J Digit Imaging (2014) 27:563–570
DOI 10.1007/s10278-014-9692-1

# A Reliable, Low-Cost Picture Archiving and Communications System for Small and Medium Veterinary Practices Built Using Open-Source Technology

**Bryan Iotti · Alberto Valazza**

Published online: 3 May 2014
© Society for Imaging Informatics in Medicine 2014

**Abstract** Picture Archiving and Communications Systems (PACS) are the most needed system in a modern hospital. As an integral part of the Digital Imaging and Communications in Medicine (DICOM) standard, they are charged with the responsibility for secure storage and accessibility of the diagnostic imaging data. These machines need to offer high performance, stability, and security while proving reliable and ergonomic in the day-to-day and long-term storage and retrieval of the data they safeguard. This paper reports the experience of the authors in developing and installing a compact and low-cost solution based on open-source technologies in the Veterinary Teaching Hospital for the University of Torino, Italy, during the course of the summer of 2012. The PACS server was built on low-cost x86-based hardware and uses an open source operating system derived from Oracle OpenSolaris (Oracle Corporation, Redwood City, CA, USA) to host the DCM4CHEE PACS DICOM server (DCM4CHEE, http://www.dcm4che.org). This solution features very high data security and an ergonomic interface to provide easy access to a large amount of imaging data. The system has been in active use for almost 2 years now and has proven to be a scalable, cost-effective solution for practices ranging from small to very large, where the use of different hardware combinations allows scaling to the different deployments, while the use of paravirtualization allows increased security and easy migrations and upgrades.

**Keywords** PACS · DICOM · Image storage and retrieval · Open source

B. Iotti (✉) · A. Valazza
Dipartimento di Scienze Veterinarie, Università degli Studi di Torino, Torino, Italy
e-mail: bryan.iotti@unito.it

A. Valazza
e-mail: alberto.valazza@unito.it

## Background

Picture Archiving and Communications Systems (PACS) are the most needed system in a modern hospital. As an integral part of the Digital Imaging and Communications in Medicine (DICOM) standard, they are charged with the responsibility for secure storage and accessibility of the diagnostic imaging data generated by a medical practice. These machines need to offer high performance, stability, and security while proving reliable in the day-to-day and long-term storage of the data they safeguard. This paper reports the experience of the authors in developing, installing and using a compact, low-cost solution based on open-source technologies in the Veterinary Teaching Hospital for the University of Torino, Italy, during the course of 1 year. The availability of open source image storage and retrieval software packages has steadily increased over the years, making it easier for a radiology department with sufficient in-house technical knowledge to deploy their own PACS solution. However, this kind of project requires a deep understanding of several aspects of information technology, ranging from networking to storage system architecture to data security. Choosing an enterprise-grade operating system as the base for such a project constitutes an important first step towards a vertical approach to system integration focusing on data integrity and system reliability. The Veterinary Teaching Hospital of the University of Torino's 3,000 new cases per year identify it as a large facility by Italian standards. Despite the considerable case load, the Diagnostic Imaging and Radiology units employ machines that are quite old by today's standards. The computed tomography scanner, for example, is a single-slice helical unit[1] with a 512×512 matrix size and uses the discontinued SGI IRIX[2] operating system for the controller unit. When we started our work, the radiology

[1] GE HighSpeed FX/I, General Electric, Fairfield, CT, USA
[2] Silicon Graphics International, Milpitas, CA, USA

department was archiving most of the imaging data as digital versatile disks (DVDs) in a separate room, organized by month and year, while the most recent records resided on the visualization workstation.[3] A copy of the imaging data for the more recent records was also sent to the hospital information system and associated to the patient records, where it could be visualized as a JPEG thumbnail or downloaded in DICOM format. Meanwhile, the CT unit was storing its data on the visualization workstations and on a network drive. This layout is shown in Fig. 1. Accessing the archived imaging data was a time-consuming process, requiring the assistance of a technician from each unit (and therefore being subject to different working schedules, for example). The amount of single points of failure that could determine data loss was extremely high, and none of the systems, with the exception of the hospital information system, was covered by either an unassisted backup or an auditing infrastructure. A proper PACS server with semiautomatic management capabilities was sorely needed. Budget constraints imposed on the authors were quite stringent, so the priority was given to data security over pure system performance.

