Published in final edited form as:

Proc SPIE. 2011 ; 7967(79670C): . doi:10.1117/12.877838.

Towards Building High Performance Medical Image Management System for Clinical Trials

Fusheng Wang^a, Rubao Lee^b, Xiaodong Zhang^b, and Joel Saltz^a

^aCenter for Comprehensive Informatics, Emory University, Atlanta, Georgia, USA

^bDepartment of Computer Science and Engineering, Ohio State University, Columbus, Ohio, USA

Abstract

Medical image based biomarkers are being established for therapeutic cancer clinical trials, where image assessment is among the essential tasks. Large scale image assessment is often performed by a large group of experts by retrieving images from a centralized image repository to workstations to markup and annotate images. In such environment, it is critical to provide a high performance image management system that supports efficient concurrent image retrievals in a distributed environment. There are several major challenges: high throughput of large scale image data over the Internet from the server for multiple concurrent client users, efficient communication protocols for transporting data, and effective management of versioning of data for audit trails. We study the major bottlenecks for such a system, propose and evaluate a solution by using a hybrid image storage with solid state drives and hard disk drives, RESTful Web Services based protocols for exchanging image data, and a database based versioning scheme for efficient archive of image revision history. Our experiments show promising results of our methods, and our work provides a guideline for building enterprise level high performance medical image management systems.

1. Introduction

Image based biomarkers are widely used for therapeutic cancer clinical trials.¹ To efficiently interpret medical images, medical images are often collected to a centralized location, anonymized, stored and read by independent domain experts such as radiologists. Reading results (lesion markups and measurements) are then analyzed and returned to clinical trial companies. The workflow can be as follows: i) Medical images are received from different clinical trial sites; ii) Images are anonymized and stored into a medical image management system; iii) Worklists are defined for each observer; iv) An observer picks a workstation installed with an image reading application, retrieves images defined in the worklist from the central medical image database to the workstation, and reads and annotates images; v) After completion, reading results are checked into the image database. These results will then be validated based on a multi-review protocol. To provide efficient assessment of medical images from multiple experts, it is critical to provide a highly efficient workflow and system to support this. Due to the large scale of DICOM images, there are many challenges.

Large scale medical images

A typical DICOM series can contain hundreds or thousands of DICOM images. With a typical size of 512KB each, a single series can be as large as 0.5GB to a couple of GBs.

Retrieving images from the database will incur significant I/O operations, and require high data throughput from storage devices.

Concurrent users

There can be many radiologists reading images from the database at the same time. This will cause extensive concurrent reading requests from the database.

Transporting data in a distributed client-server environment

While high speed Ethernet is becoming popular – 1Gbps becomes common, the protocols for exchanging data can still significantly impact the performance. Traditional Web service based protocols are commonly used for data exchange over the Web. However, for exchanging binary image data, such protocols encode the data with much overhead and the performance can deteriorate significantly.

Versioning of data

Clinical trial systems have to be compliant to 21 CFR Part 11 regulation,² and all the history of records and their access have to be preserved. DICOM images can be changed for de-identification purpose, and reading results may also be modified. All the snapshot history of images and results have to be kept as audit trails. By simply storing the versions as snapshots will immediately lead to significant redundancy on storage.

The high data throughput requirement is limited by hard disk drives (HDD). While HDD size keeps on increasing and the price keeps on dropping, HDD suffers from long latencies of handling random accesses due to its mechanic nature. Recently, flash memory based Solid State Drive (SSD) technologies are emerging. Based on semiconductor chips, SSD provides many advantages, especially its high performance for random data accesses.³ Although it is much more expensive, SSD can be used as an intermediate layer for transactions, with potential high data throughput. In this paper, we propose to use a hybrid approach of managing medical images – SSD for transactions, and HDD for archive.

In Web services, REpresentational State Transfer (REST)⁴ is a key software architecture style that takes a stateless client-server architecture and the Web services are viewed as resources and can be identified by their URIs. DICOM images can be naturally taken as resources consumed by radiologists, thus well represented through the RESTful interfaces. Moreover, REST is directly based on HTTP protocol, which is far more efficient than SOAP based data transportation protocol, where binary files are encoded as text for transportation purpose.

PACS systems do not support versioning of images. Although traditional source code control systems (e.g., CVS,⁵ subversion⁶) appear attractive, these systems suffer from several limitations: i) These systems are focused on text based files, and not optimized for binary files; ii) Version control systems are OS file system based, thus will not support generic metadata based search of images, other than traversing by folder names; iii) Such systems have very limited interfaces and error handling mechanism, are neither flexible nor robust to support enterprise level applications.

