Migrating a Digital Library to a Private Cloud

Jian Wu*, Pradeep Teregowda†, Kyle Williams*, Madian Khabsa†, Douglas Jordan†,
Eric Treece*, Zhaohui Wu†, and C.Lee Giles*†
*Information Sciences and Technology
†Computer Science and Engineering
Pennsylvania State University
Email: jxw394@ist.psu.edu

***Abstract*—A private cloud deployment of an infrastructure as a service (IaaS) cluster is a cost effective solution to many small and intermediate digital libraries and maybe companies. As a working online digital library search engine, the physical infrastructure of CiteSeerX represents many of the clusters for a typical digital library in terms of size and functionalities. CiteSeerX used to run on a cluster consisting of eighteen loosely coupled physical machines. In this work we share the experiences and lessons learned through migrating CiteSeerX into a private cloud environment using virtualization technique. We also discuss alternative solutions including a public cloud deployment using Amazon EC2 and EBS services. We found that the private cloud via virtualization is a better model for a digital library system like CiteSeerX. We also report system status, activities and proposed variations after the new system has been running for over half a year.**

I. Introduction

Cloud computing has emerged as an attractive paradigm for both personal and large scale computational and service based projects. It features elastic resource allocation and on-demand scalability without a huge upfront investment. Successful and most popular large cloud services include Amazon EC2 and Google App Engine. Cloud computing can be generally categorized into three service models [1], i.e., Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) with several deployment models, i.e., public, private, hybrid (public+private) and virtual private (offered by Amazon Web Services).

As cloud computing gains more popularity, there has been active research in the past years on this topic. For example, an auction-based approach was proposed by Zhang et al. [2] to schedule computational resources interactively in a cloud service. There have also been attempts to migrate existing systems into the cloud infrastructure. For instance, Teregowda & Giles discussed the feasibility of moving the extraction system of the CiteSeerX digital library into Amazon EC2 cloud [3]. Chauhan & Barbar reported the lessons learned from migrating a service-oriented system to a cloud environment [4].

Cloud computing has many advantages which attract individual users, companies and enterprises to move their existing systems into it. The first is elasticity. For example, Amazon EC2 charges users on a cost per use bases, so individual users can "shutdown" their virtual machines when they are offline with no (or little) additional charge. The second is relatively low cost. For instance, [5] did a case study and concluded that the cost of hosting a source code repository using Amazon EC2/S3 was lower than hosting it locally. The third is the on-demand self-service. Most of time, a consumer can avail computing resources in an automated fashion without resorting to human interactions. Finally, using cloud services can save tremendous amount of time on system maintenance. Most public cloud services are off-premise and maintained by profesional IT. Users are not responsible for handling hardware failure, storage and cooling issues.

To our knowledge, most of research work was focused on the big public cloud services such as Amazon EC2, e.g., [6], [7]. There is a lack of publications giving principles and practical guidance on migrating small or medium size server clusters into a private cloud.

CiteSeerX is a digital library search engine which provides free access to over three million academic documents crawled from the public web. CiteSeerX used to run on a cluster of 18 loosely coupled physical servers. This is a typical size for many small or medium size service-oriented clusters in digital libraries or other related projects. Most nodes in this cluster were already running for many years (5–6) which is a property of many research systems. We had experienced occasional hard drive or controller failures, which caused permanent loss of data, delay of research progress and even downtime for online services. In addition, as CiteSeerX scales up, the existing storage and computational resources have become bottlenecks to sustain the system growth. Instead of moving each server to an individual new machine, migrating the system to a cloud infrastructure is a promising solution for both system maintainability and scalability.

The major contribution of this work is two-fold. On one hand, we rationalize the feasibility to move the system to the private cloud and list challenges during the migration project. These challenges can be common when moving any peer digital library into a cloud. On the other hand, we provide suggestions and lessons learned through the migration steps. These suggestions and lessons can be helpful for IT managers to evaluate the difficulty of their projects and decide better approaches when migrating a real system like ours.

This paper is organized as follows. In Section II, we give an introduction to the frontend and backend of the CiteSeerX digital library and describe its properties which are common for small and medium size digital libraries. In Section III, we rationalize the decision to choose a private cloud as a solution instead of a public cloud or a simple hardware replacement. In Section IV, we first list the challenges we were facing and how we tackled them in the context of detailed migration steps. We then describe several post-migration issues that we experienced, which inspired us on improving the system

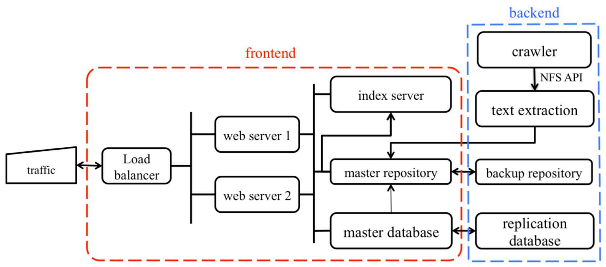


Fig. 1. The architecture of the CiteSeerX system and their main jobs. Arrows indicate data flow directions. Red dashed lines enclose the frontend; blue dashed lines enclose the backend.

design. We have a brief discussion on possible alternatives and variations of architecture in Section V and conclude in Section VI.

II. CiteSeerX As a Digital Library

As a typical digital library, CiteSeerX includes the following components. The frontend contains a web search interface, a database, an index and a large repository; the backend plays roles of information acquisition (focused crawling), metadata extraction, filtering and ingestion.

From the users' perspective, CiteSeerX provides over 3 million (after migration, updated in September, 2013) downloadable academic papers in PDF or postscript formats from which over 2.2 million are unique (after clustering similar documents). There are over 15 million unique records (documents+citations). As most of scholarly search engines as Google Scholar, the users can perform full text searches by querying keywords in a search box. A user is also allowed to create a personal account and add favorite papers into his/her personal collection. The paper summary page contains metadata extracted from the original papers including titles, authors, abstracts and citations. CiteSeerX offers a user-correction feature, in which registered users can make corrections to metadata errors. CiteSeerX also designs special interfaces for author and table searches. Most of authors are disambiguated using techniques described in [8], which results in over 300,000 unique disambiguated authors. In addition to submitting queries from the search box, users may also obtain direct links by searching general search engines such as Google or Bing. Users are also encouraged to submit URLs of crawlable PDF files to get them indexed.