## Methods

### Previous Works and Introduction

Previous work by other authors outlines the possibility of building an open source PACS solution that delivers commercial quality results without incurring in licensing costs [1]. The authors decided to evaluate the possibility of building a similar system on low-cost, over-the-counter hardware using the more common x86 platform instead of the SPARC architecture used in other projects [1]. Our main goal for this project was the consolidation of the available imaging data behind a single, easy to use user interface.

### Hardware Configuration Overview

The hardware powering the PACS server is a HP ProLiant Microserver,[4] model N40L, with 8 GB error correction and control (ECC) random access memory (RAM) and several internal hard disks. The processor is a dual-core 1.7Ghz AMD Turion.[5] The system itself is physically small and very quiet (measured noise at normal operation is about 22 dB). It is secured against physical intrusion through keyed access to the case and is set up in an office that is closed when not in use. The power cord is chained in place both on the case of the server and the power outlet to prevent accidental disconnections. The system Basic Input/Output System (BIOS) is programmed to restore the system to the previous state in case of a loss of power.

### PACS Software

DCM4CHEE[6] is a DICOM-compliant, feature-complete open-source PACS solution that can offer levels of service on par with commercial solutions [1]. It is written in the Java[7] programming language, making it multiplatform. The Weasis DICOM viewer[4] is installed as a centrally deployed application on the PACS server.

### Database

We chose the PostgreSQL database,[8] version 9.1.

### Operating System

We chose OpenIndiana[9] as the operating system for the PACS. This distribution is a derivative of OpenSolaris[7] and uses the Illumos Kernel.[10]

### System Security

The system is protected by a local software firewall based on IPsec, the standard Solaris system firewall, as well as the perimeter hardware firewalls of the university network.

The system uses the auditing technologies provided by both the PACS server software and the operating system, in the form of an auditing database and a system fingerprint.

### System Subdivision

The DCM4CHEE server resides in a paravirtualized local zone. The current layout of the PACS server is shown in Fig. 2.

### Storage

The disks contained in the PACS server are divided into two groups: system disks and data disks. The system disks are two 250 GB 7200RPM SATA-II disks in software RAID-1 mirror configuration and independently bootable (each disk has a

[3] Agfa Mimosa Vips 1206, Agfa-Gevaert NV/SA 96, Munich, Germany
[4] Hewlett Packard, Palo Alto, CA, USA
[5] Advanced Micro Devices, Sunnyvale, CA, USA
[6] DCM4CHEE, http://www.dcm4che.org, last accessed 08/03/13
[7] Oracle Corporation, Redwood City, CA, USA
[8] PostgreSQL, http://www.postgresql.org, last accessed 08/03/13
[9] OpenIndiana Project, http://openindiana.org, last accessed 08/03/13
[10] Illumos Project, http://wiki.illumos.org, last accessed 08/03/13

