1} l_i(x_i) \le T \tag{5}$$

$$W = \max\{w_i(x_i)/1 \le i \le q+1\} \le T \tag{6}$$

$$x_i > 0,\ 1 \le i \le q+1 \tag{7}$$

3.3. Solution

Let MAXTASK(T) be the optimal solution of the previous problem. The next theorem provides a close form solution for the MP problem in homogeneous platform.

Theorem 1. *Let T be a real nonnegative numbers, $w_i=w$ and $l_i=l$ for all i and*

$$y = w^{-1}(T) \tag{9}$$

$$q = \lfloor T / l(y) \rfloor \tag{10}$$

$$T_\varepsilon = T - q \times l(y) \tag{11}$$

Then

$$\text{MAXTASK(T)} = qy + \max\{0, l^{-1}(T_\varepsilon)\} \tag{12}$$

The previous theorem has a possible extension to the heterogeneous problems. At this moment the best solution is an approximation theorem.

Theorem 2. *Let T be a real nonnegative number, p be a positive integer. The method:*

1. *Sort the workers by increasing communication times. Renumber them so that $l_1 \le l_2 \le \ldots \le l_p$.*
2. *Let $y_i=w^{-1}(T)$ for $1 \le i \le p$ and q the largest index so that $\sum_{i=1}^{q} l_i(y_i) \le T$. If $q<p$, let $T_\varepsilon = T - \sum_{i=1}^{q} l_i(y_i)$; otherwise let $T_\varepsilon=0$.*

Return the values to construct the following inequalities:

i. $$\left|\text{MAXTASK}(T) - \sum_{i=1}^{q} y_i + \max\{0, l_{q+1}^{-1}(T_\varepsilon)\}\right| \le \frac{\sum_{i=1}^{q+1} L_i}{l_{q+1}}$$

ii. *If the period T is large enough*

$$\left|\text{MAXTASK}(T) - \sum_{i=1}^{q} y_i + \max\{0, l_{q+1}^{-1}(T_\varepsilon)\}\right| \le \frac{L_{q+1}}{l_{q+1}}$$

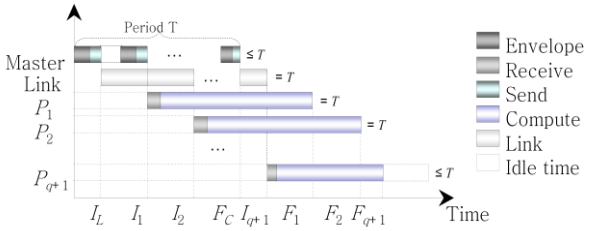


Figure 2. Gantt Char interpretation to the proposes solution

The theorem 2 gives an approximate to the optimal solution to the MP problem. In section 4 this approximation is used as a restriction to formulate a makespan minimization problem.

Figure 2 is a graphical representation to the solution given by the theorem 2. This solution follows the principle of bandwidth-centric [3, 4] because the priorities do not depend on the workers computation capabilities, only on their communication capabilities. Elsewhere, the number of select processor is directly affected by the computation capacity. In Grid computing the computational capacity depends on the local CPU utilization. This tendency is estimated using prediction strategy described in section 6

3.4. Last round modification

A common condition to get an optimal schedule is that all processor finishes the work at the same time. Modifications of the last round are used to impose the condition that all processors end operating at the same time[1]. This last round modification introduces a constant makespan reduction of $1/2T$.

4. Makespan minimization

The makespan minimization problem constrained to maximal production (MMP-MP) solution approximation and the workers order described in theorem 2 is formulate as follow:

Minimize $\mu(T) = (M+1/2)T + l_1(Y)$

subject to $$X = v\left(\sum_{i=1}^{q} y_i + y_{q+1}\right) \tag{13}$$

$$T = w_i(y_i), \qquad 1 \le i \le q \tag{14}$$

$$T = \sum_{i=1}^{q+1} l_i(y_i) \tag{15}$$

$$T \ge w_{q+1}(y_{q+1}) \tag{16}$$

$$y_i \ge 0, \qquad 1 \le i \le q \tag{17}$$

$$p \ge q+1 \tag{18}$$

The problem is solved by Lagrange multipliers [9]. This solution is stated in the next theorem.

Theorem 3: *Let X be a nonnegative real number and q a positive integer (q+1 ≤ p) . Then the solution to the MMP-MP problem without restriction* 16 *with q+1 processor is,*

$$T = \frac{1}{a(q)}\sqrt{\frac{b(q)w_1X}{\frac{1}{2}w_1 + l_1}} + \frac{b(q)}{a(q)} \qquad (19)$$

where $$a(q) = \sum_{j=1}^{q+1}\frac{1}{w_j} + \frac{1}{l_{q+1}}\left(1 - \sum_{j=1}^{q+1}\frac{l_j}{w_j}\right) \qquad (20)$$

and $$b(q) = \sum_{j=1}^{q+1}\frac{W_j}{w_j} + \frac{1}{l_{q+1}}\left(\sum_{j=1}^{q+1}L_j - \sum_{j=1}^{q+1}\frac{l_jW_j}{w_j}\right) \qquad (19)$$

the theorem 3 permit reformulates the MMP-MP, in the problem to finding the minimal value of $\mu(T)$ with *i* ranging over the subset the *i* that satisfying

$$w_{i+1}(y_{i+1}) \le T \qquad (20)$$

5. CPU prediction strategy

In Grid computing typically the resources are non-dedicated, that is, the availability of the full processing speed is no guaranteed. Let *S*, the full processor speed. The local task execution generates a CPU utilization, if the *Utilization* can be predicted them the *ActualSpeed* can be computed as follows:

ActualSpeed = *S* * (100%-*Utilization*) (21)

This *ActualSpeed* is used as retro-alimentation information to the static scheduler. To predict the CPU load and utilization is used a time series prediction approach [10, 11] that has been probed the effective empirically.

The idea of this prediction strategy is based on the assumption that if the current value increases, the next value will also increase, and if the current value decreases, the next value will also decrease.
Formally, we can write:

```
If (U_{T-1} < U_T)
     IncrementValueAdaptation()
   P_{T+1} = U_T + IncrementValue
Else if(U_{T-1} > U_T)
   DecrementFactorAdaptation()
   P_{T+1} = U_T × IncrementValue
```

Where,
U_T: the measured utilization at measurement T,
P_{T+1}: the predicted utilization for measurement U_{T+1},
H: the number of historical data points used in the prediction.

Increment value and decrement factor can be calculated as:

```
Procedure: INNCREMENTVALUEADAPTATION()
Mean = (1/n)∑i
RealIncValue =U_T − U_{T-1} ;
NormalInc = IncrementValue + (RealIncValue −
          IncrementValue) × AdaptDegree;
if (U_T < Mean)
  IncrementValue = NormalInc;
Else
   PastGreater = (number of data points>UT)/H
   TurningPointInc=IncrementValue×PastGreater
   IncrementValue=Min(NormalInc,
                 TurningPointInc)
```

AdaptDegree can range from 0 to 1 and expresses the adaptation degree of the variation. The best values for input parameters such as AdaptDegree and DecrementFactor are determined empirically.

6. Experiment

In table 1 the group of numerical values selected to perform the simulation is presented. The values are used to predict the corresponding SCOW-MP and UMR version develop in [13]. These predictions are made using the mathematical equations underlying them and a random variable to CPU simulation is also generated using gamma function

Table 1: Simulation Parameter

Parameter	Values
Number of workers	p=10,20,30,40
Agglomerated tasks	X = 1000
Computational rate	S_i=.5+*randonvariavle* $w_i = 1/S_i$
randonvariavle	is also generated using gamma function with fixed arrival time landa=.5 and beta = 1
Transfer rate	B_i = 1.1p to 1.1p + 1; l_i =1/b
Computational latency	$cLat_i$ = 0:03; W_i = cLat;
Communication latency	$nLat_i$ = 0:03; L_i = nLat

7. Result

It is worth remarking that UMR methods and the optimal number of rounds, and perform no discretization on the amount of work per round. Thus, in order to make the comparisons possible, SCOW discretizations are made on the number of rounds and not on the amount of agglomerated tasks per round. The randomly chosen values are shown in Table1.

The numerical prediction of the performances of UMR and SCOW are shown in Table 2.

Table 2: Comparison Between SCOW and UMR

	SCOW-MP	UMR	UMR2
Normalized Make-span	1.000	1.012	1.032
Normalized Workers CPU Utilization	1.000	1.008	1.009

Table 2 shows the comparison between SCOW-MP, UMR and UMR2, averaged over similar (in the number of workers) experiment. All the scheduler was improved by a last round modification in order keep consistent the comparison.
The first row shows the ratio of make-span achieved for the 3 schedulers. The second row shows the similar ratio for the system utilization, but at this time the ratio is inverted, because the maximal values is the best. The main observation is that SCOW-MP outperforms UMR and UMR2 on average. The SCOW-MP is the best algorithm in Make-span and system utilization.

8. Conclusions

Many researches in maximal throughput in lineal model can be found in the literature. These results are developed to the problem formulated for fixed sizes tasks. In this research the problem is exported to affine model and in contrast the goal is maximize the production, that is, the total number of tasks processed.

The contribution of this research includes an optimal solution in the homogeneous case and approximate solution in the heterogeneous case, integrate with a CPU prediction strategy to perform a scheduler reliable in a Grid environment. The makespan and system utilization of two algorithms is also compared.

The results show that the proposed SCOW-MP algorithms outperform the competitors. Future work includes the development of a strategy to predict the network utilization; due to SCOW is a bandwidth-centric algorithm.

9. Acknowledgments

This work was made possible by funding from the Caribbean Computer Center of Excellence (CCCE) under NSF Award number CNS-0940522.
Thanks are due to Dr. Juan F Arratia and Dr. Oliva Primera-Pedrozo from the Universidad Metropolitana-Cupey for their support.

References

1. L. de la Torre, "Scheduling divisible tasks under production or utilization constraints", PhD diss., University of Puerto Rico, Mayaguez, Puerto Rico 2010.
2. Y. Yang, K. van der Raadt, H. Casanove, Multiround Algorithms for Scheduling Divisible Loads, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 11, 2005.
3. C. Banino, O. Beaumont, L. Carter, J. Ferrante, A. Legrand and Y. Robert, Scheduling Strategies for Master-slave Tasking on Heterogeneous Processor Platforms, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No. 4, pp. 319-330, 2004.
4. M. Drozdowski and P. Wolniewicz Optimum Divisible Load Scheduling on Heterogeneous Stars with Limited Memory, *European Journal of Operation Research*, Vol. 172, No. 2, 2006.
5. N. Jones and P. Pevzner An Introduction to Bioinformatics Algorithms, MIT Press, 2000.
6. V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi. Scheduling Divisible Loads in Parallel and Distributed Systems. *IEEE Computer Society Press*, 1996.
7. O. Beaumont, H. Casanova, A. Legrand, Y. Robert, Y. Yang: Scheduling Divisible Loads on Star and Tree Networks: Results and Open Problems. *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 3, 2005, 207-218.
8. D. Bertsekas. Constrained Optimization and Lagrange Multiplier Methods. *Athena Scientific, Belmont, Mass.*, 1996.
9. D. Bertsekas, editor. Constrained Optimization and Lagrange Multiplier Methods. Athena Scienti_c, Belmont, Mass., 1996.
10. L. Yang, J.M. Schopf, and I. Foster, Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decision in Dynamic Environments, *SuperComputing 2003*, Phoenix, Arizona USA November 2003.
11. L.Yang, I. Foster, and J.M. Schopf, Homeostatic and Tendency-Based CPU Load Predictions, *International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice,France, April 2003.
12. Yang Yang and Henri Casanova, UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads; *Proceeding of the International Parallel and Distributed Processing Symposium* (IPDPS'03), Nice, France, April 2003.
13. Said Elnaffar and Nguyen The Loc. "Enabling Dynamic Scheduling in Computational Grids by Predicting CPU Utilization"; WSEAS Transactions on Communications Issue 12, Volume 4, pages 1419-1426. December 2005.

SESSION

GRID MIDDLEWARE + RESOURCE DISCOVERY

Chair(s)

TBA

Grid Resource Discovery using Tree Data Structure for Multi-Trait Requests

Leyli Mohammad Khanli[1], **Saeed Kargar**[2], **Ali Kazemi Niari**[2]
[1] CS Department, University of Tabriz, Tabriz, Iran
[2]Department of Computer Engineering, Islamic Azad University, Tabriz Branch, Tabriz, Iran

Abstract *- Grid is an extensive environment in which different resources are dispersed geographically. A user may need a resource or a combination of resources in order to solve a problem. The task to find such a resource is borne by resource discovery algorithms. Therefore, the resource discovery algorithms are of high importance in grids. The methods proposed to resource discovery so far have not suggested a method to discover several resources simultaneously in the form of a request.*

In this paper, we have proposed a method that is able to discover simultaneously the desired number of the resources for the user. In our proposed algorithm, the cost of the resource discovery is very low. By means of this method, a user will be able to request several resources simultaneously in one format.

The results of simulations indicate that fewer numbers of nodes meet in the resource discovery stages in this method than that in the other suggested methods. Compared to other methods, this method also creates less traffic in the network.

Keywords: Grid, Resource Discovery, Multi-Trait Requests

1 Introduction

Grid is a new technology that enables the users to share different resources from long distance by using network and communication infrastructures. These resources can be heterogeneous and far from one another geographically [1]. Different methods have been suggested for resource discovery. Centralized methods [2-4] are among the methods that have been used. These methods have a central server that manages all nodes. In such environments as grids where there is a large number of users, there has been mounted a bottleneck in the server region, which reduces the system efficiency. The other methods are decentralized. There is not a centralized server in these methods which can manage all nodes. Flooding-based and Random-based are instances of this method. Although these methods have removed many faults of the previous methods, the system efficiency reduces with the increase in the number of nodes and with the variation in the resources.

Recently, there have been introduced distributed methods that use tree structure for resource discovery. These methods are more optimal in terms of the number of the produced traffic, etc. However, in none of these methods occurs the discovery of several resources simultaneously in one format. “A resource discovery tree using bitmap for grids” [5] and “FRDT: Footprint Resource Discovery Tree for grids” [6] and the methods proposed in [7-8] are instances of this method.

This paper proposes a method for resource discovery that uses a weighted tree structure as the method [6] does with this difference that the former makes it possible for the user to search for several resources simultaneously. The simulations show that the algorithms suggested in this paper find one or more resource for users without recourse to unnecessary and extra nodes, creating less traffic.

Below are discussed some of the works done with regard to the resource discovery so far. Section 3 is concerned with the explanation of the method suggested in this paper. Section 4 is associated with the results of the simulations. Finally, section 5 concerns Conclusion and further studies.

2 Related work

Various methods have been proposed as regards the resource discovery in the grid. Below are presented some of these methods.

Matchmaking is one of these methods [9] in which matchmaking service find a correspondence between requests and entities. Most methods use this algorithm [10-13].

Another group of methods uses a Semantic Communities among the nodes in the grid [14-17].

Juan Li. [18] has proposed a resource discovery method based on the Semantic Communities. In this method, a Semantic structure is used to group the similar nodes; therefore, the request for the resource discovery is sent to the related nodes only.

There is another method suggested recently for the resource discovery which makes use of tree structure [5]. A series of bitmaps have been used in the nodes. Upon the resource discovery, the user's requests are transformed to these formats and delivered to one of the nodes existing in the environment. These nodes utilizes AND operation to discover the resources required by the users.

In the previous work by the authors [6], a weighted tree structure had been used for the resource discovery. In this method are used a series of bitmaps that maintain the path to

the target in addition to keep the information of the resources existing in the environment.

In contrary to all previous methods, the method proposed in this work is able to discover a combination of the resources for the users at the same time. Another advantage of this method is that it is able to perform resource discovery without recourse to unnecessary nodes.

3 Our proposed method

As mentioned earlier, this method is based on a weighted tree structure. The information of the resources in the nodes will be stored in the form of a tree data structure called "*Resource-Tree*" (RT). Through RT, the information of the combined resources will be stored in the nodes, and the user's requests will be guided to the appropriate paths in the environment. To get more familiar with this method, the general structure of RT and the format that is stored in each field RT will be discussed in the later subsection, and then the resource discovery will be discussed in next subsections.

3.1 Resource-Tree (RT)

As pointed out before, the method proposed in this paper uses a tree data structure called RT. The size of RT depends on the type of the resources in the environment. RT includes fields in which the information related to the local node resource and/or the information of the children of this node will be stored. Fig. 1 shows an instance of RT. This RT is devised for an environment which shares 3 kinds of Operating Systems (OS) and 2 kinds of RAM. It is noted that the general structure of RT is known for all nodes in the environment. Not all nodes in the environment will use all fields in RT, but they will use some of these fields depending on the resources at hand. A sample of field RT is shown in Fig. 2. This field consists of two columns called "*Resource*" and "*Children*". The meaning of the numbers stored in these columns is explained through an example.

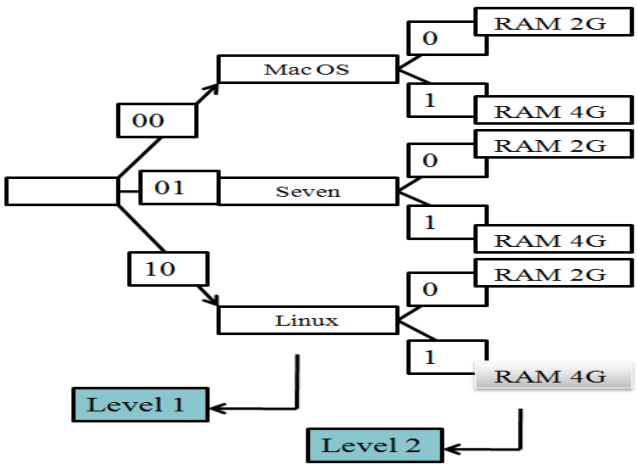


Fig. 1: An example of Resource-Tree (RT).

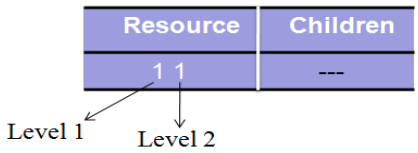


Fig. 2: The content of the highlighted field in Fig. 1.

Assume that the field in Fig. 2 has been stored in the place highlighted in Fig. 1. In *Resource* column, the number 11 has been stores. This means that this node possesses the resource level 1 (OS) and the resource level 2 (RAM). Considering the place where is stored in RT, it possesses Linux and RAM 4G. For better comprehension, you can look at Fig. 3 and Fig. 4. Fig 3 illustrates the assumed environment of our grid on a weighted tree structure. As seen in the figure, each node shares a resource or a combination of resources in the environment. How the resource information is stored inside some of the nodes is clearly seen in Fig. 4.

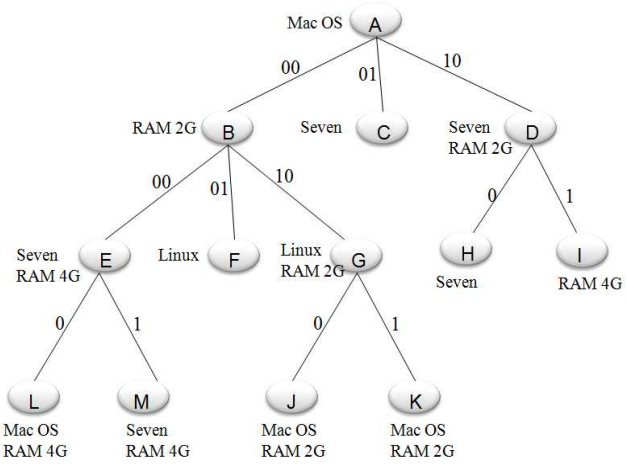


Fig. 3. An example of typical grid environment on a weighted tree.

Each part of Fig. 4 is explained subsequently. Just for simplification, O1, O2, O3, R1 and R2 will be used to refer to MacOS, Seven, Linux, RAM 2G and RAM 4G respectively. In Fig. 4(a), the RT stored in the nodes J and K are shown. As both nodes share the resources Mac OS and RAM 2G in the environment, they have similar RTs. Number 11 stored in *Resource* column means that in this place exists the information related to a combinational resource that possesses both OS and RAM, and they are Mac OS and RAM 2G with consideration of the place where they are stored. The mark "---" in the *Children* column indicates that these resources are the local resources of the node itself.

