Evaluation of power consumption in programming models based on map reduce in shared memory systems

Zahra khoshmanesh[1]
[1]CSE and IT Department, Shiraz University, Shiraz, Iran

Abstract **-** *One of the most common models in parallel programming is Map reduce. In companies which use Map reduce framework, at each time a lot of computers are executing Map and Reduce functions. These functions are executing many times and we can estimate the effect of any small changes in response speed or consumed power in each execution of the Map reduce model to be very high. On the other hand, power and energy became an important challenge in computer systems with high performance, so that criteria such as consumed Power are as important as performance criteria. Nowadays, because of the generated heat and also because of decrease in energy sources, saving the consumed power is very important. So, finding the parts of the programs which needs more power is of prime Importance, because by finding these parts, we can find the ways to investigate and improves them in terms of power. In this paper, we investigate the available Map Reduce framework and programs in this regard from the point of consumed power, which are implemented in multi-core environment with common memory.*

Keywords: *consumed power, map reduce, multicores, parallel programs, efficiency*

1 Introduction

According to Moore law [5], numbers of transistors on a chip are doubled each two years. As the size of the transistors is reduced, more of them can be placed on the chips, so we can have more cores on a chip. But the problem which arises is the consumed power. One transistor has a low consumed power by itself, but here we have a big number of transistors which cause high amount of the heat. On the other hand, transistors cannot be turned off completely, this means even if they are completely turned off, they will have current leakage, and some currents will pass through them, even when they are turned off. And this will cause gradual loose of energy and the power of connection wires will be too high. Most power saving mechanisms like doing the task slowly, will affect the performance. For two reasons, increasing the speed of the processors by increasing the frequency is not possible. These two challenges are memory wall and heat wall. Memory wall is related to difference in speed between memory and main processor and heat wall related to the fact that as the execution speed of the computer increases, consumed power which is proportional to cubic root of the frequency, also will increase and subsequently more heat will be generated [7]. To overcome these problems and to increase computation power of the system, parallel architecture was recommended among which multi-core architecture was designed as the practical way of overcoming problems. So, we have a number of cores on a processor and in order to be able to use the power of all cores, we should use multi-node programming. But the problem which arises is that common multi-node programming leaves the whole control to the user, and this is a disadvantage.

By increasing the use of information technology and popularity of the issues such as automation and use of digital equipment in business processes, in all cases we face data generation, nowadays we come across with the problem named data volume explosion. For example EBay Company has announced that, it has more than 6.5 petabyte data. This figure is 10 petabyte for Yahoo. The need to analyze the raw data for different uses in increasing which in turn demand appropriate and effective solutions for data analysis.

In 2004, Google introduced its programming model for use in environment with several processor units which is called Map reduce. This model which is inspired from functional programming model does all the operations related to passing, distributing and gathering data between computers and just demands the computing core of the program from user. In today world, in which we deal with a high amount of the data, this programming model is very useful. If we consider each computer in a distributed system as a core in multi-core processors, we come with the conclusion that models such as Map reduce is a good choice for use in this processing unit.

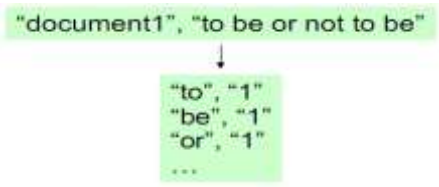
This model is a simple programming model which is used for solving computation problems in big scales and in distributing form. Map reduce was developed by Google in 2003 and is a software framework which provide a safe and scalable bed for development of distributing uses and is implemented in different languages.

In fact, it contains a set of library functions which hide the details and sophistication from the user. These details include: automatic paralleling of the tasks, data load balancing, optimization of network and disk transferring, management of faults in machines. Moreover, each improvement in library will be applied to all the places

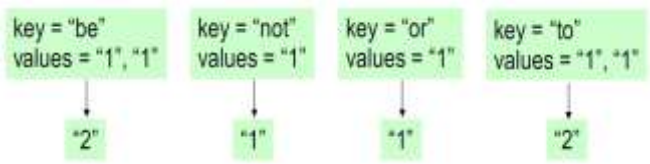
which this library has been used.In this method, two main steps exist: Map and Reduce.