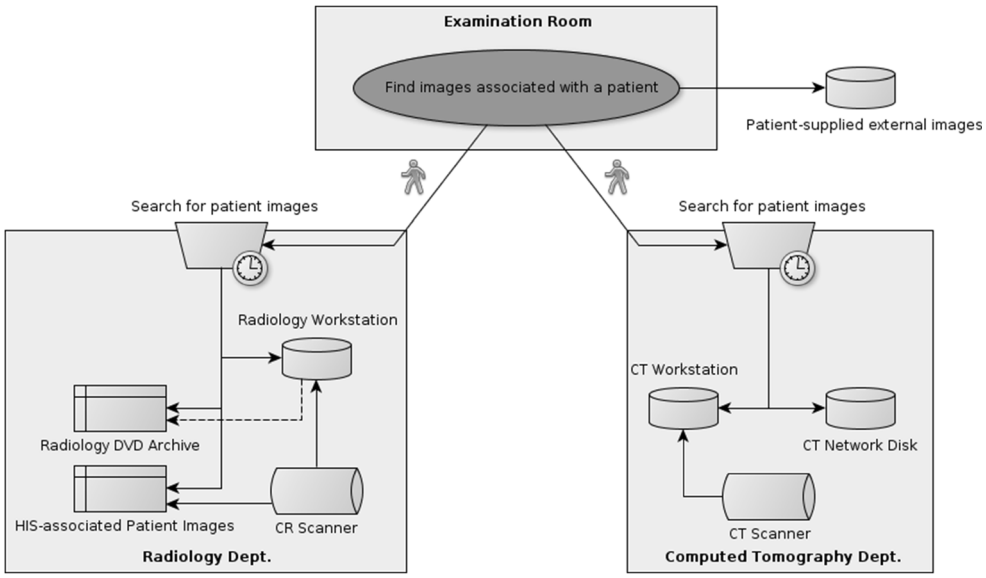


**Fig. 1** The diagnostic imaging workflow before the installation of the PACS system

complete bootloader, and the BIOS is configured to attempt booting from the second one if the first one fails to respond). The data disks are two 1 TB 7200RPM SATA-II disks, also in software RAID-1 mirror configuration. Due to the low cost of storage and the need to keep a native DICOM copy of the files, together with the adequate throughput of the network and the limited CPU resources, we opted to store the files as lossless DICOM images, devoting all available space to online storage. Upon installation of the system, the available archive data for the various modalities was migrated to the server, accounting for roughly 550 GB of the total used storage.

System Tuning

The filesystem read cache (L2ARC) was limited to 4 GB using the kernel configuration files.

Backup Strategy

The system performs automatic unassisted backups of the image data to a network drive on the other side of the building every night and updates a series of log files with the results of the operation. Since the configuration of the system itself is

