An Approach Using Parallel Architecture to Storage DICOM Images in Distributed File System

Tiago S Soares, Thiago C Prado, M A R Dantas
Informatics and Statistic Department
Federal University of Santa Catarina
Florianopolis, Brazil
E-mail: steinmetz@telemedicina.ufsc.br, coeio@inf.ufsc.br, mario@inf.ufsc.br

Douglas D J de Macedo
Post-Graduate Program of Knowledge Engineering and Management
Federal University of Santa Catarina
Florianopolis, Brazil
E-mail: macedo@inf.ufsc.br

Michael A Bauer
Department of Computer Science
University of Western Ontario, UWO
London, Canada
E-mail: bauer@uwo.ca

Abstract. Telemedicine is a very important area in medical field that is expanding daily motivated by many researchers interested in improving medical applications. In Brazil was started in 2005, in the State of Santa Catarina has a developed server called the CyclopsDCMServer, which the purpose to embrace the HDF for the manipulation of medical images (DICOM) using a distributed file system. Since then, many researches were initiated in order to seek better performance. Our approach for this server represents an additional parallel implementation in I/O operations since HDF version 5 has an essential feature for our work which supports parallel I/O, based upon the MPI paradigm. Early experiments using four parallel nodes, provide good performance when compare to the serial HDF implemented in the CyclopsDCMServer.

1. Introduction
Since the HDF release many scientific and general purpose applications facing difficulties when storing large amounts of data started to use it as an efficient alternative to store and access large files. HDF is used from clinical applications for managing large collections of X-ray images to oil companies when storing and accessing large 3D seismic soundings. Developed by HDF Group’s of University of Illinois, the primary goal of HDF project was accessibility over multi-platforms, later other goals like access ability, efficient storing of large objects and data viewing tools [1].
In Brazil, the Federal University Of Santa Catarina’s Telemedicine Laboratory (LABTELEMED) [4] have two main projects, the first is called CyclopsDCMServer which is a DICOM medical image facility developed by The Cyclops Group [6] to provide DICOM image storage and access by using wide area networks (WAN) and the second one is a large-scale telemedicine network called Santa Catarina Telemedicine Network (RCTM - Acronyms for "Rede Catarinense de Telemedicina" in Portuguese language) [2]. RCTM connects different health institutions such as hospitals and basic care instalations encompassing 286 municipalities in the state of Santa Catarina. This network provide access for more than 10 DICOM modalities like electrocardiograms (ECG), magnetic resonance (MRI), computed tomography (CT) and computed radiography (CR) [3].

The RCTM network uses a Picture Archiving and Communication Systems (PACS) proper developed to aquire all patient’s clinical information such as DICOM images and personal data to the CyclopsDCMServer. Figure 1 show the PACS distribution in Santa Catarina state. A DICOM based PACS consist in hardware and software designed for medical use and is supported by major medical imaging equipment manufacturers. It embraces digital image acquisition devices, digital image archives and workstations. Its basic function starts when the image is aquired by the proper equipment (e.g. Tomograph) the DICOM server provides segmentation service over the incoming information, processing and storing the images in a centralized database.

![Figure 1. PACS Network.](image)

Currently the CyclopsDCMServer stores all PACS information in a PostgreSQL object-relational database with an aproximate amount of data of 50 terabytes [3]. Such magnitude leads to problems like scalability, information distribution, operational cost and ability to use high performance system techniques. To overcome these issues a new server architecture based on PVFS and HDF5 was conceived. Parallel virtual file system (PVFS) is a distributed file systems designed to scale petabytes of data and provide high access rates [5], and Hierarchical Data Format 5 is a data model to handle high volume and complex data. This paper is structured in background with a summary of the CyclopsDCMServer application. The next two sections introduces the DICOM standard and the HDF5 definitions while section 4 describe our methodology follow by obtained results, our conclusions and future work.

2. Background
The CyclopsDCMServer was developed for both scenarios, medical image storage and workstation manipulation, currently it supports DICOM based PACS equipments distributed over hospitals and radiology clinics and communication between the server and medical equipment is performed through TCP/IP. The server aims at storage and retrieval of indexed DICOM files which are
store in a relational PostgreSQL database. In addition for CyclopsDCMServer, a new research started in 2008, to explore new architecture for DICOM images using a distributed file systems. The goal is to address issues of telemedicine environments based on relational database systems such as scalability, information distribution, ability to use high performance system techniques and operational costs. Among the usual procedures to avoid the scalability issue, the project design to use high performance distributed systems, like clusters and grids [3]. The first approach distinguish from the usual system is to store all information hierarchically, such as organize and store in HDF5 data format. The second step was to use a distributed file system to keep the HDF file distributes across the nodes.

