Journal of KOSSE. (2019. 12) Vol. 15, No. 2 pp. 66-71
DOI: https://doi.org/10.14248/JKOSSE.2019.15.2.066
www.kosse.or.kr

ISSN (print) : 1738-480X
ISSN (online) : 2288-3592

Big Data Management System for Biomedical Images to Improve Short-term and Long-term Storage

Shamweel Qamar,[1] Eun Sung Kim,[2),3)] Peom Park[1),2),3)*]

1) Systems Biomedical Informatics, Ajou University
2) Industrial Engineering, Ajou University
3) Humintec, Co. Ltd.

Abstract : In digital pathology, an electronic system in the biomedical domain storage of the files is a big constrain and because all the analysis and annotation takes place at every user-end manually, it becomes even harder to manage the data that is being shared inside an enterprise. Therefore, we need such a storage system which is not only big enough to store all the data but also manage it and making communication of that data much easier without losing its true from. A virtual server setup is one of those techniques which can solve this issue. We set a main server which is the main storage for all the virtual machines(that are being used at user-end) and that main server is controlled through a hypervisor so that if we want to make changes in storage overall or the main server in itself, it could be reached remotely from anywhere by just using the server's IP address. The server in our case includes XML-RPC based API which are transmitted between computers using HTTP protocol. JAVA API connects to HTTP/HTTPS protocol through JAVA Runtime Environment and exists on top of other SDK web services for the productivity boost of the running application. To manage the server easily, we use Tkinter library to develop the GUI and pmw magawidgets library which is also utilized through Tkinter. For managing, monitoring and performing operations on virtual machines, we use Python binding to XML-RPC based API. After all these settings, we approach to make the system user friendly by making GUI of the main server. Using that GUI, user can perform administrative functions like restart, suspend or resume a virtual machine. They can also logon to the slave host of the pool in case of emergency and if needed, they can also filter virtual machine by the host. Network monitoring can be performed on multiple virtual machines at same time in order to detect any loss of network connectivity.

Key Words : Hypervisor, XML-RPC, HTTP, JAVA Runtime Environment, Tkinter, Python

Received: December 4, 2019 / **Revised:** December 22, 2019 / **Accepted:** December 30, 2019

* Corresponding author : Peom Park, ppark@ajou.ac.kr

1. Introduction

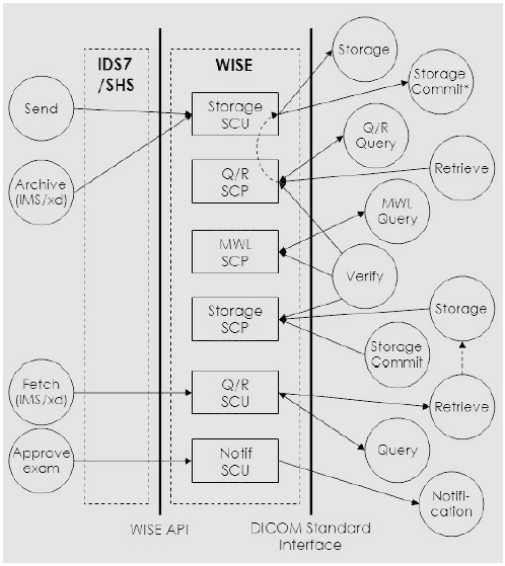
Pathology slides are extracted in the form of software image formats like .svs. The problem with those pathology slides is their inability to be shared among entities because their mag-nification is very high, and the size goes up to 1 gigabyte. To be able to diagnose the tumorous region, their quality cannot be compromised. So, we imported from the slide scanners to the database server containing WiSe and different type of entity databases. Images are accessed by the workstations through a image servers and image servers store and sequence the data to be reachable to the workstations. Architecture flow of the management system is shown in Figure 1.

Database is an important component in making this system work and there are two main components in that respect that we should talk about. The workflow of WISE database and SHS application are very necessary clogs of our application to make sure it has compatible efficiency.