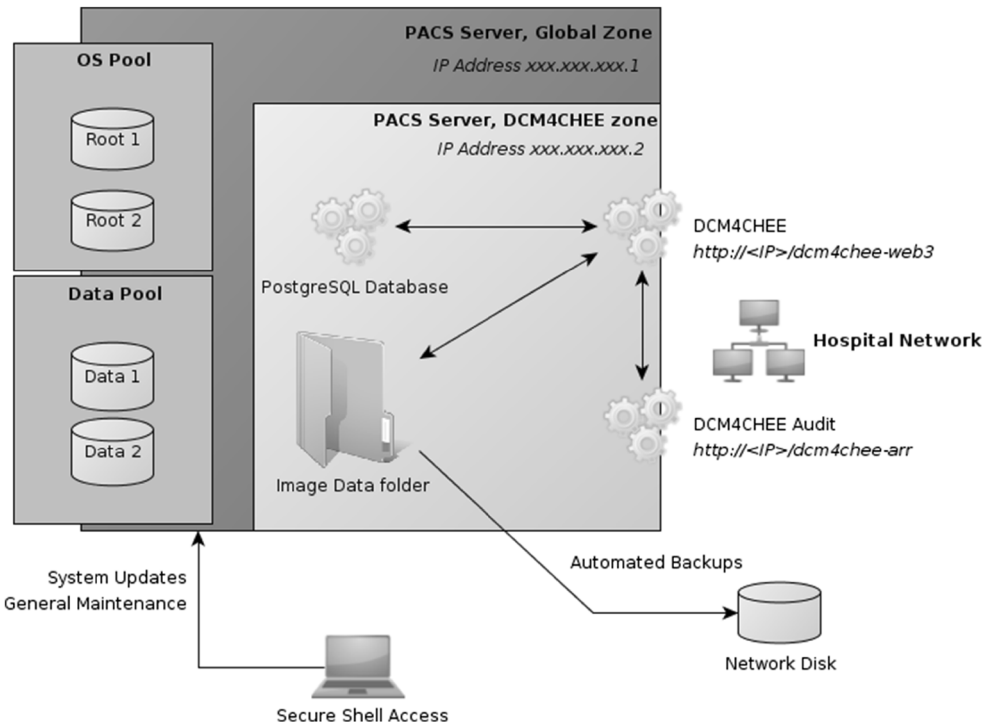


**Fig. 2** Current system subdivision and layout of the PACS server

static, it is backed up before and after performing system updates and upgrades by a competent technician, with the backup data stored on a removable drive in a separate location.

System Monitoring

The system automatically logs the results of actions such as disk data consistency checks, SMART checks, audit logs, and backups to a series of log files, easily accessible by the system administrator.

Total Cost

Total cost for the system as installed was 800,00 € including VAT at the time of purchase (August 2012). The network disk we use for backups had already been purchased and was repurposed. If acquired separately it would have cost 399,00 € including VAT.

## Results

The system has been in active use for almost 2 years now. It has overcome minor situations like power outages successfully, without the loss of data and restoring the services it provides without the need for operator assistance. No unplanned downtime has occurred so far, performance has been better than expected and constant. System log reports have never reported a fault either in disk health or data consistency. The web-based interface is ergonomic to use, loads quickly, and allows role-based access to all the required functions. At the moment of writing this paper, the system contains CT, MR, and CR modality images amounting to roughly 51,000 studies for 20,000 patients, totaling about 700 GB of data and growing by roughly 60 GB per year. The faculty's medical staff can easily access the imaging data and review patient progress by comparing past and present examinations with ease, across multiple modalities, while the system allows students working on their thesis to find and collect all the necessary case data in a matter of a few hours. The current workflow is shown in Fig. 3.

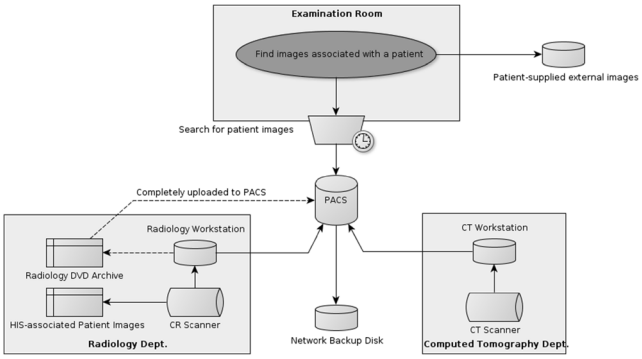


**Fig. 3** The current diagnostic imaging workflow

## Discussion

PACS Software

*Role-Based Approach*

One of the key features of this PACS server is that it allows a role-based approach to user management. We therefore have four main user groups with different privileges: administrators, staff, thesis, and student. Administrators can access the image data, the auditing interface and the system configuration, including the definition of application entity (AE) titles and other DICOM network parameters. Staff users can access the image data, view it and have it sent to an external viewer, but cannot delete it. Thesis students have the same privileges as the staff group; the distinction is made for management and auditing reasons. Student users are the basis for the future development of teaching files, where the creation of special, educational datasets will allow interested students to enhance their knowledge of both physiological and pathological anatomy viewed through different digital imaging modalities. The teaching staff will prepare anonymous datasets of various anatomical parts and conditions. This role is currently not enabled. All operations performed on the web interface, including logins, and the imaging data by the users are subject to monitoring by the auditing infrastructure present in DCM4CHEE.