For another example, look at Fig. 4(b) related to the node G in Fig. 3. This node which receives information from its own children in addition to its own local information will store all this information in its RT as shown in Fig. 4(b). This node itself consists of O1 and R1 which will store the information of which as 11 in *Resource* column and mark "---" in *Children*

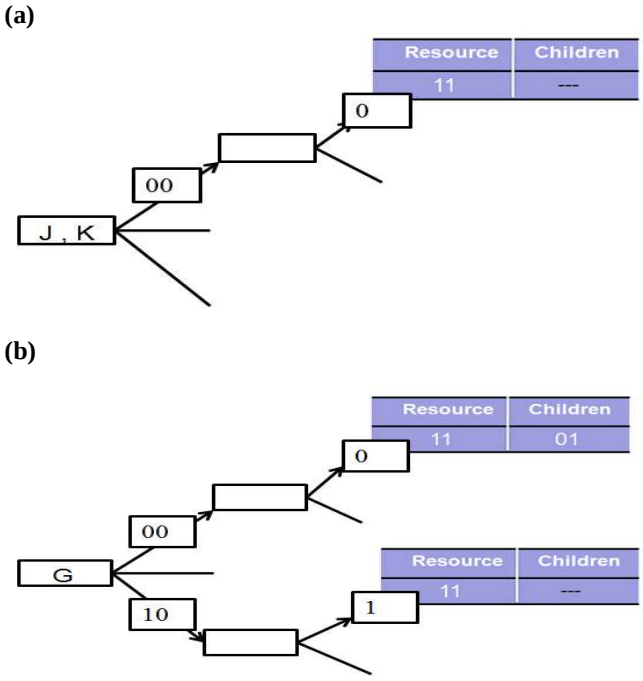


Fig. 4. The stored RTs in the (a) nodes J and K; (b) node G.

column, but will store the information related to children (O1 and R1 from both children) in the related place (number 11 in *Resource* column). Number 01 stored in *Children* column means that because this node has two children; therefore, it allocated at least 2 bits to each node, which is 0 and 1 as in Fig. 3. Since it receives similar information from both its children, the children's weight; that is 0 and 1 is written beside Children column (01).

It is pointed out that the information the method proposed here is stored in nodes distributary. This reduces the volume of the information stored in the nodes.

3.2 Multi-resource discovery

As seen in Fig. 5, there is a sample of the request form. The request form consists of two columns, *Location* and *Resource*. *Resource* Column resembles the column with the same title in RT, and its bits indicate the existence or non-existence of resources. The other column; that is, *Location* column, indicates a field to which referral will be made in every node in the course of the resource discovery.

Request	
Location	Resource
XX0	01

Fig. 5. A sample of Request form.

For example in Fig. 5, when the user needs a resource R1, the information related to R1 in RTs may be stored in each of three fields at the address of 000, 010 and 100 (Fig. 1). That is to say, for the applicant R1, the resource level 1; i.e., OS is not important, and only the second path ending in RAM is of importance.

As such, in *Location* column, sign XX (X means an unimportant state) is stored, and any field that receives this request searches for three fields at 000, 010 and 100.

In Fig. 6, a sample of the resource discovery is shown. As seen in this figure, a user needs the resources O1 and R1 simultaneously, and delivers a requested form as shown in this figure to the node D in the tree. Receiving this form, this node immediately refers to the same column in its RT using the position written in *Location* column, and compares *Resource* columns. But as it is seen, this position does not exist in the RT of node D. Thus, it passes the request to its parent node. Referring to a place in its RT, node A compares *Resource* column of the request with *Resource* column in the related place and finds a correspondence in the second line and sends the request to a child with weight 00. Node B, too, repeats this process, and delivers the request form to node G. Finding a correspondence in the related field and referring to *Children* column, node G notices this resource in children with edge 0 and 1 and sends the request to one of the nodes selectively (node J here). Finally, node J finds the requested resource for the user and reserves O1 and R1, and then sends a successful response to the origin node. As seen, the resource discovery method suggested in this paper is simple and does not meet any extra nodes.

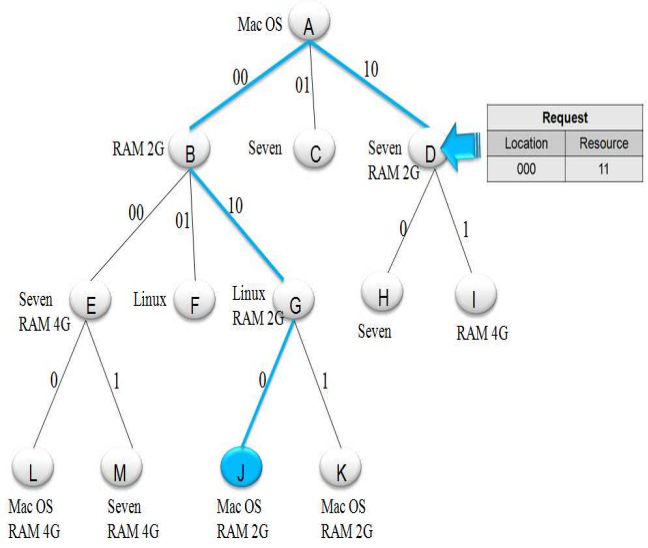


Fig. 6. An example of resource discovery in our method.

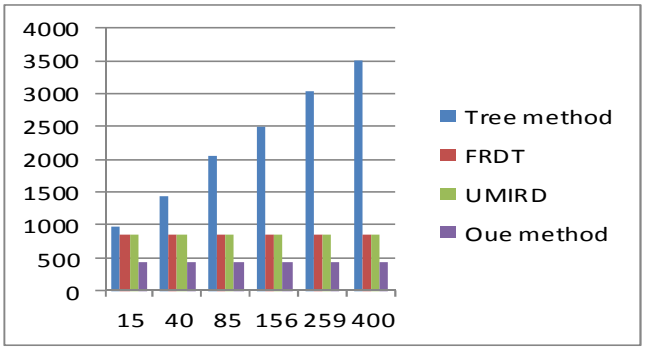
4 Simulation results

The simulations required for this work have been performed in MATLAB environment. The resources have been distributed randomly in this environment, and the requests, too, have been delivered to every tree node randomly. The height of the trees has been assumed 4 as in [5,6,19].

Since we did not find a method that can discover several resources in one format at the same time, we compared our method with other available methods with one resource. To compare multi-resources, we assumed that other methods discover the users' requested resources altogether in one place.

In the first simulations, we compared our method with "A resource discovery tree using bitmap for grids "[5] (which is called tree method), "Using Matrix indexes for Resource Discovery in Grid Environment" [8] (which is called UMIRD) and " FRDT: Footprint Resource Discovery Tree for grids "[6] methods. In Fig. 7, we supposed that the user requested different number of resources. In these tests, it is supposed that the 100 of the users, requested different number of resources. In our method, 100 requests will be sent but in other ones, for example for request six resources (Fig. 7(b)), 600 separate requests should be sent. As observed in the Fig. 7, the number of the nodes visited in our method is lower than other methods.

(a)



(b)

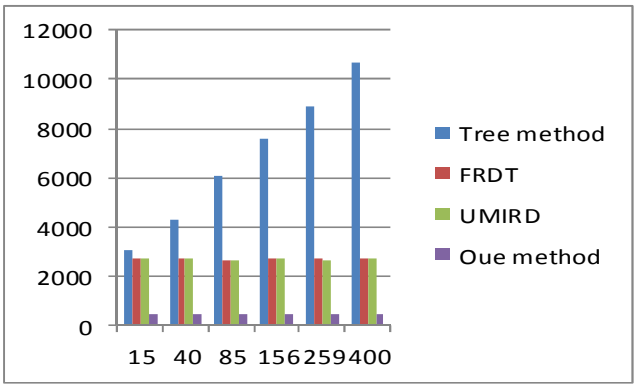
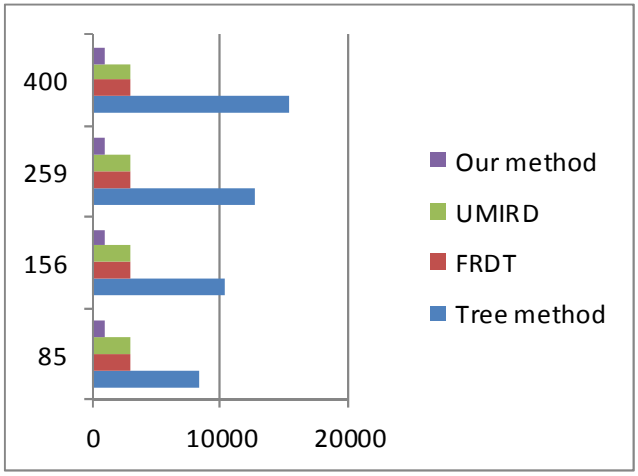


Fig. 7. The number of the nodes met by the users' requests during the resource discovery that the user requests; (a) two resources; (b) six resources.

In the next simulations, the traffic produced by the methods tree method, UMIRD, FRDT and our method upon resource discovery was compare, which is shown in Fig. 8. In these tests, it is supposed that the 300 of the users, requested different number of resources. As shown, our method is able to discover a desired number of resources for the user producing the least traffic and not referring to unnecessary nodes. Therefore, this method is more effective in the grid environment with many resources.

(a)



(b)

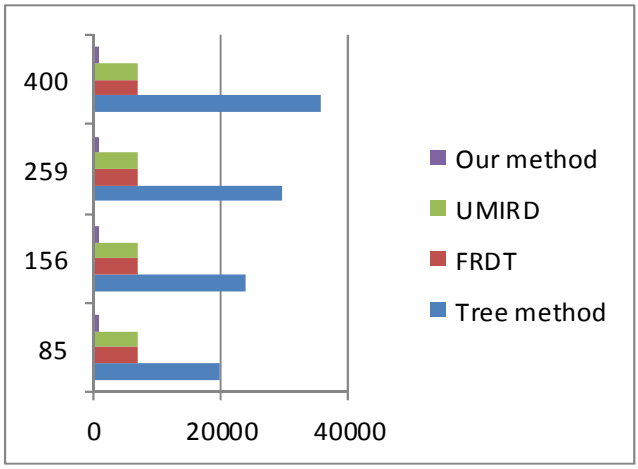


Fig. 8. The traffic produced by the different methods during the resource discovery for 300 users that each user requests: (a) three resources; (b) seven resources.

In the last tests, our method was compared with methods flooding-based, MMO [20-21], Tree method [5] and FRDT [6]. In this experiment, the mean of the number of met nodes was compared in different methods. It was assumed that any user would request only one resource (Fig. 9).

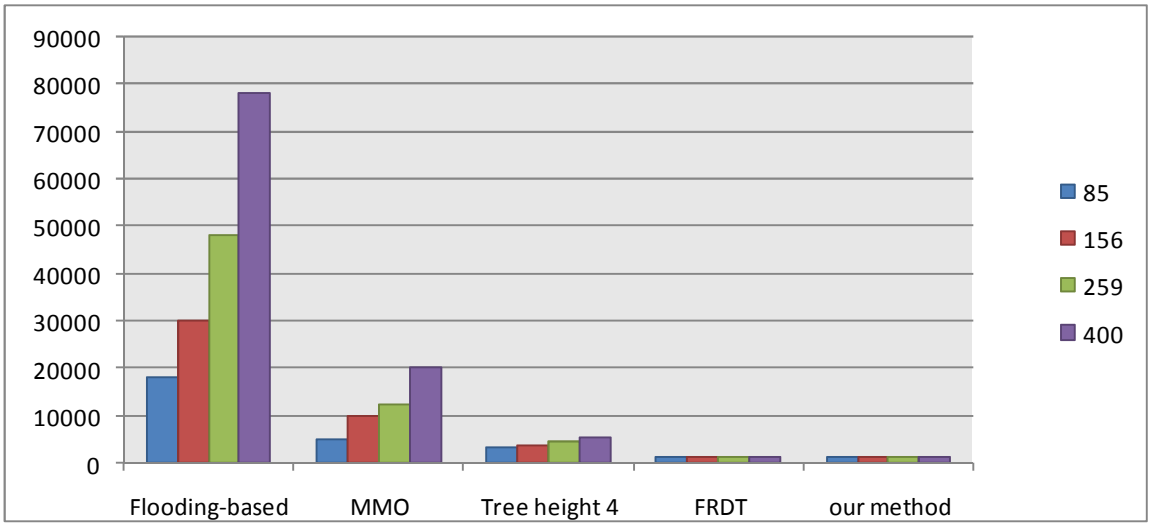


Fig. 9. The mean of the number of met nodes by different methods.

5 Conclusions and future work

As discussed earlier, we have proposed a method that is able to discover simultaneously the desired number of the resources for the user. In our proposed algorithm, the cost of the resource discovery is very low. The results of simulations indicate that our method is more efficient than other methods. In the future, the researchers will try to suggest a method which takes into account such factors as the cost, geographical distance, etc. for the resource discovery.

6 References

[1] I. Foster and C. Kesselman. "The Grid 2: Blueprint for a New Computing Infrastructure". Morgan Kaufmann Publishers Inc., San Francisco, CA, 2003.

[2] A. A. Chien, B. Calder, S. Elbert, K. Bhatia, "Entropia: Architecture and performance of an enterprise desktop grid system", J. Parallel Distrib. Comput. (Elsevier), vol. 63, pp. 597-610,2003.

[3] F. Berman, et al., "Adaptive computing on the grid using AppLeS", TPDS, vol. 14, pp.369-382, 2003.

[4] M.O. Neary, S.P. Brydon, P. Kmiec, S. Rollins, P. Capello. "JavelinCC: Scalability issues in global computing". Journal of Future Gener. Comput. Syst. (Elsevier), Vol. 15, pp. 659-674, 1999.

[5] Chang, R-.S. and M-.S .Hu. "A resource discovery tree using bitmap for grids". Future Generation Computer Systems (Elsevier), vol. 26, pp. 29-37, 2010.

[6] L.M Khanli, and S. Kargar. "FRDT: Footprint Resource Discovery Tree for grids". Future Gener. Comput. Syst. (Elsevier), vol. 27, pp. 148–156, 2011.

[7] L.M Khanli, A. Kazemi Niari and S. Kargar. "An Efficient Resource Discovery Mechanism Based on Tree Structure". In the 16th International Symposium on Computer Science and Software Engineering (CSSE 2011), p. 48-53, 2011.

[8] Leyli Mohammad Khanli, Saeed Kargar, Ali Kazemi Niari. "Using Matrix indexes for Resource Discovery in Grid Environment". In the 2011 International Conference on Grid Computing and Applications (GCA'11), Las Vegas, Nevada, USA, pp. 38-43, 2011.

[9] R. Raman, M. Livny, M. Solomon, "Matchmaking: distributed resource management for high throughput computing", In the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC-7'98), pp. 140, 1998.

[10] Ye Zhu, Junzhou Luo, Teng Ma, "Dividing Grid Service Discovery into 2-stage matchmaking", ISPA 2004 (LNCS), vol. 3358, pp. 372–381, 2004.

[11] Sanya Tangpongprasit, Takahiro Katagiri, Hiroki Honda, Toshitsugu Yuba, "A time-to-live based reservation algorithm on fully decentralized Resource Discovery in Grid computing", Parallel Computing (Elsevier), vol. 31, pp. 529-543, 2005.

[12] Simone A. Ludwig, S.M.S. Reyhani, "Introduction of semantic matchmaking to Grid computing", J. Parallel Distrib. Comput. (Elsevier), vol. 65, pp.1533 – 1541, 2005.

[13] Ami T.Choksi, Devesh Jinwala, "Improving Semantic Matching of Grid Resources Using refined Ontology with Complement Class", Journal of AICIT, vol. 2, no. 5, pp.129-139, 2010.

[14] J. Li. and Son Vuong, "Grid resource discovery using semantic communities". In the proceedings of the 4th International Conference on Grid and Cooperative Computing, Beijing, China, 2005.

[15] Juan Li, Son Vuong, "Semantic overlay network for Grid Resource Discovery", In Grid Computing Workshop, 2005.

[16] Cheng Zhu, Zhong Liu, Weiming Zhang, Weidong Xiao, Zhenning Xu, Dongsheng Yang, "Decentralized Grid Resource Discovery based on Resource Information Community", Journal of Grid Computing (Springer), vol. 2,no. 3, pp. 261-277,2004.

[17] Thamarai Selvi Somasundaram, R.A. Balachandar, Vijayakumar Kandasamy, Rajkumar Buyya, Rajagopalan Raman, N. Mohanram, S. Varun, "Semantic based Grid Resource Discovery and its integration with the Grid Service Broker", In the proceedings of 14th International Conference on Advanced Computing & Communications (ADCOM 2006), pp. 84–89, 2006.

[18] J. Li, "Grid resource discovery based on semantically linked virtual organizations". Future Gener. Comput. Syst. Vol. 26, pp. 361–373, 2010.

[19] Mastroianni, C., D. Talia and O. Versta. "Evaluating resource discovery protocols for hierarchical and super-peer grid information systems". In the proceedings of the 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, PDP'07, February 7– 9, pp. 147–154, 2007.

[20] Marzolla, M., M. Mordacchini and S. Orlando. "Resource discovery in a dynamic environment". In the proceedings of the 16th International Workshop on Database and Expert Systems Applications, DEXA'05, September 3–7, pp. 356–360, 2005.

[21] Marzolla, M., M. Mordacchini and S. Orlando. "Peer-to-peer systems for discovering resources in a dynamic grid". Parallel Comput. Vol. 33, pp. 339–358, 2007.

ITU-GRAM+: A WEB Based Grid Middleware

G. Bengi TURKEL
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
bengi.sendur@avea.com.tr

D. Turgay ALTILAR
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
altilar@itu.edu.tr

Abstract— **This paper describes the Grid middleware (ITU-GRAM+) which is implemented by using only open source software based on trend web technologies. The ITU-GRAM+ allows users to access computational and data resources, submit their jobs and track job status via standard interfaces. Main goal of ITU-GRAM+ is mapping the job resource requests to the resources in a way that will satisfy both the application users and resource owners. In order to satisfy this requirement ITU-GRAM+ is designed to have Resource Discovery, Resource Allocation, Job Submission, Job Execution and Job Monitoring components. Some of these components in ITU-GRAM+ work with SOA architecture to facilitate a communication between resources and resource manager which also gain flexibility to the system in terms of adding new resources and new users. For this communication ITU-GRAM+ exposes different web services to run both on resources and resource manager. In this paper all details about components are described with their additional functionality comparing to existing middleware.**

Keywords-Grid Computing, Grid Middleware, Resource Manager, Web Service

I. INTRODUCTION

Grid middleware are large software which provide a set of basic functionalities, each one implemented by a separate component. Such functionalities include data storage, authentication and authorization, resource monitoring, resource management and job management.[1]

One of the most challenging areas of Grid application development is the construction of portals to allow computational scientists, researchers and high performance computer/application users to access the resources via an easy to use web page interface. The aim is to develop common components that can be used by portal developers to allow them to build a web application that can securely authenticate users to remote resources and help them make better decisions for scheduling jobs by allowing them to view pertinent resource information.

In this paper, a new Grid Middleware (ITU-GRAM+) system that contains these components is designed and implemented using standard interfaces and protocols through new technologies which are open source to all people. The ITU-GRAM+ responds both data and computational intensive requirements of users. In order to meet these requirements in such environment, Service Oriented Architecture (SOA) principles have gained great importance. Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services [2]. For ITU-GRAM+, SOA is the main architecture to provide a communication between resources. Due to allow users to reach these resources, a web application is designed and developed. Getting job specific information by users and allow them to reserve and allocate the resources according to these requirements is main goal of this web application. But resources in grid system are not only belongs to system, they have to maintain their daily work issues. For this reason, to keep update information about resources in such system, Grid Information System is implemented.[3] The aim of this component to retrieve resource information such as *available cpu, available storage, available memory* which could be changed dynamically. The challenging question is that how the system informs the resources about releasing job after reservation of resource is completed. The Job Execution Component is developed to send job information to resources via web services which is exposed by resource. So far the components of ITU-GRAM+ are described, now we will have a look state of todays grid middleware.