Map step: main node, take the input and divides it into smaller sub-problems and then distribute them between nodes which are responsible for doing the tasks. It is possible that, this node repeats the task and if so, we would have a multi-surface architecture. Finally, these sub-problems are processed and response is sent to the output.

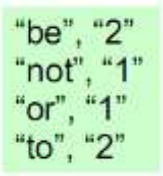
Reduce step: for generating an output, the responses and results which are received by main node, will be merged together. To do so, some operations like filtering and conversion may be applied on the data.
These two main operations are done on a regular pair (key, value). Map function, take a regular pair of the data and convert it to a list of regular pairs. Then, Map reduce framework, gather all the pairs with the same keys from all lists and produce a group. For each generated key, one group is produced. And the reduce function act on all groups. Now, map reduce framework, convert one list of (key, value) to a list of values.
As an example, a framework called Mars is designed for graphical processors for programs based on Map reduce. Also some programs which are designed and developed by Google based on the map reduce and focus on web based search programs for ordinary CPU are tested and implemented GPUs with high computation power and broad band widths.
One of the common examples for solving problems by Map reduce, is finding the number of occurrence of a word in a document. Here document referee to web page. In this problem, input is a file which has a text in each row. Map function takes (key, value) pairs. In this case, key is the address of the web page, and value is web content .[8] Output of the map function will be a list of other regular pairs :(number of occurrence, word) same as figure below:



Now map reduce framework gathers all the pairs with common keys. Then reduce function, merges the value of the pairs with common keys and assign a new value for that which in this case is sum of the values.



And finally, output will be like this:



Word count pseudo codes are shown in following:

```
map(String key, String value):
 // key: document name
 // value: document contents
 for each word w in value:
  EmitIntermediate(w, "1");
reduce(String key, Iterator values):
 // key: a word
 // values: a list of counts
 int result = 0;
 for each v in values:
  result += ParseInt(v);
 Emit(AsString(result));
```

Algorithm 1 .word count pseudo-code

Studies show that direct monthly energy costs and expenses for data center is about 23% of the total monthly operational costs [9]. If we consider costs like power distribution and cooling structure which affect the monthly energy costs indirectly, it will be about 42% of monthly operational costs. Trends show that performance of the processors increase each 18 month in terms of number of cores, while performance is doubled in each wat in each two years. [10]. so, it would not be surprising if a previous study has estimated that servers in USA include 3% of the total consumed energy in 2011. One reason for the high cost of the energy of the servers is that nodes in clusters environment are used 20 to 30% and the performance of the energy in this range is below 50%. This reveals the fact that 42% of monthly operational cost contribute to power, which decrease in it will increases the energy performance. [11]
Since map reduce framework can be used in computers in a data center and its power can be determinable, investigation and analysis of power is necessary in these environments. [12, 13]

2. Previous works

In this part a brief description of the works which have been done before, is presented. In [12] Map reduce programming model for systems with common memory called Phoenix is implemented. Creation of the nodes, dividing the data, dynamic work timing and fault tolerance between nodes of the processors are done automatically by Phoenix. In this paper, codes written by low level APIs such as P-thread were compared with those written by Map Reduce. Conclusion

was that performance of the Map Reduce programming model on systems with common memory is as good as simpler parallel codes. Despite, run time overloads, Phoenix yield the same performance results for most of the applicable programs. Obviously, there still exist programs which give better results in P-thread than in Map Reduce model.