*Integrated DICOM Viewer*

The Weasis DICOM viewer is a multiplatform, multimodality DICOM-compliant image viewer written in Java and centrally deployed (after the initial installation, any update will be delivered to the client computers automatically). Weasis played a crucial role in allowing any computer inside the faculty to be used for basic visualization and measurements, resulting in a diminished workload for the specialized workstations. The tools available in Weasis are considerably superior to those available on the older workstations, especially for more structured measurements like fixed right-angles and Cobb's angle. The Osirix DICOM viewer[11] is used when three-dimensional reconstructions are necessary, and in these cases, the PACS is accessed through the DICOM network using the standard DICOM Query commands (C-GET, C-MOVE).

[11] Antoine Rosset, Geneva, CH

### Database

The DCM4CHEE PACS server comes with interfaces for many popular databases, like MySQL, PostgreSQL, Microsoft SQL Server, and IBM DB2. We chose PostgreSQL because it has several desirable characteristics: it is completely open source, well known, powerful, and reliable. It supports atomic transactions, ensuring data consistency, and is well suited to vertical scaling (i.e., move to a more powerful machine, versus horizontal scaling which indicates a move to a cluster-type deployment). PostgreSQL version 9.1 is available as a pre-built package for OpenIndiana and comes with its own service manifests for the Service Management Facility.

### Operating System

#### *Why Solaris?*

The Sun Solaris operating system is known in the enterprise world for its stability, security, and reliability [1]. It offers some of the most refined and complete administration and maintenance tools available and is built on cutting edge technology that has been field tested and proven in the most demanding mission-critical environments [1]. Historically, it has been associated only with large businesses due to the elevated licensing costs. In June 2005, Sun Microsystems created a mixed open/closed source version of Solaris for the community, called OpenSolaris. This operating system gained a substantial following in fields that could benefit from the Solaris features but could not afford the expensive support contracts associated with it. When Oracle acquired Sun Microsystems in 2009, it also became the owner of the rights to the Solaris operating system. Oracle stopped the development and distribution of OpenSolaris, but the community had forked just in time, creating an open-source version of the Solaris Kernel with the Illumos Project. By combining the traditional OpenSolaris GNU userland with the Illumos kernel, the OpenIndiana distribution was born. This distribution aims to continue the OpenSolaris legacy as both a desktop and server general-purpose operating system by providing a drop-in replacement for existing OpenSolaris installations. We chose to use OpenIndiana as the operating system for the PACS because it features the Zettabyte File System (ZFS), numerous virtualization possibilities, an advanced networking stack, the Service Management Facility (SMF), and the DTrace framework. OpenIndiana also offers the possibility of a desktop graphical user interface if needed (other derivatives like OmniOS and SmartOS are text-only), allowing the authors to develop and test the system on a workstation that provides an identical software stack as the final product but with the additional ease of use of a graphical user interface.

#### *ZFS*

ZFS has long been one of the most advanced file systems available [2, 3]. By separating the concept of volumes (called *pools*) and physical drives, it allows flexible management of the storage assets of a server. Software Redundant Array of Inexpensive Disks (RAID) solutions can be implemented and changed while a system is online, without requiring downtime. ZFS assumes that the drives themselves will eventually fail or generate errors. It therefore writes a checksum for each and every block of data written to the disks and then checks their correspondence both when the data is accessed, and periodically through the use of *scrubs*, a maintenance operation that checks each and every used data block without offlining the system. ZFS is a copy-on-write journaled file system. This means that when a block of data is written to the disk, each pool contains a log of the write operations, called a ZFS Intent Log or ZIL. Should the power be cut accidentally before the write operation is complete, this ZIL will contain the necessary information for replaying the transaction, ensuring data consistency. Also, in a copy-on-write filesystem, the data that is read and modified is not rewritten to the same physical block, but to another part of the disk and its inode reference changed. This allows the implementation of point-in-time versioning through the use of *snapshots*, in essence keeping a copy of the complete pool or filesystem and only marking the changes made to a new filesystem with very little overhead. Snapshots are the bases for the ZFS send/receive commands that are used for backing up and cloning filesystems. Should the data pool need to be transferred to another server, for example, the system administrator could take a snapshot of the data filesystem and start to copy it in its exact condition, including access permissions and file times. While the data is being copied, the server continues to function regularly, and new data is received without touching the snapshot. After the transfer is complete, the PACS server software would be stopped, a new snapshot taken, and the data sent incrementally, bringing the new copy up to date. The result is a greatly decreased offline time. System updates follow a similar logic, through the use of boot environments (BEs) that represent a complete, working copy of the system software. When updates are applied, the system administrator can request the creation of a new BE for the updated software, ensuring that the system can be restarted from the previous, known-good copy should trouble arise. Another useful feature of ZFS is the availability of four different built-in compression schemes that can be added on a per-file system basis. Proper use of this technology could, for example, allow the use of compression on a separate pool or filesystem to implement compressed nearline storage and increase the available storage capacity. The computational cost of compression on modern higher-tier processors is negligible, considering the improvement gained in disk input/output (which is the main bottleneck

