PH5WRAP: A Parallel Approach To Storage Server of Dicom Images

PDPTA

Tiago S. Soares*, Thiago C. Prado†, M.A.R Dantas‡, Douglas D.J. de Macedo§, Michael A. Bauer¶
*†‡*Informatics and Statistic Department*
Federal University of Santa Catarina
Florianópolis, Brazil
Email: steinmetz@telemedicina.ufsc.br, coeio@inf.ufsc.br, mario@inf.ufsc.br
§*Post-Graduate Program of Knowledge Engineering and Management*
Federal University of Santa Catarina
Florianópolis, Brazil
Email: macedo@inf.ufsc.br
¶*Department of Computer Sciencet*
University of Western Ontario
London, Canada
Email: bauer@uwo.ca

***Abstract*—Nowadays has been observed a large increase in the volume of images generated by medical devices. The manipulation of medical images has one known as DICOM standard that allows interoperability between these images in different devices. The server CyclopsDCMServer was developed in order to work with Hierarchical Data Format (HDF5) for manipulation of medical images (DICOM) on a distributed file system. This work aims to improve the performance of the server through an approach that uses the functions of reading and writing parallel. The experimental results indicate a gain of the proposed approach with respect to the original environment.**

***Keywords*-HDF5; MPI; PVFS; DICOM; Parallel I/O;**

I. Introduction

Databases alternative to conventional database for data storage has been a solution more sought by researchers and companies when it faces to large volumes of data in systems of high performance computing. Among the alternatives that are available and active development, currently the Hierarchical Data Format (HDF) has been highly seeking by researchers. HDF5 is the fifth version of this format, and was developed by the group "HDFGroup" from the University of Illinois in the United States. Since the release of HDF many general purpose and scientific began using it as an efficient alternative and high performance for storing and accessing large files. As examples, NASA use HDF to store data from global monitoring, clinical applications for managing large collections of images of X-rays and oil companies that store large amounts of data from 3D seismic surveys [1].

CyclopsDCMServer is a work being done by the group INCoD at Federal University of Santa Catarina in Brazil [2] and it aims to supply storage of DICOM images [3] provided by medical devices connected on *Rede Catarinense de Telemedicina* (RTCM) [4]. RTCM connects different health institutions such as hospitals and primary care that encompasses 286 cities in the state of Santa Catarina. This network provides service access to more than 10 DICOM modalities, among them are: electrocardiogram (ECG), magnetic resonance imaging (MRI), computed tomography (CT) and computed radiography (CR) [5]. For this integration occur between server and DICOM modalities, the RCTM use Picture Archiving and Communication System (PACS) [6] which includes hardware and software support for most medical equipment masking all part of communication, safety and accessibility. In Santa Catarina, the system is available almost over the state, and trend in few years cover the entire state.

A research mode of this server uses HDF5 as the data format to DICOM images and a distributed file system PVFS (Parallel Virtual File Systems), in order to focus on storage performance, DICOM image queries and solve the problem of scalability generated by storing vast amounts of files generated by PACS system. This work involves on a version of this server that includes the parallelization of the reading and writing DICOM files in HDF5. This study focuses on presents the architecture and experiments of reading and writing in parallel , with the goal of achieving better performance in access to the database and collect communication delays. This work has as motivated some parallel writing function tests performed in work [7], which demonstrated improvements in writing time.

This work is structured in six sessions. The next session begins with a base of how the CyclopsDCMServer works, and in section 3 presents the introduction of the approach of proposed architecture. In Section 4, we describe some related work. In section 5 depicts the hardware and software used in this project, the experiments and results. Finally, in Chapter 6, we ended the study with conclusions and future work.

II. CyclopsDCMServer

CyclopsDCMServer was developed in order to provide service integration of DICOM in PACS environment, in order to provide medical imaging storage and accessibility for manipulating these images through workstations. The application is multiplatform and can run on the Linux machine, Windows and other systems. Currently, in real system, the application CyclopsDCMServer stores all information provided by PACS system in a relational database called PostgreSQL. The DICOM files generated by available modalities on PACS system, generate files with different sizes, ranging from 300 Kbytes to 600Mbytes per image, and should remain stored on the server for at least 10 years. In measuring the increase in volume of data they generate problems such as scalability, latency in queries and maintenance costs.