Currently, the Globus Toolkit is the de facto standard for grid computing because of its wide acceptance and deployment worldwide, even though several alternatives do exist, like Legion and Condor[4] . Web services-based GT4 which is one of the latest deliveries of Globus team provides significant improvements over previous releases in terms of robustness, performance, usability, documentation, standards compliance and functionality. A set of service implementations provide useful infrastructure services. These services address such concerns as execution management (GRAM), data access and movement (GridFTP , RFT, OGSA-DAI), replica management (RLS , DRS), monitoring and discovery (Index,Trigger, WebMDS), credential management (MyProxy, Delegation ,SimpleCA), and instrument management (GTCP). Most are Java Web services but some are implemented in other languages and use other protocols. Grid Resource Allocation and Management (GRAM) service addresses these issues, providing a Web services interface for initiating, monitoring, and managing the execution of arbitrary computations on remote computers.[5] During ITU-GRAM+ design phase, all Globus components are investigated and analyzed. Most of Globus components have corresponding component in ITU-GRAM+ such as GRAM, GridFTP , Index that's why we called the with + suffix. It have additional components plus to GRAM.

Condor is the one of alternative resource management system which is designed to support high-throughput computations by discovering idle resources on a network and

allocating those resources to application tasks. Legion is the other alternative to existing resource management system which is a reflective, object-based operating system for the Grid. It offers the infrastructure for grid computing. Legion provides a framework for scheduling which can accommodate different placement strategies for different classes of applications.

Limitations of the existing Grid middleware which does not take into account the needs of everyday scientific and business users. Day-to-day computer users and small to medium sized organizations often do not use clusters, and thus for them setting up Grids using the existing middleware is complex [6]. Furthermore, Grid enabling their applications is nearly impossible, as they are not easily supported, and this poses a massive barrier to the pervasive adoption of Grid computing by these communities. Grid Middleware is also complex to setup and necessitates a steep learning curve. But ITU-GRAM+ has basic interface to use and it is not necessary to adapt your application for grid system. You can submit your jobs as it is.

The paper is organized as fallowing.Section2 will describe the reasons that motivate me to do this work.. Section 3 describes main functionalities of ITU-GRAM+ and details of the components are explained in Section 4. Last section includes future works and conclusion.

II. MOTIVATION

Many Grid Portals can provide a customizable interface allowing scientists and researchers to perform Grid operations such as remote submission of their own programs, staging input and output files, and querying resources and queues information. Some of them is widely used by most of people and contains all necessary components to support grid infrastructure. According to my best knowledge there is no such a grid middleware is implemented is only using open source software .From the perspective of developers , it is easy to create their own middleware which provides basic infrastructure to add new functionalities and improve the system themselves. From the perspective of day-to-day computer users, scientists and business users it is easy to start work on it without having computational knowledge. These are the key points of my work to motivate me.

III. ITU-GRAM+ OVERVIEW

ITU-GRAM+ main functionalities are Resource Discovery, Resource Allocation, Job Submission, Job Execution and Job Monitoring. In this section it will be given brief description of these components respectively.

Computational or data-intensive applications use the resources according to the requests from the user in order to achieve results quickly and efficiently. The resource management system must take into account not only computational resources such as the amount of available CPU, memory, data storage capacity but also starting time of job, the financial value of the resource and the efficiency of the resource in order to find these resources. Source information on the resource can change dynamically, so to make the right choices these information should be kept constantly up to date. Grid Information Service component is implemented on ITU-GRAM+ using web-service and batch job technologies in order to satisfy this update requirement of grid nature.

Job management component is used to submit, cancel and monitor jobs for execution on available resources. Users can submit their jobs using job description language (jdl) which is a high-level, user oriented language [7]. It has lot of information about content of job. It is based on XML structure which is a standard to exchange data between systems. After submitting jobs, it is possible to monitor job steps. The resource will update the grid resource manager on critical points of the process with unique id which is provided by Grid Resource Manager in the beginning of job submission.

Resource management is used to allocate suitable resources to jobs. As in the economy, if we think that there are unlimited demands and limited number of resources, scheduling of resources is the one of the challenge problem on Grid systems. In this work resources selection is done not only the physical properties of the resources but also the distances from each other is considered. At the same time this distance varies depending on the type of job is relatively. All of these components will be analyzed one by one in next section.

IV. ITU-GRAM+ ARCHITECTURE AND IMPLEMENTATION

The above components which are given a brief description will be more detailed in this section. In Figure1 describes the architecture of ITU-GRAM+. ITU-GRAM+ includes a web application which allow users to use a standard interfaces in order to submit their jobs. The application container is the Tomcat for both web services and web application. Apache Tomcat is an open source webserver and servlet container developed by the Apache Software Foundation [8] .Java is the language used to develop these components. Java is currently one of the most popular programming languages in use, particularly for client-server web applications [9]. MySQL officially, but also commonly the world's most used relational and open source database management system [10]. ITU-GRAM+ choose the MYSQL to store all the information about users, resources, jobs and reservations. Hibernate is one of the free software that facilitated the storage and retrieval of Java domain objects via Object/Relational Mapping [11]. Java Server Pages (JSP) is a technology that helps software developers to serve dynamically generated web pages based on HTML , XML, or other document types [12]. Apache Struts 2 is an elegant, extensible framework for creating enterprise-ready Java web applications [13]. Apache Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. [14]

Figure1 shows that all the relations between these technologies and which of them are used for which component of ITU-GRAM+. From the Grid Middleware point of view there are three different part of system. The Web Application which interacts with end user, the web service which is exposed to get update status of job and the batch jobs which are running at background to retrieve update resource information and submit jobs.

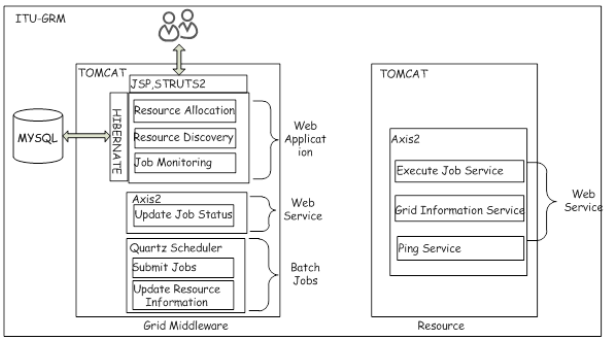


Figure 1. ITU-GRAM+ Architecture

From the resource point of view, only one web service is exposed includes three different methods inside.

Figure 2 shows a sample grid job life beginning from user submission to the completion of job in grid environment as an overview. The resources showed in the figure are small subset of in all resources. Client (user) gives the requirements inside job description language file (jdl) and user interface. Resource manager shows all available resource list and allows to user select one of them. System allocate resources during job execution time (Step1).A batch job waits for starting time of the job and invoke a web service to send job to computing resource (Step2). If the input file is stored in remote resource then it retrieves the file from File resources (Step3). After execution of job in computing resource, if output file is produced and if it is needed to keep in remote resources then the computing resource will send the file to storage resources.-(Step 4). The computing resource will inform resource manager for each step of job execution (Step5). Users could track their jobs via user interface (Step6).

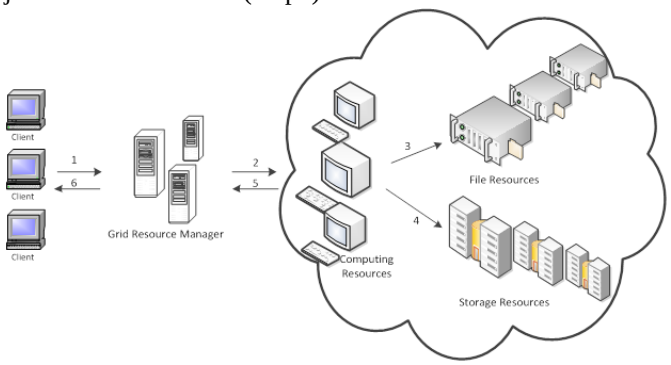


Figure 2.Grid Job Life

A. *Grid Information Service*

The resource information on Grid system can be changed dynamically because of grid nature. The nodes are not just a computational or data resource for Grid they also have their own work. So grid system should be aware of the changes on the resources. In order keep track these changes a web service is implemented which is running on the resources. The batch job on grid resource manager runs in a periodic time to retrieve the number of available processors, free memory and free disk space.

B. *Get User Requirements*

The users enter their requirements using both web user interface and job description language (jdl) file. The sample jdl file is showed in Figure 3. Some of them are clear to understand the meaning considering the tags but some of them are not. *gridType* could be *DataIntensive* or *Computing-Intensive* based on which resource type is needed more comparing to other types. *ioType* could be *Read* or *Write* based on input file size is greater or smaller than output file size relatively. Considering this comparison between output and input files, computing resource will be chosen to be closer to (as distances) file resource or storage resource in order to decrease the time during network transfer. *inputFileType* or *outputFileType* are the tags which are valuable to understand where input file will be found and where the output file will be kept. These tags values could be *Local*, *Remote* or *Expected*. *Local* means that input or output file exists on local system of user that submits jobs. *Remote* means that the user knows where input file could be found or output file could be kept and they are not on local. *Expected* means that user does not know where input file could be found and asks to grid system to find it. According to *gridType*, *ioType*, *inputFileType*, *output-FileType* values, some tags become mandatory. Based on combination of these tags the algorithm and resource types could differ.