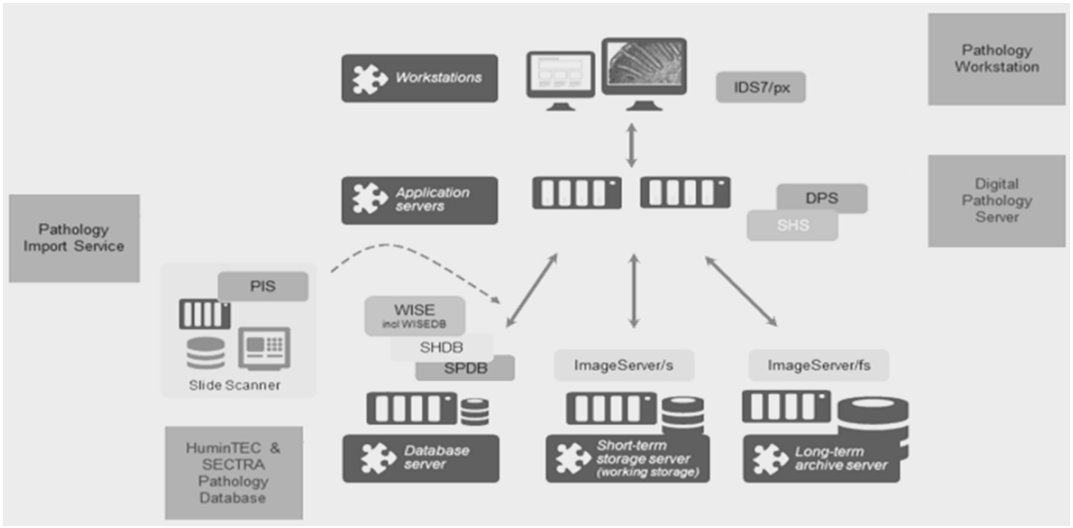
1.1 WISEDB Flow Diagram

WISE is an already build-in database tech-niques for storing text and images as well. It's exchange of data and information with storage and IDS7/SHS is shown in Figure 2. It provides (among other things) the following features:

- It replies on communication tests from remote applications.
- It allows remote applications (modalities



[Figure 2] WISE database flow diagram showing exchange with IDS7/SHS and storage device



[Figure 1] System architecture showing the flow of data from databases to workstation

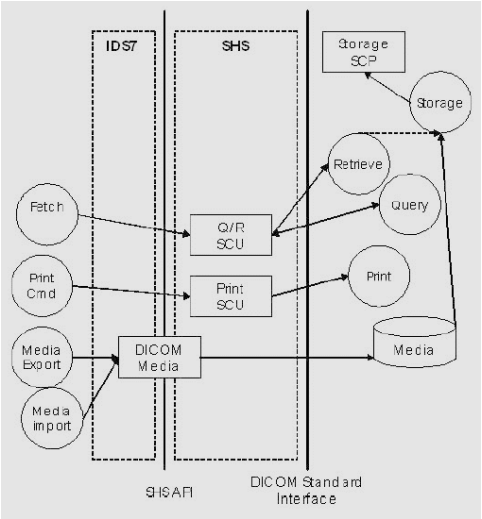
and image workstations) to send images to it.
- It allows remote applications to commit storage of sent images.
- It allows remote applications to query the WISE database and retrieve images.
- Send images to a remote application (e.g. a workstation or a DICOM archive).
- Fetch images from remote applications (typically a DICOM archive.

1.2 SHS Application Flow Diagram

SHS provides (among other things) the following features:

- Store and retrieve of images stored on a CD media. Media import and export is residing on IDS7.Media export is also an operation on SHS when CD/DVD Production Center is used.
- Print images.
- Send images to a remote application (e.g. a workstation or a DICOM archive) via WISE.
- Fetch images from remote applications (typically a DICOM archive) via WISE.

SHS contains one Application Entity (AE), Q/R SCU which only has one instance. It's exchange of data and information with storage and IDS7 is shown in Figure 3.