in many systems). We opted not to use filesystem compression because of the machine's limited processing power and have dedicated all available space to online storage.

*Filesystem Caching*

ZFS features both read and write caches. The write cache is the ZIL, as previously mentioned. Its performance can be improved by moving it to a separate drive, maybe based on high-speed flash storage (otherwise it resides on the pool itself, mixed with the data). The read cache is divided in two levels: ARC and L2ARC. ARC stands for adaptive replacement cache. Being adaptive, this read cache "learns" from the interaction between the pool and the operating system and "warms up" by storing the most frequently used blocks of data in RAM. Since RAM is two orders of magnitude faster than hard disks (access times are in nanoseconds instead of milliseconds), accessing the cached data greatly improves the system response time. The ARC also stores the deduplication tables, should deduplication be enabled on a pool. Deduplication means that the operating systems writes data to the disks and ZFS finds identical blocks of data across a stream and only writes them once, referencing said blocks in other locations. Depending on the data being stored, this can improve storage capacity by as much as twenty times [4]. However, deduplication tables must be stored in RAM or the system response times will become unacceptable. ZFS requires 1–2 GB of RAM per TB of deduplicated data. Should a system not be able to install enough RAM to hold the deduplication tables together with the application memory requirements, one can resort to using a solid state drive (SSD, flash storage) to store the tables and ARC data on. This is the L2ARC. SSDs feature a random seek capacity that is one order of magnitude slower than RAM, but one order of magnitude faster than traditional hard disks. In our system, we did not use data deduplication because the limited CPU resources would not have been able to deal with the increased workload. The default behavior of ZFS is to allocate all system memory to the ARC except one GB, which remains for the OS. Now, this memory can be freed, but if the applications require the use of large pages of memory, the ARC can be slow to release the used memory, resulting in slower performance and memory fragmentation. We therefore limited the maximum amount of memory used by the ARC by profiling the system after several weeks of regular use. By analyzing the most commonly used size for the ARC, we found it to be at most 4 GB and limited it to that value, as per instructions provided by the ZFS tuning guide.[12] This left a sufficient amount of free memory for both the OS and the applications. This dynamic caching ability is very important for achieving high performance, since it represents a user-transparent way of mimicking the behavior of enterprise-level tiered storage (i.e., storage solutions where infrequently accessed data resides on slow, high capacity disks and recent or often requested data resides in RAM or fast solid state drives) for a fraction of the cost.

