Cloud Computing: A Digital Libraries Perspective

(To appear in IEEE Cloud 2010)

Pradeep Teregowda, Bhuvan Urgaonkar
Computer Science and Engineering,
Pennsylvania State University,
University Park, PA 16802, USA

C. Lee Giles
Information Sciences and Technology,
Computer Science and Engineering,
Pennsylvania State University,
University Park, PA 16802, USA

***Abstract*—Provisioning and maintenance of infrastructure for Web based digital library search engines such as CiteSeer^x present several challenges. CiteSeer^x provides autonomous citation indexing, full text indexing, and extensive document metadata from documents crawled from the web across computer and information sciences and related fields. Infrastructure virtualization and cloud computing are particularly attractive choices for CiteSeer^x, which is challenged by both growth in the size of the indexed document collection, new features and most prominently usage. In this paper, we discuss constraints and choices faced by information retrieval systems like CiteSeer^x by exploring in detail aspects of placing CiteSeer^x into current cloud infrastructure offerings. We also implement an ad-hoc virtualized storage system for experimenting with adoption of cloud infrastructure services. Our results show that a cloud implementation of CiteSeer^x may be a feasible alternative for its continued operation and growth.**

I. Introduction

Cloud computing [4] offers information retrieval systems, particularly digital libraries and search engines, a wide variety of options for growth and reduction of maintenance needs and encourages efficient resource use. These features are particularly attractive for digital libraries, repositories, and search engines such as CiteSeer^x [12], [8]. The dynamic and elastic provisioning features of a cloud infrastructure allow rapid growth in collection size and support a larger user base, while reducing management issues.

The pace of growth of information available on the Web and the challenge in finding relevant and authoritative sources of information make information retrieval systems such as CiteSeer^x very useful. CiteSeer^x is an application instance of SeerSuite [22], a framework for building digital libraries, repositories and search engines. SeerSuite was developed as a result of extensive research and development with the goal of enabling efficient dissemination of scientific information and literature. The autonomous citation indexing feature [11], exhaustive document metadata and the size of documents indexed make CiteSeer^x an effective and popular tool in research.

SeerSuite adopts a service-oriented architecture and takes advantage of numerous open source applications, modules and components. Prominent among these are Apache Solr [2] for indexing documents and relations, the java spring framework for applications and user interfaces, and metadata extraction modules written in perl and java. SeerSuite uses state of the art, machine learning methods for automated metadata extraction, autonomous citation indexing, author disambiguation and other proposed features. The use of these application components have helped reduce feature and software development time and improve efficiency of maintenance and operations. As such, an examination of cloud computing infrastructure is particularly relevant for SeerSuite application instances as in CiteSeer^x.

The rest of this paper is arranged as follows, We begin with background and related work in section II and describe the architecture of SeerSuite and its deployment in CiteSeer^x. We discuss cloud computing infrastructure in the context of SeerSuite in section III, and the cost estimation for hosting CiteSeer^x on cloud offerings in section IV. We examine in detail optimizing cloud hosting by placing selected services in the cloud in section V. Virtualization can be part of SeerSuite implementations and is discussed in sections VI and VII. We discuss future work in VIII and conclusions in IX.

II. Background and Related Work

Cloud computing and infrastructure have been discussed in the context of information retrieval systems which are related to grid computing and distributed computing [15]. In these discussions, the focus has been either on cloud computing for data storage infrastructure or the compute infrastructure. A discussion of cloud computing for scalability in preprocessing, harvesting, transformation and storage is provided by [19] in the context of crawling the Web with Sindice.

In the following sub sections, we briefly overview the architecture of the SeerSuite application framework and its deployment as CiteSeer^x. This section is meant to provide an understanding of the application requirements that are crucial for understanding the challenges and the need for infrastructure virtualization and abstraction.

A. SeerSuite Architecture

Figure 1 provides an overview of SeerSuite architecture. SeerSuite includes components of both web search systems [18] and digital libraries [3], [5]. SeerSuite consists of several components and services loosely coupled together with REST and SOA interfaces. We group components of the framework into Web application, data storage, extraction, ingestion, and maintenance systems. These components can function as standalone applications.