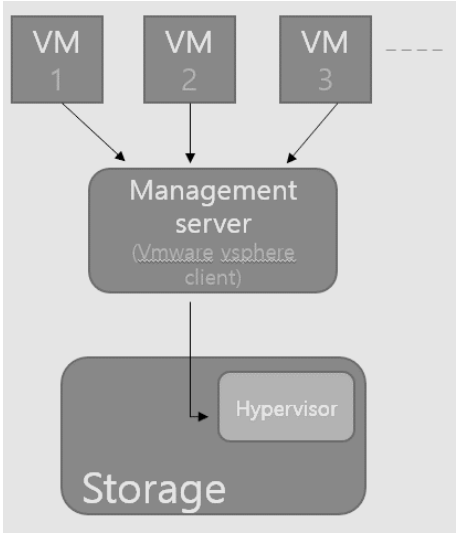
[Figure 3] SHS flow diagram showing exchange with IDS7 and storage device

2. Application

2.1 Setup of VMware Data Center

The connection from the main physical data center and the virtual machines (VM) is in multiple layers (shown in Figure 4). Main storage is set through a hypervisor which controls the data center to make it virtual. From the user's perspective, a management server is set for them to access the virtual machines through putting their IP addresses.

Data center has several constituents that have some important customized factors according to our big data system. One of them is hyper-visor, a layer to control and allocate the storage array in order to design a data center for virtualization. There were two options for the



[Figure 4] Data center setup from storage device to the virtual machines

hypervisor to use for our system. First is XEN server which has an open source access and the setup is easy to manage. It is free of cost on some scale and enterprise readiness is questionable. The other option is XEN server pool comprised of multiple XenServer hosts bound together as a single managed entity. Then virtual machines (VMs) can be started on any XenServer host in the pool that has sufficient available resources such as CPU or memory. And apparently, XenServer is the obvious choice in our case.

2.2 XEN server requirements

A resource pool is a homogeneous aggregate of one or more XenServer hosts. A XenServer pool is homogeneous when the CPUs on the server joining the pool are the same (in terms of vendor, model, and features) as the CPUs on servers already in the pool. When we join a host to a pool, XenServer will enforce additional constraints, in particular: It is not a member of an existing resource pool, it has no shared storage configured ,there are no running or suspended Virtual Machines on the XenServer host you are joining to the pool and there are no active operations on the VMs in progress such as restart or shutdown. When we want to join a new host :Clock of the host joining the pool is synchronized to the same time as the pool master (for example, by using NTP) ,its primary management interface is not bonded (you can configure this once the host has successfully joined the pool) and its management IP address is static (either configured on the host itself or by using an appropriate con-figuration on your DHCP server).

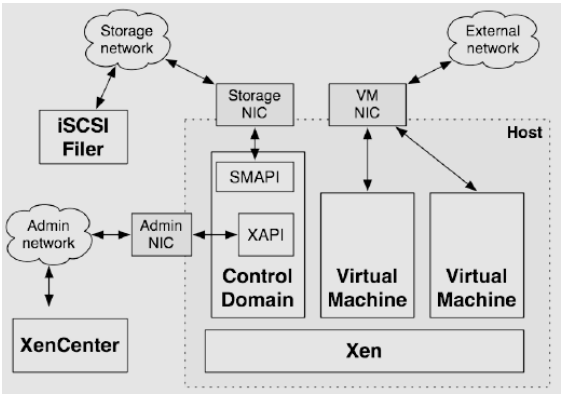
2.3 Creating a XenServer resource pool

Resource pools can be created using either the XenCenter management console or the command-line interface (CLI). When you join a new host to a resource pool, the joining host synchronizes its local database with the pool-wide one, hosted and managed by the pool master, and inherits some settings from the pool:

1. VM, local, and remote storage configuration is added to the pool-wide database.
2. The joining host inherits existing shared storage repositories in the pool and appropriate PBD (Physical Block Devices) records are created, so that the new host can access existing shared storage auto-matically
3. Networking information is partially inherited by the joining host. The structural details of NICs, VLANs, and bonded interfaces are all inherited, but policy information is not. This policy information, which must be reconfigured, includes the IP addresses of management NICs, which are preserved from the original configuration, the location of the primary management interface and dedicated storage NICs