As a result, user provides the details of the job in this sample xml file. Deadline, execution time of job, budget, execution file and other related information will be taken via user interface as shown in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<RequirementList>
    <gridType>DataIntensive</gridType>
    <ioType>Read</ioType>
    <inputFileType>Expected File</inputFileType>
    <inputFileName>earthquake.txt</inputFileName>
    <inputfilePath>D:/Data/Inputs</inputfilePath>
    <remoteMachine>192.168.1.1</remoteMachine>
    <isWrited>No</isWrited>
    <os>windows</os>
    <cpu>1</cpu>
    <memory>3</memory>
    <storage>200</storage>
    <outputFileType>Local</outputFileType>
    <outputFileName>output.txt</outputFileName>
    <outputfilePath>D:/Data/Outputs</outputfilePath>
    <remoteOutputMachine>192.168.1.1</remoteOutputMachine>
</RequirementList>
```

Figure 3.JDL File

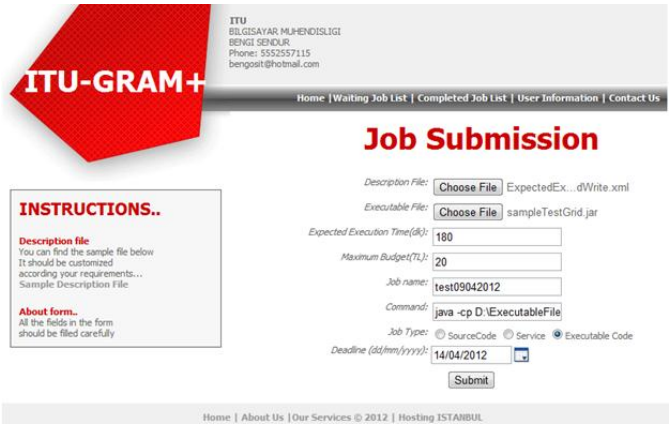


Figure 4.Webpage of job submission

C. List Available Resources

As soon as the users pass their requirements to web application, the system checks firstly what kind of resources (such as computational, file or storage) are required and which of them meet the physical requirements of job. Expected execution time of job is important to find available interval time of resources. Figure 4 shows the one of sample scenario which contains the all resource types defined in ITU-GRAM+. For each resource type the system finds time interval which is blank and add to list. Each minute in time table is represented with "1" and "0". For the example in Figure 5, the time table list is constructed as following.

Deadline-Job Submission Time=25 m

CR1={1,1,1,1,1,0,0,0,0,0,**0,0,0,0,0,**0,0,0,0,1,1,1,1,1,1,1}
SR1={0,0,1,1,1,1,1,1,0,0,0,**0,0,0,0,0**,0,1,1,1,1,1,1,1,1,1,1}
FR1={0,0,0,0,0,0,1,1,1,1,1,**0,0,0,0,0**,0,0,0,0,0,1,1,1,1,1,1}

List of ResourceList = { CR1SR1FR1, CR1SR2FR1,
CR1SR2FR2…. CRx SRy FRz }

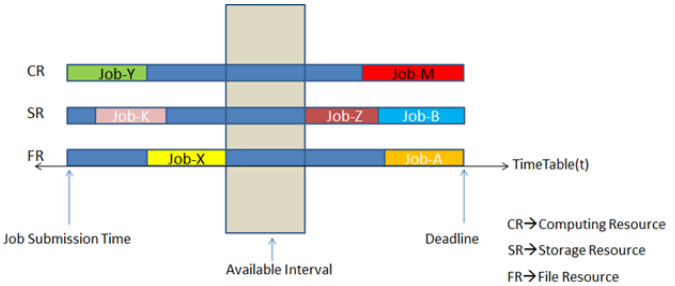


Figure5-List Available Intervals

But this list could be so much to show the user. So when the list is populated it will be filtered in terms of budget which is one of the input passing by user. Every resource has own cost which is calculated by the system based on their physical properties. A discount will be applied to calculated cost according to some conditions such as total load on the system during job execution or how long before the deadline the job is submitted or how many times the user submit a job to check loyalty of user. All these different situations cause a change on calculated cost. Despite of removing some triples from list based on budget there are still some triples are inefficient comparing the other triples. For example the input file of the job is larger than output file this means that CR should be closer to FR instead of SR in order to decrease time during transfer of the file through network.

If Distances(CR1-SR1) > Distances(CR1-SR2) then CR1SR1 should be removed from the list. The distances is not a physical distances between resources. As showed in Figure 1 every resources has *pingService* which takes the *ip* of other resource as an input in order to send a ping request and waiting for response. It calculates duration and send back to resource manager. It compares these results and filters triples which will not worth to select them. As a result, maximum 10 different options should be presented to the customer. A sample resource list is showed in figure 6.

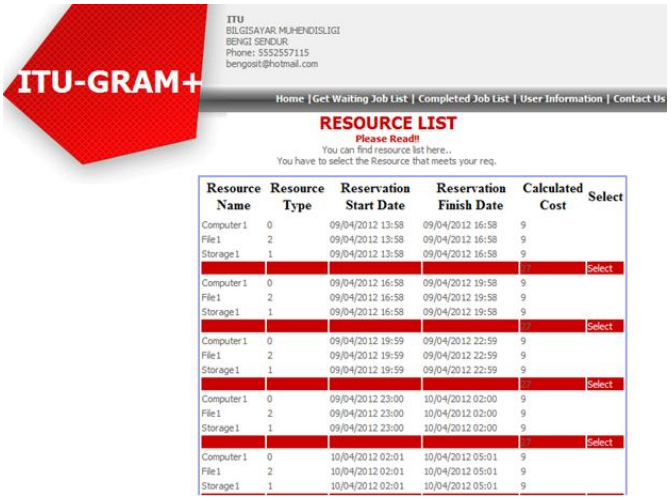


Figure 6-Available Resource List

D. Allocate the Resource

Best 10 resource triples at maximum are displayed to the user. The system asks users to choose only one of them. As showed in Figure 6, users can select resources for specified time interval. But the system actually does not reserve all resources in same period. If Expected Input File Size or Expected Output File Size is defined in jdl file, in order to increase the efficiency of resources, the system allocates the resources in different period. Execution time of job could be retrieved from job information but the user could not estimate the time during transferring the files through network. While allocating the resources, this information also should be considered. In figure 7, T and Z zones shows actual usage of resources. In available interval, there are some idle time for SR and FR so new jobs can be scheduled on this interval. Ratio of X to Z and T regions vary depending on the type of grid. If the grid is data intensive grid the large part of the work will be on SR or FR but if it is computational intensive grid then the large part of the work will be on CR. Since these calculations are done approximately Z and T zones should be considered as calculated with buffered time to prevent conflict with other jobs.

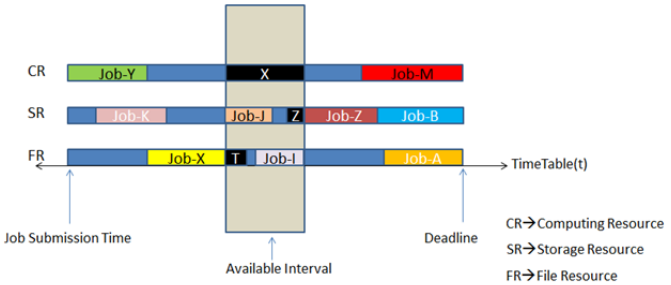


Figure7-Allocate Resources

E. Submit Jobs to Resources

This section describes how the system will submit jobs to resources. This component of ITU-GRAM+ works as a batch job in background to listen database for reservation jobs. Before submitting the job, execution file of the job should be transferred to CR. For this purpose, another batch job is

defined to check the database and retrieve jobs which will start immediately. This batch job works in a periodic time with multithread to send files using secure FTP protocol. Submitting jobs to resources is done by using web service technologies. All necessary parameters to execute job will be passed to resources via web service. In figure 8 shows interactions between resources and resource manager during execution of job. In first step job object of *sendJob* method includes parameters listed below. These parameters will be used by resource in different steps of job execution as showed following.

Job→ {resIdList, remoteInputIp, remoteInputPath, inputFileName, userId, jobName, command, remoteOutputIp, remoteOutputPath, outputFileName, executableFile, ioType, inputType, outputType, gridType}

After this submission, ITU-GRAM + waits update from the resources, after that time, control is on resources. If we go over the same example is described previous sections the steps on execution of job is showed in Figure 8. These states have been defined in the system in order to track jobs in every step. *UpdateJobStatus* is an another web service which is exposed by Grid Resource Manager to retrieve the status of job from resources. *resIdList* parameter is passed to resource in step1 while submitting the job so the resources have this information. Since the web service is a stateless protocol it is important for grid system to restore the parameter on database and when any update comes from resources it matches the job with this *resIdList* parameter.

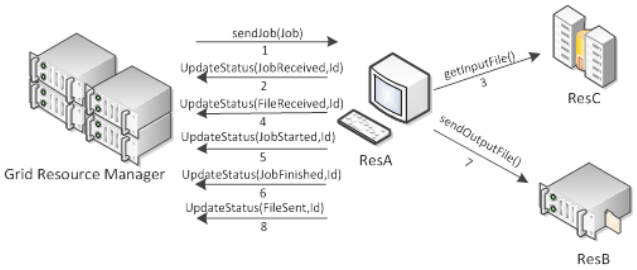


Figure 8-Submit Job

F. Monitoring Job Status

This section describes how to user can track their jobs on ITU-GRAM+. For this purpose a new web page is implemented to show the jobs status to users. ITU-GRAM+ has capability to allow users to cancel their jobs before starting job execution. But in this case penalty will be charged on users in order to keep stable the performance of the system. Figure 9 shows the all states are described in ITU-GRAM+ and which transactions are possible between these states. These states of job are stored on the database table which is called "ReservationJob". This table keeps the information which resources will be matched which jobs in which time period. All possible transition between these states are showed with letters below.

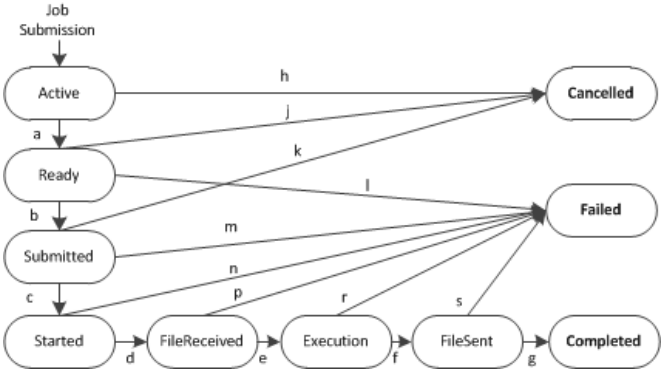


Figure9-State Diagram of Job

a→The status is changed *Active* to *Ready* when execution file transferred to CR.

b→The status is changed from *Ready* to *Submitted* when web service of *sendJob* is invoked.

c→The status is changed from *Submitted* to *Started* when resource received job information.

d→ The status is changed from *Started* to *FileReceived* when input file is retrieved.

e→ The status is changed from *FileReceived* to *Execution* when resource started to run job.

f→ The status is changed from *Execution* to *FileSent* when output file is sent.

g→ The status is changed from *FileSent* to *Completed* when job finished successfully.

h,j,k→The status is changed from *Active, Ready or Submitted to Cancelled* when the users want to cancel job.

l,m,n,p,r,s→ The status is changed from *Ready, Submitted, Started , FileReceived*, *Execution , FileSent to Failed* when any error occurs during job execution

Resource states are also kept in ITU-GRAM+ as showed in Figure10. ITU-GRAM+ works with reservation method so these states are valuable for the system in order to prevent conflicts.

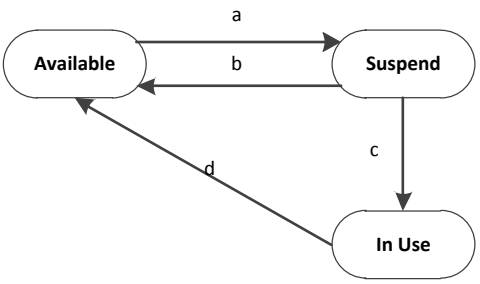


Figure 10. State Diagram of Resource

a→ The status is changed from *Available* to *Suspend* when resources are displayed to user.

b→ The status is changed from *Suspend* to *Available* when user releases or timeout is reached.

c→ The status is changed from *Suspend* to *In Use* when user selects resources that meets requirements

d→ The status is changed from *In Use* to *Available* when job is completed.

To display completed and waiting jobs to user, specific web pages are implemented. It is also possible to cancel or monitor jobs using these pages.

V. CONCLUSION AND FUTURE WORK

ITU-GRAM+ is a grid middleware which allows to users lot of capabilities to execute their jobs. The important feature of the system is that it is implemented using all open source software. In addition to this the other functionalities on the components are valuable for future work such as allocation resources considering not only availability of all resources but also the resource utilization. Besides using web service technology to ensure communication between resources and middleware is added value for Grid system. ITU-GRAM+ is not only support data or computational grids it works with fine both of them. As a result ITU-GRAM+ helps developers to create their own middleware based on open source software and provide standard and basic interfaces to users to submit their jobs. Behind of these functionalities basic user access protocols has been developed but other security related issues are kept out of the scope of this paper. In addition to this recover mechanism for failure states is handled with basic implementation. Since the ITU-GRAM+ is free software it is open any improvement for future works.

REFERENCES

[1] Job submission and management through web services: the experience with the CREAM service C Aiftimiei, P Andreetto, S Bertocco, D Cesini, M Corvo,S Dalla Fina, S Da Ronco, D Dongiovanni, A Dorigo, A Gianelle,C Grandi, M Marzolla, M Mazzucato, V Miccio, A Sciaba',M Sgaravatto, M Verlato and L Zangrando, 2008

[2] Service-Oriented Architecture: Concepts, Technology, and Design, Thomas Erl Prentice Hall PTR Upper Saddle River, NJ, USA ©2005 ISBN:013185858

[3] Ten actions when Grid scheduling: the user as a Grid scheduler, Kluwer AcademicPublishers Norwell,MA,USA ©2004 table of contents ISBN:1 -4020-7575-8

[4] A Taxonomy and Survey of Grid Resource Management Systems, Chaitanya Kandagatla,2003

[5] Globus Toolkit Version 4:Software for Service-Oriented Systems,Ian Foster,2006

[6] Job Submission Description Language (JSDL) Specification, Version1.0, 2005 http://www.gridforum.org/documents/GFD.56.pdf

[7] From Grid Middleware to a Grid Operating System, Arshad Ali, Richard McClatchey, Ashiq Anjum, Irfan Habib, Kamran Soomro, Mohammed Asif, Ali Adil, Athar Mohsin ,2006

[8] Available on 2012, http://tomcat.apache.org/

[9] Available on 2012, http://www.oracle.com/technetwork/java/

[10] Available on 2012, http://www.mysql.com/

[11] Available on 2012, http://www.hibernate.org/

[12] Available on 2012, http://www.oracle.com/technetwork/java/javaee/jsp/index.html

[13] Available on 2012, http://struts.apache.org/2.x/

[14] Available on 2012, http://axis.apache.org/axis2/java/core/

SESSION

COMPUTATIONAL GRID AND SCHEDULING

Chair(s)

Prof. Hamid Arabnia

Survey of Real Time Divisible Load Scheduling Algorithms in computational grid

Mohamed Youssri A. El Nahas [1], Nahed M. El Desouky [2], Sahar A. Gomaa[2]and Naglaa Mostafa [2]
[1]Faculty of Engineering , Al-Azhar University(girls) Cairo, Egypt.
[2]Departement of Mathematics, Computer Branch, Faculty of Science ,
Al-Azhar University(girls), Cairo, Egypt

Abstract - *Cluster computing has emerged as a new paradigm for solving large-scale problems. These workloads represent a broad variety of real-world applications in cluster and grid computing, such as BLAST (Basic Local Alignment Search Tool) [2], a bioinformatics application, and high energy and particle physics applications in ATLAS (A Toroidal LHC Apparatus) [6] and CMS (Compact Muon Solenoid) [10] projects. To provide performance guarantees in cluster computing environments, various real-time scheduling algorithms and workload models have been investigated. Computational loads that can be arbitrarily divided into independent pieces represent many real-world applications. Divisible load theory (DLT) provides insight into distribution strategies for such computations. However, the problem of providing performance guarantees to divisible load applications has not yet been systematically studied. This paper provides a survey and compares different algorithms for a cluster environment that provide a solution for the real time divisible load applications. And provide different parameters that affect the performance of these algorithms and scenarios when the choice of these parameters has significant effects are studied. It provides a taxonomy of the different scheduling methods, and considers the various performance metrics that can be used for comparison purposes*.

Keywords: grid computing, divisible load theory.

1 Introduction

Real-time Divisible Load Theory (RT-DLT) holds great promise for modeling an emergent class of massively parallel real-time workloads. However, the theory needs strong formal foundations before it can be widely used for the design and analysis of hard real-time safety-critical applications. In[1] the general problem of obtaining such formal foundations, by generalizing and extending recent results and concepts from multiprocessor real-time scheduling theory.

Cluster computing has become an important paradigm [2] for solving large-scale problems. However, as the size of a cluster increases, so does the complexity of resource management and maintenance. Therefore, automated performance control and resource management are expected to play critical roles in sustaining the evolution of cluster computing.

Real-time divisible load scheduling is a well researched area [3, 4, 5, 6, 7, 8]. Focusing on satisfying QoS (Quality of services), providing real-time guarantees, and better utilizing cluster resources, existing approaches give little emphasis to scheduling efficiency. They assume that scheduling takes much less time than the execution of a task, and thus the scheduling overhead is ignored. When the processors in a cluster platform all become available at the same instant in time, the issue of scheduling a real-time divisible workload on such platforms is pretty well understood. However, the reality in many multiprocessor environments is that all the processors do not become available to a given workload at the same instant (perhaps because some of the processors are also being used for other purposes).

Broadly speaking, computational loads submitted to a cluster are structured in two primary ways: *indivisible* and *divisible*. An indivisible load is essentially a sequential job and thus must be assigned to a single processor. The divisible loads are comprised of tasks that can be executed in parallel and can be further divided into two categories: *modularly divisible* and *arbitrarily divisible* loads. *Modularly divisible* loads are divided a priori into a certain number of subtasks and are often described by a task (or processing) graph. *Arbitrarily divisible* loads can be partitioned into an arbitrarily large number of load fractions. Examples of arbitrarily divisible loads can be easily found in high-energy and particle physics as well as bio-metrics. For example, the CMS (Compact Muon Solenoid) [18] and ATLAS (A Toroidal LHC Apparatus) [19] projects, which are asso-ciated with the LHC (Large Hadron Collider) at CERN (European Laboratory for Particle Physics), execute cluster-based applica-tions with arbitrarily divisible loads.

There are three important decisions for an algorithm of real-time scheduling to schedule divisible loads. The first is to adopt a scheduling policy to determine the order of execution for tasks. The second decision is to determine the number of processing nodes (n) to allocate to each task and the third is to choose a strategy to partition the task among the allocated n nodes.

Description of the distributed system and assumption used by different algorithms of real time divisible load are discussed in section 2. Section 3 studies different scheduling policies that classify the algorithms used in real time divisible load. Different algorithms that are used to decide the number of nodes must be assigned to each task and method of portioning the task itself is discussed in section 4. The metrics used to measure the real time performance of different scheduling algorithms are explained in section 5 .Conclusion is given in section 6**.**

1.1 Instructions for authors

An electronic copy of your *full camera-ready paper* must be uploaded (in PDF format) to Publication Web site before the announced deadline. Please follow the submission instructions shown on the web site. The URL to the website is included in the notification of acceptance that has been emailed to you by Prof. Arabnia.

2 Tasks and System Assumption

The model of the investigated system used by different algorithms and the assumptions are as follow:

System model: a cluster consists of a head node, denoted by p_0, and connected via a switch to *n* processing nodes, denoted by $p_1, p_2, p_3, ...p_n$. And assuming that all processing nodes have the same computational power and all links from the switch to the processing nodes have the same bandwidth. The system model assumes a typical cluster environment in which the head node does not participate in computation. The role of the head node is to accept or reject incoming tasks, execute the scheduling algorithm, divide the workload and distribute data chunks to processing nodes. Since different nodes process different data chunks, the head node sequentially sends every data chunk to its corresponding processing node via the switch. And data transmission does not happen in parallel, although it is straightforward to generalize this model and include the case where some pipelining of communication may occur. There is an assumption for the divisible loads that is task and subtasks are independent. Therefore, there is no need for processing nodes to communicate with each other.

According to divisible load theory, linear models are used to represent processing and transmission times [10]. In the simplest scenario, the computation time of a load σ is calculated by a cost function $cp\ (\sigma) = \sigma c_{ps}$, where c_{ps} represents the time to compute a unit of workload on a single processing node. The transmission time of a load σ is calculated by a cost function $cm(\sigma) = \sigma c_{ms}$, where c_{ms} is the time to transmit a unit of workload from the head node to a processing node.

Tasks Assumption: Assuming that a real time aperiodic task model in which each a periodic task *Ti* consists of a single invocation specified by the tuple (A_i, σ_i, D_i), where $A_i \geq 0$ is the arrival time of the task, $\sigma_i > 0$ is the total data size of the task, and $D_i > 0$ is its relative deadline. The absolute deadline of the task is given by $A_i + D_i$ **.** Task execution time is dynamically computed based on total data size σ_i , resources allocated (i.e., processing nodes and bandwidth) and the partitioning method applied to parallelize the computation.

3 Scheduling Policies

There are three scheduling policies to determine the execution order of tasks that are investigated in different algorithms: FIFO, EDF (earliest deadline first) and MWF (Maximum Workload derivative First)[1,2,9,10,12,20]. The FIFO scheduling algorithm executes tasks following their order of arrival. EDF, a well-known real-time scheduling algorithm, orders tasks by their absolute deadlines. MWF is a real-time scheduling algorithm for divisible tasks. The main rules of MWF are:

1) A task with the highest workload derivative (DC_i) is scheduled first.

2) The number of nodes allocated to a task is kept as small as possible ($n_i^{\min}$) without violating its deadline. Here, we review how MWF determines task execution order and define the workload derivative metric, DC_i, where $W_i(n)$ is used to represent the workload (cost) of a task T_i when n processing nodes are assigned to it.