Based on these problems described above, studies were carried out by Macedo et al [5] seeking to circumvent issues of telemedicine environments based in server with relational database systems. These issues considered are scalability, distribution of information, ability to use techniques for high system performance and operating costs. The result of this research led to a new server architecture based on PVFS and HDF5. Among the usual procedures of the current approach, the contribution was storing all the information hierarchically, such as organize and store data in HDF5 format. The second step was to use a cluster with distributed file system in order to seek high-performance disk access. Another contribution was to create an object called H5WRAP to handling with HDF5 interface.

A. H5WRAP

Due to lack of a procedure that incorporate a DICOM image in HDF5, it was necessary to create a library called H5WRAP, which converts a DICOM file in the format HDF5 data. The library contains an object that is used to create, find, collect and store information related to DICOM images in HDF5 files and this information is represented in Figure 1. Among various metadata contained in a DICOM file [8], for the H5WRAP, were collected only metadata that are importance to the PACS system used in Santa Catarina.

HDF5 files are organized in a hierarchical structure. It is observed that were selected three DICOM layers, each layer represent a group of study, serie and image, where the hospital attributes and patient studies are contained in the study layer. Their primary structures are two datasets and groups [1] . An HDF5 group is a grouping structure that can contain zero or more instances of groups or datasets, while an HDF5 dataset is a multidimensional array of data and also contains metadata that describes the dataset. In the next section, we explain how the read and write in parallel was built in H5WRAP

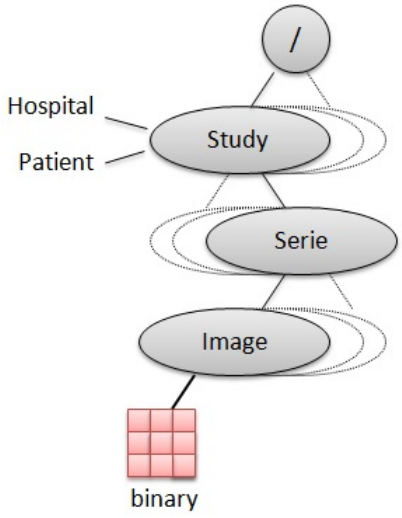

Figure 1. DICOM layers in HDF format

B. PH5WRAP

The PH5WRAP is an architecture oriented to work in reading and writing parallel only the binary. The reason for this restriction is due to the fact that the binary image representing on average 90% of a DICOM file, ignoring the need to parallelize small disk accesses to metadata, which could result in a longer process, due to communication necessary to distributed these layers. This architecture is designed to work in the same environment previously proposed in H5WRAP, based in PVFS and HDF5, with the addition of the Message Passing Interface (MPI).

The parallel access is provided by processes that are initiated at each node. The parallel access comes from the HDF5 API, which uses ROMIO interface to access the file system. ROMIO is a portable implementation of MPI-IO access which was created to provide high performance access to distributed file system. Its development is restricted to support some types of file system and among the most widely used, we have the PVFS and NFS [9]. In terms of characteristics of parallel access, the Parallel Virtual File Systems designed to support multiple access models, such as collective and independent access, as well as non-contiguous access patterns and structured. PVFS has three important structures: the metadata node, data node and client node. A node can represent all the structures simultaneously. The metadata nodes are controllers of permission file, directories and file names. Data nodes store physically data files and the client makes requests to the file system commands using POSIX or through APIs .

The Figure 2 illustrates the architecture of PH5WRAP, which illustrates the communication between the usual H5WRAP with the parallel application. The implementation of parallel application will be used by the server as it needs to perform a read operation or write a binary image DICOM.