2.4 XenServer API Setup

The software layer of hypervisor boots first which runs in 64-bit mode. Next, the control domain boots, which is a 32-bit Linux-based embedded distribution. The control domain is a normal XenServer VM and as shown in Figure 5, it runs inside the control domain. XenServer includes a XML-RPC based API, which are transmitted between computers using HTTP. There are five SDKs available, one for each of



[Figure 5] API setup of XenServer controlling virtual machines

C, C#, Java, PowerShell, and Python. VI Java API for prior project with vSphere ESXi is used which is a set of Java libraries that sits on top of existing vSphere SDK Web Services interfaces.It provides a full managed object model and run-time type checking, resulting in a dramatic productivity boost. The Source Code contains Tkinter library to develop GUI for easily managing XenServers and Pmw megawidgets library (which itself uses Tkinter as a base). Python Binding is utilized for XML-RPC based API for managing, monitoring, and performing operations on virtual machines. Classes used in this system are both for architecture perspective and from the user's perspective like session, subject, event, VM etc.

3. XenServer application features

Multiple purpose features of our system allow user to connect to a pool of XenServers via a GUI application. User can filter VMs in the pool by host and they can perform some administrative functions like restart VM, suspend VM, resume VM, etc. In-case of emergency, user can logon to the slave host of the pool. One of the advanced monitoring features is that user can set advanced network monitoring on multiple VMs at the same time. If network monitoring detects a loss of network con-nectivity for a VM, a snapshot of that VM is automatically taken.

4. Conclusion

There are existing systems which can be manipulated as pioneer work for our requirement. The key is to know our system's requirement i.e. how much data is going to be employed, the number of patients, the load of users from multiple virtual machines etc. If practically installed, the system can extremely cost effective as we do not have to install hardware system at every work station and because of the use single server, the workload management is effectively convenient for the system. From the literature review, it is obvious that visu-alization and sharing of images can be easily done even when there are multiple users on multiple virtual machine.

Acknowledgement

This research was supported by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number : HI18C0316)."

References

1. Khushi M, Carpenter JE, Balleine R, Clarke

CL. Development of a data entry auditing protocol and quality assurance for a tissue bank database. Cell and Tissue Banking ePub Feb 18, 2011.
2. Potters L, Kattan MW, Fearn P. A chrono-logical database to support outcomes research in prostate cancer. Int J Radiat Oncol Biol Phys 56:1252, 2003.
3. DICOM Conformance Statement Sectra PACS and Sectra VNA Sectra PACS, Version 19.3, October 2017.
4. Allan,C. et al. (2012) OMERO: flexible, model-driven data management for experimental biology. Nat. Methods, 9, 245.
5. de Chaumont,F. et al. (2012) Icy: an open bioimage informatics platform for extended reproducible research. Nat. Methods, 9, 690-696.
6. Kvilekval,K. et al. (2010) Bisque: a platform for bioimage analysis and management. Bioin-formatics, 26, 544-552.
7. Lobet,G. et al. (2013) An online database for plant image analysis software tools. Plant Methods, 9, 38.
8. Carpenter JE, Miller JA et al (2007) The Caisis system for biorepository data requirements—Breast cancer tissue bank, Australia. Cell Preserve Technol 5(1):51-52.
9. Fan J-W, Friedman C (2008) Semantic reclassification of the UMLS concepts. Bio-informatics 24(17):1971-1973.
10. Fearn P, Regan K et al (2007) Lessons learned from Caisis: anopen source, web-based system for integrating clinical practice and research. Computer-based medical systems, 2007. CBMS '07. In: Twentieth IEEE international symposium on 2007.
11. Mare´e, R. et al. (2013a) Extremely randomized trees and random subwindows for image clas-sification, annotation, and retrieval. Invited chapter indecision Forests in Computer Vision and Medical Image Analysis, Advances in Computer Vision and Pattern Recognition, pp. 125-142.