*Paravirtualization*

Paravirtualization is a unique feature of the Solaris OS, similar to FreeBSD's Jails. The kernel itself allows partitioning of memory and resources (which can be limited or "throttled" if needed) into software *zones*. Zones can be shared or dedicated. By relying heavily on the ZFS snapshot feature, they are extremely lightweight (a few hundred megabytes per instance). Each zone behaves like a single separate entity (they can for example have a different network address) and has no access to either the data or the memory maps of the global zone (host computer). Due to their light weight and little to no performance impact, zones represent an ideal way of running multiple separate services on a single machine. Should a zone be compromised, the attacker would only have access to the data contained inside it, and no control over the global zone. Our PACS installation hosts the DCM4CHEE server and the PostgreSQL database in a single zone at the present time. Zones are very easy to migrate across machines, consisting only of a zone manifest and an associated ZFS filesystem. Should we decide to upgrade the hardware on the PACS, we could shut down the zone containing the PACS server, export a configuration manifest, edit it if necessary, and use the ZFS send/receive commands to stream the complete zone to the new machine. When the zone is started again, nothing has changed from its point of view and service resumes with little interruption.

*Networking*

Solaris features a very advanced networking stack that allows the creation of inter-zone dedicated high-speed networks. This does much to improve the security of a system based on this technology, limiting the exterior visibility of the services provided. Also, the system allows the aggregation of different physical links into a larger logical one, with to several different modes of operation, or the partitioning of an existing link into smaller VLANs.

*Service Management Facility*

Services in Solaris are managed through the Service Management Facility or SMF. Here, each service carries a manifest that identifies it, places it in the right part of the boot and shutdown sequence for the OS, handles dependencies (some software might require to be started only after the database it

[12] ZFS Tuning Guide, http://www.solarisinternals.com, last accessed 08/03/13

runs on has been correctly started, for instance), and monitors service health periodically, logging any malfunctions and startup error codes. Services placed under SMF are automatically restarted, for example, should they stop working. In our system, the local zone has its own list of SMF services, including the PostgreSQL database and JBoss server (hosting the PACS server application). The global zone starts the local zone itself as an SMF service.

*DTrace*

The inherent complexity of a modern operating system can make it very difficult to troubleshoot. Just gathering the necessary data to understand the problem can be a daunting task. The management tools available are numerous and often intimately tied in to the platform they were developed for. DTrace is a very advanced dynamic tracing technology that can effectively visualize different aspects of a running, unmodified production system, providing great insight into potential performance problems [5]. Since it is also available on Apple Mac OS X and FreeBSD, it has become a common ground for troubleshooting and objective-driven benchmarking.

Auditing Technologies

The installation guidelines of the PACS server software encourage the system administrator to create an auditing database, which will contain a record for each and every operation performed on the PACS (e.g., image send/retrieve, user login attempts, and whether they are successful or not). This database can then be examined periodically by system administrators to detect any unusual behavior. The Solaris operating system features a program called BART, an acronym for *basic audit reporting tool*, that allows the rapid acquisition of a system fingerprint: in essence, what files are stored where, their size, date of last access and modification, and access permissions. When run, the output of the tool should be saved to a text file, so that the program can later use it to perform an accurate comparison with the future state of the system. When updating the system, the administrator verifies the current fingerprint against the stored one to ensure that no unauthorized modifications have been made before proceeding. After all updates have been applied, a new fingerprint is acquired and stored.

Important Hardware Features

Despite being a very small server, the HP Proliant Microserver features error correction control (ECC) random access memory (RAM). This is important because it can recognize corruption of the in-memory contents as it happens and report this to the operating system, preventing the corrupted data from being written to disk as if it was correct. ECC RAM represents a very important cornerstone in a data security scheme based on ZFS: employing the best file system does not matter if the data that the system processes becomes corrupted in RAM and this is not recognized.

Skills Needed

Despite the best attempts to make management of such a system as easy to use as possible, the flexibility provided means that a quite steep learning curve is involved. The IT personnel managing this type of machine should be well-versed in the Solaris OS (which is similar to Linux in many respects, but managed differently) as well as Java and the management of a JBoss[13] Web workflow.