From the developers/administrators' perspective, the architecture of CiteSeerX is presented in Fig. 1. At the backend, the crawler downloads PDF files and stores them in the crawl repository. The documents are passed to the text extraction server through an API. The text content of these documents is parsed and filtered so that only documents classified as academic are kept. The ingestion system, which runs on the repository server, imports the retained documents into the master production repository and writes the metadata into the database. Documents are clustered and new documents are indexed by Solr. At the frontend, the online requests (queries or direct links) access through a load balancer. The traffic is redirected to one of the web servers. The repository is mounted to one of the web servers via a global network block device (GNBD). Searching results are returned by the index server. Documents can be downloaded from the repository server. All metadata are retrieved from the database server.

TABLE I. Physical production servers.

Alias	#cores	Memory	Storage	Functionality
Web (x2)	4@1.0GHz	16GB	240GB	Web server
LB (x2)	2@1.0GHz	2GB	80GB	Load Balancer
DB-M	24@2.7GH	48GB	1.5TB	Master database
DB-R	8@2.0GHz	16GB	1.3TB	Replication database
Rep-P	4@2.0GHz	16GB	13TB	Production repository
Rep-B	8@2.0GHz	32GB	15TB	Backup repository
Index-P	4@1.0GHz	16GB	240GB	Paper index
Index-AT[1]	4@1.0GHz	16GB	100GB	Table and author indices
Ext-P	2@2.4GHz	32GB	1.5TB	Primary text extraction
Ext-A[1]	2@1.0GHz	3.5GB	1.4TB	Auxiliary extraction
Crawl[1]	8@2.8GHz	32GB	15TB	Crawler
Crawl Web[1]	4@1.0GHz	8GB	62GB	Crawler web/API
Crawl DB[1]	8@2.8GHz	32GB	394GB	Crawler database
Staging[1]	8@2.0GHz	16GB	7TB	Feature testing
DOI[1]	2@1.0GHz	2GB	80GB	DOI server
Static[1]	2@1.0GHz	3.5GB	80GB	Static web[2]

[1] Note included in Fig. 1.
[2] Static web pages like team information is hosted separately.

This architecture used to be implemented by 18 physical production servers listed in Table I. From the descriptions above, we can see that CiteSeerX represents a digital library with the following properties.

a) Medium Size: CiteSeerX repository contains about 2.6 million documents (before migration and hereafter), which takes about 4 tera-bytes of disk space. The database is 130GB (on disk before dump) and the Solr index is 70GB (after optimization). This is a medium size compared to large academic search engines such as Google Scholar and Microsoft Academic Search (about 50 million according to Wikipedia) although these giant repositories include a fraction of metadata-only papers without free full-text. As we will note later, the relatively large (and growing) repository makes it challenging to replicate and backup.

b) Steadily Growing: CiteSeerX steadily increases its collection size by crawling the web. At least 2,000 new documents are ingested daily, with the associated citations. With revised crawling policies and new hardware, we expect to reach at least 10,000 new documents daily, which is at least 3 million a year. In addition, one web server is generating on average 500MB access logs every day. These require the system to be scalable.

c) Loosely Coupled Components: The repository, index, database and crawler are hosted on separate servers. Data are pulled or pushed by RESTful or other types of APIs. Backend servers mostly run batch jobs and do not need to work as a closely bonded cluster. This allows certain servers detached without affecting the functionality of other servers. For example, the crawler repository can be unmounted from the extraction server which only stops text extraction but ingestion can continue (from extraction server to the master repository). At the frontend, if only the database server is offline the search function is still available through the index server. This gives us more flexibility to move less dependent units one at a time and makes it easy for testing and error tracking.

d) Sub-Mission Critical: Although CiteSeerX has an average traffic of 2 million hits per day (including spiders), it is different from a commercial server, e.g., a game server, which allows (almost) zero downtime. In those cases, an in-memory state migration should be considered to minimize the downtime to sub-second [9]. Empirically, a downtime of a few minutes to a few hours was acceptable. We can temporarily disable user registration and error correction features without complaints, which gives us less constraints to synchronize data.

e) Small Maintenance Team: CiteSeerX has a small maintenance team of 3–5 people, which is typical for a digital library in a research institute. Most of them are graduate students that cannot dedicate on this project. With limited funding and flow of human resources, an long-term economical system design is required to reduce operational cost. In addition, a good documentation is essential to minimize learning time for new people.

f) High Data Throughput: CiteSeerX has an average traffic of 2 million hits per day and an average downloading rate of at least 10 per second [10]. This yields an average outbound data transfer rate of up to 25TB per month. In addition, the ingestion rate is about 2,000 per day which turns up to 4GB to the repository, database and index.

The properties of such a digital library as CiteSeerX imply both degrees of freedom and constraints when performing any major upgrade. As the system components get aged, multiple issues emerged, such as hardware failures, scalability bottlenecks, computing resource deficiency, and increase of maintenance time. All of these factors motivate us to upgrade the system to keep it sustainable. We discuss three possible choices in the next section.

III. Rationale

A. System Requirements

We had three choices to upgrade our system.

1) Replace old machines.
2) Move the system to Amazon EC2.
3) Move the system to a private cloud using virtualization.

Whichever choice we make, the new hardware must have sufficient resources for computing and storage. Specially, the storage should be scalable/extendible to hold the data so that no major upgrade is necessary for at least 2 years. We rationale the changes of each server below:

Load Balancers Because the load balancer only distributes the requests but do not actually process them, a light weighted server is sufficient.

Web Servers These are the servers where CiteSeerX is deployed and actually processes incoming requests. The physical web server only has 240GB and is not sufficient to store fast increasing log files (500MB/day). Therefore, we allocate 1TB space to it so that it can hold logs up to 4–5 years.

Database Servers The size of the dumped database file is 65GB and it takes about 130GB of space after being imported into the MySQL server. The disk storages for database servers are then set to 400GB, which can always be extended when needed.

TABLE II. Baseline computing resources of new servers.