$$DC_i = W_i\left(n_i^{\min}+1\right) - W_i\left(n_i^{\min}\right) \qquad (1)$$

That is, $W_i(n) = n \times \xi(\sigma_i, n)$, where $\xi(\sigma_i, n)$ denotes the task's execution time. Therefore, DC_i is the derivative of the task workload $W_i(n)$ at $n_i^{\min}$ (the minimum number of nodes needed by *Ti* to meet its deadline).

4 Real time divisible algorithms

Different algorithms that investigate the real time divisible load are studied during the past few years. OPR which searches optimally to find the minimum number of processing nodes that satisfies the real time requirement of the load that is proposed in [1, 2, 9, 10, 20]. Another way to solve this problem follows the idea of equally portioning the load among the processing nodes that presented in [1, 2, 9, 10]. Task Waiting Queue (TWQ) algorithms are divisible load

scheduling algorithms [3, 4, 5, 6, 7], that perform the schedule ability test. The admission controller generates a new schedule for the newly arrived task and all tasks waiting in TWQ, this decision module is referred to as the admission controller. When processing nodes become available, the dispatcher module partitions each task and dispatches subtasks to execute on processing nodes. And finally there are algorithms depend on the time which processing nodes are available at it. There are two algorithms of this idea first when processing nodes have equal ready and when processing nodes have different ready time, these algorithms presented in [7,14, 15]. In the following subsections a brief details of these algorithms are presented.

4.1 Optimal Partitioning Rule (OPR) Algorithm

Divisible load theory states that optimal execution time is obtained for a divisible load if all processing nodes allocated to the task complete their computation at the same time instant [11]. This is called the *Optimal Partitioning Rule* (OPR). In divisible load theory, normally all *n* nodes of a cluster are allocated to a task. Then, following the OPR, the task load is partitioned such that all nodes finish processing at the same time. In contrast to this approach, first computing the minimum number of processing nodes needed to meet the task's deadline given its schedule, and then partition the task following the OPR (using at least the minimum number of nodes required to meet the deadline). The execution time of a task is then trivially computed as the difference between its completion and start times. The following notations, partially adopted from [11], are used in these computations.

- $T=(A,\sigma,D)$: A divisible task, where *A* is the arrival time, σ is the data size, and *D* is the relative deadline.
- $\alpha=(\alpha_1,\alpha_2,.....,\alpha_n)$: Data distribution vector, where *n* is the number of processing nodes allocated to the task, α_j is the data fraction allocated to the j^{th} node, i.e., $\alpha_j\sigma$, is the amount of data that is to be transmitted to the j^{th} node for processing, $0\prec\alpha_j\leq 1$ and $\sum_{j=1}^{n}\alpha_j=1$.
- c_{ms} : Cost of transmitting a unit workload.
- c_{ps} : Cost of processing a unit workload.

The following cost functions describe that: the data transmission time on the j^{th} link is $c_m(\alpha_j\sigma)=\alpha_j\sigma c_{ms}$ and the data processing time on the j^{th} node is $c_p(\alpha_j\sigma)=\alpha_j\sigma c_{ps}$.

Figure 1 shows an example task execution time diagram following OPR when *n* nodes are allocated to process the task. Let ξ denote *Task Execution Time*, which is a function of σ and *n*.

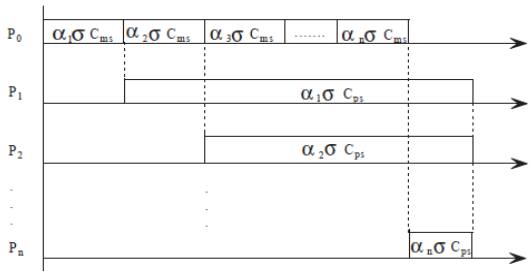


Figure 1: Time Diagram for OPR-Based Partitioning**.**

By analyzing the diagram, we have

$$\xi(\sigma,n)=\alpha_1\sigma c_{ms}+\alpha_1\sigma c_{ps} \quad (2)$$

$$=(\alpha_1+\alpha_2)\sigma c_{ms}+\alpha_2\sigma c_{ps} \quad (3)$$

$$=(\alpha_1+\alpha_2+\alpha_3)\sigma c_{ms}+\alpha_3\sigma c_{ps} \quad (4)$$

$$\cdots$$

$$=(\alpha_1+\alpha_2+\alpha_3+.....+\alpha_n)\sigma c_{ms}+\alpha_n\sigma c_{ps}\ . \quad (5)$$

To specify the minimum number $n^{\min}$ of nodes that required to meet a task's deadline**,** **a**ssuming that the task $T(A,\sigma,D)$ has a start time s, then the task completion time is C (n) = s + $\xi(\sigma,n)$, which must satisfy the constraint that C(n) ≤ A + D. Lin et al. [14, 15, 16] derived the task execution time function $\xi(\sigma,n)$ and the minimum number $n^{\min}$ of nodes that the task needs at time *s* to meet its deadline are

$$\xi(\sigma,n)=\frac{1-\beta}{1-\beta^n}\sigma(c_{ms}+c_{ps}) \quad (6)$$

$$n^{\min}=\left\lceil\frac{\ln\gamma}{\ln\beta}\right\rceil \quad (7)$$

where $\beta=\dfrac{c_{ps}}{c_{ms}+c_{ps}}$ and $\gamma=1-\dfrac{\sigma c_{ms}}{A+D-s}$.

4.2 Equal Partitioning Rule (EPR) Algorithm

Equal Partitioning Rule (EPR) is based on a common practice of dividing a task into *n* equal-sized subtasks when the task is to be processed by *n* nodes. An example task execution time diagram following the EPR is shown in figure 2. By analyzing the diagram, we have

$$\xi(\sigma,n)=\sigma c_{ms}+\frac{\sigma c_{ps}}{n}\ . \quad (8)$$

Similar to the analysis for DLT-based OPR, Lin et al. [1, 2 ,9, 10] derived the minimum number $n^{\min}$ for EPR. The minimum number of processing nodes that the task needs at time s to complete before its deadline is

$$n^{\min} = \left\lceil \frac{\sigma c_{ps}}{A + D - s - \sigma c_{ms}} \right\rceil$$

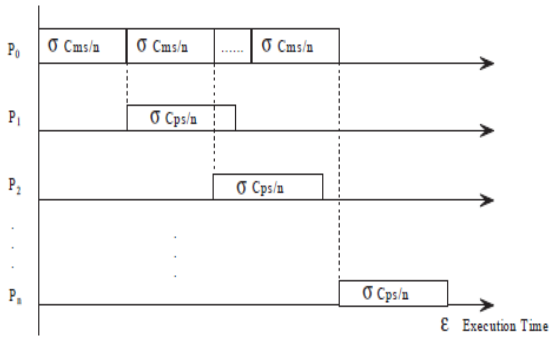


Figure 2: Time Diagram for EPR-Based Partitioning.

4.3 Task waiting queue (TWQ) algorithms

In these algorithms [10] *Mamat, Ying Lu, Jitender Deogun and Steve Goddard*, when a task arrives, the scheduler determines if it is feasible to schedule the new task without compromising the guarantees for previously admitted tasks. Only those tasks that pass this schedule ability test are allowed to enter the *task waiting queue* (TWQ). This decision module is referred to as the *admission controller*. When processing nodes become available, the *dispatcher* module partitions each task and dispatches subtasks to execute on processing nodes. In the Modules, admission controller and dispatcher, run on the head node. For existing divisible load scheduling algorithms [3, 4, 5, 6, 7], in order to perform the schedule ability test, the admission controller generates a new schedule for the newly arrived task and all tasks waiting in TWQ. If the schedule is feasible, the new task is accepted; otherwise, it is rejected. For these algorithms, the dispatcher acts as an execution agent, which simply implements the feasible schedule developed by the admission controller.

There are two factors that contribute to large overheads of these algorithms. First, to make an admission control decision, they reschedule tasks in TWQ. Second, they calculate in the admission controller the minimum number $n^{\min}$ of nodes required to meet a task's deadline so that it guarantees enough resources for each task. The later a task starts, the more nodes are needed to complete it before its deadline. Therefore, if a task is rescheduled to start at a different time, the $n^{\min}$ of the task may change and needs to be recomputed. This process of rescheduling and re-computing $n^{\min}$ of waiting tasks introduces a big overhead.

The dispatching algorithm [10] is rather straightforward. When a processing node and the head node become available, the dispatcher takes the first task τ (A, σ, D) in TWQ, partitions the task and sends a subtask of size $\hat{\sigma}$ to the node, where $\hat{\sigma} = \min\left(\frac{A + D - CurrentTime}{c_{ms} + c_{ps}}, \sigma\right)$. The remaining portion of the task τ (A, σ-$\hat{\sigma}$,D) is left in TWQ. The dispatcher chooses a proper size $\hat{\sigma}$ to guarantee that the dispatched subtask completes no later than the task's absolute deadline $A + D$. Following the algorithm, all subtasks of a given task complete at the task absolute deadline, except for the last one, which may not be big enough to occupy the node until the task deadline. By dispatching the task as soon as the resources become available and letting the task occupy the node until the task deadline, the dispatcher allocates the minimum number of nodes to each task.

4.4 Case of Processor Ready Times

These algorithms can solve the real time divisible load by depending on the time which processing nodes are ready at it. This approach contains two kinds algorithms, algorithms when the processing nodes are equal ready time and algorithms when different ready time. The following subsections describe briefly the ideas of these algorithms.

4.4.1 Processors with Equal Ready Times

In [14, 15, 18], it is assumed that all the processors, upon which a particular job will be distributed by the head node, are available for that job over the entire time-interval between the instant that the head-node initiates data transfer to any one of these nodes, and the instant that it completes execution upon all the nodes. Under this model of processor availability, it is known that the completion time of a job on a given set of processing nodes is minimized if all the processing nodes complete their execution of the job at the same instant. This makes intuitive sense – if some processing node completes before the others for a given distribution of the job's workload, then a different distribution of the workload that transfers some of the assigned work from the remaining processing node to this one would have an earlier completion time. Figure 6 depicts the data transmission and execution time diagram when processors have equal ready times.

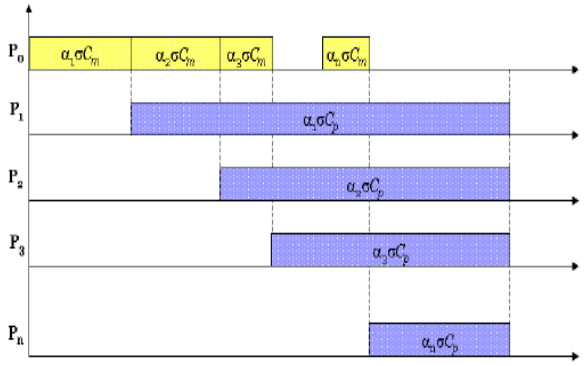


Figure 6: Data transmission and execution time diagram when processor have equal ready times

For a given job (A, σ, D) and a given number of processing nodes *n*, let $\sigma_i \times \alpha$ denote the amount of the load of the job that is assigned to the j^{th} processing node, $1 \leq j \leq n$. Since data-transmission occurs sequentially, the node $_i P$ can only receive data after the previous $(i-1)$ nodes have completed receiving their data. Hence, each p_i receives its data over the interval $\left[c_m \sum_{j=1}^{i-1} \alpha_j \sigma, c_m \sum_{j=1}^{i} \alpha_i \sigma \right)$

And therefore completes execution at time-instant $c_m \sum_{j=1}^{i} \alpha_j \sigma + c_p \alpha_i \sigma$ Then time execution time is assignment by equation (6) and to determine a minimum number of processors needed is computed from equation (6) by setting this completion time to the job's deadline (A+ D) in Equation (6), and making "*n*" — the number of processors — the variable. (Since the number of processors is necessary integral, it is actually the ceiling of this value that is the minimum number of processors.)

4.4.2 Processors with Different Ready Times Algorithm

In [16, 17], Lin et al. allow for the possibility that all the processors are not immediately available. To determine the completion time of a job upon a given number of processors in this more general setting, Lin et al.[16, 17, 21] adopt a *heuristic* approach that aims to partition a job so that the allocated processors could start at different times but finish computation (almost) simultaneously.

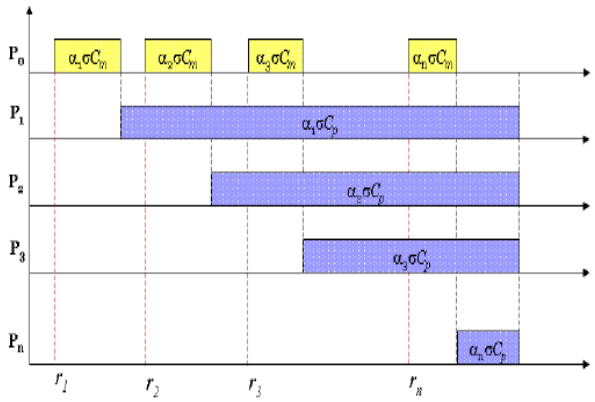
To achieve this, they first map the given homogenous cluster with different processor available times $r_1, r_2, ...r_n$ (with $\forall r_i \leq r_{i+1}$) into a *heterogeneous* model where all *n* assigned nodes become available simultaneously at the time-instant r_n, but different processors may have different computing capacities. Intuitively speaking, the i^{th} processor has its computing capacity inflated to account for the reality that it is able to execute over the interval $[r_i, r_n)$ as well. Figure 7 depicts the data transmission and execution time diagram when processors have different ready times.

Figure 7: Data transmission and execution time diagram when processors have different ready times

In Lin et al [16, 17] , this heterogeneity is modeled by associating a different constant c_{ps_i} with each processor p_i, with the interpretation that it takes c_{ps_i} time to complete one unit of work on the processor p_i . The formula for determining c_{ps_i} , as given in Lin et al [16, 17] , is

$$c_{ps_i} = \frac{\xi(\sigma, n)}{\xi(\sigma, n) + r_n - r_i} \qquad (11)$$

Where $\xi(\sigma, n)$ denotes the completion time if all processors are immediately available in the original



(homogenous) cluster these c_{ps_i} values are used to derive formulas for computing the fractions of the workload that are to be allocated to each heterogeneous processor such that all processors complete at approximately the same time, and for computing this completion-time.

4.5 Least Cost Methods

G.K.Kamalam and Dr.V.Murali Bhaskaran [25] introduce a decentralized job scheduling algorithms which performs intra cluster and inter cluster (grid) job scheduling. They apply Divisible Load Theory (DLT) and Least Cost Method (LCM) to model the grid scheduling problem involving multiple resources within an intra cluster and inter cluster grid environment. The LCM method, the jobs are allocated to the resource with the least allocation cost [26]. The algorithm reduces the total processing time and the total cost and the resource utilization is more and the load is balanced across the grid environment.

Xin Liu et al [23,24] proposed another algorithm in which they tried to obtain minimum cost by perturbing the schedule of some tasks from minimum time solution. They proposed min-time algorithm to find the minimum completion time and the min-cost algorithm to find the minimum cost without considering the deadline constraint. Their proposed algorithm is a hybrid scheduling algorithm to minimize some of the tasks lying to the first of the list follow min time and the remaining tasks in the list follow min cost algorithm. This is called as perturbation degree. Their proposed algorithm stated that the task from the list is allowed to evaluate the minimum completion time and if it is greater than the deadline, then there is no possibility of getting feasible solution, if the minimum completion time is less than the deadline then binary search is used recursively for largest perturbation degree, such that the current or the next perturbation degree is smaller than the deadline. Now the cost and perturbation degree is obtained and returned as schedule with minimum cost and finished before deadline.

5 Metrics of real time divisible load for cluster scheduling

To measure the performance and distinguish between algorithms, different metrics are used. These metrics are used to measure the effects of parameters on these algorithms and it also merits between them.

The DC Ratio, task reject ratio, processing speed and number of nodes are the main performance metrics used by OPR, EPR and ready time processor algorithms. While task reject ratio, system utilization and scheduling overhead are used by TQW algorithms to measure their performance. The following subsections give brief descriptions of these metrics.

5.1 Effect of Task Reject Ratio

There is new metric *Task Reject Ratio* can use it to specify the real-time scheduling algorithm is better or not , which is define as the ratio of the number of tasks rejected by a real-time scheduling algorithm to the total number of tasks arriving at the cluster The smaller the *Task Reject Ratio*, the better the real-time scheduling algorithm. The *Task Reject Ratio* of the four algorithms: EDF-OPR-MN, EDF-EPR-MN, EDF-OPR-AN, and EDF-EPR-AN. Observe that EDF-OPR-MN always leads to a lower *Task Reject Ratio* than EDF-EPR-MN. Similarly, observe that EDF-OPR-AN always achieves a lower *Task Reject Ratio* than EDF-EPR-AN. These simulation results confirm the hypothesis [10] that it is advantageous to apply DLT in real-time, cluster-based scheduling algorithms. DLT provides an optimal task partitioning, which leads to minimum task execution times, and as a result the cluster can satisfy a larger number of task deadlines.

5.2 Effects of DCRatio

[1,2] There are another metric that effect on the real time algorithms that is DCRatio which is defined as the ratio of mean deadline to mean minimum execution time (cost), that is $\frac{\text{AvgD}}{\xi(\text{Avg}\sigma,\text{N})}$, where $\xi(\text{Avg}\sigma,\text{N})$ is the task execution time computed with Eq (6) assuming the task has an average data size $\text{Avg}\sigma$ and runs on all N processing nodes.

To study the effects of the DCRatio, on the real time algorithms of divisible load, observe that by increasing DCRatio, the performance of EDF-EPR-AN becomes closer to that of EDF-OPR-AN. This is because the higher the DCRatio, the looser the task relative deadlines are. Consequently, the worse execution times caused by a non-optimal partition, like EPR, will have less impact on the algorithms' performance. In particular, when DCRatio is extremely high (100), the two algorithms perform almost the same.

5.3 Effects of Processing Speed

By studying effects of processing speed, the algorithm with OPR [9,10] partitioning (EDF-OPR-MN) still outperforms the algorithm with EPR partitioning (EDF-EPR-MN). However, as the processing speed decreases, i.e., c_{ps} increases, the difference between the two algorithms becomes less and less significant. In particular, when the computation is extremely slow (c_{ps} = 10000), the curves for the two algorithms are almost overlapped, indicating non-differentiable *Task Reject Ratios*. Therefore, OPR and EPR will perform the same in this case. From the aforementioned intensive experiments, then the conclusion is no matter what the system parameters are, the algorithms with DLT-based partitioning (OPR) always perform better than the ones with the equal-sized partitioning heuristic (EPR). This shows that it is beneficial to apply divisible load theory in real-time, cluster-based scheduling.

5.4 All nodes N versus $n^{\min}$ Nodes

The performance of real time divisible load algorithm [9,10,20] difference in algorithms assigning all N nodes to every task (ALG-AN) v.s. those assigning the minimum number $n^{\min}$ of nodes needed to meet a task's deadline (ALG-MN). Where the relative performance of EDF-OPR-MN v.s. EDF-OPR-AN is noteworthy that in contrast to the results in [12] comparing MWF (-MN) and FIXED (-AN) algorithms, the initial data seem to indicate that EDF-OPR-AN outperforms EDF-OPR-MN most of the time. To gain insight into the performance results, Carry out rigorous analysis of a simplified scenario where a scheduling algorithm always assigns K nodes (K < N) to a periodic divisible task. This analysis sheds new light on possible scenarios where algorithms assigning $n^{\min}$ nodes (ALG-MN) perform better than those assigning all N nodes (ALG-AN).

5.5 Scheduling Overhead and cost

This metrics investigate the effect of scheduling overheads and deadline constrain. Theses algorithms try to minimize the overhead affected by the scheduling algorithms and meet the deadline constrain.

6. Conclusion

In this paper, the real-time divisible load distribution problem in computational grid is investigated. We try to present the progress and developing efforts to determine the best mechanisms, policies and analysis to use in these systems. Different matrices and constrains can be compromised by building systems using approaches that lack the necessary theoretical underpinnings. Ultimately, computational grid will be used in high integrity real-time systems, and consequently, timing failures could affect safety. The paper study different scheduling algorithms; scheduling policies, and hybrid algorithms. Comparisons of fewer algorithms with various factors influencing Grid system are explained. An investigation on various factors that influence the scheduling in grid has been made and shown in this paper. This is an effort made to find the silver lining in the dark clouds which could paint an idea about the scheduling policies applied to the real-time divisible load problem in computational environment.

References

[1] Suriayati bt chuprat," The deadline-based scheduling of Divisible Real-Time Workloads on Multiprocessor

Platforms", PHD degree of Doctor of Philosophy (Mathematics), Faculty of science, university technology Malaysia 2009.

[2] Computer Science and Engineering, Department of Computer Science and Engineering: "Real-Time Divisible Load Scheduling For Cluster Computing", Theses Dissertations, and Student Research University of Nebraska - Lincoln Year 2011.

[3] S. Chuprat and S. Baruah. " Scheduling divisible real-time loads on clusters with varying processor start times" In 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '08),pages 15–24, Aug 2008.

[4] S. Chuprat, S. Salleh, and S. Baruah. "Evaluation of a linear programming approach towards scheduling divisible real-time Loads", In International Symposium on Information Technology, pages 1–8, Aug 2008.