Another important contribution is the HDF5 Wrapper Library (H5WL). It was created to supply the scarcity of DICOM applications, once usually only support drivers for standard DBMSs. The library contains a wrapper object which is used to create, locate, collect and store information related to DICOM images on HDF5 files. The hierarchical data structured is organized is based on DICOM information hierarchy [7] and an example of the structure is shown in Figure 2, where the data is organized around six layers: root, hospital, patient, study, series and image.

When CyclopsDCMServer receive a DICOM file, the hierarchy is created through H5WL methods, creating groups that provide information to identify an image inside the file. It is important to note that image (p. e. tomography picture) represents almost the full size of a DICOM file and normally represents as the last hierarchical structure. Basically, the image structure is represented in binary and can easily been converted to usual image like JPEG [7] and representing the compress DICOM image.

Based on 25 experiences with the system, it showed an average improvement in storage of about 16% when compared with the usual system using DBMSs. However, in term of retrieval operations, there was a drop in performance, with average decrease in performance of around 21%. This was due to mechanisms used to provide similar behavior to that of standard DBMSs when retrieving information. This paper addresses this problem by proposing an extension to the architecture to take more advantage of a parallel environment. This architecture is detailed in Section VI.

2.1. DICOM

The main proposed of Digital Imaging and Communications in Medicine (DICOM 3.0) is to standard the medical digital images as visualization, storage and impression. Previously was known as ACR_NEMA Standards Publication PS3 and created in 1983 by the American College of Radiology, conceiving a unique standard for medical devices and facilitating the expansion of digital images [7]. Before this standard was established, each manufacture created their own solution for visualization, storage and impression of digital images, resulting in incompatibility between medical devices.
Completed in September 1992, DICOM third version came with major revision, supplying increasing variety of digital devices and their communications protocols. This version was called DICOM 3.0, as it followed two earlier ACRNEMA editions. Taking many important features from earlier and other standards, the early versions of focused on the improvement and correction of some issues, and where those publications provided specifications related to hardware interfaces, it introduced a set of data format and commands for software packages. Nowadays the standard is reviewed annually and updated with new supplements if necessary [7]. Our current CyclopsDCMServer supports eight DICOM modalities: computed radiography; computed tomography; magnetic resonance; nuclear medicine; ultrasound; X-ray angiography; electrocardiograms; DICOM structured reporting [8].

2.2. HDF5
The HDF group at the University of Illinois, initially in the 90s, developed a new product aiming to work with large scientific data, including high performance data manipulation supporting random access, number encoding in native format, data compression, individual data set encryption, and storage strategies for parallel I/O and multidimensional data structures; the current version is HDF5. There is two important feature of HDF. First is that files can contain binary data as multi-dimensional arrays and allow direct access to parts of the file without first parsing the entire contents [1].

Second feature is supporting for standard parallel I/O interfaces. Known as The Parallel Hierarchical Data Format 5 (Parallel HDF5), for work with it is required MPI/IO interface, which is supported through MPICH ROMIO [9]. Now days, ROMIO has support to most common file system, but only offer bracing to PVFS as distributed file system. The purpose of Parallel HDF5 is to make it easy for users to use the library, providing compatibility with serial HDF5 file. One approach is to read and write data by hyperslab [1], i.e., a multidimensional array that can be spread by rows, columns, patterns and chunks, and a hyperslab selection could be a logically contiguous collection of points, or it can be a regular pattern of points or blocks, depending on the type used.

When works with parallel I/O, the properties of communication that will realize the I/O operation is important to synchronise the nodes. The Parallel HDF5 library has available two types of properties (collective and independent data access) and four hyperslab model (Contiguous Hyperslab, Regularly Spaced Data, Pattern and Chunk) [1]. In HDF5 there are two essential structures which form the base for the library: dataset and group. Dataset is a multi-dimensional array of datatype; HDF stores and organize all kinds of data from atomic to

**Figure 3.** Example of DICOM imag...
composed types, similar to the C structure construct. Special array operations, such as chunks, compression and extendibility are available through the HDF library and can be applied to a dataset. Other important structure is the dataspace. Through a dataspace is possible to require components of dataset or even an attributes are defined, as well as array ranks, sizes and types. The group is similar to UNIX directories, though cycles are allowed. Every file is started with a root group, represented as /, and could be followed by the name of another group or a dataset as show in figure 2.