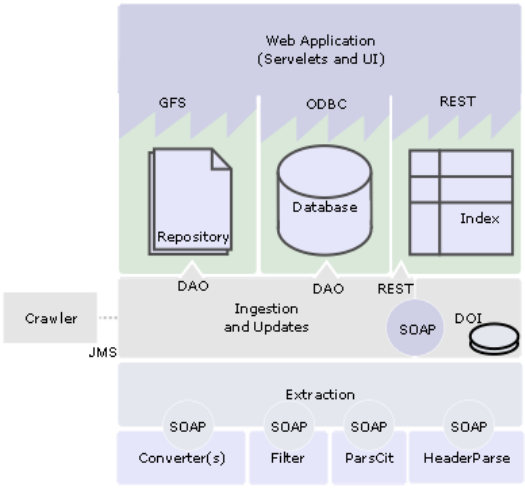

Fig. 1. SeerSuite Architecture

1) Web Application: The primary interface between users and SeerSuite is the Web application. Users can search, browse, and traverse the collection through this interface. Queries in SeerSuite are supported through an interface over REST to an index, an inverted file data structure which allows for fast and accurate retrieval. Results obtained from the index are processed by the application and with the database displays search results. Users can view document metadata from summary pages. These summary pages allow a user to download a cached copy or be directed to another source, view citations from the document, view tags created by users, and track changes to the document.

The autonomous citation indexing feature enables users to rank documents based on citations to the document inside the collection as well as browse the collection through citation links. These features require the citation graph structure of the collection to be stored in the database. In addition to providing user access through web pages, SeerSuite supports an Open Archives and an Application Programming Interface, which provide programmatic access to the database.

CiteSeer^x supports myCiteSeer, a personalization portal which with support of the database, allows users to store queries, document portfolios, tag documents and make corrections to document metadata.

2) Crawling and Metadata Extraction: The document acquisition process of SeerSuite includes two major tasks. The first is to obtain documents relevant to the collection. A focused crawler [7] traverses the web to find documents relevant to a particular topic. In SeerSuite, a focused crawler scans web pages from a crawl list and fetches documents, mostly PDF, embedded in these pages. Before metadata extraction can take place, the incoming documents are filtered and converted into text by filtering and document conversion modules. Conversion modules use standalone off the shelf applications such as PDFTet or PDFBox for converting documents. SeerSuite adopts state of the art metadata extraction methods to extract document metadata using heuristics and machine learning methods. The header parser [17] extracts document title, author, abstract and affiliation information. A reference parser, ParsCit [9] extracts reference information including citations and citation metadata. Documents can be processed individually or through a batch process, using Business Process Execution Language(BPEL), orchestration of individual service oriented modules.

3) Ingestion: Processed documents from the extraction system are ingested into the collection. In the first step, the metadata extracted from the documents and metadata from the crawl are aggregated into a single file, which serves as the source for updates to the database. Ingestion involves the database, the repository and the extraction systems. This step adds information about the incoming document and embedded objects to the database and updates the citation graph structure. The document metadata, processed files with metadata extracted by different modules, and original document are placed in the repository.

4) Maintenance: Updates to the index, generating charts and statistics are part of the maintenance service. The maintenance flow generally involves the database, the index and the repository.

5) Federated Services: While SeerSuite offers several features for document/citation search and indexing, services such as table, algorithm, figure search are also valuable to users. However, such services may have their own stack of components and interfaces. The table search feature in CiteSeer^x is one such service. The components are hosted along with existing CiteSeer^x services, but the interface and storage services for table search are independent of the main index and database. Such a federated approach is useful for services still being researched or developed.

6) Backup and Replication: CiteSeer^x maintains both offline and onsite backups for the repository, database and index.

B. Deployment

CiteSeer^x indexes more than one and half million documents currently serving two million hits every day. It utilizes several terabytes of storage for storing the cached copies of crawled documents, database, index and information extracted from documents. Services supporting CiteSeer^x are distributed across several machines. Infrastructure for CiteSeer^x services is off the shelf and heterogeneous in nature.

Figure 2 shows the deployment of SeerSuite for CiteSeer^x. The components are labeled as follows. The group of machines labeled 1 refer to the load balancers in a hot-cold cluster. These direct traffic arriving to the Web server cluster labeled 2. The index, based on Apache Solr, for documents and tables are grouped under label 3. The Web servers are supported by the database in group 4. CiteSeer^x uses a MySQL database. The document cache requests and storage for the system are handled by the repository labeled 5. This repository is shared with Web servers using the Redhat Global File System (GFS) over Global Network Block Device (GNBD). The block labeled 6 is the ingestion system, which operates from the

machine hosting the repository. It interacts with the database, repository and DOI service. The crawler labeled 8, makes use of an NFS based storage labeled 7, shared with the extraction system at 9.

The infrastructure consists of heterogeneous systems; extraction systems with dual core processors, 3 GB of RAM, web servers and index with two dual core CPUs, 16 GB of RAM, and storage systems with two dual core CPUs, 16 GB of RAM and 11 TB of storage space. Services are distributed based on expected load and functionality.