[5] W. Y. Lee, S. J. Hong, and J. Kim., "On-line scheduling of scalable real-time tasks on multiprocessor systems" , Journal of Parallel and Distributed Computing 63(12):1315–1324, 2003.

[6] X. Lin, Y. Lu, J. Deogun, and S. Goddard, "Real-time divisible load scheduling with different processor available times", In Proceedings of the 2007 International Conference on Parallel Processing (ICPP 2007).

[7] X. Lin, Y. Lu, J. Deogun, and S. Goddard.," Real-time divisible load scheduling for cluster computing" In Proceedings of the 13th IEEE Real-Time and Embedded Technology and Application Symposium pages 303–314, Bellevue, WA, April 2007.

[8] A. Mamat, Y. Lu, J. Deogun, and S. Goddard,"Real-time divisible load scheduling with advance reservations", In 20th Euromicro Conference on Real-Time Systems pages 37–46, July 2008.

[9] Anwar Mamat, Ying Lu, Jitender Deogun, Steve Goddard, "An Efficient Algorithm for Real-Time Divisible Load Scheduling", Department of Computer Science and Engineering University of Nebraska – Lincoln Lincoln NE 68588 {anwar, ylu, deogun, goddard}@cse.unl.edu

[10] B. Veeravalli, D. Ghose, and T. G. Robertazzi., "Divisible load theory: A new paradigm for load scheduling in distributed systems Cluster Computing", 6(1):7–17, 2003.

[11] D. Swanson. Personal communication. Director, UNL Research Computing Facility (RCF) and UNL CMS Tier-2 Site, August 2005.

[12] D. Isovic and G. Fohler. "Efficient scheduling of sporadic periodic, and periodic tasks with complex constraints", In Proc. of 21st IEEE Real-Time Systems Symposium,pages 207–216, Orlando, FL, November 2000.

[13] B. Veeravalli, D. Ghose, and T. G. Robertazzi, "Divisible load theory: A new paradigm for load scheduling in distributed systems Cluster Computing", 6(1):7–17, 2003.

[14] Lin, X., Lu, Y., Deogun, J. and Goddard, S., " Real-time divisible load scheduling for cluster computing", Technical Report UNL-CSE-2006-0016 (2006a), Department of Computer Science and Engineering, The University of Nebraska at Lincoln.

[15] Lin, X., Lu, Y., Deogun, J. and Goddard, S., "Real-time divisible load scheduling for clusters. Proceedings of the Real-Time Systems Symposium",(2006b).– Work-In-Progress Session pages 9–12, Rio de Janerio, December.

[16] Lin, X., Lu, Y., Deogun, J. and Goddard, S., " Real-Time Divisible Load Scheduling with Different Processor Available Times", Proceedings of International Conference on Parallel Processing (ICPP), (2007b), Xian, China, September.

[17] Lin, X., Lu, Y., Deogun, J. and Goddard, S., " Enhanced Real-Time Divisible Load Scheduling with Different Processor Available Times", Proceedings of 14th International Conference On High Performance Computing (HiPC), (2007c) Goa, India, December.

[18] Compact Muon Solenoid (CMS) Experiment for the Large Hadron Col-lider at CERN (European Lab for Particle Physics), Cms web page, http://cmsinfo.cern.ch/Welcome.html/.

[19] ATLAS (A Toroidal LHC Apparatus) Experiment, CERN (European Lab for Particle Physics), Atlas web page, http://atlas.ch/.

[20] Xuan Lin, Anwar Mamat, Ying Lu_, Jitender Deogun, Steve Goddard," Real-time scheduling of divisible loads in cluster computing environments" , J. Parallel Distrib. Comput. 70 (2010) 296_308

[21] Kijeung Choi 1, Thomas G. Robertazzi , "An Exhaustive Approach to Release Time Aware Divisible Load Scheduling", (IJIDCS) International Journal on Internet and Distributed Computing Systems. Vol: 1 No: 2, 2011.

[22] G. Murugesan and C. Chellappan, "An Economic Allocation of Resources for Divisible Workloads in Grid Computing Paradigm", European Journal of Scientific Research, ISSN 1450-216X Vol.65 No.3 (2011), pp. 434-443 © EuroJournals Publishing, Inc. 2011

[23] Xin Liu, Chunming Qiao, Wei Wei, Xiang Yu,Ting Wang, Weisheng Hu, Wei Guo, and Min-You Wu, "Task Scheduling and Lightpath Establishment in Optical Grids", Journal Of Light Wave Technology, 2009, p 1796-1805.

[24] Dr. D.I. George Amalarethinam1, P. Muthulakshmi , "An Overview of the Scheduling Policies and Algorithms in Grid Computing" , International Journal of Research and Reviews in Computer Science (IJRRCS) Vol. 2, No. 2, April 2011.

[25] Syed Nasir Mehmood Shah, Ahmad Kamil Bin Mahmood, and Alan Oxley 2010, "Hybrid Resource Allocation for Grid Comp uting", in Proceed in gs of the IEEE Second International Conference on Computer Research and Development, 426 – 431.

[26] G.K.Kamalam,and Dr.V.Murali Bhaskaran, "An Effective Approach to Job Scheduling in Decentralized Grid Environment", International Journal of Computer Applications (0975 – 8887) Volume 24– No.1, June 2011.

Proactive Economical Task Scheduling Algorithm for Grid

Piyush Chauhan, Nitin, Rashmi Sharma and Ved Prakash Bhardwaj
Department of Computer Science & Engineering and Information Technology,
Jaypee University of Information Technology, P.O. Waknaghat,
Solan-173234, Himachal Pradesh, India

***Abstract*—Task Duplication technique can reduce the makespan of any application designed for the decentralized grid. In this paper, we have presented an algorithm called Proactive Economical Task Scheduling Algorithm; it can decrease computations in Economical Task Scheduling Algorithm for Grid (EDS-G). It works proactively for a grid. In this approach, each node proactively keeps a hierarchical ordered list of the chosen computation resources. The resources are selected randomly. These resources have specific optimization criteria. It is based on their computation capability and communication cost.**

Keywords-scheduling algorithms; Decentralized Grid; Task Duplication heuristics; Directed Acyclic Graphs;

I. INTRODUCTION AND MOTIVATION

Latest generation of grids, consist of multiple heterogeneous and minute size clusters. Sometimes a grid has only single machine at one physical location. In case of huge grids composed of plentiful minute clusters, relying on Meta scheduler or single central scheduler is not practical. Scheduling jobs on such grids straddling across multiple organizations becomes vital. Many researchers projected peer-to-peer solution in their algorithms for the grid scheduling problem. Finding best possible schedule for multiprocessor scheduling [1], [2], [3] problem is NP complete problem [4].Unlike existing peer-to-peer scheduling algorithms [5],[6], [7], [8], our approaches do the proactive future allocation of computational resources even before task is generated at any grid node. Before the generation of tasks resources are allocated. The allocated resources are arranged in decreasing order, based on optimization criteria.

An optimization criterion depends upon computation capability and bandwidth constraints related to the node. CYCLON protocol [9], [10] is used to establish superior version of shuffling. We add features of hierarchically arranging nodes, which are shuffled by CYCLON protocol. These nodes are arranged in non-increasing order of their computation capability and bandwidth. Weighted directed acyclic task graph are often used to represent a distributed and parallel applications [11], [12], [13]. In DAG node symbolizes application task. Edges of DAG stand for data dependencies between various application tasks of DAG. Here ordered list of allocated computational nodes tend to be useful in reducing assessment for best fit node for tasks of DAG. Duplication based scheduling [4], [14], [15], [16] is one of classification of task scheduling heuristic [17], [18], [20],[12] for DAG application. Predecessor task node for any task node is taken into consideration. Now makespan of an application is reduced by duplicating in ideal time slots between two already scheduled tasks the predecessor task.

In Grid computing environment duplication based scheduling give excellent results. Down side of duplication based scheduling is excessive duplication. This excessive duplication yield extra using up of nodes (computation resources) in grid. Duplication based scheduling is more useful for fine grain task graphs [4], [14]. In addition, duplication based scheduling is very useful for grids with higher CCRs (CCR means Ratio of average communication cost to average computation cost on selected Grid.).Efficiency of scheduling algorithms has a great tendency to decrease with rise in heterogeneity of grid computational resources. By duplicating only important tasks, we prevented some extent of degradation of scheduling algorithms for grid caused by heterogeneity. Task duplication decreases task finish time on computation resources of grid. EDS-G [9] algorithm investigates effect of duplicated tasks over makespan. This way EDS-G algorithm improves schedule by eliminating unproductive duplicated tasks whose removal does not affect the makespan. EDS-G algorithm compares all computational resources for each application task of DAG.

EDS-G finds out which duplicated parent task can be removed from computational nodes of grid such that makespan will not get affected. In proactive economical task scheduling algorithm, we arrange the nodes hierarchically of a grid obtained by running CYCLON protocol. Ordering of grid nodes is done before task is generated at grid node. Shuffling is done as it is explained in CYCLON Protocol. These nodes are arranged in non increasing order, based on optimization criteria. Ordered list is called Empty processor list (EPL).The advantage of hierarchical arrangement of grid computation resources is that we do not need to do task node's comparison with all shortlisted nodes of grid. First task node of DAG will be assigned to first Processor from ordered list. Next task node will be assigned to either second processor from EPL list or to first processor. Out of these two processors, processor capable of finishing second task node fastest will be chosen. Instead of comparing with all shortlisted grid resources here, we compared only two processors for task assignment. Thus, less

time was consumed to assign task. Similarly, rest of task nodes will be assigned to best suited grid resources.

A lot of researchers have proposed task duplication based scheduling technique for computational grid system. But they compare entire shortlisted subset of grid nodes for each task node such that best fit grid node is obtained for each task node. In our research we have reduced the number of comparisons by ordering grid resources proactively. Rest of paper is structured as follows. Section II, explores the preliminaries and background. Section III, explains proposed Proactive economical task scheduling algorithm for grid in two parts. Part A mention modified version of CYCLON and part B explain Proactive Economical Task Scheduling Algorithm for Grid. Section IV, gives conclusion and future scope of work.

II. Related Work

Many researchers have recognized need of efficient task scheduling algorithm [15] for decentralized grid's computational resources. One example of using duplication technique to have efficient scheduling algorithm for heterogeneous computational system is explained in [11].Authors of [11] introduced Heterogeneous Limited Duplication (HLD) scheduling algorithm for heterogeneous computing environment.HLD schedules tasks, based on their precedence constraints. HLD [11] avoid redundant replication by confining duplication to most essential immediate predecessor tasks. HLD is modification of Selective Duplication (SD) Algorithm [12] for heterogeneous computational systems. Selective duplication algorithm is quite effective for homogeneous computing systems. SD [12] algorithm completes in triple phases. Firstly, we arrange task sequence using critical path based priority. Secondary phase is used to select set of candidate processors for candidate task. Last phase of SD algorithm apply candidate task to processor which apply it at earliest, by means of insertion based duplication approach.

One of latest scheduling algorithm for decentralized grid computing system is EDS-G [4].As explained in HLD algorithm EDS-G algorithm also generates priority-based task sequence. Task sequence is generated by arranging tasks in decreasing order of their communication and computation cost along longest directed path from the concerned task to the exit task in directed acyclic graph. Now, initial unscheduled task in the task sequence is chosen and scheduled on a grid computing resource that can end its computation at the earliest by means of duplication approach. This algorithm place the task in a former most idle period amid two already scheduled tasks on the decentralized grid's computational resource. Task N_i on computational grid start working once data arrived from all parent nodes of task nodeN_i.Hence, grid resource may remain idle yielding scheduling holes. When start time of task $N_i$on grid resource is restricted by data arrival from its most important immediate parent (Most important immediate parent of task $N_i$is its ancestor whose data arrive last of all parents of task N_i.) scheduling hole is generated. This scheduling hole may be exploited to duplicate tasks to minimize data arrival time. Start time and finish time of task N_i and all tasks is calculated which help in obtaining makespan. Now two lists A &B are created having record of original task with links to dependent tasks and record of duplicated tasks in non-increasing order of earliest start time respectively. List B is modified if removal of duplicated task from list B does not affect makespan. In addition, list A removes those tasks, which have already been duplicated and hence are not providing any output to any immediate descendant task. Our algorithm Proactive Economical Duplication for Decentralized Grid uses partially similar algorithm but need less number of comparisons to assign tasks to best fit corresponding nodes. This becomes possible due to proactively arranging grid resources in hierarchy of performance. Out of list of z resources, we can choose random x resources $(x < z)$.

This selection is done with help of CYCLON [5] protocol. CYCLON is gossip based protocol [6], [19]. In CYCLON, each peer knows small constantly changing set of other peers. All peers in grid occasionally contacts neighbor peer to shuffle caches and this neighbor peer is chosen whose information was the earliest one to have been injected in the grid.

III. Proposed Proactive Economical Duplication Scheduling Algorithm for Grid

In this section, we will present our proposed algorithm in a structured manner. In first step, we will focus how the resources can be ordered in a proper manner. In next step, it will be illustrated that how we can improve the quality of schedule and get scheduling process better than EDS-G.

A. Proactive Ordering of Selected Neighbour Nodes in a Grid

Proactive ordering of resources is done by modified version of CYCLON [9],[10].Overlay is formed and connected by means of epidemic algorithm [10], [7]. Each node knows a petite set of neighbor nodes, which are continuously changing. This node occasionally contacts a neighboring node whose information was the first one to have been injected in the network and exchanges some of their neighbors. Each node maintains a neighbor list in a small, fixed size cache of $\boldsymbol{T}$ entries. Neighboring node's computation capability, communication cost, IP and port address are kept in Cache entry. Nodes start neighbor swap periodically, however not synchronized, at a fixed time periodΔt. Hence, other than step of ordering neighbors, rest of steps are like CYCLON [9] protocol. The ordering and shuffling [5] of computational resources is done when initiating peer$\boldsymbol{I}$ carry out following steps: Select a random subset of $\boldsymbol{R}$ neighbors (Length of list $\boldsymbol{R}$ is $\boldsymbol{l}(1 \leq l \leq T)$) from a set of neighbors of $\boldsymbol{I}$ (initiating peer).

1. In LIST $\boldsymbol{R}$ Increment by 1 the age of all neighbors.
2. Select neighbor $\boldsymbol{H}$ havingmaximum age among all neighbors in list$\boldsymbol{R}$, and$\boldsymbol{l}-1$ other random neighbors.
3. Replace$\boldsymbol{H}$'s entries with entry of age zero (i.e. new one) and with address of $\boldsymbol{I}$.
4. Sent updated subset to $\boldsymbol{H}$node.
5. Receive from $\boldsymbol{H}$ a subset of no more than $\boldsymbol{l}$of its own entries.
6. Discard entries pointing at $\boldsymbol{I}$ and entries already contained in $\boldsymbol{I}$'s cache.

7. Update ***I***'s cache to include all remaining entries, by utilizing initially empty slots in cache (if any),and secondly substitute entries among the ones sent to ***H***.
8. List **EPL** is formed by arranging neighbors in ***R*** in a non increasing order of their computation capability and communication cost.

Node ***H*** answers by sending back a random subset of at most***l*** of its neighbors, and renew its own cache to house all received entries. However, it does not increase any entry's age until its own turn comes to run above stated protocol. This hierarchical ordering of neighboring nodes in grid helps us in reducing computations involved in Task Scheduling algorithms like EDS-G for grid.

B. Proactive EDS-G Algorithm

This section of the paper represents Proactive Economical Task Duplication based Scheduling Algorithm. This algorithm is split into two parts. Our work is modification of [4].In proactive EDS-G algorithms, first part gives a method for scheduling based on technique of insertion based task duplication. Second part of algorithm removes tasks whose termination does not have an effect on makespan. The algorithm's pseudo code is shown in figure1.Once we have orderly placed computational resources before task is generated, we can use it when task is generated at any grid node. When task is generated at grid node, we split it into interdependent sub-tasks and use DAG to show their interdependence. Grid computing system can be characterized as $G = (P, B)$ where $P(p_1, p_2, p_3 \dots p_x, p_{x+1}, \dots p_z)$ are grid's computing resources. Grid's resources are connected by various communication channels B. Because grid's resources are of heterogeneous nature, hence same task's computing cost on different processing nodes will be different. The nature of communication channel between nodes of grid is also heterogeneous type.

Here DAG's tasks are of non-preemptive nature. Along with communication computation happen in parallel, (This is possible because each node of grid contain co-processor for communication.).If any two tasks are scheduled on same grid node, we consider communication cost between these two tasks to be negligible. On finishing of task in any grid node, that node sends data in parallel to all child tasks. Weighted DAG is used to represent application of grid. $D = (Dn, e, T, C)$ here task node gets symbolized by Dn. T is a computation cost matrix [14]. At$t_{ij} \in T$ represent expected time to execute task of DAG Dn_i on grid nodep_j.e is set of communication edges. C is communication cost matrix.Expected time to communicate data from task Dn_i to Dn_j represented by $c_{ij} \in C$. Task$Dn_i's$ Mean computation cost represented by $\overline{t_i}$[1],[5] is calculated as follows:

$$\overline{t_i} = \frac{\sum_{j=1}^{z} t_{ij}}{z} \text{ for all } 1 \le i \le n \quad (1)$$

Mean communication cost $\overline{c_{ij}}$[11], [14] between task $Dn_i$and task $Dn_j$is calculated as follows:

$$\overline{c_{ij}} = \frac{c_{ij}}{meandata \text{ transport } rateover \text{ entire } linksofgrid}$$

$$\forall 1 \le i \ne j \le n \quad (2)$$

Priority based task sequence is generated by ordering task in decreasing order of their computation and communication cost which is obtained via computing mean cost parameter(MCP) recursively as follows:

$$MCP_i = \overline{t}_i + Max[MCP_j + \overline{c}_{ij}] \forall Dn_j \in successor\ (Dn_i) \quad (3)$$

Successor(Dn_i) denote set of immediate child nodes in DAG of task Dn_j.Task sequence's first unscheduled task is selected and scheduled on a grid's computational resource which finish it first using task duplication approach. Now node of grid, which is first in EPL list, will be chosen and added to UP list. We will add processor in UP list from EPL list only if no fresh processor is available in UP list. It implies that addition of new processor/node for next task is done to UP list because all existing processors in UP list are having some task to execute. Next task from priority based task sequence will be chosen. Now we check out on which processor of UP list minimum makespan is possible using task duplication approach. Makespan is calculated as:

$$Makespan = maximum[F_{ix}] \quad (4)$$

F_{ix}[14] is finish time of task Dn_ion resourse P_x.

$$F_{ix} = S_{ix} + t_{ix} \quad (5)$$

(S_{ix} is start time of task i on grid resource P_x).Formula to compute S_{ix}[4], [14] is as follows:

$$S_{ix} = maximum[DT(M_i, p_x), minimum[p_x^R, G_p^S]] \quad (6)$$