3. Related work
The works related are suchlike to our work, concerning with the use of parallel I/O as a solution for I/O bottlenecks access for large amounts of stored data. The first related work presented in Nikhil Laghave [10], is very similar to our work. This work is focused on the use of a parallel I/O library for scalability issues involving fermion dynamics for nuclear structure (MFDn). This work used the HDF5 parallel version for parallel I/O, testing with collective and independent models. As result once file sizes increasing above 20 GB, parallel HDF5 becomes more cost-effective than sequential I/O for sufficiently large datasets than sequential binary I/O.

The work of A. Adelmann [11] focused on using parallel I/O for particle-based accelerator simulation which involved vast quantities of data and dimensional arrays. He used parallel I/O performance for MPI code as well parallel HD5 by developed API call H5part. H5part is a portable high performance parallel data interface for particle simulation. He compared read and write performance in simulations between H5part, mpi-io and one file per process. HDF5 showed good performance in writing, though mpi-io showed better results.

Finally, H. Yu [12] even not using parallel HDF as solution, he demonstrated an interesting work which dealt with large earthquake simulations. He faced scalability issue and I/o bottleneck, due earthquake simulation required large file to storage. To solve his problem, he developed an application using parallel I/O strategies through MPI I/O to address his needs. The results was considerable to remove the I/O bottleneck and also hide pre-processing costs.

4. The parallel architecture
Our work comes to propose a new architecture for Macedos approach, in order to avoid I/O bottlenecks and get better performance using parallel data access to HDF files stored in the PVFS distributed file system. To accomplish our work, we configured MPI environment to work with Parallel HDF5. As cited in session 2.3, the Parallel HDF5 library requires a parallel MPI/IO interface through ROMIO and when working with MPI, it is necessary to design it to running in a cluster environment. It is important to note the requirement to use the mpirun shell script to run any MPI application, which attempts to hide the differences in starting jobs for various devices from the user [13]. We created an additional procedure to work with the CyclopsDCMServer. This procedure should be called every time when is required to retrieve or store some medical information.

For now, we build an application concerning with I/O access, which our application just responsible to direct access a file for reading and writing a dataset. Figure 4 illustrates how the architecture works. It was crucial to modify the H5WL functionality. Instead of having the H5WL responsible for reading and writing the binary information created by PACS, it will be treated as a new parallel application. When some client requests to store a DICOM files, it necessary to initiate the parallel application by calling mpirun shell script. After, all communication between them will be made by socket connections.

The communication is done by the master process (represented by MPI process zero) and H5WL, which the messages is represent function parameters, like the location that is the target of an operation (group path), the image buffer and the number of MPI processes. In write functions, per example, the H5WL will first receive the DICOM file, create a new hierarchy
of the image based on DICOM file layers, get the path location for new image (JPEG image) and then call `mpirun` procedure to start the MPI application. The master process will delegate the communication with H5WL to retrieve the function to perform, get the location (group) of image in the HDF5 structure, the arguments for the job and the stream of images to be stored. Receiving the stream, the master node has to distribute the memory buffer to each process in small buffer equal to number of nodes. Finally, once the jobs have executed, the main process will return to the wrapper the status of reading or writing the buffer.

5. Experiments
Our experiments are based on the Parallel HDF5 architecture, adapted to use PVFS, MPI and sequential CyclopsDCMServer. The parallel environment consists in four node cluster, as specified in Table 1 and one dedicated computer for DICOM server. Unfortunately, the environment is non-dedicated cluster and belongs to Telemedicine Laboratory and share a connection network 100 Mbs Ethernet. The operating system installed on all nodes is CentOS with kernel 2.6.32, and there only one metadata PVFS node. Each node has a PVFS client for access to the PVFS file system and MPI compiled. It is important to note that our results do not take into consideration external factors, like computer users, using the same network.

| Name   | CPU             | Memory | HD    |
|--------|-----------------|--------|-------|
| Node1  | AMD athlon x2 2.1 GHz | 2 Gb   | 20 Gb |
| Node2  | AMD athlon x2 2.8 GHz | 3 Gb   | 20 Gb |
| Node3  | Intel PentiumR Dual 1.80 GHz | 1 Gb   | 20 Gb |
| Node4  | Intel Core i5 3.2 GHz | 3 Gb   | 20 Gb |

All the experiments were conducted with CyclopsDCMServer sequential and parallel architecture and involve only comparison of storing a new DICOM file in HDF file, discarding all delay from communication between server and master node; future comparisons will compare...
file retrieval. This experiment measures the time spent write an image buffer into a HDF file repeating 25 times with different DICOM files for each test. The selection of the files used is the same files were used for the both experiments. The time collected is the time required to write an image, ignoring sequential information like patient name, hospital and etc. The reason that this measure was chosen is because an image represents most of a DICOM file, nearly 50 Mb. All these 25 images were created by CT equipment that generates monochromatic images with 512x512 pixels with 16 bits per pixel.