Alias	#cores	Memory	Storage
Web (x2)	4@2.5GHz	16GB	1TB
LB (x2)	2@2.5GHz	4GB	20GB
DB (x2)	8@2.5GHz	16GB	400GB
Rep (x2)	4@2.5GHz	16GB	10TB
Index (x2)	8@2.5GHz	16GB	150GB
Ext (x2)	4@2.5GHz	8GB	4TB
Crawl	4@2.5GHz	32GB	30TB
Crawl Web	2@2.5GHz	8GB	100GB
Crawl DB	4@2.5GHz	32GB	500GB
Staging	8@2.5GHz	16GB	7TB
DOI	1@2.5GHz	2GB	40GB
Static	1@2.5GHz	8GB	40GB
Total	80@2.5GHz	250GB	68.7TB

Repository Servers Repository servers host all the PDF/postscript documents, therefore disk I/O is the major bottleneck. We allocate 16GB of memory because about 80% of memory were used by system to cache frequently used files in the physical server. Although the CiteSeerX repository size is about 4TB, it grows at a rate of 2TB annually based on the current ingestion rate, so 10TB can sustain over two years before we expand it or go for another solution. Note that the ingestion also eats some disk space to store temporary files.

Index Servers The current index size is 80GB. Assuming the index size grows linearly with documents, 150GB should be sufficient for now. To speed up the indexing speed, we need at least 4GB for Solr heap memory. Because optimization may consume more memory, CPU, and disk space. We allocate 16GB of memory and 8 cores to the indexing servers.

Extraction Servers Text extraction is a CPU expensive job. We tentatively allocate 4 cores and 8GB of memory which is sufficient for the single threading case. More CPU cores and memory may be needed for multi-thread processing (Section IV-C.2). The 4TB space is allocated to store temporary files.

Staging Staging machine is a platform where we test new features before implementing them to production. It is an all-in-one machine which integrates the functionalities of web, database, repository, index, and extraction servers. As a result, we give it sufficient computing resources to hold the current repository and perform all kinds of experiments. The data on the staging server do not need to be up-to-date.

We decide to exclude the crawl-related servers from putting into the cloud. The crawler web server just provides a web interface to view the crawl progress and serves an API and the crawler database is not large (10GB). The machines hosting them were only 2 years old so they should be durable for the next 2-3 years. The crawl machine requires a huge storage which can almost occupy all the storage of a server hosting virtual machines (VMs). If we host other VMs and the crawler VM on the same physical machine, they have to share the bandwidth, which may slow down the crawling speed. In addition, as we show later, the disk I/O on VMs is in general slower than physical counterparts, which may reduce the crawl speed. The DOI server and static web servers are both light weighted. We can host them on the author/table index server, which does not have a heavy workload.

TABLE III. PRICE QUOTES FROM DELL.COM.

Server	#cores	Memory	Storage[1]	Price[2]
Web (x2)	4@2.4GHz	16GB	1TB	\$5928
LB (x2)	4@2.4GHz	4GB	250GB	\$4110
DB (x2)	8@2.4GHz	16GB	500GB	\$7188
Rep (x2)	6@2.5GHz	16GB	10TB	\$20536
Index (x2)	8@2.4GHz	8GB	250GB	\$6526
Ext (x2)	8@2.4GHz	8GB	4TB	\$11900
Staging	8@2.4GHz	16GB	7TB	\$8102
Total	84@2.4GHz[3]	152GB	39TB	\$64290

[1] Disk space after RAID 5.
[2] After multiplied by duplicate factor, e.g., x2.
[3] Average frequency.

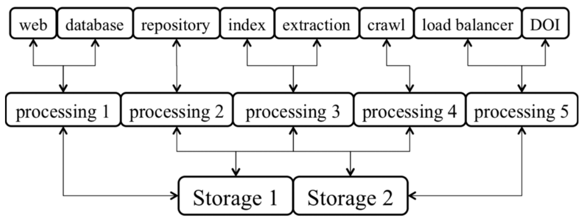


Fig. 2. Three-layer model of the cloud architecture.

Based on the these requirements and the usage history of CiteSeerX a baseline specifications for each new machine are tabulated in Table II.

B. Cost Analysis

In this section, we compare the three upgrade models through a cost analysis based on the current hardware and public cloud market.

Table III lists the price quotes of rack server PowerEdge R620 we obtained on Dell.com . We try to match the specifications in Table II, but certain items may vary. Table III indicates that the hardware cost of the new system is about \$65,000. Note that we have used all the chassises on the repository server meaning we cannot expand the storage by adding more hard drives. Also these hardware is just enough for servers in Table II. It costs extra money to purchase new servers.

We also quote the prices of moving all or part of the systems to Amazon EC2 which is a public cloud service. The migration cost to Amazon EC2 was estimated in [10], but that cost was only based on moving the *existing* data. The provision for future up scaling was not considered, so we re-do the estimation here. We use the *reserved instance* model which gives us the maximum saving. This model requires an upfront payment but with very low monthly rates. Again, we try to match the computing resources specified in Table II. To reduce the cost to the lowest level, we only implement the frontend production servers and apply a linear grow model to the disk space requested, i.e., we start from a basic level and increase the storage monthly based on data growth rates. Table IV implies that the cost for a 3-year reservation is \$177.2k and for a 1-year reservation is \$55.6k. Note that the major part of the monthly rate is the big repository storage and a high outbound data transfer rate (document downloads) which is a common property of digital libraries.

TABLE IV. PRICE QUOTES FOR AMAZON EC2.

Server	API name	Monthly (3 yrs)	Monthly (1 yr)
Web (x2)	m2.xlarge	165.44	187.40
LB (x2)	m1.medium	121.52	128.84
DB	m3.xlarge	117.86	120.78
Rep	m3.2xlarge	169.83	196.91
Index	m3.xlarge	117.86	120.78
EBS Volume	Starting Storage	Monthly Growth	Monthly Cost
Rep	5TB	512GB	512+51.2M[1]
DB	250GB	10GB	25+M
Index	100GB	5GB	10+0.5M
Data Transfer	Rate	Monthly (3 yrs)	Monthly (1 yr)
Data Transfer OUT	10MB/s[2]	2611.11	2611.11
Data Transfer IN	20GB/day[3]	0	0
AWS Support		448.23	444.02
Monthly Total Over Year(s)		**163.7k**	**47.6k**
One-time Fee Total		**13.5k**	**8.0k**
Final Payment to Amazon		**177.2k**	**55.6k**

Estimation as done using Amazon Simple Monthly Calculator. Machines are reserved for 1 or 3 years with Red Hat Enterprise Linux installed on all servers. All prices are in US dollars.
[1] M is the count of months, starting from 0.
[2] Assuming an average PDF document size of 1MB. Average downloading rate is 10 doc/s [10]. Web page access is neglected in this calculation.
[3] Assuming an ingestion rate of 10,000 per day (upper limit after migrating to the new system) and a 2MB of disk space is used per document ingested.