Our Contribution

In this paper, we propose and evaluate a high performance medical image management system (MIMS) for clinical trials, with following salient features:

- Hybrid storage of SSD and HDD for cost effective high data throughput;

- RESTful based protocol for convenient image data access and highly efficient data transportation;
- An effective database based versioning management for efficient storage;
- An architecture that brings all the components together;
- Efficiency of the system demonstrated in our comprehensive evaluation.

2. Overview of our Approach

2.1 System Architecture

MIMS consists of three major components: database server, application server, and RESTful client APIs library (Figure 1). On the database server, metadata are managed through a relational database, and medical image files are managed through OS file system. Metadata include: i) location metadata, which tracks the storage paths of images located in the file system; ii) image content metadata, such as protocolID, patientID, SOP InstanceUID, etc; and iii) versioning metadata. Versioning manager provides management of images, and prefetching manager prefetches images for retrieval operations.

2.2 Hybrid Storage

We take a hybrid storage approach on managing image files. While HDD storage is used for long term archive, SSD storage is used for retrieval transactions. When a worklist is defined, images contained in the worklist will be prefetched into SSD space on the database server. Note that it is not possible to prefetch images to workstations, due to the dynamic nature of schedules of experts, and the random selection of workstations to assess images. When radiologists begin to retrieve images from worklists, images are already prefetched onto SSDs through a database prefetching manager. Thus for transactions, images are retrieved from SSD storage.

2.3 Transferring Data Using RESTful Based Web Services

REST is a software architecture style for distributed hypermedia systems such as the Web. REST is lightweight in representation, and easy to build applications. For example, with RESTful Web Services, we can define an URL syntax to retrieve an image series, for example, <http://myimageserver.com/series/1.3.6.1.4.1.9328.50.3.95047>, by replacing the corresponding entity into its binary value. In particular, based on HTTP, RESTful Web Services are very efficient on transporting data over the Web.

2.4 Versioning of DICOM Images

To version DICOM images while maintaining high efficiency on retrieval, snapshot of current objects is stored separately from their history, as only current objects are actively used for image readings. For past versions of an object, no snapshot will be preserved; instead, a binary diff is preserved. A past version of an object is only used for audit trails and can be regenerated by applying the patch to its next version snapshot – which can be recursively generated by applying patches to current object.

3. Hybrid Storage

By using a hybrid storage, we take advantage of the cost efficiency of traditional hard disk drives for long term image archive and the I/O efficiency of solid state drives for fast transactions.

3.1 Introduction of Solid State Drives

In the last two decades, researchers worked continuously on addressing several open issues of Hard Disk Drive (HDD), such as long latencies of handling random accesses, excessively high power consumption, and uncertain reliability. Most issues are attributed to the mechanic nature of HDD. Recently, flash memory based Solid State Drive (SSD) emerges and received strong interest in both academia and industry.^{7, 8, 9} Unlike traditional rotating media, SSD is based on semiconductor chips, and comes with many strong technical merits, including low power consumption, compact size, shock resistance, and most importantly, extraordinarily high performance for random data accesses.

The building block of SSD is flash memory. There are two types of flash memories, NOR and NAND. NOR flash memory supports random accesses in bytes, and NAND flash memory is designed for data storage with denser capacity and only allows access in units of sectors. Most SSDs available on the market are based on NAND flash memories. In this paper, flash memory refers to NAND flash memory specifically. NAND flash memory can be classified into two categories, Single-Level Cell (SLC) and Multi-Level Cell (MLC) NAND. A SLC flash memory cell stores only one bit, while a MLC flash memory cell can store two bits or even more. Compared to MLC, SLC NAND usually has a 10 times longer lifetime and lower access latency. However, considering cost and capacity, most low-end and middle-level SSDs tend to use high-density MLC NAND to reduce production cost. In this paper, we have used an SLC-based SSD for our experiments. A flash memory package is composed of one or more dies (chips). Each die is segmented into multiple planes, and each plane typically contains thousands (e.g., 2048) of blocks and one or two registers of a page size as an I/O buffer. A block usually contains 64 to 128 pages, and each page has a 2KB or 4KB data part and a metadata area (e.g., 128 bytes) for storing Error Correcting Code (ECC) and other information.