In [8] a framework called Mars is designed for graphical processors for Map reduce based programs. Also some programs are designed and developed by Google based on the map reduce which focus on web based search programs for ordinary CPU. Are tested and implemented in this framework and for CPUs with higher computation power and higher band widths and then are compared with Phoenix which is a modern and updated Map Reduce framework on multi-core CPUs. Above mentioned framework hides the sophistications of GPU programming with map reduce interface. And finally, they came with 16 time faster execution of 6 common web programs compare to executions on a CPU with four cores.

In [14] is focused on power and energy for clusters which use Map Reduce programming model and propose techniques to decrease the consumed energy. This technique is turning off the nodes and attention goes towards the number of nodes to be chosen to be off and have direct impact on the consumed energy. Majority of the works which have been done in this paper is systematic consideration of different strategies for turning off the nodes in Map Reduce model and their impacts on total consumed energy and workloads response time. Two methods investigated are CS & AIS. In the first method some of the nodes with lower loads become off in low load period and the second method is turning off all the nodes in low use period which is proposed by the authors. These two methods are compared to each other and conclusion is that the second method which is proposed by them, give better results in both saving consumed energy and قresponse time which is shown by analytical models and laboratory results.

It is worth noting that in all the papers mentioned above just the performance aspect is taken into account and consumed energy and power are not considered at all and in [14] just the consumed power of Map Reduce programs in distributed environment is investigated. None of the papers dealt with consumed power of Map Reduce base programs in environments with common memory

3. Laboratory results

In this part first a brief explanation will be given about the framework in which the test is done and then the measurement results in multi-core environment with common memory related to Phoenix Map reduce will be presented.

3.1 The Phoenix System

Phoenix implements Map Reduce for shared-memory systems. Its goal is to support efficient execution on multiple cores without burdening the programmer with concurrency management. Phoenix consists of a simple API that is visible to application programmers and an efficient runtime that handles parallelization, resource management, and fault recovery. [12]

The current Phoenix implementation provides an application-programmer interface (API) for C and C++. The API includes two sets of functions. The first set is provided by Phoenix and is used by the programmer's application code to initialize the system and emit output pairs. The second set includes the functions that the programmer defines .Apart from the Map and Reduce functions; the user provides functions that partition the data before each step and a function that implements key comparison. The API is quite small compared to other models. The API is type agnostic. The function arguments are declared as void pointers wherever possible to provide flexibility in their declaration and fast use without conversion overhead.

The API guarantees that within a partition of the intermediate output, the pairs will be processed in key order. This makes it easier to produce a sorted final output which is often desired. There is no guarantee in the processing order of the original input during the Map stage.

3.2 Basic Operation and Control Flow

Figure 1 shows the basic data flow for the runtime system. The runtime is controlled by the scheduler, which is initiated by user code. The scheduler creates and manages the threads that run all Map and Reduce tasks. It also manages the buffers used for task communication. The programmer provides the scheduler with all the required data and function pointers through the scheduler args t structure.

After initialization, the scheduler determines the number of cores to use for this computation. For each core, it spawns a worker thread that is dynamically assigned some number of Map and Reduce tasks.

To start the Map stage, the scheduler uses the Splitter to divide input pairs into equally sized units to be processed by the Map tasks. The Splitter is called once per Map task and returns a pointer to the data the Map task will process.

The Map tasks are allocated dynamically to workers and each one emits intermediate <key, value> pairs. The Partition function splits the intermediate pairs into units for the Reduce tasks. The function ensures all values of the same key go to the same unit. Within each buffer, values are ordered by key to assist with the final sorting. At this point, the Map stage is over. The scheduler must wait for all Map tasks to complete before initiating the Reduce stage. [12]

Reduce tasks are also assigned to workers dynamically, similar to Map tasks. The one difference is that, while with Map tasks we have complete freedom in distributing pairs across tasks; with Reduce we must process all values for the same key in one task. Hence, the Reduce stage may exhibit higher imbalance across workers and dynamic scheduling is more important. The output of each Reduce task is already

sorted by key. As the last step, the final output from all tasks is merged into a single buffer, sorted by keys. The merging takes place in log2 (P/2) steps, where P is the number of workers used. While one can imagine cases where the output pairs do not have to be ordered, our current implementation always sorts the final output as it is also the case in Google's implementation [8].