In above formula p_x^R stands for ready time of processor P_x. G_p^S is start time of first suitable and available time slot $G_p$that can accommodate task Dn_i on resource P_x,if it exist then only we calculate F_{ix} . $DT(M_i, p_x)$ [4] is data arrival time for most important immediate parent$M_i$of task Dn_i on P_x. Formula to calculate $DT(M_i, p_x)$is given by:

$$DT(M_i, p_x) = maximum\ [\ minimum[\ F_{jx}\ , F'_{jx} + c_{ji}]\ ] \quad (7)$$

Proactive EDS-G Algorithm

Begin

1: Sort nodes in non increasing order of performance of processors. EMPTY PROCESSOR LIST (**EPL**);

2: Construct a priority based task sequence***β***;

3: Make a empty list of used Processors (**UP**);

4: **do {**

5: Select the first unscheduled task Dn_iin the task sequence$\boldsymbol{\beta}$.

6: **if** (no fresh Processor in **UP** list)

7: add 1st processor from **EPL** to **UP** list & remove it from **EPL** list;

8: **for** (all $\boldsymbol{p_x}$processors in list **UP**)
{
9: Sort the list of immediate parents of $\boldsymbol{Dn_i}$in non-increasing order of data arrival time;

10: **for** all immediate parents, select the first immediate parent $\boldsymbol{Dn_j}$ from the list at step 9
{
11: **if** duplication of$\boldsymbol{Dn_j}$can reduce finish time $\boldsymbol{F_{ik}}$of $\boldsymbol{Dn_i}$ on $\boldsymbol{P_x}$Duplicate $\boldsymbol{Dn_j}$;
12 :}

13: Compute earliest finish time$\boldsymbol{F_{ix}}$ of $\boldsymbol{Dn_i}$on$\boldsymbol{P_x}$ using eq. (5);
14: }

15: Find the minimum earliest finish time of $\boldsymbol{Dn_i}$;

16: AssignDn_ion resource $\boldsymbol{P_x}$ with minimum $\boldsymbol{F_{ix}}$in schedule $\boldsymbol{S}$;

17 :} **while** (there are unscheduled tasks in task sequence **β**);

18: Maintain a list **A** of origin tasks which have been duplicated later with their successor links to other tasks and list **B** of duplicated tasks in non-increasing order of their earliest start time;

19: **for** (each duplicated task$\boldsymbol{Dn_i}$ in list **B**)
{
20: **if** (no change in makespan of schedule $\boldsymbol{S}$ after Removing duplicated task $\boldsymbol{Dn_i}$)

21: Remove this duplicated task Dn_i from the schedule $\boldsymbol{S}$and update list A;
22: }
23: **for** (each task $\boldsymbol{Dn_i}$in list **A**)
{
24: **if** (task$\boldsymbol{Dn_i}$has no dependent task in schedule$\boldsymbol{S}$due to its duplication later on different processors)

25: remove this task$\boldsymbol{Dn_i}$from the schedule $\boldsymbol{S}$;
26: }

End

Fig.1. Pseudo Code of Proactive economical task scheduling algorithm for grid.

Once schedule is obtained from above shown algorithm, List A and B are maintained. Like EDS-G algorithm list A has record of original tasks and links of these tasks to their corresponding dependent tasks. Duplicated task are stored in decreasing order of earliest start time in list B. This Schedule will be modified if there is no change in makespan on removal of duplicated task from list B. From list A, tasks which do not provide results to immediate successor because of their duplication elsewhere are also erased. This updated schedule will not only contain optimum number of duplications like EDS-G algorithms but also it obtains this optimum schedule in less number of comparisons of best fit processors for task nodes using task duplication approach.

IV Conclusion and Future Work

To improve the performance of distributed and grid systems duplication based strategy is widely used. This limited duplication based approach helps in improving performance of the grid computing system in economical way. Scheduling algorithms for the grid computing system have high communication cost. Previously, existing EDS-G algorithm improves makespan of the Task graph with precedence constraints. Our approach simply reduces computations involved in EDS-G. In Proactive economical task scheduling algorithm for computational grid system reduced computation and comparison is achieved to duplicate task on best fit grid resource. This deduction in the computation of best grid node for any given task node is due to proactively making EPL list.EPL list contain detail of the chosen grid nodes in non increasing order of their performance. Obviously, in our algorithm we need not compare task node with all chosen nodes of grid like in EDS-G. Instead, task node is simply assigned or duplicated to fresh node or any other node in UP list. Number of grid nodes in UP list is always less than random number of grid nodes chosen in EDS-G. Hence Proactive economical task scheduling algorithm for grid will not only give better results in terms of time but also in terms of computation and communication cost. This new approach can be further enhanced to include multiple bags of task applications.

References

[1] J. Liou and M. Palis, A Comparison of General Approaches to Multiprocessor Scheduling, Proceedings of the 11th International Parallel Processing Symposium, pp. 152–156, 1997.

[2] A.A. Khan, C. McCreary and M.S. Jones, A Comparison of Multiprocessor Scheduling Heuristics, ICPP, pp. 243–250,1994.

[3] B. Olivier, B.Vincent and R. Yves, The Iso–level Scheduling Heuristic for Heterogeneous Processors, Proceedings of 10th Euromicro workshop on Parallel, Distributed and Network Based Processing, pp.335–342, 2002.

[4] A. Agarwal and P. Kumar, Economical Task Scheduling Algorithm for Grid Computing Systems, Global Journal of Computer Science and Technology10(11), pp. 48–53, 2010.

[5] A. Stavrou, D. Rubenstein, and S. Sahu, A Lightweight, Robust P2P System to Handle Flash Crowds, IEEE Journal on Selected Areas in Communications22(1), pp. 6–17, 2004.

[6] A. Ganesh, A.M. Kermarrec, and L. Massouli´e, Peer–to–Peer Membership Management for Gossip–based Protocols, IEEE Transaction on Computers52(2),pp. 139–149, 2003.

[7] S. Voulgaris and M. van Steen, An Epidemic Protocol for Managing Routing Tables in Very Large Peer–to–Peer Networks, In 14th

IFIP/IEEEWorkshop on Distributed Systems: Operations and Management, Heidelberg, Germany, 2003.

[8] J. Risson and T. Moors, Survey of Research towards Robust Peer–to–Peer Networks: Search Methods, Technical Report UNSW–EE–P2P–1–1, University of New South Wales, Sydney, Australia, 2004.

[9] S. Voulgaris, D. Gavidia, and M. van Steen, CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays, Journal of Network and Systems Management 13(2), 2005.

[10] Amit Agarwal and Padam Kumar, Economiacal Duplication based Task Scheduling for Hetrogeneous and Homogeneous Computing systems, IEEE International AdvanceComputing Conference, pp.87–93, 2009.

[11] Savina Bansal, Padam Kumar and Kuldip Singh, Dealing with Heterogeneity Through Limited Duplication for Scheduling Precedence Constrained Task Graphs, Journal of Parallel and Distributed Computing65(4), pp. 479–491, 2005.

[12] Savina Bansal, Padam Kumar and Kuldip Singh, An Improved Duplication Strategy for Scheduling Precedence Constrained Graphs in Multiprocessor Systems, IEEE Transaction on Parallel and Distributed Systems14(6),pp.533–544, 2003.

[13] Y.K. Kwok and I. Ahmed, Benchmarking the Task Graph Scheduling Algorithms, IPPS/SPDP, pp. 531–537, 1998.

[14] A. Agarwal and P. Kumar, Economiacal Duplication based Task Scheduling for Hetrogeneous and Homogeneous Computing systems, IEEE International Advance Computing Conference, pp.87–93, 2009.

[15] F. Dong and S.G. Akl, Scheduling Algorithms for Grid Computing: State of the Art and Open Problems, Technical Report No. 2006–504, School of Computing, Queen's University, Kingston, Ontario, 2006.

[16] K.C. Lai and C.T. Yang, A Dominant Predecessor Duplication Scheduling Algorithm for Heterogeneous Systems, Journal of Supercomputing44(2), pp. 126–145, 2008.

[17] H. Topcuoglu, S. Hariri and M.Y. Wu, Performance–Effective and Low–Complexity Task Scheduling for Heterogeneous Computing, IEEE Transaction on Parallel and Distributed Systems13(3), pp. 260–274, 2002.

[18] C. Ernemann, V. Hamscher, and R.Yahyapour, Economic Scheduling in Grid Computing, Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processingof Lecture Notes in Computer Science 2537, Springer Verlag, pp. 128–152, 2002.

[19] M. Jelasity and O. Babaoglu, T–Man: Fast Gossip–based Construction of Large–scale Overlay Topologies, Technical Report UBLCS–2004–7, University of Bologna, Department of Computer Science, Bologna, Italy, 2004.

[20] A.A. Khan, C. McCreary and M.S. Jones, A Comparison of Multiprocessor Scheduling Heuristics, ICPP, pp. 243–250, 1994.

Evaluation of Electric-Grids Performance Using Robust Estimators

Ibrahim Habiballah, EE Dept., King Fahd University of Petroleum & Minerals, Saudi Arabia

Abstract

This paper evaluates the performance of electric-grids due to imperfect prediction of the state variables of these grids. Currently, system control centers are supported by supervisory control and data acquisition (SCADA) systems that report the status of circuit breakers as well as voltage, current, and power levels. This paper presents different types of robust state estimators in order to predict the best status of the grid. Different set up of bad measurements and bad leverage measurements are introduced to evaluate the performance of these estimators. Independent and confirming bad measurements are considered. The performance of the different estimators is evaluated by comparing the summation of the absolute residuals between the estimated and the actual values. The IEEE 14-bus DC example is used to illustrate the properties of these estimators.

Keywords: Robust Estimators, Electric Grid, Bad Measurements.

1. Introduction

Opportunities for improving the functioning and reliability of electric-grids arise from technological developments in sensing, communications, control, and power electronics. These technologies can enhance efficiency and reliability, increase capacity utilization, enable more rapid response to remediate contingencies, and increase flexibility in controlling power flows on transmission lines. If properly deployed and accompanied by appropriate policies, they can facilitate the integration of large volumes of renewable and distributed generation, provide greater visibility of the instantaneous state of the grids, and make possible the engagement of demand as a resource [1].

Power systems require a level of centralized planning and operation to ensure system reliability. System operators at control centers carry out many of these centralized functions in support of operations, including short-term monitoring, analysis, and control. System operators use various displays and alarms to develop awareness of the state of the system. Raw data reported to control centers, such as SCADA, are analyzed using computer tools, e.g., state estimators that can give insight to the current and future state of the grid. This suite of tools is collectively known as an energy management system.

Estimators are used when unknown states in a given mathematical model must be determined from available measurements. Usually, there are more measurements than are strictly needed to define the unknowns and the problem is called over-determined. This type of problem is variously referred to as state estimation, parameter estimation, multivariate regression, and curve fitting. All these terms essentially describe the same computational process.

Various robust estimators have been proposed, which try to combat these problems by processing the measurements so that the outliers have little or no effect on the estimated states [9-13]. Specialised algorithms have generally been developed to solve robust estimation problems. However, it has been found that all the robust estimators considered in this paper can be solved using standard mathematical programming algorithms. Some of these formulations are original and can provide accurate solutions with good computational efficiency.

This paper evaluates the performance of some robust state estimation methods using general-purpose mathematical programming algorithms. IEEE 14-bus DC example is used to illustrate the properties of these methods.

By looking at the IEEE 14-bus DC example, it is possible to see how these estimators work, and perhaps more importantly understand conditions where some algorithms may produce unexpected results. The detection of outliers and the elimination of their effects on the estimates can provide measurement fault detection and measurement fault tolerance. The presented estimators are evaluated by comparing the summation of the absolute errors between the actual and estimated values, i.e., the summation of the residuals.

2. Mathematical Formulation of Different Estimators

2.1 Least Squares (LS) Estimator

Assume a multivariate linear model:

$$A\underline{x} = \underline{b} \qquad (1)$$

where A is a known (m x n) matrix
$\underline{x}$ is an unknown state vector (n x 1), and
$\underline{b}$ is an (m x 1) vector of values which can be measured

Normally their are more measurements than states, so m > n, and it is expected that each measurement will include some unknown error:

$$A\underline{x} = \underline{b} + \underline{e} \quad (2)$$

where $\underline{b}$ is a vector of known measurements
$\underline{e}$ is a vector of unknown measurement errors

The well known least squares estimate can be formulates as

$$\underset{\underline{x},\,\underline{e}}{\text{Min}} \quad \underline{e}^T\underline{e} \quad (3)$$

Subject to the equality constraint of equation (2)

Equations (2) and (3) can readily be solved as a linearly constrained quadratic program [7].

2.2. Least Absolute Values (LAV) Estimator

The existence of unexpectedly large residuals is associated with the presence of outliers in the measurement set. The least squares principle could be said to give outliers excessive weight by squaring the value of the residual. An alternative approach is to minimise the sum of absolute values of residuals. By taking the absolute value (or modulus) of the residual, the effect of outliers on the estimate is reduced. A property of Least Absolute Value (LAV) estimates is that at least 'n' of the measurements will be fitted exactly (with zero residuals) [7].

An efficient algorithm for LAV estimation is via the solution of the following linear program:

$$\underset{\underline{x},\,\underline{e},\,\underline{f}}{\text{Min}} \quad \Sigma\,(e_i + f_i) \quad (4)$$

Subject to:
$$A\underline{x} - \underline{e} + \underline{f} = \underline{b} \quad (5)$$
$$\underline{e} \geq \underline{0}\,,\ \underline{f} \geq \underline{0} \quad (6)$$

where $\underline{e}$ and $\underline{f}$ are non-negative vectors of unknown measurement errors

2.3. Least Median of Squares (LMS) Estimator

Rousseeuw [3] introduced a new robust estimation principle referred to as Least Median of Squares (LMS). This is a generalisation of the idea that the median of a set of real values is a more robust estimate than the mean. This idea is generalised to the multivariate estimator problem by finding an estimate that minimises the median of the squared residuals. Roughly speaking, the median is unaffected even if up to half of the residuals are very high [7]. (When larger problems with more than two state variables are considered, it is customary to minimise the (n+m+1)/2 ordered squared residual, since non-zero residuals only exist for m > n.) [4]. A characterisation of an LMS estimate is that it seeks a regression that minimises the value of a tolerance 't' whereby the majority of the measurements fall within tolerance. This interpretation motivates an original implementation of an LMS estimator via a mixed integer program, formulated as follows:

$$\underset{\underline{x},\,\underline{k},\,t}{\text{Min}} \quad t \quad (7)$$

Subject to:
$$\underline{b} - t - M\,\underline{k} \leq A\underline{x} \leq \underline{b} + t + M\,\underline{k} \quad (8)$$
$$k_1 + k_2 + \ldots + k_m \leq K \quad (9)$$

where
$\underline{k}$ is an unknown binary integer vector
(each element is either 0 or 1)
t is an unknown scalar tolerance
M is a specified arbitrary large positive scalar
K is specified as m/2 (if m is even)
(m-1)/2 (if m is odd)

In the above mixed integer linear program, the binary integer variables $\underline{k}$ allow some of the measurements to be 'switched off' or 'rejected' (in other words to be outside the tolerance 't'). The value of 'M' is chosen to be large enough so that when a measurement is switched off (by k_i being 1) the expanded tolerance 't + M $\underline{k}$' is large enough to avoid that measurement having any effect on the estimate $\underline{x}$. The specification of K, together with constraint (9), is such that a majority of the measurements cannot be switched off.

2.4. Least Trimmed Squares (LTS) Estimator

The principle of Least Trimmed Squares (LTS), also proposed by Rousseeuw [3], is to consider the sum of squared errors for the (m-K) smallest residuals only. Equivalently, the K largest residuals are rejected and the remaining residuals are considered in a least squares objective. An original mathematical programming formulation for LTS is as follows:

$$\underset{\underline{x},\,\underline{e},\,\underline{k}}{\text{Min}} \quad \underline{e}^T\underline{e} \quad (10)$$

Subject to:
$$\underline{b} - M\,\underline{k} \leq A\underline{x} - \underline{e} \leq \underline{b} + M\,\underline{k} \quad (11)$$
$$k_1 + k_2 + \ldots + k_m \leq K \quad (12)$$

where
$\underline{k}$ is an unknown binary integer vector
(each element is either 0 or 1)
M is a specified arbitrary large positive scalar
K is a specified number of measurements that may be rejected
$\underline{e}$ is a vector of unknown measurement errors

In this formulation, the binary vector $\underline{k}$ allows up to K of the measurements to be switched off. At the solution, any measurement that is switched off (k_i being 1) will have its associated e_i at zero (and not contributing to the objective function) [7]. This formulation is a Mixed Integer Nonlinear Program, which can be efficiently solved via the

NEOS server, using the MINLP algorithm of Fletcher and Leyffer [5].

2.5. Least Measurements Rejected (LMR) Estimator

Whereas the LMS method pre-determines the number of measurements to reject and then seeks a regression that minimises the tolerance on the retained measurements, a new approach has been proposed by M. Irving [7], which follows the converse principle. This has been implemented using a genetic algorithm in reference [6] and as a mathematical program in. This approach requires the user to pre-specify a tolerance for each measurement and then seeks a regression that minimises the number of measurements unable to satisfy their tolerance. The tolerance value of each measurement should be chosen according to the range of error within which the measurement can still be regarded as 'good'. For example, a temperature measurement of 12.5 ^{o}C might have a tolerance of $\pm$ 1.0 ^{o}C. This is compatible with the usual engineering approach for specifying transducer accuracy. The new approach is referred to as Least Measurements Rejected (LMR) [7].

The mathematical programming formulation for LMR is as follows:

$$\underset{\underline{x},\,\underline{k}}{\text{Min}} \quad \Sigma\, k_i \qquad (13)$$

$$\text{Subject to: } \underline{b} - M\,\underline{k} - \underline{t} \le A\underline{x} \le \underline{b} + M\,\underline{k} + \underline{t} \qquad (14)$$

where

$\underline{k}$ is an unknown binary integer vector (each element is either 0 or 1)

M is a specified arbitrary large positive scalar

$\underline{t}$ is a specified vector of tolerances on measurement errors

As before, the binary vector $\underline{k}$ allows some measurements to be 'switched off', but in this case the solution will be a regression which maximises the number of measurements that are within tolerance (i.e. minimises the number of measurements which need to be switched off with $k_i = 1$). This formulation is a Mixed Integer Linear Program, which can be efficiently solved via the NEOS server, using the MINTO algorithm.

3. IEEE 14-Bus DC Example

To illustrate and evaluate the solutions of the above five estimators, the IEEE 14-bus DC example illustrated in [8] is considered with some modification on the values of generating units, the loads, and the measured meters. The actual values of all power injections and line flows as obtained from a load flow solution are given in Table 1. This will be used as a benchmark for comparison with the five estimators. The selected measurement meters with their values are shown in Table 2. Normal Gaussian noises are introduced to these measurements. The redundancy of the selected measurements is 2.1.

Six linear regression cases have been considered; a summary of which is shown in Table 3. The cases are selected to evaluate the performance of the estimators for different scenario of bad measurements and bad leverage measurements (i.e., bad measurements on leverage points). The solutions have been obtained using the general-purpose non-linear programming package MINOS by Murtagh and Saunders, available via the NEOS public-domain web-service [3].