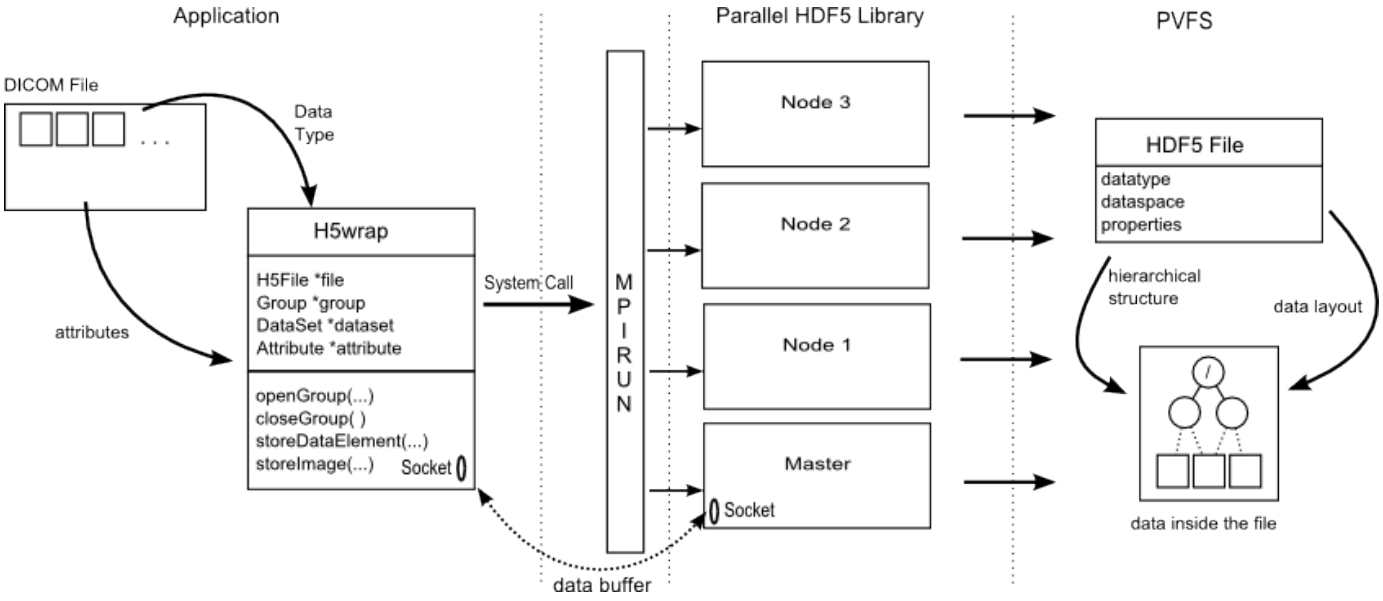

Figure 2. Ph5wrap architecture

Basically, the dicom file is unpacked to the data type and attributes, wrapping these contents into HDF5 format and throwing the binary image to parallel application. In this application, two main procedures are performed, the reading and writing functions. In case of Figure 3, it illustrates the division of binary image between the application processes when the master node receives the buffer from H5wrap. The division it's make simple, based on number of parallel nodes, which more nodes, smaller the buffer to distributed to other nodes.

The next two sections, it's describe the functionality of the both main function.

1) Write function: The write function begins with a request for some DICOM modality available on the PACS for storage an image in CyclopsDCMServer. As the object H5WRAP performs file analysis , it is create indexes in Clucene index table [10]. Before writing the metadata, it is checked whether the study groups, series and image metadata are already available in the HDF5 file, otherwise it is create the layers. Stored metadata, the server creates a connection through socket with the master process, sending the type of operation, the operation path and binary buffer, awaiting return the status of communication process. The Master process distributes the binary partition for nodes according to number of nodes in the system, and they store their respective buffer in system PVFS.

2) Read function: The process of reading function is similar to the writing process. Start with a client application consulting an image. This query is performed on server via Clucene index table where DICOMS images can be filtered by modalities study, series and images. The retrieve for an image is performed after the step above, that returns the

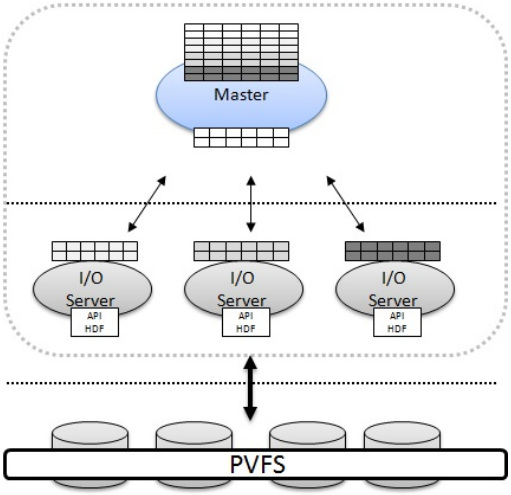

Figure 3. Shows the binary buffer distribution between nodes

object path that will be accessed in HDF5 file. The reading of metadata is performed by H5WRAP, while the binary image is performed by the parallel application. To performing the reading of the binary, H5WRAP create connection with master process, sending only the type of operation and location of the binary access. The Master node distributes to other nodes the location, which each node retrieves a region of binary. After reading their region, the nodes return a binary portion to master process. The Master, taken up all the regions, it sends the binary to H5WRAP. The H5WRAP ends the DICOM file, returning to the client.