The third choice is to purchase a small number but large powerful machines to build a private cloud cluster providing IaaS using virtualization.

The cloud architecture (Fig. 2) is composed of three layers: the storage layer, followed by a processing layer and finally an OS/application layer. The storage layer is composed of two servers whose sole purpose is to act as storage for virtual machines. The processing layer consists of five powerful servers which are connected to the storage level. The system/application layer consists of various virtual machines running on the processing layer while data and the virtual machine themselves are stored on the storage level. We use VMware ESXi version 5.1 as the hypervisor which acts like a status monitor and a coordinator dealing with all interactions between the storage and the processing layers.

The advantages of this architecture are three fold. First, it increases the server reliability. If one processing server fails, the hypervisor can respond and move VMs on that server to another processing server. For example, moving a VM containing 4 cores and 4GB memory takes about 85 seconds and a VM with 8 cores and 16GB memory takes 180 seconds. The second advantage is a smaller footprint in the datacenter which equates to less physical space used in racks as well as a lower operating temperature and thus more efficient use of power. This allows us to add physical servers to our cluster should we need more processing power or storage. We then can move more mission critical VMs to the newly added physical servers while keeping the old servers for less critical work such as research or experiments. The third advantage is flexibility to create and delete a new server. By using a template-based workflow in a virtualized architecture, setup time has been reduced from a day, not including the time for a vendor to deliver a system, to a matter of minutes.

The plan is to purchase five processing servers and two storage servers. The computing resources and and costs are

TABLE V. Specifications and cost of production servers.

Server Type	#cores[1]	Memory	Storage[2]	Quantity	Sub-total
Processing	12	96GB	1TB	5	$35k
Storage	12	32GB	30TB	2	$24k
HW Total	84	640GB	65TB	7	$59k
Other	Power[3]	Network[4]	License[5]	Sub-total	**Total**
1-year	$4980	$6000	$2111	$8.3k	**$70k**
3-year	$14939	$18000	$2843	$21.4k	**$95k**

[1] CPU frequency 2.5GHz.
[2] After RAID 5 for each unit.
[3] Assuming a PUE value of 1.17, including electrical power and cooling.
[4] Estimated by assuming 100Mbps.
[5] Quoted from vmWare with Standard Basic support.

listed in Table V. Besides the hardware cost, we also consider electrical power, cooling, bandwidth and hypervisor license. While a university usually pays the bills for these, they are not negligible in general to build a data center. The electrical power is estimated by assuming an upper limit of energy consumption of 700W for each server and 10 cents per kWh. The cooling cost depends on the type of cooling methods, desired temperature and rack postions. A rough estimation can be made by assuming an average PUE (Power Usage Effectiveness), which is defined as the total facility power divided by IT equipment power. This value is about 1.16 for Google, 1.08 for Yahoo and 1.07 for Facebook. We assume $\text{PUE} = 1.16$. The bandwidth (network) cost is estimated by assuming $5 per Mbps per month. In addition, we need a virtual platform which allows us to build a set of virtual machines on top of it. We choose VMware vSphere for its support to Red Hat Linux Enterprise (RHEL), past reviews and usage experiences [11]. The itemized cost for each item is listed in Table V.

Comparing the three choices, Choice 3) (private cloud) is better than 1) (physical) because with a comparable cost (hardware + license), the private cloud choice provides a lot more memory and disk space than the physical infrastructure. Besides, the power consumption is much less by the cloud due to the reduced number of physical servers. Comparing 3) and 2) (public cloud), although in the short term, the public cloud is a more economical choice, in the long term (3 years or longer), the public cloud choice almost doubles the cost of building a private cloud. This reflects the elastic nature of the Amazon EC2 service. In short, we choose the private cloud solution because in the long term, with the lowest cost, it can fit all the servers demanded and still have plenty of extra resources for expansion.

IV. Migration to a Private Cloud

A. Challenges

Although virtual platform has been used for university lab machines and we have been familiar with using cloud services offered by Amazon EC2, moving such an real online service, we still face many challenges. These challenges may be common for other digital libraries and similar projects deciding to make the same move.

a) Lack of Documentation: Although the SeerSuite package [12] is shipped with a documentation folder, the information was usually limited and fragmented. A significant portion of technical details were not addressed. Like many research systems, CiteSeerX has been running for years and is mostly maintained by graduate students and postdoctoral scholars. The depart of students and lack of documentation make it difficult for new people to handle all cases, including installing from scratch. Frequent duplicate communication and extensive error-and-trials are required for new people to get adapted to the working environment and technical details. This motivates us to write a complete document on our project including all components, operations, and troubleshooting, which is an invaluable resource for future people.

b) Resource Allocation: The challenge is to find out what kinds of products/parts we should order and how many cores, memory, storage should be allocated to each new machine. First, an analysis of current usage should be performed to understand if the current computing resources are sufficient, and if not, how much more are expected. The processing power and storage roughly scales with the size of input data. For CiteSeerX, the document volume from our focused crawling basically (but not definitively) determines the growth rate of the entire dataset and further determines the hardware. After implementing a whitelist policy [13] and using Heritrix plus an importing middleware, our crawling rate increased from a few thousand documents a day to about 50,000 a day. The extraction and ingestion hardware need to be upgraded accordingly to process these documents on time. The parallelization on the roadmap also requires multi-core servers and high memory. The storage need to be large enough to fit the current datasets *and* the growth of entire system. Table II gives out the results of investigation. Note that those allocated resources can be adjusted according to their actual usage (see Section IV-C.2) which is an advantage of using virtual infrastructure.

c) System Compatibility: Like many legacy systems, CiteSeerX was designed years ago, and has been optimized on RHEL5. While this OS is still under support at the date we plan to upgrade, its full support ended on January, 2013 and the regular life cycle will end on 2017. How to install all components of the digital library based on legacy codes on RHEL6 is a challenge. CiteSeerX web apps was mostly written in Java, the extraction was mostly in perl, the web service is deployed by Tomcat and it uses MySQL as the database manager. With the newest versions of MySQL and perl, the database and extraction are all running on RHEL6. The load balancers are still on RHEL5 due to a compatibility conflict of the load balancer we were running with RHEL6. We found this by creating four *temporary* light weighted testing VM servers with two for the web deployment and two for load balancing loaded with RHEL6. The legacy system uses *heartbeat-ldirectord* as a load balancer, which is provided in the EPEL repository. We use it because it is widely used across many users and provides many good features such as session persistence, port grouping and standby take-over, which are not provided by other load balancing tools, as far as we know. We found after many attempts that *heartbeat-ldirectord* is not compatible with RHEL6 so load balancing servers have to be kept on RHEL5. Testing servers are also used when setting up the repository cluster which includes a production repository and two web servers. This cluster allows web servers to perform I/O operations to files stored in the drive exported from the repository server. Such a drive is formated to GFS but in order to export this drive to web servers over TCP/IP, the global network block device (GNBD) module must be loaded to the Linux kernel. We found that this module could not be installed to RHEL6 kernel so both of web and repository servers must be kept to RHEL5.