The measurements in Cases 1-3 are similar to the ones given in Table 2 with the introduction of negative reading (due to improper switching connection) on some of the meters associated to a leverage point: bus 4. Case 1 is an independent meter: P4 (power-injection on bus 4). Case 2 assumes three confirming bad-data: power-injection P4, line-flows P3-4 and P4-7. Case 3 assumes five confirming bad-data: power-injections P3, P4 and P5, line-flows P3-4 and P4-7.

The measurements in Cases 4-6 are similar to the ones given in Table 2 with the introduction of negative reading on some of the meters that are not associated to a leverage point: bus 14. Case 4 is an independent meter: P14 (power-injection on bus 14). Case 5 assumes two confirming bad-data: power-injection P14, and line-flow P13-14. Case 6 assumes three confirming bad-data: power-injections P13 and P14, and line-flow P13-14.

The estimated power-injections and line-flows for Case 1 are presented in Table 4 for the five estimators. Table 4 shows the actual values of the power-injections and line-flows as well as the summation of the absolute residuals. The location of the meters is highlighted on the first column of the table. It can be observed that all the measurements can be fitted reasonably well by all estimators, except the LS. The LAV, LMS, and LMR estimators performed better than the other estimators. The LS estimator could not reject the bad-measurement, while the others have successfully rejected the introduced bad-measurement.

The estimated power-injections and line-flows for Case 2 are presented in Table 5 for the five estimators. It can be observed that all the measurements can be fitted reasonably well by all estimators, except the LS, and LAV. The LMR estimator performed better than the LAV, LMS estimators. The LS, and LAV estimators could not reject the introduced multiple confirmed bad-measurements, while the others have successfully rejected these bad-measurements.

The estimated power-injections and line-flows for Case 3 are presented in Table 6 for the five estimators. It can be observed that measurements are not fitted well by all estimators. The five estimators could not reject the introduced multiple confirmed bad-measurements. This is

a breakdown case were the number of the introduced multiple confirmed bad-measurements become the number of critical measurements (a critical measurement is defined as the measurement which if removed then the system become unobservable).

The estimated power-injections and line-flows for Case 4 are presented in Table 7 for the five estimators. It can be observed that all the measurements can be fitted reasonably well by all estimators, except the LS. The LMS, and LMR estimators performed better than the other estimators. The LS estimator could not reject the bad-measurement, while the others have successfully rejected the introduced bad-measurement.

The estimated power-injections and line-flows for Case 5 are presented in Table 8 for the five estimators. It can be observed that all the measurements can be fitted reasonably well by all estimators, except the LS. The LMR estimator performed the best among the other estimators. The LS, estimator could not reject the introduced multiple confirmed bad-measurements, while the others have successfully rejected these bad-measurements.

The estimated power-injections and line-flows for Case 6 are presented in Table 9 for the five estimators. It can be observed that measurements are not fitted well by all estimators. The five estimators could not reject the introduced multiple confirmed bad-measurements. This is a breakdown case were the number of the introduced multiple confirmed bad-measurements become the number of critical measurements.

It can also be observed that the availability of multiple confirmed bad-measurements may introduce bad estimate to the power-injections and/or power-flows of meters with good readings. For example, in case 2, bus 9 which has a good meter reading, has a bad estimate with the LAV estimator. This could be attributed to the fact that bus 9 is directly connected to bus 4. Another interesting observation is that, in general, estimators perform better when bad-measurements are associated with locations that are not leverage points.

Table 10 presents a summary of the five estimator performance for all cases. For example, the LAV estimator performs better in rejecting bad-measurements associated with locations that are not leverage points. Table 11 shows which estimator has successfully rejected the bad-measurements. It can be observed that the LS estimator takes into account all the measurements regardless of their status, and need bad-data identification step in order to reject the bad-measurements.

In summary, it can be observed, without loss of generality, that the LMR estimator has the best performance among the other estimators. It can also be observed that the LS and LTS estimators minimize the square of the residuals and need non-linear programming packages. Their performance is relatively lower than the remaining estimators when bad leverage measurements are introduced. On the other hand, the LMS, LTS, and LMR estimators have successfully rejected the introduced bad-data measurements for most of the cases as shown in Table 11. The LS estimator has failed to reject all the introduced bad-data measurements in all cases.

4. Conclusions

This paper presented different robust power system state estimators. It introduced the original formulations that allow solutions to be obtained using general-purpose mathematical programming algorithms. The IEEE 14-bus DC example was used to illustrate the properties of these estimators. Different set up of bad-data measurements were introduced to evaluate the performance of the estimators. The LMR estimator is among the best estimator in terms of the estimated values. The LMS, LTS, and LMR estimators are among the best in term of rejecting bad-data measurements.

Acknowledgments

The authors acknowledge the support of the EE Department, King Fahd University of Petroleum & Minerals, Saudi Arabia under the project number IN101032.

References

[1] MIT Study Report, "The Future of the Electric Grid", http://web.mit.edu/mitei/research/studies/the-electric-grid-2011.shtml, 2011.

[2] A. Abur and A. G. Exposito, *Power System State Estimation Theory and Implementation*, Marcel Dekker, ISBN 0-8247-5570-7, 2004.

[3] P.J. Rousseeuw, "Least median of squares", *Jnl. of American Statistical Association*, 79, pp 871-880, 1984

[4] A. Monticelli, *State Estimation in Electric Power Systems A Generalized Approach*, Kluwer, ISBN 0-7923-8519-5, 1999

[5] NEOS, "NEOS sever for optimisation", http://neos.mcs.anl.gov/neos, Feb. 2008

[6] A. K. Al-Othman and M. R. Irving, "Robust state estimator based on maximum constraints satisfaction of uncertain measurements", *Measurement*, Vol. 40, pp. 347 - 359, 2007

[7] M. R. Irving, "Robust state estimation using mixed integer programming", *IEEE trans. Power Systems*, 23, 3, pp 1519-1520, August 2008

[8] G.N. Korres, "A New Method for Treatment of Equality Constraints in Power System State Estimation", *Proceeding PSCC Conference*, Glasgow, UK, paper no. 107, July 2008

[9] E. Handschin, M. Langer, and E. Kliokys, "An interior point method for state estimation with current magnitude measurements and inequality constraints", IEEE Power Industry Computer Application Conf., pp. 385–391, 1995

[10] R.A. Jabr, "Power system Huber M-estimation with equality and inequality constraints", Elec. Power Syst. Res., 74, (2), pp. 239–246, 2005

[11] R.A. Jabr and B.C. Pal, "AC network state estimation using linear measurement functions", The Institution of Engineering and Technology, IET Gener. Transm. Distrib., 2, (1), pp. 1–6, June 2008

[12] H. Singh F.L. Alvarado, "Fast Approximations to LAV Solutions for State Estimation of Power Systems", Proceedings of PSCC Avignon France Aug. 30-Sep 3, 1993

[13] T. Dhadbanjan, and S. S. K. Vanjari, "Linear Programming Approach for Power System State Estimation Using Upper Bound Optimization Techniques", International Journal of Emerging Electric Power Systems, Volume 11, Issue 3, Article 2, 2010

Table 1: Load Flow's Power-Injections and Line-Flows

P1	0.2500	**P13**	-0.0900	**P6-11**	0.0368
P2	0.1500	**P14**	-0.1250	**P6-12**	0.0688
P3	0.0750	**P1-2**	0.0669	**P6-13**	0.0676
P4	0.0600	**P1-5**	0.1831	**P7-8**	-0.0950
P5	-0.0750	**P2-3**	0.0086	**P7-9**	0.1239
P6	-0.0750	**P2-4**	0.0922	**P9-10**	0.0982
P7	-0.0150	**P2-5**	0.1162	**P9-14**	0.1486
P8	0.0950	**P3-4**	0.0836	**P10-11**	0.0432
P9	-0.0450	**P4-5**	0.0240	**P12-13**	-0.0012
P10	-0.0550	**P4-7**	0.0439	**P13-14**	-0.0236
P11	-0.0800	**P4-9**	0.1678		
P12	-0.0700	**P5-6**	0.2483		

Table 2: Normal Noise Measurements on Selected Locations

P1	0.23920	**P10**	-0.05583	**P4-7**	0.04411
P2	0.14832	**P11**	-0.07976	**P5-6**	0.25197
P3	0.07552	**P12**	-0.07003	**P6-12**	0.06885
P4	0.05994	**P13**	-0.09086	**P7-8**	-0.09538
P5	-0.07506	**P14**	-0.12273	**P7-9**	0.12364
P6	-0.07442	**P1-2**	0.06582	**P9-10**	0.09872
P7	-0.01475	**P1-5**	0.18324	**P10-11**	0.04351
P8	0.09544	**P2-3**	0.00864	**P12-13**	-0.00119
P9	-0.04521	**P3-4**	0.08235	**P13-14**	-0.02321

Table 3: Summary of the Considered Cases

Case #	Meter Location	Measurements
1	P4	-0.05994
2	P4, P3-4, P4-7	-0.05994, -0.08235, -0.04411
3	P3, P4, P3-4, P4-7	-0.0750, -0.05994, -0.08235, -0.04411
4	P14	0.12273
5	P14, P13-14	0.12273, 0.02321
6	P13, P14, P13-14	0.09086, 0.12273, 0.02321

Table 4: Estimated Power-Injections and Line-Flows for Case 1

	Actual	**LS**	**LAV**	**LMS**	**LTS**	**LMR**
P1	0.2500	0.2535	0.2482	0.2491	0.2431	0.2489
P2	0.1500	0.1643	0.1522	0.1490	0.1493	0.1488
P3	0.0750	0.0869	0.0755	0.0748	0.0746	0.0750
P4	0.0600	-0.0401	0.0573	0.0632	0.0700	0.0646
P5	-0.0750	-0.0580	-0.0751	-0.0758	-0.0742	-0.0756
P6	-0.0750	-0.0703	-0.0744	-0.0743	-0.0754	-0.0749
P7	-0.0150	-0.0063	-0.0148	-0.0155	-0.0152	-0.0153
P8	0.0950	0.0997	0.0954	0.0947	0.0952	0.0949
P9	-0.0450	-0.0342	-0.0452	-0.0459	-0.0456	-0.0453
P10	-0.0550	-0.0510	-0.0558	-0.0560	-0.0559	-0.0559
P11	-0.0800	-0.0770	-0.0798	-0.0805	-0.0803	-0.0803
P12	-0.0700	-0.0670	-0.0700	-0.0693	-0.0705	-0.0705
P13	-0.0900	-0.0860	-0.0909	-0.0902	-0.0912	-0.0914
P14	-0.1250	-0.1144	-0.1227	-0.1234	-0.1240	-0.1232
P1-2	0.0669	0.0708	0.0658	0.0665	0.0634	0.0663
P1-5	0.1831	0.1827	0.1824	0.1825	0.1797	0.1826
P2-3	0.0086	0.0121	0.0086	0.0082	0.0073	0.0079
P2-4	0.0922	0.1110	0.0928	0.0913	0.0892	0.0909
P2-5	0.1162	0.1120	0.1166	0.1160	0.1162	0.1163
P3-4	0.0836	0.0990	0.0842	0.0831	0.0819	0.0830
P4-5	0.0240	0.0009	0.0238	0.0247	0.0271	0.0254
P4-7	0.0439	0.0252	0.0433	0.0445	0.0447	0.0445
P4-9	0.1678	0.1438	0.1672	0.1683	0.1694	0.1686
P5-6	0.2483	0.2376	0.2477	0.2475	0.2488	0.2487
P6-11	0.0368	0.0367	0.0370	0.0377	0.0367	0.0370
P6-12	0.0688	0.0659	0.0688	0.0683	0.0691	0.0691
P6-13	0.0676	0.0647	0.0675	0.0673	0.0676	0.0677
P7-8	-0.0950	-0.0997	-0.0954	-0.0947	-0.0952	-0.0949
P7-9	0.1239	0.1186	0.1239	0.1238	0.1247	0.1241
P9-10	0.0982	0.0913	0.0986	0.0988	0.0995	0.0992
P9-14	0.1486	0.1369	0.1473	0.1473	0.1490	0.1483
P10-11	0.0432	0.0402	0.0428	0.0428	0.0436	0.0433
P12-13	-0.0012	-0.0012	-0.0012	-0.0010	-0.0014	-0.0014
P13-14	-0.0236	-0.0225	-0.0246	-0.0239	-0.0250	-0.0250
Sum of Residuals		**0.3612**	**0.0222**	**0.0224**	**0.0482**	**0.0238**

Table 5: Estimated Power-Injections and Line-Flows for Case 2

	Actual	**LS**	**LAV**	**LMS**	**LTS**	**LMR**
P1	0.2500	0.2459	0.2409	0.2506	0.2430	0.2482
P2	0.1500	0.1450	0.1483	0.1495	0.1492	0.1488
P3	0.0750	0.0282	0.0755	0.0767	0.0751	0.0751
P4	0.0600	-0.0321	-0.0599	0.0641	0.0706	0.0610
P5	-0.0750	-0.0615	-0.0751	-0.0739	-0.0743	-0.0746
P6	-0.0750	-0.0732	-0.0744	-0.0756	-0.0754	-0.0739
P7	-0.0150	0.0259	0.0369	-0.0159	-0.0155	-0.0153
P8	0.0950	0.1157	0.0954	0.0943	0.0950	0.0949
P9	-0.0450	-0.0181	0.0143	-0.0464	-0.0457	-0.0457
P10	-0.0550	-0.0439	-0.0552	-0.0569	-0.0560	-0.0553

P11	-0.0800	-0.0717	-0.0631	-0.0809	-0.0803	-0.0803
P12	-0.0700	-0.0673	-0.0700	-0.0712	-0.0705	-0.0695
P13	-0.0900	-0.0838	-0.0909	-0.0904	-0.0913	-0.0904
P14	-0.1250	-0.1091	-0.1227	-0.1239	-0.1241	-0.1232
P1-2	0.0669	0.0735	0.0658	0.0670	0.0634	0.0663
P1-5	0.1831	0.1724	0.1751	0.1836	0.1797	0.1819
P2-3	0.0086	0.0305	0.0098	0.0077	0.0070	0.0081
P2-4	0.0922	0.0891	0.0951	0.0922	0.0892	0.0914
P2-5	0.1162	0.0989	0.1093	0.1166	0.1163	0.1156
P3-4	0.0836	0.0587	0.0853	0.0844	0.0822	0.0833
P4-5	0.0240	0.0097	0.0142	0.0244	0.0272	0.0242
P4-7	0.0439	-0.0118	-0.0087	0.0460	0.0451	0.0439
P4-9	0.1678	0.1179	0.1150	0.1703	0.1697	0.1676
P5-6	0.2483	0.2194	0.2234	0.2508	0.2489	0.2472
P6-11	0.0368	0.0254	0.0196	0.0380	0.0367	0.0373
P6-12	0.0688	0.0627	0.0665	0.0695	0.0691	0.0685
P6-13	0.0676	0.0582	0.0630	0.0678	0.0677	0.0674
P7-8	-0.0950	-0.1157	-0.0954	-0.0943	-0.0950	-0.0949
P7-9	0.1239	0.1297	0.1236	0.1243	0.1246	0.1236
P9-10	0.0982	0.0903	0.0987	0.0999	0.0995	0.0982
P9-14	0.1486	0.1393	0.1542	0.1483	0.1491	0.1472
P10-11	0.0432	0.0464	0.0435	0.0430	0.0436	0.0429
P12-13	-0.0012	-0.0045	-0.0035	-0.0017	-0.0014	-0.0011
P13-14	-0.0236	-0.0302	-0.0314	-0.0244	-0.0250	-0.0240
Sum of Residuals		**0.6133**	**0.4675**	**0.0341**	**0.0500**	**0.0189**

Table 6: Estimated Power-Injections and Line-Flows for Case 3

	Actual	**LS**	**LAV**	**LMS**	**LTS**	**LMR**
P1	0.2500	0.2394	0.2392	-0.2152	0.2392	0.2397
P2	0.1500	0.1411	0.1483	0.1483	0.1483	0.1488
P3	0.0750	-0.0755	-0.0755	-0.0755	-0.0755	-0.0750
P4	0.0600	-0.0372	-0.0599	-0.0599	-0.0599	-0.0594
P5	-0.0750	0.0729	0.0751	0.0751	0.0751	0.0756
P6	-0.0750	-0.0745	-0.0744	-0.0744	-0.0744	-0.0739
P7	-0.0150	0.0238	0.0493	-0.0148	0.1163	0.0484
P8	0.0950	0.1147	0.0954	0.0954	0.0954	0.0959
P9	-0.0450	-0.0203	0.0206	-0.0452	-0.0452	0.0170
P10	-0.0550	-0.0462	-0.0552	-0.0558	-0.0558	-0.0553
P11	-0.0800	-0.0765	-0.0792	-0.0798	-0.0798	-0.0794
P12	-0.0700	-0.0683	-0.0700	0.1247	-0.0700	-0.0697
P13	-0.0900	-0.0855	-0.0909	0.2999	-0.0909	-0.0904
P14	-0.1250	-0.1078	-0.1227	-0.1227	-0.1227	-0.1222
P1-2	0.0669	0.0916	0.0904	-0.1018	0.0895	0.0906
P1-5	0.1831	0.1478	0.1488	-0.1135	0.1497	0.1491
P2-3	0.0086	0.0840	0.0853	0.0446	0.0844	0.0853
P2-4	0.0922	0.0925	0.0950	0.0137	0.0932	0.0957
P2-5	0.1162	0.0562	0.0584	-0.0117	0.0602	0.0584
P3-4	0.0836	0.0085	0.0098	-0.0309	0.0089	0.0103
P4-5	0.0240	-0.0363	-0.0366	-0.0254	-0.0330	-0.0372
P4-7	0.0439	-0.0127	-0.0211	-0.0441	-0.0455	-0.0202
P4-9	0.1678	0.1129	0.1025	-0.0076	0.1207	0.1040
P5-6	0.2483	0.2406	0.2457	-0.0756	0.2520	0.2458
P6-11	0.0368	0.0360	0.0357	0.1029	0.0390	0.0365
P6-12	0.0688	0.0661	0.0685	-0.1259	0.0695	0.0684
P6-13	0.0676	0.0640	0.0670	-0.1271	0.0690	0.0670
P7-8	-0.0950	-0.1147	-0.0954	-0.0954	-0.0954	-0.0959
P7-9	0.1239	0.1257	0.1236	0.0365	0.1662	0.1241
P9-10	0.0982	0.0868	0.0987	0.0326	0.0966	0.0982
P9-14	0.1486	0.1316	0.1481	-0.0489	0.1451	0.1469
P10-11	0.0432	0.0405	0.0435	-0.0232	0.0408	0.0429
P12-13	-0.0012	-0.0022	-0.0015	-0.0012	-0.0005	-0.0013
P13-14	-0.0236	-0.0237	-0.0254	0.1717	-0.0223	-0.0247
Sum of Residuals		**1.0457**	**1.0368**	**4.0037**	**1.0870**	**1.0297**

Table 7: Estimated Power-Injections and Line-Flows for Case 4

	Actual	**LS**	**LAV**	**LMS**	**LTS**	**LMR**
P1	0.2500	0.2377	0.2472	0.2491	0.2436	0.2500
P2	0.1500	0.1395	0.1483	0.1490	0.1500	0.1478
P3	0.0750	0.0664	0.0755	0.0748	0.0759	0.0750
P4	0.0600	0.0433	0.0599	0.0607	0.0611	0.0604
P5	-0.0750	-0.0807	-0.0751	-0.0758	-0.0735	-0.0750
P6	-0.0750	-0.0879	-0.0744	-0.0751	-0.0752	-0.0739
P7	-0.0150	-0.0311	-0.0147	-0.0150	-0.0147	-0.0152
P8	0.0950	0.0873	0.0954	0.0953	0.0954	0.0954
P9	-0.0450	-0.0845	-0.0452	-0.0445	-0.0330	-0.0447
P10	-0.0550	-0.0742	-0.0558	-0.0551	-0.0552	-0.0553
P11	-0.0800	-0.0961	-0.0798	-0.0790	-0.0798	-0.0803
P12	-0.0700	-0.0782	-0.0700	-0.0699	-0.0704	-0.0695
P13	-0.0900	-0.1019	-0.0909	-0.0901	-0.0912	-0.0904
P14	-0.1250	0.0603	-0.1204	-0.1244	-0.1329	-0.1243
P1-2	0.0669	0.0646	0.0658	0.0665	0.0636	0.0673
P1-5	0.1831	0.1731	0.1814	0.1825	0.1800	0.1827
P2-3	0.0086	0.0097	0.0077	0.0083	0.0071	0.0082
P2-4	0.0922	0.0858	0.0909	0.0914	0.0901	0.0914
P2-5	0.1162	0.1086	0.1156	0.1160	0.1163	0.1155
P3-4	0.0836	0.0761	0.0832	0.0831	0.0830	0.0832
P4-5	0.0240	0.0227	0.0247	0.0246	0.0262	0.0241
P4-7	0.0439	0.0421	0.0429	0.0434	0.0424	0.0436
P4-9	0.1678	0.1404	0.1665	0.1671	0.1656	0.1673
P5-6	0.2483	0.2238	0.2466	0.2473	0.2490	0.2473
P6-11	0.0368	0.0534	0.0369	0.0361	0.0351	0.0372
P6-12	0.0688	0.0536	0.0684	0.0686	0.0697	0.0686
P6-13	0.0676	0.0289	0.0669	0.0674	0.0690	0.0676
P7-8	-0.0950	-0.0873	-0.0954	-0.0953	-0.0954	-0.0954
P7-9	0.1239	0.0983	0.1235	0.1237	0.1231	0.1237
P9-10	0.0982	0.1169	0.0987	0.0980	0.1000	0.0983
P9-14	0.1486	0.0373	0.1460	0.1483	0.1557	0.1480
P10-11	0.0432	0.0427	0.0429	0.0429	0.0447	0.0430
P12-13	-0.0012	-0.0247	-0.0016	-0.0012	-0.0007	-0.0010
P13-14	-0.0236	-0.0977	-0.0256	-0.0239	-0.0229	-0.0237
Sum of Residuals		**0.7924**	**0.0314**	**0.0148**	**0.0668**	**0.0139**

Table 8: Estimated Power-Injections and Line-Flows for Case 5

	Actual	LS	LAV	LMS	LTS	LMR
P1	0.2500	0.2384	0.2472	0.2506	0.2431	0.2495
P2	0.1500	0.1399	0.1483	0.1496	0.1507	0.1488
P3	0.0750	0.0663	0.0755	0.0753	0.0762	0.0750
P4	0.0600	0.0424	0.0599	0.0612	0.0615	0.0604
P5	-0.0750	-0.0788	-0.0751	-0.0738	-0.0732	-0.0756
P6	-0.0750	-0.0839	-0.0744	-0.0757	-0.0753	-0.0749
P7	-0.0150	-0.0324	-0.0147	-0.0153	-0.0144	-0.0153
P8	0.0950	0.0866	0.0954	0.0966	0.0956	0.0949
P9	-0.0450	-0.0882	-0.0452	-0.0440	-0.0446	-0.0447
P10	-0.0550	-0.0755	-0.0558	-0.0546	-0.0555	-0.0553
P11	-0.0800	-0.0965	-0.0798	-0.0810	-0.0800	-0.0803
P12	-0.0700	-0.0734	-0.0700	-0.0713	-0.0705	-0.0705
P13	-0.0900	-0.0914	-0.0909	-0.0921	-0.0915	-0.0914
P14	-0.1250	0.0466	-0.1204	-0.1255	-0.1220	-0.1208
P1-2	0.0669	0.0657	0.0658	0.0671	0.0635	0.0668
P1-5	0.1831	0.1728	0.1814	0.1835	0.1796	0.1827
P2-3	0.0086	0.0107	0.0077	0.0083	0.0072	0.0082
P2-4	0.0922	0.0877	0.0909	0.0919	0.0907	0.0914
P2-5	0.1162	0.1071	0.1156	0.1165	0.1162	0.1160
P3-4	0.0836	0.0770	0.0832	0.0836	0.0835	0.0832
P4-5	0.0240	0.0194	0.0247	0.0246	0.0254	0.0246
P4-7	0.0439	0.0445	0.0429	0.0436	0.0431	0.0436
P4-9	0.1678	0.1433	0.1665	0.1685	0.1673	0.1669
P5-6	0.2483	0.2204	0.2466	0.2507	0.2480	0.2477
P6-11	0.0368	0.0574	0.0369	0.0366	0.0365	0.0368
P6-12	0.0688	0.0508	0.0684	0.0699	0.0689	0.0688
P6-13	0.0676	0.0283	0.0669	0.0685	0.0673	0.0672
P7-8	-0.0950	-0.0866	-0.0954	-0.0966	-0.0956	-0.0949
P7-9	0.1239	0.0987	0.1235	0.1249	0.1242	0.1233
P9-10	0.0982	0.1147	0.0987	0.0990	0.0991	0.0988
P9-14	0.1486	0.0391	0.1460	0.1504	0.1478	0.1467
P10-11	0.0432	0.0392	0.0429	0.0444	0.0436	0.0435
P12-13	-0.0012	-0.0225	-0.0016	-0.0014	-0.0016	-0.0017
P13-14	-0.0236	-0.0856	-0.0256	-0.0249	-0.0258	-0.0259
Sum of Residuals		**0.7596**	**0.0314**	**0.0284**	**0.0387**	**0.0214**

Table 9: Estimated Power-Injections and Line-Flows for Case 6

	Actual	LS	LAV	LMS	LTS	LMR
P1	0.2500	0.2343	0.2392	0.2502	0.2436	0.2491
P2	0.1500	0.1341	0.1483	0.1493	0.1500	0.1488
P3	0.0750	0.0619	0.0742	0.0752	0.0759	0.0749
P4	0.0600	0.0350	0.0599	0.0610	0.0611	0.0640
P5	-0.0750	-0.0854	-0.0751	-0.0740	-0.0735	-0.0756
P6	-0.0750	-0.1095	-0.0744	-0.0754	-0.0750	-0.0749
P7	-0.0150	-0.0378	-0.0148	-0.0148	-0.0147	-0.0150
P8	0.0950	0.0839	0.0954	0.0963	0.0954	0.0953
P9	-0.0450	-0.0997	-0.0452	0.1469	0.1510	-0.1664
P10	-0.0550	-0.0859	-0.0558	-0.0552	-0.0552	-0.0553
P11	-0.0800	-0.1162	-0.1008	-0.0805	-0.0797	-0.0802
P12	-0.0700	-0.0973	-0.0700	-0.0700	-0.0702	-0.0700
P13	-0.0900	0.0442	-0.0072	0.0919	0.0909	-0.2169
P14	-0.1250	0.0384	-0.1738	-0.5008	-0.4994	0.1223
P1-2	0.0669	0.0664	0.0637	0.0668	0.0636	0.0663
P1-5	0.1831	0.1679	0.1755	0.1834	0.1800	0.1827
P2-3	0.0086	0.0124	0.0086	0.0081	0.0071	0.0079
P2-4	0.0922	0.0867	0.0915	0.0915	0.0901	0.0908
P2-5	0.1162	0.1014	0.1119	0.1166	0.1164	0.1164
P3-4	0.0836	0.0743	0.0828	0.0834	0.0830	0.0829
P4-5	0.0240	0.0147	0.0204	0.0251	0.0263	0.0256
P4-7	0.0439	0.0450	0.0444	0.0431	0.0424	0.0439
P4-9	0.1678	0.1362	0.1694	0.1677	0.1655	0.1681
P5-6	0.2483	0.1986	0.2328	0.2510	0.2492	0.2492
P6-11	0.0368	0.0804	0.0579	0.0359	0.0349	0.0363
P6-12	0.0688	0.0353	0.0568	0.0699	0.0699	0.0694
P6-13	0.0676	-0.0267	0.0436	0.0697	0.0695	0.0687
P7-8	-0.0950	-0.0839	-0.0954	-0.0963	-0.0954	-0.0953
P7-9	0.1239	0.0912	0.1250	0.1246	0.1231	0.1241
P9-10	0.0982	0.1217	0.0987	0.0997	0.1000	0.0992
P9-14	0.1486	0.0060	0.1505	0.3394	0.3395	0.0266
P10-11	0.0432	0.0358	0.0429	0.0445	0.0449	0.0439
P12-13	-0.0012	-0.0620	-0.0132	-0.0002	-0.0004	-0.0007
P13-14	-0.0236	-0.0444	0.0232	0.1614	0.1599	-0.1489
Sum of Residuals		**1.2062**	**0.3261**	**1.1477**	**1.1654**	**0.7622**

Table 10: Performance of the Five Estimators for All Cases

Case #	LS	LAV	LMS	LTS	LMR
1	0.3612	0.0222	0.0224	0.0482	0.0238
2	0.6133	0.4675	0.0341	0.0500	0.0189
3	1.0457	1.0368	4.0037	1.0870	1.0297
4	0.7924	0.0314	0.0148	0.0668	0.0139
5	0.7596	0.0314	0.0284	0.0387	0.0214
6	1.2062	0.3261	1.1477	1.1654	0.7622

Table 11: The Best Estimators that Rejected Bad-Data

Case #	LS	LAV	LMS	LTS	LMR
1	NOT	Rejected	Rejected	Rejected	Rejected
2	NOT	NOT	Rejected	Rejected	Rejected
3	NOT	NOT	NOT	NOT	NOT
4	NOT	Rejected	Rejected	Rejected	Rejected
5	NOT	Rejected	Rejected	Rejected	Rejected
6	NOT	NOT	NOT	NOT	NOT