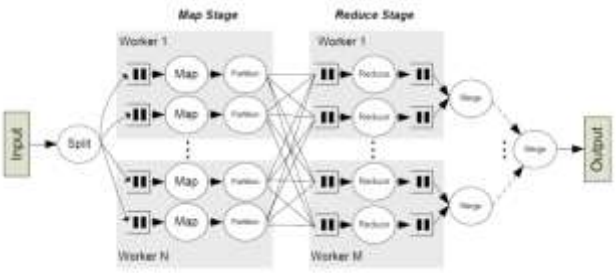


Fig. 1 the basic data flow for the phoenix runtime

For power consumer measurement, we use a power measurement device, applications of study with a brief description of them include:

Word Count: It counts the frequency of occurrence for each word in a set of files. The Map tasks process different sections of the input files and return intermediate data that consist of a word (key) and a value of 1 to indicate that the word was found. The Reduce tasks add up the values for each word (key).
Reverse Index: It traverses a set of HTML files, extracts all links, and compiles an index from links to files. Each Map task parses a collection of HTML files. For each link it finds, it outputs an intermediate pair with the link as the key and the file info as the value. The Reduce task combines all files referencing the same link into a single linked-list.

Matrix Multiply: Each Map task computes the results for a set of rows of the output matrix and returns the (x,y) location of each element as the key and the result of the computation as the value. The Reduce task is just the Identity function.

String Match: It processes two files: the "encrypt" file contains a set of encrypted words and a "keys" file contains a list of non-encrypted words. The goal is to encrypt the words in the "keys" file to determine which words were originally encrypted to generate the "encrypt file". Each Map task parses a portion of the "keys" file and returns a word in the "keys" file as the key and a flag to indicate whether it was a match as the value. The reduce task is just the identity function

.

KMeans: It implements the popular kmeans algorithm that groups a set of input data points into clusters. Since it is iterative, the Phoenix scheduler is called multiple times until it converges. In each iteration, the Map task takes in the existing mean vectors and a subset of the data points. It finds the distance between each point and each mean and assigns the point to the closest cluster. For each point, it emits the cluster id as the key and the data vector as the value. The Reduce task gathers all points with the same cluster-id, and finds their centroid (mean vector). It emits the cluster id as the key and the mean vector as the value.

PCA: It performs a portion of the Principal Component Analysis algorithm in order to find the mean vector and the covariance matrix of a set of data points. The data is presented in a matrix as a collection of column vectors. The algorithm uses two Map Reduce iterations. To find the mean, each Map task in the first iteration computes the mean for a set of rows and emits the row numbers as the keys, and the means as the values. In the second iteration, the Map task is assigned to compute a few elements in the required covariance matrix, and is provided with the data required to calculate the value of those elements. It emits the element row and column numbers as the key, and the covariance as the value. The Reduce task is the identity in both iterations.

Histogram: It analyzes a given bitmap image to compute the frequency of occurrence of a value in the 0-255 range for the RGB components of the pixels. The algorithm assigns different portions of the image to different Map tasks, which parse the image and insert the frequency of component occurrences into arrays. The reduce tasks sum up these numbers across all the portions.

Linear Regression: It computes the line that best fits a given set of coordinates in an input file. The algorithm assigns different portions of the file to different map tasks, which compute certain summary statistics like the sum of squares. The reduce tasks compute these statistics across the entire data set in order to finally determine the best fit line.[12]

3.3 Measurement of consumed power of the programs

In this part each program with different set of the data (small, medium or large) have been executed and the consumed power of the different phases including splitters, map, reduce, partition, sort and hash is measured, if available, and the results are shown in the following graphs.