The last example is the index powered by Apache Solr. The legacy CiteSeerX relies on Solr 1.3. The interface in the code is not compatible with the newest Solr (v4.6.0), which has a different index format. To avoid introducing system complexity, we decide to comply with the old Solr by using the existing code and postponing the Solr upgrade for future work.

System compatibility is always a problem when upgrading a legacy system. In our case, we learned that it is effective to use testing machines to find out the system compatibility issues before moving the whole units into production. In addition, our main goal is to ensure the system is migrated and runnable, components upgrades can be performed later. This may save a significant amount of time.

d) Migration Plan: Our initial plan was to migrate the system without major modifications of the architecture (components and connections). However, the complexity of the system makes it challenging to make the migration *complete* and *seamless*. Here *Completeness* means that no data should be lost during this migration. *Seamlessness* means that there should be no or minimum down time for the service.

The VM cluster uses a different virtual IP (VIP) from the physical cluster, the new system should be fully functional before we change the DNS table to map the domain to the new VIP.

We considered two possible migration plans: synchronization vs. snapshot. Both of these plans only apply to the frontend (red circled area in Fig. 1). The initial thought of the synchronization plan was to first create the database, index and repository servers in the cloud and synchroniz them with their physical counterparts before load balancers and web servers are migrated. Ingestion can continue, i.e., new papers and metadata can be injected and are copied to the VM servers using *rsync*. A database backup can be setup using the master-slave replication in MySQL. New users can continue to register and make corrections to wrong metadata. For the incoming traffic size and ingestion rate of CiteSeerX, the slave database is almost always synchronized with the master database, i.e., any changes in the master database is reflected in the slave copy in less than a few seconds. However, a real-time synchronization is hard to be achieved for the repository due to its large size. It takes 3–4 days for the *rsync* command to finish one cycle but during the same time, the master repository is already updated with new documents. As a result, the repository in the cloud is always out of date. To achieve an absolute synchronization, the ingestion has to be halted for at least 3–4 days before we deploy the VM web servers and switch the ingestion destination to VMs. Because real-time synchronization cannot be really achieved, this is not a practical plan.

Therefore, we consider a "snapshot" plan. In this plan, the ingestion process is halted and the user registration and correction are disabled so no new data are written into the system. The databases are dumped and imported to the VM server. The repository and indexes are simply copied from the physical production server. Meanwhile, we deploy the web servers and load balancers, configure the web servers to point to the new data servers and test them until everything is ready before changing the DNS table. Note that CiteSeerX is a sub-mission critical project, so a temporary halt of ingestion is permitted. Conceptionally, the snapshot plan is simple. The database replication only needs to be setup once and no *cron* jobs are needed to synchronize data between physical and VM servers. We will present the detailed migration steps in the next section.

Migrating the backend systems, i.e., text extraction and ingestion, are less challenging because they are *loosely* coupled with the other components and has little influence on frontend production services. We only need to install the dependent applications, copy the codes and change paths.

e) Redundancy and Data Backup: Redundancy and data backup are crucial for a digital library especially for legacy data which are difficult or impossible to re-gain and large chunk of secondary data which takes too long to generate. Several strategies are carried out to protect data against unexpected system failures. First, the disk array of each physical server is built to RAID 5 to allow single disk failure. RAID 5 requires at least three physical drives, but we recommend to reduce the number of disks when customizing a server because a large number of disks reduces the ratio of "losable" drives to total drives and thus increases the chance of loosing all data. We had experiences in which two drives failed in a big array and all data were lost. Second, the storage we got from the storage machines is sufficient for us to create a backup server for repository, index and database, respectively. The vSphere hypervisor offers a feature to automatically move the data on a failed processing server to another processing server. However, these are all performed *inside* the cloud. An *external* backup to high priority data is necessary. These external storage may not be computationally powerful. For datasets less than 1TB, a portable hard drive or a workstation can take the job. For large datasets such as the repository, a NAS storage can be used. The repository, database and index data are periocally copied to the external storage. We use *github*, which is an online project hosting service, to backup the CiteSeerX code.

f) Configurations: After migration, the extraction and ingestion speed may be boosted significantly. As a result, the existing database and index configurations may not be proper for the scaled data input. Adjustments should be made to allocate more memory to MySQL buffers to ensure fast responses. This should be done *before* importing the database. At least 50% of memory of the database server should be allocated to MySQL buffers. Similarly, at least 50%–80% of memory on the index server should be allocated to Solr. For web servers,the memory allocated should be no less than 80%. In most cases, the direct cause to a web server crash is not CPU overload, but memory deficiency.

g) Security: In a physical cluster, each server has its own public IP and may not be in the same sub-domain. After moving to the cloud, we assign each public IP to be in the same sub-domain, which are all behind a universal firewall. This simplifies the firewall configuration and prohibits malicious activities such as port scanning. We place the entire cluster inside a Demilitarized Zone (DMZ) by applying a Linux virtual service (LVS) on the load balancer. This disallows any access to cluster nodes from outside and makes the virtual IP address the only point exposed to the internet. We configure *iptables* of the load balancers to limit the number of hits per minutes per IP address to avoid brute force attacks and excessive crawling.

h) Backward availability: After migration, we keep the physical cluster running for at least a few months for two reasons. First, the DNS servers around the world may not update their tables frequently. As a result, internal network may be the first to "know" the new mapping, while external network may still attempt to resolve the DNS name as the old virtual IP address. Second, for a big move to a complex system, we should always have a "B" plan in case the new system fails. Keeping the physical cluster running allows us to switch the DNS table back to match the old IP in case of any unexpected problem which causes unavailability of the new site. Although there could be some data inconsistency, this ensures availability.