Flash memory supports three major operations, read, write, and erase. Read is performed in units of pages, and each read operation may take $25\mu\text{s}$ (SLC) to $60\mu\text{s}$ (MLC). Writes are normally performed in page granularity, and pages in one block must be written sequentially. Each write operation takes $250\mu\text{s}$ (SLC) to $900\mu\text{s}$ (MLC). A unique requirement of flash memory is that a block must be erased before being programmed (written). An erase operation can take as long as 3.5ms and must be conducted in block granularity. Thus, a block is also called an erase block. Flash memory blocks have limited erase cycles. A typical MLC flash memory has around 10,000 erase cycles, while a SLC flash memory has around 100,000 erase cycles. After wearing out, a flash memory cell can no longer store data. Thus, flash memory chip manufacturers usually ship with extra flash memory blocks to replace bad blocks.

Since an individual flash memory package only provides limited bandwidth (around 40MB/sec), flash memory based SSDs are normally built on an array of flash memory packages. As logic pages can be striped over flash memory chips, similar to a typical RAID-0 storage, high bandwidth can be achieved through parallel access.¹⁰ A serial I/O bus connects the flash memory package to a controller. The controller receives and processes requests from the host through connection interface, such as SATA, and issues commands and transfers data from/to the flash memory array. When a page is being read, the data is first read from the flash memory into the register of the plane, then shifted via the serial bus to the controller. A write is performed in the reverse direction. Some SSDs are also equipped with an external RAM buffer to cache data or metadata. A critical component—the Flash Translation Layer (FTL)—is implemented in the SSD controller to emulate a hard disk and exposes an array of logic blocks to the upper-level components. The FTL plays a key role in SSD for logic block mapping, garbage collection, and wear leveling. Many sophisticated mechanisms are adopted to optimize SSD performance.

Our previous experiments³ demonstrated the exceptional performance of SSD for handling random reads. In general, our findings show that significant advances have been made in SSD hardware design, providing high read access rates combined with reasonable write performance under many regular workloads. This motivates us to use SSD to support high throughput operations, and leave long term archive of images to cost effective traditional HDD.

3.2 Prefetching of Images onto SSDs

A worklist manager (additional software system, not shown in the architecture) provides lists of datasets to be annotated. The database prefetching manager prefetches images from HDD storage to SSD storage, based on the images defined in the worklist for next day reading. When a client workstation sends an image retrieval query to the database, the database first tries to locate images on the SSD storage. If images are available on SSD storage, they are read and sent to the client by the application server. If the requested images are missed on the SSD storage, they are retrieved from the original HDD storage. The prefetching manager is implemented as a service on the database server and starts periodically before each working day.

4. Transferring Data with Restful Based Web Services

Service Oriented Architecture (SOA) is widely used in enterprise application systems, and has been adopted in the health-care world as well. Web access to DICOM objects has been discussed and proposed in DICOM Supplement 148: WADO Web Services.¹¹ Web services standards can be classified into two categories: REST-oriented and SOAP-oriented standards.^{4,12} While SOAP is a general protocol, REST is an architectural style, representing a navigational style of design. Existing resources on the server can be represented as unique URIs and navigated by the clients. Although RESTful architectures can be based on other application layer protocols, HTTP is the mostly used protocols.

In a summary, a RESTful Web service is a simple web service implemented using HTTP and the principles of REST. It is a collection of resources, with three defined aspects: i) the base URI for the web service; ii) the MIME type of the data supported by the web service, such as JSON, XML, binary images or others; iii) the set of operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE). The advantages of the REST-based approach include the potential scalability, and the lightweight access to its operations due to the limited number of operations and the unified address schema.

We take the RESTful based Web service approach, and prototyped APIs for locating DICOM images. Multiple types of REST URI types for image retrieval are defined:

- Image: locate a DICOM image based on image instance UID.
For example: `http://myimageserver.com/image/1.3.6.1.4.1.9328.50.3.122487`
- Series: locate images as a zip object based on a series instance UID.
For example: `http://myimageserver.com/series/1.3.6.1.4.1.9328.50.3.95047`
- Study: locate images as a zip object based on a study instance UID.
For example: `http://myimageserver.com/study/1.3.6.1.4.1.9328.50.3.95021`
- Image header: locate DICOM image header as an XML document based on an image UID.
For example: `http://myimageserver.com/image/1.3.6.1.4.1.9328.50.3.122487`

Additional APIs are defined for uploading and modification of images, and manipulating image annotations.

RESTful Web service implemented using HTTP is very efficient on exchanging images. We performed a comparative study of REST, FTP, and SOAP based implementations with Apache Axis and Tomcat (Figure 2) on the transportation cost versus number of image files. REST performs best, while SOAP based implementation with images as attachments performs worst – almost five times slower than REST, due to the complexity of the protocol and the much increased size of encoded context of images.