III. Related Work

The first two works are themes that seek to resolve I/O performance issue of large amounts of stored data, using the parallel HDF5 library. The first related work [11], is very similar to ours where this work focus on the use of parallel I/O library for scalability problems involving many fermions dynamics for nuclear structure (MFDn). This study used a parallel version of HDF5 and performed tests on the independent and collective models. The results showed that files with size up to 20 GB, parallel HDF5 became more productive than sequential. The work [12] focuses in using parallel-IO for particle-based accelerator simulations that involved large amounts of data and dimensional arrays. They used two different API, the first is well-known MPI-IO and the second is the parallel HDF5 called H5part. He compared the simulations performance of read and write file operation between the API H5part, MPI-IO and the primary (file per process). HDF5 showed good performance in writing, although MPI-IO showed better results in both operations.

The works [13] [14] [15] are interesting because they focused in working with comparisons and performance improvements of I/O in platforms such as Jaguar and Red Storm. These are supercomputing platforms, which provide computer services, such as GTC for spectrum, Parallel Ocean Program (POP) ocean modeling, physics calculations program and others. The work proposed by Yu et al has some interesting comparisons with different types of I/O. They propose to improve and enhance access in a parallel application called Jaguar of Oak Ridge National Laboratory and aimed at several objectives such as, the characterization of performance parallel-IO on storage unit and scalability of the entire system. The results showed a better performance and less scalability in parallel-IO in collective mode and showed that using technique called FLASH I/O, the performance can improve by 34% with certain adjustments on collective I/O .

About the file systems, the work [16] conducted three experiments in an atmospheric model system, distinguishing the access mode on disk. The experiments deal with applying a single application with a differential in system architecture that adds Threads on its processes. The written tests with local disk had the best performance by not requiring communication between the nodes of the file system. The environment with threads had a better performance when used the parallel virtual file system (PVFS), because it has facilities when use MPI interface. The experiments were performed using 30 clients in one PVFS environment with four nodes. As each process accesses the file system constantly to write the atmospheric results, when you have more than one process per machine, they compete for access disk, reducing the execution time. This does not occur with the application threads, since it runs one process per machine and performs better memory used.

IV. Environment And Experimental Results

The environment used for the experiments is a virtualized with eight virtual machines, both PVFS nodes and nodes of the parallel application. Another virtual machine is used to host the server CyclopsDCMServer. For virtualization platform environment, we used VMware vSphere 5. The host machine has an Intel Xeon E5310 with 4 cores at 1.6 GHz each and a total of 10Gbytes of memory and 460 Gbytes of disk. Each virtual machine has virtualized 1077 Gbytes of memory and 23:26 Gbyte disk.

To assemble the virtual cluster, it was necessary to set up an environment with PVFS, MPICH2and HDF5, and their arguments are meant for installation work with parallel features. Each of the parallel application process will also be a PVFS data storage node. We use the same number of PVFS data nodes as parallel application processes in order to use the maximum use of the environment. The graphs bellow shows the number of attempts on y axis and time spent in seconds on x axis to complete the experiments.

In reading experiments were total conducted 24 experiments, 16 collecting only the read time of binary from two DICOM images in HDF file, discarding the communication delay and assembly of the DICOM file. The other eight experiments represent the total value of reading the images, plus the assembly of the DICOM file. The first binary image has 92.16Mbytes while the second has 52.42 Mbytes. Experiments were performed with the serial server, three more experiments with 2, 4 and 8 parallel nodes. These experiments were performed twice, resulting in a total of eight experiments per image. Each reading experiment was repeated 25 times on the same image. From these results, it is collected the average and presented in the comparison chart of the averages in Figure 4.

It can be observed in the first average graph of image 1 on the first attempt, a difference up to 6 milliseconds of average serial reading with 4 parallel nodes. In the second attempt, the parallel reading with 8 nodes have underperformed with a difference of approximately 7 milliseconds compared to the serial reading and near twice worse compared with 4 nodes. The same can be observed in image 2, although the results are better than eight nodes serial reading, the difference between the applications with four nodes is almost twice. Finally, the Figura 6 graph is a noteworthy the impact of the runtime application when used CyclopsDCMServer with 8 nodes.