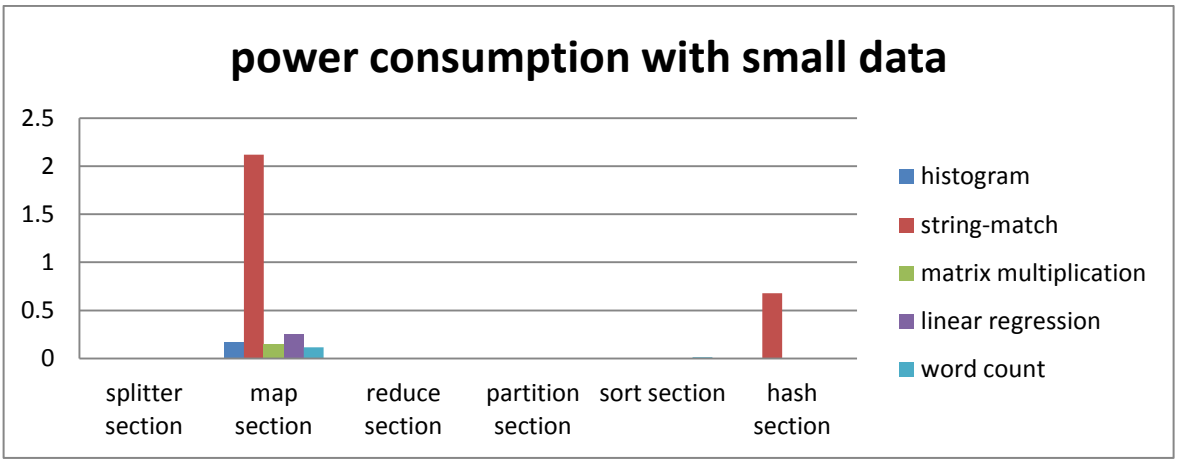


Fig .2 power consumption in small data

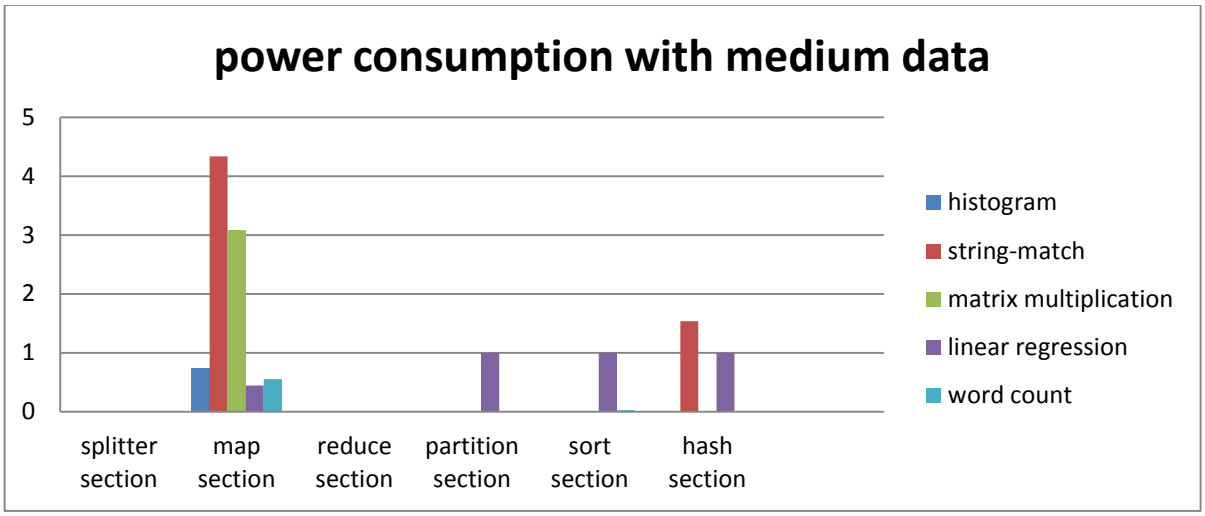


Fig .3 power consumption with medium data

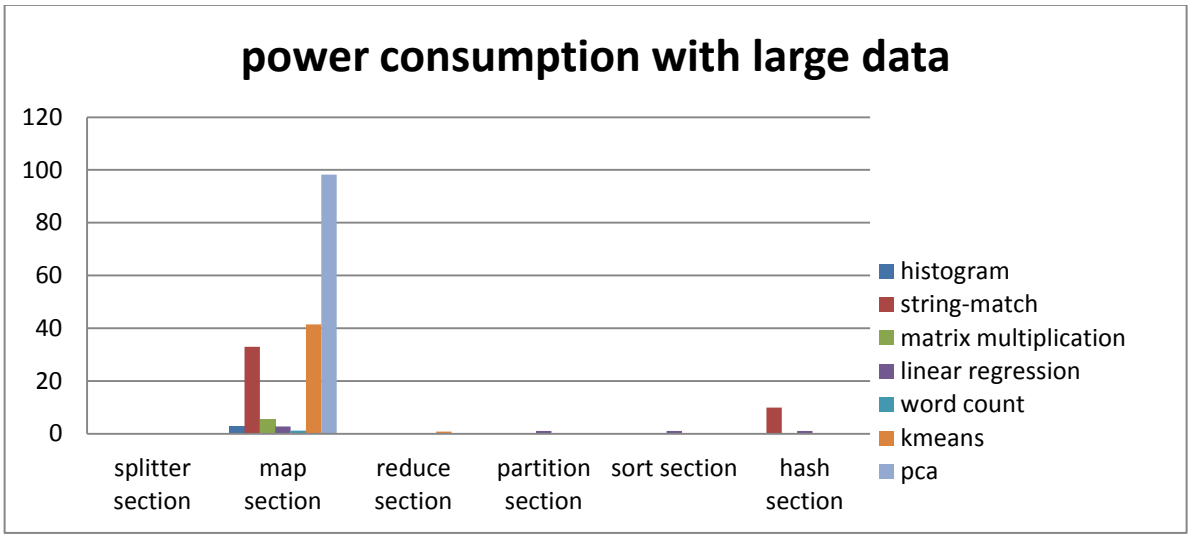


Fig .4 power consumption with large data

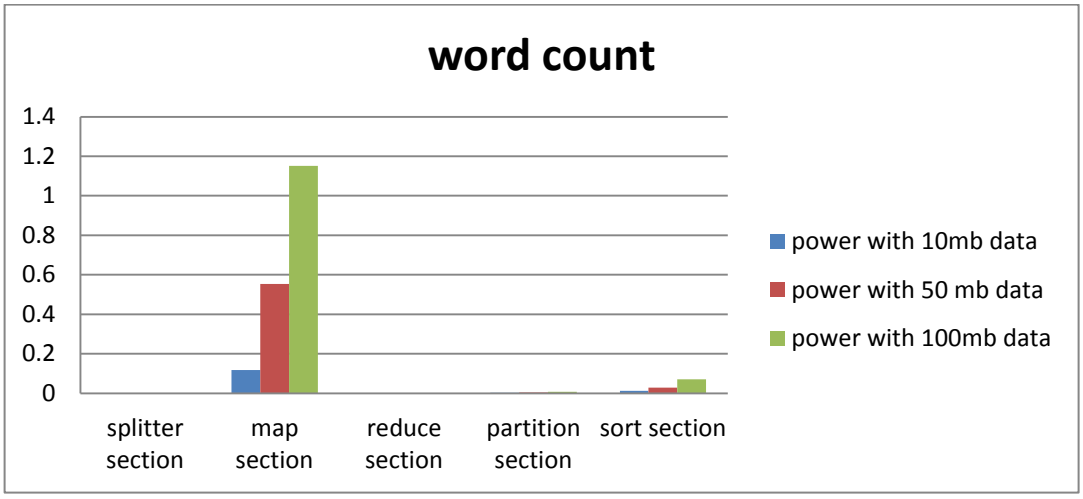


Fig.5 power consumption in word count application

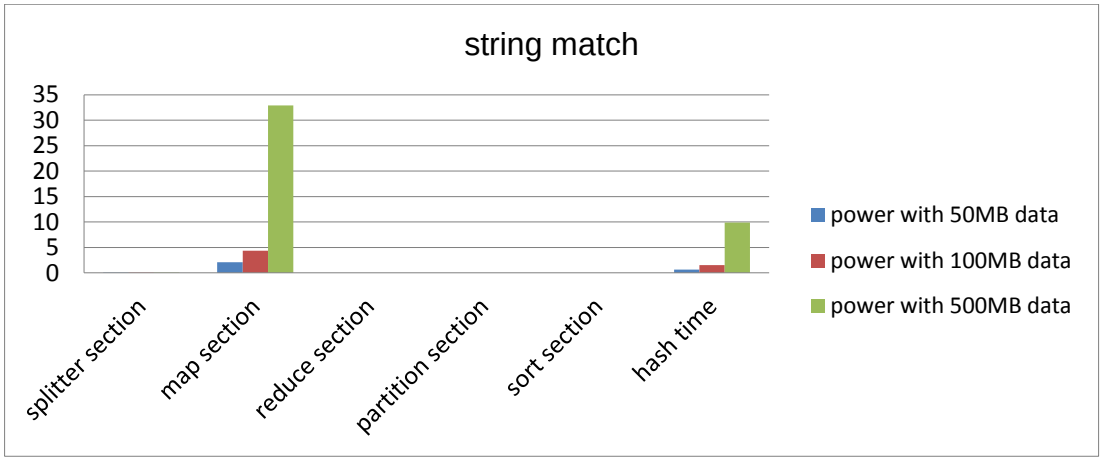


Fig .6 power consumption in string match application

First, data is divided into 3 categories: small, medium and large. And consumed power of each category for all programs under study is measured and recorded. Tables 2 to 4 show these measurements. To see the consumed power better, two more common and more important programs named "word count" and "string match" which cover all parts of the work are shown separately.

It is obvious from the above graphs and tables that most of the activities occurred in the map part and consumed power in map part of different programs is far from other phases of each program. In "word count" program most of the consumed energy is allocated to map. In sort part by changing the size of the data, by doubling the