5. Image Versioning

Audit trails are required for keeping track of database activities, including database audit trails and image audit trails, and the latter is much challenging due to the scale of data. DICOM images could be updated a couple of times during the whole process of image assessment. The simplest approach is to keep a snapshot for each versions, which could quickly multiply the storage size of data.

Versioning is not supported by major PACS systems, although content management systems such as Documentum¹³ provide versioning support. Source code control systems such as CVS provide comprehensive support of versioning of text based source codes. Versioning information is tracked through file systems, thus searching files in a large scale is not efficient. And metadata based search is also very limited.

Since versioning of images for audit trails is much simpler than versioning for source code control, we can take advantage of the methodology of source code control system, and implement a database based versioning approach for audit trails. To provide both search efficiency and flexibility, and storage efficiency, we propose a database approach to manage versioning of image data based on chaining of diffs between versions.

In our method, a snapshot of current version of image objects is always physically preserved to avoid additional I/O and computation cost to reconstruct the snapshot, as current snapshots are most likely to be retrieved. The current version of objects are stored together, separate from their history – this could provide a better clustering of current data. For past versions of an object, no snapshot will be preserved; instead, a binary diff is kept. A past version of an object is only used for audit trails and can be regenerated by applying the patch to its next version snapshot – which can be recursively generated by applying patches to current object.

The version history is managed through version metadata in a table, as shown in Figure 3. Figure 3 shows a versioning example of a DICOM object A. Current objects are managed together in a current object table. The object history table has version metadata that keeps track of the corresponding version object – a snapshot object of current object or a reversal diff object in the file system. To retrieve the initial version (Version 1) of object A, Version 2 is first regenerated by patching delta2-3 to current object A, and then Version 2 is further patched with delta1-2 to generate Version 1.

This approach is highly storage efficient and without losing retrieving performance for reading current images. For example, the delta between two snapshots after anonymizing the image (edited with Sante DICOM Editor¹⁴) from its header is less than 0.5 percent of the snapshot size; the delta between a modified image (blinding a text region on the image) and its original image is less than 2 percent. The test is based on the bsdiff tool.¹⁵

6. Evaluation

We performed experiments on data throughput in a simulated distributed client-server environment with multiple concurrent users. The goal is to test the I/O or network bottlenecks under different settings. We conducted comprehensive studies on disk reading throughput from the server and overall data throughput from the server to clients with multiple concurrent users. We used a 1Gbps Ethernet between the testing server and clients. The server machine is a Linux machine installed with Fedora 11 (32bits), with Core 2 Duo Processor E7300 at 2.66GHz, 3GB memory, and we used an Intel X25-E SSD (32GB) and a traditional HDD (Western Digital WDC WD1600JS-60M, 160GB, 7200rpm). For datasets, since images from different modalities vary in sizes (for example, 512KB or 58KB for a single DICOM image), in our experiments, we used three data sets: Dataset 1 with 4000 512KB images, Dataset 2 with 30000 58KB images, and Dataset 3 with 4000 512KB images mixed with 4000 58KB images. Caching is invalidated between each test, and results are averaged based on multiple tests.

Figure 4 shows the server reading throughput for three datasets on HDD and SSD respectively, with varying numbers of concurrent retrievals (1, 2, 4, 5, 8, 10, 16, and 20 users respectively). The best throughput for HDD is about 27.5MBps, and it remains similar for different number of concurrent users. The reading throughput for smaller files on HDD decreases due to the overhead of random access. SSD has significant performance advantage on reading throughput: the best throughput is 218MBps for 20 concurrent users, almost eight times more than that of HDD, and it still not yet saturated. Clearly, the disk reading throughput based on HDD is bounded by the limitations of HDD, thus it is difficult to support high throughput image retrieval with traditional HDD storage in a cost effective way.

Figure 5 demonstrates the overall throughput over the network from the server to clients based on HDD and SSD storage respectively. Three datasets are used for 1, 12 and 24 concurrent retrievals respectively, with increasing file sizes. For HDD, the overall throughput is limited by the I/O bottleneck – it never reaches 40MBps; SSD based storage provides much higher overall throughput – maximum at 110MBps, which is eventually bounded by the network bandwidth. Based on this, to retrieve a series of 1000 DICOM images (e.g., 512KB each) for a single user, we can estimate that it takes about 4.6 seconds for SSD based approach, and about 16 seconds for HDD based approach.

We can conclude that with SSD as the storage device for image retrieval, we can fully take advantage of the network bandwidth in a conventional 1Gps Ethernet to achieve best data transmission throughput. Thus, the data throughput bottleneck is shifted from disk I/O (as the case for HDD based storage) to network bandwidth, which can be further improved when a high speed network is used.