B. Migration Steps

In this section, we present the detailed migration steps we carried out to move CiteSeerX into a private cloud. It follows the "snapshot" model we discussed above. All or some of the steps are applicable to many digital libraries like ours.

1) **Preparation.** It is fundamental to investigate and evaluate the existing working systems for both documentation and hardware budget estimation. This includes but not limited to monitoring the system vital signs (CPU and memory usage), studying the disk space occupancy and data growth rate to estimate the provisioned disk space, learning to install the system, trouble shooting and documenting. A comparison of different upgrade models like above can help us to decide the most feasible model. A system blueprint in regard to computing resources of each VM server should be made prior to ordering any hardware. From the blueprint, we can get a resource allocation in Table II. Based on this table, we can decide the total number of processing servers and storage servers. Possible improvements and refinements can also be proposed and discussed.
2) **Hardware Purchase.** This includes choosing the most appropriate server models, CPUs, memories and hard drives. We chose Intel Xeon E5649 because it had 6 cores and supported hyper-threading technique. Processing servers must have high memories because VM servers run on them. For our purposes, each processing server has 96GB of memory. Storage servers, instead, do not need as much memory as processing servers. Because the disk array is built in RAID 5, we try to select the maximum capacity hard drives to minimize the total number of hard drives.
3) **Testing.** This includes all kinds of testing including but not limited to installing the CiteSeerX web application, importing database, testing new version of Solr, and testing load balancing software on RHEL6. The goals are to understand system compatibility and pre-determine any conflicts between applications and OS's. For example, we tried to install *Piranha*, which is a load balancing tool based on RHEL6 and try to configure it to surrogate *heartbeat-ldirectord*. However, we were not able to use the persistency feature[1] in *Piranha* so we chose to stick to *heartbeat-ldirectord*. Testing servers leave small footprints on the physical server. If the testing fails, it is easy to create a new testing server within a time scale of minutes. The testing phase seems to delay the migration. In fact, it avoids us to play with configurations on production and makes a clean production deployment. Another thing we found through the database testing is that the MySQL on the slave should have an equal or higher version. The opposite way will fail due to incompatibility issues.
4) **Migrate Components.** Components can be migrated in the following order.
   a) There are two tightly coupled clusters in the frontend: the high availability (HA) cluster, and the GNBD cluster (see Fig. 3). The two load balancers and two web servers form the HA cluster. These four production servers are first created and the CiteSeerX web application is deployed on two web servers. The goal of this step is to make sure that incoming traffic is distributed across two real (web) servers and the front page is displayed. Searching is unavailable yet. However, we correctly configure index, repository and database servers.
   b) The GNBD cluster includes the two web servers and the repository server. The whole repository is copied to the VM repository server. After mounting the drive from the repository server to the web servers as a GNBD device, we should be able to download documents directly. Note that ingestion is still allowed so the VM repository may not contain exactly the same content as the physical repository.
   c) The ingestion is stopped. The user registration and correction features are also disabled. These ensure that no new data are injected into the existing datasets. An *rsync* command is run on the VM repository server to ensure that it contains the up-to-date documents. Because the majority of documents are already in place, this synchronization does not take long.
   d) The index VM is created and the index data is copied. Solr is deployed within Tomcat. After this step, users should be able to make queries from the VM web

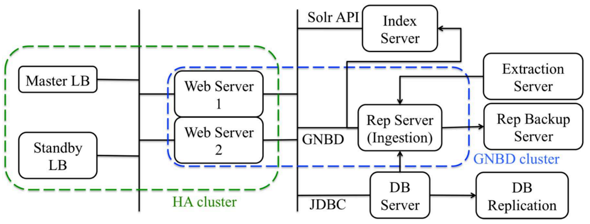


Fig. 3. HA cluster (green) and GNBD cluster.

[1]The persistency allows connections to be bonded to one real server in a single session. It also enforces the load balancer to group *http* and *https* requests from the same IP to be processed by the same real server.

TABLE VI. Connection speed testing.

Location	Physical (ms)	VM (ms)
University Park, PA	5.2±1.7	7.3±6.5
Beijing, China	442.4±1.5	444.5±4.2
Cape Town, South Africa	343.8±133.3	339.6±112.5

interface.

e) Meanwhile, all databases are dumped. For a 100GB database like ours, dumping only takes a few hours. The VM database server is created, and proper configuration should be performed *before* importing the dumped file (see Section IV-A.6). At the end of this step, the paper summary page should be available and the digital library search engine should be fully functional.
f) The whole VM system should be run for at least a week and undergo some pressure tests to ensure sustainability to the current traffic volume. Apache JMeter is one of the tools that can be used.
g) The DNS table is updated so that the domain name is mapped to the new VIP. This makes the new CiteSeerX publicly visible. The user registration and corrections features are enabled.
h) The backup servers are created in the cloud and *cron* jobs are scheduled to periodically backup data.
i) The text extraction servers and staging servers are moved to the cloud. The ingestion system is installed and run on the repository server. The text extraction, ingestion start again.

The component migration alone may take up to two weeks with most of time spent on copying data and testing. There is essentially no noticeable downtime except that the user registration and correction are disabled for that period which is a common action for website maintenance.

C. Post-Migration

CiteSeerX has been running on the private cloud infrastructure for over seven months since it was migrated. We periodically monitor system performance by checking vital signs such as CPU and memory usage, network traffic and disk usage. In general, the performance is comparable or better than the physical system. Here we share some experiences and cases we encountered.

a) Connection Speed: After the migration was completed, we performed test connections to the CiteSeerX front pages on the physical cluster and the cloud using Apache Benchmarking tool (ab[2]) in three different locations in the world. In each test, we submit in total 10000 requests with 100 concurrent requests. The mean connection time (in millisecond) is tabulated in Table VI, which indicates the response time of the cloud system is comparable to the physical one.

b) System Vital Signs: After migration, we continuously monitor the VM production servers using the *sar* tool provided in the *sysstat* package. The frontend servers are monitored during the first 24 hours after the DNS table is updated. The extraction server is monitored when multiple metadata extraction processes are running. The values of vital signs are tabulated in Table VII, which indicates that CPUs in most

[2]http://httpd.apache.org/docs/2.4/programs/ab.html

TABLE VII. System resource usage after migration.