One important factor that causes the loss of performance of parallel application with 8 nodes in the three graphs is due to be working with one processor with four cores and using 8 vitual machines, running one process. This process will disputed to use one core, therefore, others have to wait. This causes saturation in processor use, rather than working with a virtual machine requires a high processing power. This

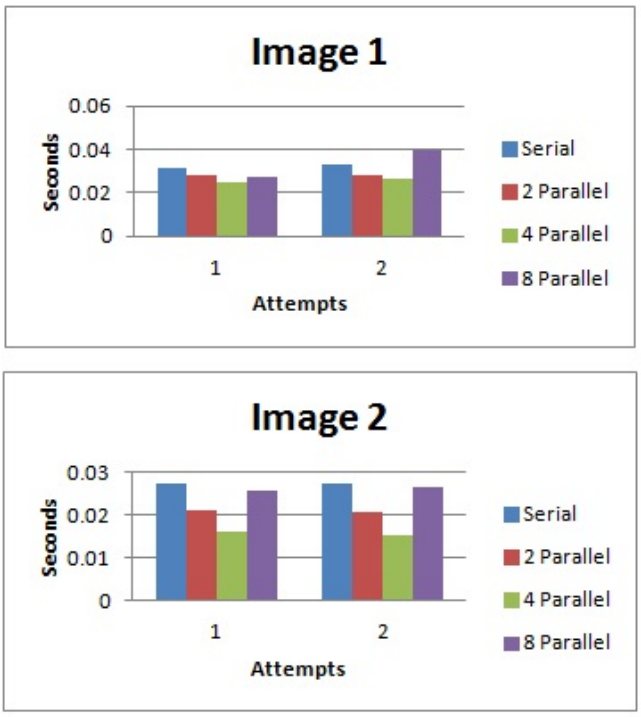

Figure 4. Reading comparisons graphics

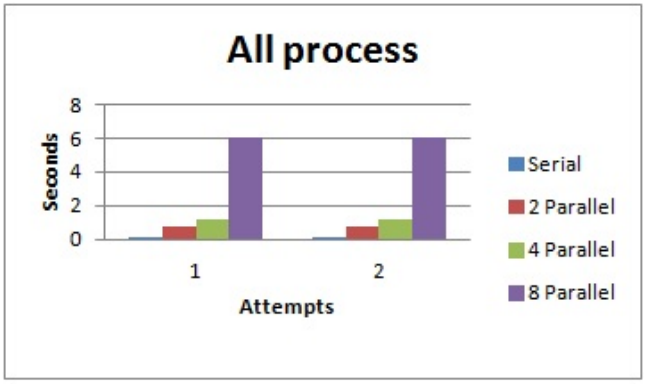

Figure 5. Server runtime

conclusion can be proved also by the fact that when we use only four nodes, no process have to wait, the performance is approximately 2 times faster. This underperformance provide by 8 nodes is also observed in Figures 5.

The next figures illustrate two graphs of average for written experiments performed with about 1000 DICOM images each attempts. These images are sent to the server CyclopsDCMServer, which is responsible to communicate with parallel application.

In Figure 6 shows the average time of writing an image in the HDF. For this first graph, we collected the time of writing only, discarding the time of the system. Among the two attempts, there is the bottleneck described above

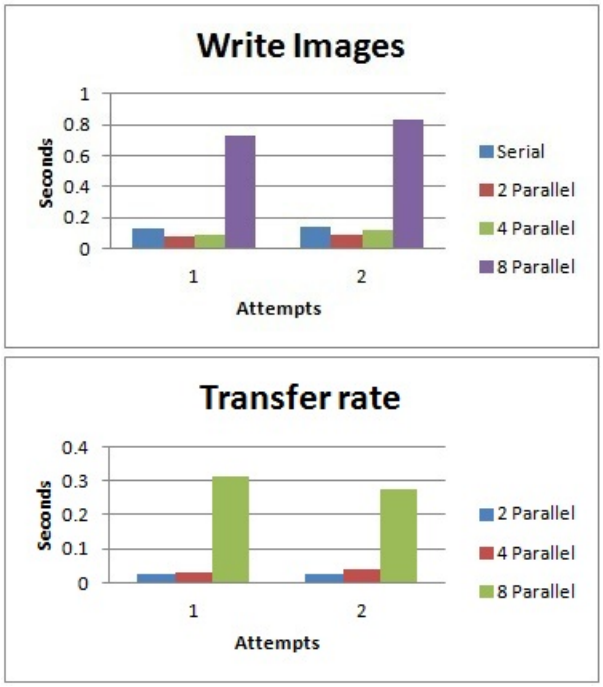

Figure 6. Written comparisons graphics and graph of transfer rate

about the 8 nodes and performance gain of 2 and 4 nodes. This gain is around 50 milliseconds and 33 respectively compared with serial mode. The transfer rate is shown in the second graph. This transfer rate is the measurement of the binary transmission from the server to the MASTER of the parallel application. Unlike the total time of the system, the transfer rate does not take into account other communications between the MASTER and the server, as well as communication between the application processes. Note that the speed of data communication is directly interconnected to the number of processes in the system.