7. Discussion

Much work has been done to efficiently support image visualization over the Internet. For example, JPIP (JPEG 2000 Interactive Protocol)¹⁶ is a compression streamlining protocol that works with JPEG 2000 to produce an image using the least bandwidth required. The Internet Imaging Protocol, or IIP,¹⁷ is an Internet protocol designed by the International Imaging Industry Association to communicate images and their metadata on top of HTTP. Such approach works for certain environments where only a portion of an image is needed, or low resolution images are acceptable for initial viewing. This is not the case for image readings for clinical trials, where the exact images have to be downloaded for viewing.

Caching or prefetching of images to clients is another common approach, when the patterns of image accesses are well known.

Our approach tries to provide a cost effective image management solution by maximizing data throughput from the server to clients. The approach is generic, and can be applied to many other applications with similar characteristics. For example, DICOM extension for pathology¹⁸ has been approved thus pathology images such as whole slide images can be stored and managed as tile series. Based on this, we can manage and provide high throughput pathology image retrieval in a similar approach, by combining image streaming protocols.

8. Conclusion

Large scale image assessment for clinical trials requires high performance data management systems to support efficient image retrieval for image assessment. We show that we can achieve high image data throughput through hybrid storage with solid state drives and hard disk drives, efficient data transportation and queries through RESTful Web Services, and effective database oriented version management. Our experiments demonstrate that our proposed solution is highly efficient. Our approach is generic and can be used to support different types of image applications.

Acknowledgments

The project is funded in part by the National Library of Medicine under Grant Number R01LM009239, the National Science Foundation under Grant Number CCF-0913050, and the National Cancer Institute, National Institutes of Health, under Contract No. HHSN261200800001E.

References

1. Clunie, David. Dicom structured reporting and cancer clinical trials results. *Cancer Informatics*. May.2007 4:33–56. [PubMed: 19390663]
2. CFR - Code of Federal Regulations Title 21.
<http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfCFR/CFRSearch.cfm?CFRPart=11>
3. Chen, Feng; Koufaty, David A.; Zhang, Xiaodong. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. *SIGMETRICS*; 2009. p. 181-192.
4. Richardson, Leonard; Ruby, Sam. *RESTful Web Services*. O'Reilly Media; May. 2007
5. CVSNT. <http://www.march-hare.com/cvsp/ro/>
6. Subversion. <http://subversion.tigris.org>
7. Lee, Sang-Won; Moon, Bongki; Park, Chanik; Kim, Jae-Myung; Kim, Sang-Woo. A case for flash memory ssd in enterprise database applications. *ACM SIGMOD*; 2008. p. 1075-1086.
8. Matthews, Jeanna; Trika, Sanjeev; Hensgen, Debra; Coulson, Rick; Grimsrud, Knut. Intel turbo memory: Nonvolatile disk caches in the storage hierarchy of mainstream computer systems. *ACM Transactions on Storage*. 4:4:1–4:24.
9. Kim, Hyojun; Ahn, Seongjun. Bplru: a buffer management scheme for improving random writes in flash storage. *Proceedings of the 6th USENIX Conference on File and Storage Technologies, FAST'08*; 2008. p. 16:1-16:14.
10. Chen, Feng; Lee, Rubao; Zhang, Xiaodong. Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing. *HPCA-17*; 2011.
11. DICOM Supplement 148: WADO Web Services.
ftp://medical.nema.org/medical/dicom/supps/sup148_pc.pdf
12. zur Muehlen, Michael; Nickerson, Jeffrey V.; Swenson, Keith D. Developing web services choreography standards: the case of rest vs. soap. *Decis Support Syst*. July.2005 40:9–29.
13. EMC Documentum. www.emc.com/domains/documentum
14. Sante DICOM Editor. http://www.santesoft.com/dicom_editor.html

15. Binary diff/patch utility. <http://www.daemonology.net/bsdif/>
16. Jpeg 2000 interactive protocol (jpip). <http://www.jpeg.org/jpeg2000/j2kpart9.html>
17. Internet imaging protocol. <http://iipimage.sourceforge.net/IIPv105.pdf>
18. DICOM Supplement 145: Whole Slide Microscopic Image IOD and SOP Classes. ftp://medical.nema.org/MEDICAL/Dicom/Final/sup145_ft.pdf

Figure 1. Architecture of the medical image management system for clinical trials

Figure 2. Performance Comparison of Web Services

Figure 3. Versioning of DICOM Objects

Figure 4. Comparison of reading throughput on the server versus number of concurrent users

Figure 5. Comparison of overall data throughput over the network versus file sizes