Server	CPU%	CPU Load	memory%	rx[1]	tx[1]
LB	$< 1\%$	< 0.01	52%	90	55
Web	2%	0.5	15%	1500	15000
DB	2%	1	60%	90	250
Rep	$< 1\%$	0.3	77%	35	400
Index	4%	0.2	10%	4	40
Ext	80%	6.5	23%	3500	50

[1] Received flux (rx) and transit flux (tx) in kB/s.

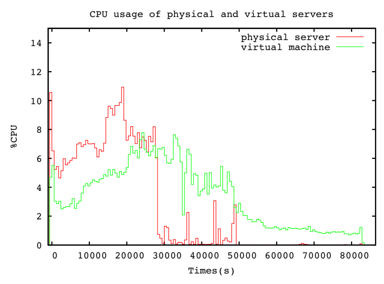
of VMs servers are under-used. In particular, the CPUs of most servers are used for less than 5%. In contrast, the text extraction server, with 4 cores, uses 80% of CPU, and the average CPU load is 6.5, indicating that the cores on this server are insufficient to process the job queues. In addition, more memories should be allocated the servers whose memory is over 60% used, e.g., DB and Rep. Therefore, we reduce the CPU on Web servers by 50%. The CPU usage increases to about 10% after this change. The CPU cores on the other web server, the database, repository and index servers are also reduced by 50%, so that more cores are released. *Lessons learned: system resource allocations should be adjusted based on their usage. The flexibility to adjust these resources is a great advantage of the cloud environment.* However, it is safer to overestimate the resource usage at first to avoid system crash.

TABLE VIII. Data transfer rates.

From	To	Database dump	Repository
Physical	Physical	6.2MB/s	4.5MB/s
Physical	VM	4.2MB/s	4.2MB/s
VM	VM	5.1MB/s	3.9MB/s

c) I/O Performance on Virtual Machines: Data I/O between VMs hosted by the same storage server can be slower than across real servers. We did experiments by monitoring the data transferring rates when copying data using *rsync* between VMs or across VMs and physical servers (Table VIII). In the first senario, we transfer a CiteSeerX database dump, which is a single 64GB file. In the second senario, we do the CiteSeerX repository backup of 3.6TB, which contains many small filers. The measurements indicates that the data transfer across VMs is in general slower than with physical servers by about 13%-17%. Transferring a dataset containing a large number of files is in general slower than a single data file. Because the sizes of data transfers in production servers are in general small, the relatively low transfer rate is not noticeable. However, the effect is more obvious when we perform repository backups. *Lessons: the backup/replication servers should be hosted in different physical servers to minimize the delays caused by a reduced data transfer rate between virtual machines.*

With the spare hardware resources in the cloud, we performed a series of crawling experiments and compared the crawling performance to a physical machine with comparable hardware. We use Heritrix 1.14.4 to crawl starting from 1000 seed URLs selected from our whitelist [13] with a depth of two and 50 threads. The CPU usage percentages and bandwidth of incoming traffic for the physical and virtual servers are shown in Fig. 4, respectively. The CPU usage percentages are comparable but the crawler on the physical server finishes sooner than the virtual server. The physical server finishes



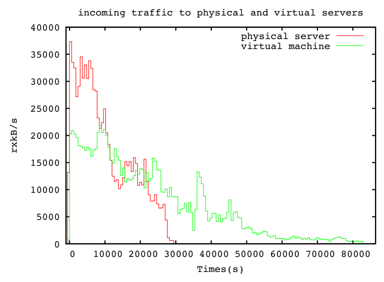


Fig. 4. Comparison of CPU usages (upper) and incoming traffic (lower) between physical and virtual servers.

crawling in about 8 hours but the virtual server is still crawling after 24 hours. We believe that the significant difference is due to the relatively inefficient I/O performance in virtual machines. Thus we exclude the crawling server from cloud migration. Currently, we are using a separate server for focused crawling.

d) System Expansion: The legacy system contains two web servers. Incoming traffic is (almost) evenly distributed to these two web servers so each server is not overloaded. This relieves the payload of a single server, and is appropriate when traffic volume is normal. However, if the total traffic is beyond what a single web server can handle, a two web server model has the potential hazard of system failure. If for any accidental reason, one of the web servers is down, all traffic has to be directed to the other one causing it to crash due to overloading. This occurs about five months after migration when CiteSeerX was heavily crawled by a small number of spiders. In addition to applying exclusion rules in the `robots.txt` and limiting connection rate in iptables, it is desirable to add the third web server to the existing HA cluster. With the under-used VM resources, no hardware purchase is required.

We are also planning to add the secondary database replication server. Currently, we only have one database replication server. For research purposes, it is often desirable to use the most updated database. Supposing we allow the users to access the current replication database, if it crashes due to a bad query or overload, the master database looses its only backup, which increases the hazard for database unavailability and data inconsistency.

Lessons learned: for a digital library in a cloud, it is always valuable to add additional storage. Multiple redundancy and backups are necessary to ensure data availability.

V. Discussion

In this section, we propose and discuss some possible variations of the current cloud architecture for CiteSeerX. That is, what improvements we can make in the future.

The first is to use a network storage rather than a repository server. The obvious advantage of this approach is to save more CPU cores, and memory. We may also be able to upgrade the Web server OS to RHEL6. Of course, the ingestion code needs to be modified so it can run on a separate server. However, we must ensure that the mounting device supports “fencing”. Fencing is the disconnection of a node from the cluster’s shared storage when the node is detected to be failed. In the current architecture, the GNBD device has built-in fencing. If we chose the network storage and mounted it to web servers using iSCSI. We either needs hardware fencing or the iSCSI target must support SCSI-3 persistent reservations. This is an alternative to replace the current GNBD model but further investigation is needed to ensure the cluster components meet all SCSI fencing requirements.