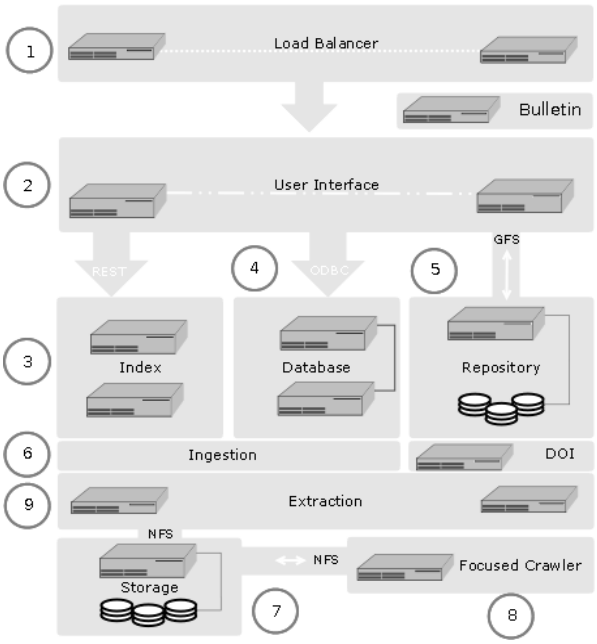

Fig. 2. CiteSeer^x Infrastructure Overview

Limitations: This setup poses limitations on future growth. Shared storage over GNBD and GFS, without an underlying storage infrastructure such as a NAS or SAN, cause locking of the storage space with disruption to other services. Additionally, growth in services offered places considerable stress on compute resources. Growth also exposes the lack of resources for management of infrastructure and services.

III. Cloud Infrastructure for CiteSeer^x

SeerSuite based applications are particularly suited for taking advantage of virtualized hardware resources and cloud infrastructure. Virtualization of resources can reduce the maintenance requirements considerably. Dynamic computational needs of processing documents and Web applications can take advantage of the elasticity provided by cloud infrastructure. With some exceptions in the Web application particularly the myCiteSeer component, most of the data stored with SeerSuite instances is freely available on the Web. This aspect of SeerSuite data enables easy adoption of cloud infrastructure.

Before such a deployment can be considered, the cost and feasibility of hosting these components and processes on the cloud need to be studied. The key questions we are interested in answering is *whether moving CiteSeer^x to a cloud would be (a) feasible and (b) cost effective.* With respect to (b), we are interested in the following refinement: instead of moving the entire application to a cloud, how cost effective would it be to move well chosen components to the cloud and host the rest locally ? In the remainder of this section, we discuss the feasibility of being able to run components in a cloud as well as the amount of effort needed for such a migration. We also conduct a qualitative discussion of cost concerns. In the next section, we conduct a quantitative analysis of the cost involved in moving CiteSeer^x components to representative cloud environments. We look at cost analysis for moving parts of CiteSeer^x to a cloud in section V.

A. Services and the Cloud

All services supporting SeerSuite can be hosted on cloud infrastructure with modifications. We study various components of CiteSeer^x and their characteristics, to help understand any advantages or disadvantages obtained by placing these components on the cloud. We discuss this issue in two contexts, the effort required to migrate a component or service to cloud infrastructure and the cost of such placement.

1) Web Application: Other then the user interface hosted on the Web servers, the database, the index and repository are part of the the application response to user requests. The variation in user load makes these services attractive as candidates for hosting this application on cloud infrastructure.

- Migration effort: In addition, the application would require modifications to state information stored as part of myCiteSeer.
- Costs: The Web application has a large footprint, in traffic passing through it, hence likely to be expensive to host this service on current cloud offerings.
- Other considerations: This service has significant data access requirements, hosting is linked to the location of the database, index and repository services.

2) Metadata extraction and Conversion: The metadata extraction and conversion services are responsible for acquiring documents for the collection. If the documents are acquired one at a time (when users submit documents) the extraction and conversion services are not resource intensive. In the case of batch processing documents, the process is resource intensive for the period of operation and benefits from elasticity. When batch processing documents, infrastructure can be dedicated to the service and once processing is complete resources can be freed.

- Migration effort: Most extraction and conversion modules, use multiple programming languages and applications, which may not be supported by most current cloud offerings. Refactoring effort is significant.
- Costs: The costs are likely to be reasonable as the traffic through this process is limited.

Thus, metadata extraction services in the batch processing mode are strong candidates for hosting on the cloud. This fact has been borne out by earlier literature covering distributed

computing [15], use of peer to peer resources [21] and cloud computing [19] efforts.

3) Ingestion and Maintenance: The ingestion system has one of the smallest footprints in the system, This process includes obtaining the digital object identifiers and methods for adding processed documents into the collection.

- Migration effort: Minimal, the code is homogeneous and supported, with minor modifications required.
- Costs: The traffic flow through this process is low. Hence costs are likely to be reasonable.
- Other considerations: If placed on the cloud by itself, some modification to other services will be required.

4) Focused Crawler and Submissions: Focused crawlers are responsible for acquiring documents from the Web for addition to the collection.