V. Conclusion And Future Work

This work was done in order to show new results for the parallel architecture introduced earlier in [7], in order to detect failures and performance in reading and writing parallel of a DICOM image in HDF format. The environment is virtualized and configured with parallel virtual file system (PVFS) and HDF5, where more tests were realized to collect more consistent results.

The results showed a gain in reading and writing parallel, but the number of nodes exceeds the number of processing from the host machine, the performance of the architecture is low. Other problem in this system is the poor time of runtime system, provided by the communication between the server and parallel approach to transfer binary buffer. One solution to overcome the time problem is to move the H5wrap object to the master process, with goal to remove the necessity of binary transfer.

As future work, we intend that run parallel architecture in a real environment and dedicated to collect results that do not depend mainly on the processing power. Analyze the performance of system access in accordance with increasing the number of nodes. Finally, more tests with different types of DICOM modalities, mainly work with larger files.

REFERENCES

[1] HDFGroup, Access: January, January 2011. [Online]. Available: http://www.hdfgroup.org

[2] "Romio: A high-performance, portable mpi-io implementation," Access: December. 2010, December. [Online]. Available: http://cyclops.telemedicina.ufsc.br

[3] O. Pianykh, *Digital Imaging and Communications in Medicine (DICOM): A practical introduction and survival guide*. Springer Verlag, 2008.

[4] J. Wallauer, A. von Wangenheim, R. Andrade, and D. De Macedo, "A telemedicine network using secure techniques and intelligent user access control," in *21st IEEE International Symposium on Computer-Based Medical Systems*. IEEE, 2008, pp. 105–107.

[5] D. De Macedo, A. Von Wangenheim, M. Dantas, and H. Perantunes, "An architecture for dicom medical images storage and retrieval adopting distributed file systems," *International Journal of High Performance Systems Architecture*, vol. 2, no. 2, pp. 99–106, 2009.

[6] T. Cradduck, "Pacs: Basic principles and applications," *Physics in Medicine and Biology*, vol. 45, p. 2444, 2000.

[7] T. Soares, D. de Macedo, M. Bauer, and M. Dantas, "A parallel architecture using hdf for storing dicom medical images on distributed file systems."

[8] "Digital imaging and communications in medicine," Access: December. 2011, December. [Online]. Available: http://en.wikipedia.org/wiki/DICOM

[9] M. Division, "Romio: A high-performance, portable mpi-io implementation," Access: March. 2011. [Online]. Available: http://www.mcs.anl.gov/research/projects/romio/

[10] Sourceforge, "Clucene," Access: December, December 2011. [Online]. Available: http://clucene.sourceforge.net/

[11] N. Laghave, M. Sosonkina, P. Maris, and J. Vary, "Benefits of parallel i/o in ab initio nuclear physics calculations," *Computational Science–ICCS 2009*, pp. 84–93, 2009.

[12] A. Adelmann, R. Ryne, J. Shalf, and C. Siegerist, "H5part: A portable high performance parallel data interface for particle simulations," in *Particle Accelerator Conference, 2005. PAC 2005. Proceedings of the*. IEEE, 2006, pp. 4129–4131.

[13] M. Fahey, J. Larkin, and J. Adams, "I/o performance on a massively parallel cray xt3/xt4," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–12.

[14] J. Laros, L. Ward, R. Klundt, S. Kelly, J. Tomkins, and B. Kellogg, "Red storm io performance analysis," in *Cluster Computing, 2007 IEEE International Conference on*. IEEE, 2007, pp. 50–57.

[15] W. Yu, J. Vetter, and H. Oral, "Performance characterization and optimization of parallel i/o on the cray xt," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–11.

[16] F. Boito, R. Kassick, L. Pilla, N. Barbieri, C. Schepke, P. Navaux, N. Maillard, Y. Denneulin, C. Osthoff, P. Grunmann *et al.*, "I/o performance of a large atmospheric model using pvfs," *Rencontres francophones du Parallélisme (RenPar20)*, 2011.