Another change to the architecture is to switch the positions of masters and replications/backups. Currently, the master database, repository and index servers contain the most up-to-date data, which ensures data freshness. However, this configuration has some potential hazard of causing data inconsistency and unavailability. For example, if the master repository server is down, the user correction still writes metadata into the database. Normally, a new version of metadata file is generated and written to the repository server, but this cannot be done in this situation, causing data inconsistency between the database and repository because the new version of metadata file does not exist in the backup. In fact, if the repository backup is not fully synchronized to its master (which is very likely), all the differences between them are lost. The inconsistency issues exist in the database servers but is less serious because the “seconds_behind_master” is usually small (a few seconds). Using the backup as the production is especially beneficial for the repository server (assuming we are not moving to the network storage) because the production server do not need to run the ingestion code any more. One drawback of this solution is that the production repository may not be up-to-date, i.e., some papers may not be able to be downloaded for a period of time. However, the inconsistency time scale is short (less than the synchronization time scale). However, this approach prioritizes the data completeness and significantly alleviates the workload on the production, thus reduces the chance of data loss and inconsistencies. This approach may not apply to the index server because during the data transfer, the index data on the production may be fragmented. The Solr version we use does not support replication. An upgrade to Solr 4 is necessary to achieve a real-time backup.

There are still several bottlenecks that are obstacles for CiteSeerX to scale up, as addressed in [14]. Migrating CiteSeerX into the VM is an important step in solving these bottlenecks. One of the issues is that as the repository grows, it takes a longer time to backup. It also increases the potential risk of losing data. A promising solution is to use the Hadoop Distributed File system, which uses commodity hardware to create multiple replicas of the same document on different machines [15]. HDFS has been under active development in the open source community, as well as by many consulting companies that provide enterprise level support, i.e., *Cloudera*. Under this approach, the application could deal with a single repository, which is on top of HDFS, and the reads and writes are handled by the file system itself.

VI. Conclusion

In this work, we discussed the motivation, requirements, feasibility of migrating CiteSeerX digital library to provide an IaaS model in a private cloud. We report the challenges we encountered prior to and during the migration. We also report the post-migration issues and possible solutions. CiteSeerX represents a typical small or medium size digital library and similar projects in terms of its size, architecture, availability, maintenance team size, and data throughput. These digital libraries are likely to have the same or similar challenges when upgrading their systems. Our experience indicates that moving to a private cloud is a cost-effective solution *in the long term* compared to a public cloud model such as Amazon EC2. The major cost of the latter is due to huge disk storage and high outbound traffic. In addition, the private cloud solution provides us more flexibility to extend the system and create/delete new VM servers. The vSphere hypervisor automatically moves failed servers to healthy ones. The major challenges include lack of documentation, resource allocation, system compatibility, a complete and seamless migration plan, redundancy/data backup, configuration, security and backward availability. The major lessons we learned through the migration are summarized below: (1) An up-to-date and complete documentation can significantly reduce the length of learning and investigation time; (2) For a digital library, the critical factor which drives the system expansion is the size of the data and its growth rate; (3) Testing machines should be used before putting changes to the production, to save a lot of time; (4) For a sub-mission critical project, a snapshot plan is a better choice; (5) External backups are necessary to protect data in the cloud; (6) New system configurations should leave sufficient room for data up-scaling; (7) After migration, old systems should keep running for a certain period of time if possible to ensure service availability; (8) Web crawlers that are hosted with other servers in the virtual environment may have a reduced crawling rate compared to a separate dedicated server. With the growth of data, it is necessary to project new approaches to store and backup the repository. The HDFS is a promising solution.

Acknowledgment

We acknowledge partially support from National Science Foundation.

References

[1] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," in *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications*, ser. AINA '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 27–33. [Online]. Available: http://dx.doi.org/10.1109/AINA.2010.187

[2] Z. Zhang, R. T. B. Ma, J. Ding, and Y. Yang, "Abacus: An auction-based approach to cloud service differentiation," in *Proceedings of the 2013 IEEE International Conference on Cloud Engineering*, ser. IC2E '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 292–301. [Online]. Available: http://dx.doi.org/10.1109/IC2E.2013.43

[3] P. Teregowda and C. L. Giles, "Scaling seersuite in the cloud," in *Proceedings of the 2013 IEEE International Conference on Cloud Engineering*, ser. IC2E '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 146–155. [Online]. Available: http://dx.doi.org/10.1109/IC2E.2013.41

[4] M. A. Chauhan and M. A. Babar, "Migrating service-oriented system to cloud computing: An experience report," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, ser. CLOUD '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 404–411. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2011.46

[5] M. Siegenthaler and H. Weatherspoon, "Cloudifying source code repositories: how much does it cost?" *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 24–28, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1773912.1773919

[6] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2011.66

[7] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of ec2 cloud computing services for scientific computing," in *Cloud Computing*, ser. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, D. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Springer Berlin Heidelberg, 2010, vol. 34, pp. 115–131. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12636-9_9

[8] P. Treeratpituk and C. L. Giles, "Disambiguating authors in academic publications using random forests," in *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, ser. JCDL '09. New York, NY, USA: ACM, 2009, pp. 39–48. [Online]. Available: http://doi.acm.org/10.1145/1555400.1555408

[9] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251223

[10] P. Teregowda, B. Urgaonkar, and C. Giles, "Cloud computing: A digital libraries perspective," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, 2010, pp. 115–122.

[11] P. Venezia, "Virtualization shoot-out: Citrix, microsoft, red hat, and vmware," *InfoWorld*, April 2011. [Online]. Available: http://www.infoworld.com/d/virtualization/virtualization-shoot-out-citrix-microsoft-red-hat-and-vmware-666

[12] P. B. Teregowda, I. G. Councill, R. J. P. Fernández, M. Khabsa, S. Zheng, and C. L. Giles, "Seersuite: developing a scalable and reliable application framework for building digital libraries by crawling the web," in *Proceedings of the 2010 USENIX conference on Web application development*, ser. WebApps'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 14–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863166.1863180

[13] J. Wu, P. Teregowda, J. P. F. Ramírez, P. Mitra, S. Zheng, and C. L. Giles, "The evolution of a crawling strategy for an academic document search engine: whitelists and blacklists," in *Proceedings of the 3rd Annual ACM Web Science Conference*, ser. WebSci '12. New York, NY, USA: ACM, 2012, pp. 340–343. [Online]. Available: http://doi.acm.org/10.1145/2380718.2380762

[14] J. Wu, P. Teregowda, M. Khabsa, E. Treece, D. Jordan, S. Carman, P. Mitra, and C. Giles, "Scalability bottlenecks of the citeseerx digial library search engine," in *Proceedings of Large-Scale And Distributed Systems for Information Retrieval (WSDM'12)*, Oct. 2012.

[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, ser. MSST '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/MSST.2010.5496972