data, consumed energy increases more than two times, unexpectedly. So, it is expected that by increasing size of input data, consumed energy will paramount. In map part by doubling the size of the data, consumed energy also is doubled approximately.

In all programs, by increase in size of the input data, consumed power will be much more evident. In matrix multiplication program, by increasing the size of the data, calculation increased and the consumed power is fixed and by doubling size of the input, consumed power increased by less than 2 times.

In "string match" program, consumed of hash part is considerable and includes about ¼ of the total consumed energy. By increasing size of the input file, the consumed power increase non-linearly.

In programs which contain sort and hash, the rule of consumed power is more evident and by increasing input data, input of these tasks becomes more complicated. consequently, sub-programs of these operations should be

improved and better algorithms should be used in these operations.
In all programs consumed power of reduce part is not considerable. But it should be noted that most of the program either haven't reduce part or a little work is done in this part.

4. Conclusion

Today, consumed energy plays an important role in the world we are living in and we look for the cases which consume less energy. This role is of the same importance in computers and its programs. Moreover, volume of the data which should be processed is increasing such that we face explosion of the data.
Map reduce programming framework is used for processing these huge data. Different implementations of this framework in different environments are proposed.
One of these implementations is in multi-core environments with common memory. Here, measurements are done in this environment.
Based on the measurement following results obtained:
Most of the activities occurred in Map part and consumed power in Map part of different programs is much more than that of other phases of each program. In programs which include sort and hash parts, role of consumed power is more evident and by increasing input data, these tasks become more complicated. So, sub-programs of these operations should be improved and better algorithms should be used for these operations.