Possible Future Optimizations

A good hardware upgrade would be the use of a system with dual power supplies (known as an $n+1$ configuration) or at least of an uninterruptible power supply (UPS battery backup). When the system was installed, it was not deemed necessary to create separate AE titles for each modality as would be generally suggested [6]. This was performed 1 year after the initial installation, since it represents a good optimization of the system and allows separate processing and routing of the different modalities. Also, the system layout could be improved by moving the PostgreSQL database to a separate zone, accessed through a virtual network that is accessible only to the DCM4CHEE server software. This would allow easier management of the database software while improving reliability and performance. The installation of a web management console like Nagios[14] would allow the system to be largely monitored by the hospital personnel, without requiring system administrator intervention unless it is needed. Choosing a system with faster processors would allow the creation of a compressed filesystem for use as nearline storage, with automatic transfer of the imaging data between tiers based on when it was last requested. Backing up the image data to an external disk is barely an acceptable solution. The optimum would be the creating a cluster configuration, as supported by both JBoss and DCM4CHEE, with two or more systems balancing their load and synchronizing the information they store. The adoption of clustering would allow higher performance, reliability, and automatic failover with no system downtime. The main issue with many external network hard disks is the lack of SMART monitoring: should a disk start to accumulate errors, they would not be reported and go unnoticed until the total failure of the device. This could expose the system to a small timeframe where the data is not covered by

[13] JBoss, http://www.jboss.org/, last accessed 08/03/13
[14] Nagios, http://www.nagios.org/, last accessed 08/03/13

the security provided by a backup. One strategy could be to plan obsolescence for the backup disks on a yearly basis, or use a dedicated small system for backups, allowing the use of SMART monitoring and reporting, possibly together with data compression and/or deduplication for increased storage. The system is currently not integrated with the hospital radiology information system (RIS), but this would allow the use of the modality worklist, reducing the possibility of user error during patient data entry [7]. All connections to and from the PACS server could be upgraded to run over a Secure Sockets Layer (SSL) encrypted protocol, thus increasing security. Thanks to the open source nature of this project, all components of this system could be redistributed as a pre-installed, ready-to-run image with good default values for most options, enabling close-to-unassisted deployments, requiring only a minimum amount of configuration.

## Conclusion

After 2 years of use, this system has proven itself a scalable, cost-effective solution for practices ranging from small to very large, where the use of different hardware combinations allows scaling to the different deployments while the use of zones allows increased security and easy migrations and upgrades.

**Acknowledgments** The authors wish to acknowledge the OpenIndiana user and developer community and the DCM4CHEE user and developer community. Their continued high-quality work makes this system possible.

**Conflicts of Interest and Financial Support** The authors declare no conflict of interest. No external financial support was received for this project.

## References

1. Marcheschi P, et al: A new approach to affordable and reliable cardiology PACS architecture using Open-Source Technology. Comput Cardiol, 2009. IEEE, 2009
2. Bonwick J, Ahrens M, Henson V, Maybee M, Shellenbaum M: The Zettabyte File System Course material for CMPS 221: Advanced Operating Systems, Fall 2006, UC Santa Cruz.
3. Zhang Y, Rajimwale A, Arpaci-Dusseau A, Arpaci-Dusseau R: End-to-end Data Integrity for File Systems: A ZFS Case Study FAST'10 Proceedings of the 8th USENIX conference on File and storage technologies, pp 3–3
4. Bonwick J: Zfs deduplication. Jeff Bonwick's Blog at Oracle (2009), last accessed 08/06/13.
5. Gregg B, Mauro J: DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X and FreeBSD (Oracle Solaris Series) ISBN 978–0132091510 Prentice Hall Professional, 2011
6. Robertson ID, Saveraid T: Hospital, Radiology and Picture Archiving and Communication Systems. Vet Radiol Ultrasound 49, No. 1, Supp. 1:S19–S28,2008
7. Ballance D: The Network And Its Role In Digital Imaging And Communications In Medicine Imaging. Vet Radiol Ultrasound 49, No. 1, Supp. 1:S29–S32,2008