![Figure 5. Time of writing operation](image)

The figure 5 compares the performance between serial CyclosDCMServer and our integrated parallel architecture. Collected 25 stored times, our architecture shows better performance than the serial with the average for parallel method being 0.0107176 seconds, while the average for serial was 0.01539956 seconds, representing an improvement of around 30 percent.

![Figure 6. Time of server process](image)

However, the figure 6 shows the server performance in all process, collecting time since receiving a DICOM image, to storage in HDF file. The experience demonstrates the difference between the architectures, in specific, the leakage of using system call for initiate MPI process. As we can note, the performance of CyclopsDCMServer with additional of parallel architecture, has a delay about twice time than serial server.
6. Conclusion and future work
The objective of this paper was to introduce a new parallel architecture designed to reduce the bottleneck I/O issues in the serial architecture provide by HDF. With the goal to get better performance in I/O access in HDF file at PVFS, we introduced an extension to the architecture for the CyclopsDCMServer previously introduced by Macedo et al. [14].

In related work, we shown many studies with similar issues in bottleneck which have used parallel I/O to avoid the problems to obtain great results.

Although our experiments involved only the writing function to compare serial and parallel architecture in the same environment, we shown that the parallel architecture result in thirty percent improvement on I/O operations, but still having performance issues from system call.

The retrieval time of a DICOM image collected from a HDF file using parallel I/O is our priority work although the aproach presented in this work doesn’t inclued the retrieval operation since the server application was not completly done.

Future work includes analyzing the significance MPI nodes numbers to measure the variance of I/O performance and minimize the communication between H5WL and master MPI node since causes performance decrease. Running performance experiments in a virtual cluster machines comparing the network performance in a virtualization enviroment over standard cluster is another goal.

7. Reference
[1] HDFGroup 2011 Access: January URL http://www.hdfgroup.org
[2] Wallauer J, von Wangenheim A, Andrade R and De Macedo D 2008 21st IEEE International Symposium on Computer-Based Medical Systems (IEEE) pp 105–107
[3] De Macedo D, Von Wangenheim A, Dantas M and Perantunes H 2009 International Journal of High Performance Systems Architecture 2 99–106 ISSN 1751-6528
[4] Labtelemed Laboratorio de telemedicina Access: January. 2011 URL http://www.telemedicina.ufsc.br
[5] PVFS Parallel virtual file system, version 2 Access: January. 2011 URL http://www.pvfs.org
[6] Romio: A high-performance, portable mpi-io implementation Access: December. 2010 URL http://cyclops.telemedicina.ufsc.br
[7] Pianykh O 2008 Digital Imaging and Communications in Medicine (DICOM): A practical introduction and survival guide (Springer Verlag) ISBN 354074570X
[8] Cyclops 2011 Cyclops project Access: January URL http://www.cyclops.ufsc.br
[9] Division M Romio: A high-performance, portable mpi-io implementation Access: March. 2011 URL http://www.mcs.anl.gov/research/projects/romio/
[10] Laghaye N, Sosonkina M, Maris P and Vary J 2009 Computational Science–ICCS 2009 84–93
[11] Adelmann A, Ryne R, Shalf J and Siegerist C 2006 Particle Accelerator Conference, 2005. PAC 2005. Proceedings of the (IEEE) pp 4129–4131 ISBN 0780388593
[12] Yu H, Ma K and Welling J 2005 Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference (IEEE) p 49 ISBN 0769521533
[13] Division M Mpiich2: High-performance and widely portable mpi Access: December. 2010 URL http://www.mcs.anl.gov/research/projects/mpiich2
[14] de Macedo D, Perantunes H, Maia L, Comunello E, von Wangenheim A and Dantas M 2008 Computers and Communications, 2008. ISCC 2008. IEEE Symposium on pp 658 –663 ISSN 1530-1346