5. Future works

Measurement in Mars framework and comparison with Phoenix will be done in the future. And we will try to look for algorithms which improve the sort and hash functions and results of implementation of these algorithms will be investigated.

6. References

[1] J. Lo, J. Emer, H. Levy, R. Stamm, D. Tullsen, and S. Eggers. Converting Thread-Level Parallelism to Instruction-Level Parallelism via Simultaneous Multithreading. *ACM Transactions on Computer Systems*, 15(3):322–353, August 1997.

[2] NVIDIA Corporation. *NVIDIA CUDA Programming Guide*. 2009

[3] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. In Proceedings of the 6th conference on Symposium on Opearting Systems Design and Implementation , San Francisco, CA, 2004.

[4] Soyean.park, weihang.jiang, yuanyuan.zhou, sarita. Managing Energy –Performance Tradeoffs for Multithreaded Applications ON Multiprocessor Architectures adve*SIGMETRICS'07* June 12–16, 2007, San Diego, California, USA.

[5] G. E. Moore, "Cramming more components onto integrated circuits", Proc of Electronics, vol. 38, pp. 114-117, April 1965.

[6] W. A. Wulf, S. A. McKee, "Hitting the memory wall: implications of the obvious", ACM SIGARCH Computer Architecture News,vol. 23, pp.20-24, March 1995
` | `
[7] Sh. Ryoo, Ch. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk and W. W. Hwu, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA", Proc. ACM SIGPLAN Symposium on Principles and practice of parallel programming, February 20-23, 2008, Salt Lake City, UT, USA

[8] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. Mars: A mapreduce framework on graphics processors. In Proceedings of the Intl. Conference on Parallel Architectures and Compilation Techniques (PACT) ,Toronto, Canada, October 2008.
[9] C. Belady. In the Data Center, Power and Cooling Costs More than the IT Equipment it Supports. *Electronics Cooling*, 23(1), 2007.

[10] Report To Congress on Server and Data Center Energy Efficiency. In *U.S. EPATechnical Report*, 2007.

[11] L. A. Barroso and U. H¨olzle. The Case for Energy-Proportional Computing.*IEEE Computer*, 40(12), 2007.
[12] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, and Christos Kozyrakis. Evaluating MapReduce for multi-core and multiprocessor systems. In Proceedings of the 13th Intl. Symposium on High-Performance Computer architecture (HPCA) Phoenix,AZ, February 2007.
[13] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. Mars: A mapreduce framework on graphics processors. In Proceedings of the Intl. Conference on Parallel Architectures and Compilation Techniques (PACT) ,Toronto, Canada, October 2008.

[14]willis.lang , jignesh m.patel ,Energy Management for Map reduce Clusters *Proceedings of the VLDB Endowment*, Vol. 3, No. 1Copyright 2010 VLDB Endowment

[15] K. Heafield, "Introduction To Hadoop," Google Inc, 2008

[16]y.chen,a.ganapathi,a.fox,r.katz,david.patterson,Statistical Workload for Energy Efficient Mapreduce, Technical Report No.UCB/EECS-2010-6,January21, 2010(http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-6.html)

[17] Apache Software Foundation. JIRA issue MAPREDUCE-776.Gridmix:Trace-basedbenchmarkforMap/Reduce. https://issues.apache.org/jira/browse/MAPREDUCE-776.

[18] J. Leverich and C. Kozyrakis. On the Energy (In)efficiency of Hadoop Clusters. In *HotPower '09: Workshop on Power Aware Computing and Systems*, 2009.

[19] Google. Data center efficiency measurements. The Google Blog, 2009.