- Migration effort: Minimal, the code is fairly homogeneous and supported in many cloud offerings.
- Costs: The traffic flow through this service is minimal. Hence costs are likely to be reasonable.

Since the load factor of the crawler is fairly elastic, it is well suited to take advantage of the underlying infrastructure. This fact is also borne out by [19].

5) Maintenance: The maintenance functions are responsible for updating the index, generating citation graphs, statistics, and inference based corrections to document metadata.

- Migration effort: Minimal, the code is fairly homogeneous and supported in many cloud offerings.
- Costs: The maintenance system by itself consumes very little traffic and does not generate significant amounts of data. Thus, we feel hosting costs are likely to be minimal.
- Other considerations: Each of the maintenance service interactions occur over large sections of data which includes those stored in the database and the repository.

Thus, the maintenance system must be based near to the database and repository. Similarly the placement of the data should be near where it is used for computation [14].

6) Federated Services: Federated services include services not part of the framework which share the main CiteSeer^x infrastructure. These services may use separate databases, index.

- Migration effort: Most services offered under federated services are still under development or in research so it will require minimal effort to adopt these services in the cloud.
- Costs: The size of the metadata processed by most of these services is much less than the size of the data processed by the production services. Hence costs are likely to be minimal.

IV. Estimation of cost

In this section, we explore the cost of hosting CiteSeer^x in the cloud. Costs are estimated from publicly available figures at the websites of cloud providers [1], [13]. We provide a break down of costs for setting up the system and hosting for a 30 day month. In these scenarios, backup and staging systems are maintained locally on an incremental basis. The cost of data transfer for these backups are not included. Federated service costs are not included. Calculations are made using figures provided in Table I.

The cost for hosting and running CiteSeer^x at present is provided gratis by the College of Information Sciences and Technology at The Pennsylvania State University. The hardware infrastructure is provided from funding by sponsors of CiteSeer^x.

We consider Amazon EC2 and the Google App Engine cloud offerings, since SeerSuite can easily be adapted to run on these services without major refactoring of SeerSuite. Refactoring is required when using Software as a Service platforms and support for libraries and frameworks are not available. Statistics are obtained for each component by using iostat or by log analysis.

Type	Value
Documents	1.5 (1,518,739) Million
Database	120 GB
Repository	1.5 TB
Index (Document and Citation)	31 GB
Index (Tables and Authors)	750 MB
Traffic (Hits - Average) per day	2 Million
Growth per week	4000 Documents
Crawled documents	5000 documents

TABLE I
CiteSeer^x Profile

A. Amazon EC2 and EBS

Amazon EC2 offers an infrastructure based approach to cloud computing. SeerSuite can take advantage of this approach, since its underlying architecture is scalable and robust. SeerSuite components are amenable to replication with very few state linked features. The amount of modifications required for the conversion and metadata extraction processes are minimal.

The application is assumed to be hosted on EC2 and the data storage on EBS. A reserved instance is considered, due to benefits in costs offered by Amazon for such instances. This estimation assumes that all services provided by SeerSuite are hosted in the cloud. Some of the services such as the DOI issue service are not included, since its footprint is very small compared to other services. Billing is determined by the data to be stored, the number of I/O operations and data transferred. Data used for these calculations are provided in the tables I and II. Table II provides the transactions per second observed for particular systems for the disk unit storing the operating system and the disk unit storing data.

The cost estimate includes a total of 6 machines with a one-to-one mapping for the metadata extraction, repository, database, index, web application, and crawler with standard large reserved instances. The initial data transfer is assumed to be free at this time. Estimates are provided in the table III. The cost of hosting the services is $7000 for the initial setup and a recurring cost of $1378.2 per month. This calculation

System Function	TPS (OS)	TPS (data)
Database	37	97
Repository	27	42
Index	40	61
Web server 1	8	12
Web server 2	7	6
Extraction	4	15
Crawler	1	4

TABLE II
CITESEER^x DISK AND APPLICATION STATISTICS

Amazon Cloud Cost Estimates				
Setup	Instance	Units	Cost/Unit	Total($)
	Large Res.	6	1400	8400
Monthly	Type	Units	Cost/Unit	Total
	Instance	6	0.15/Hour	648
	Data In	150 GB	0.10/GB	15
	Data Out	3 TB	0.15/GB	450
	Storage	1717 GB	0.10/GB	171.70
	I/O	321.4 + 614.3 M	0.1/1M	32.1+ 61.4
	Total			1378.2

TABLE III
COST ESTIMATION FOR HOSTING CITESEERX ON AMAZON EC2 WITH DATA STORED ON AMAZON EBS

is based on the assumption that multiple instances will be required. If the number of instances are reduced, the cost of the deployment would reduce to $1400 for the initial setup with a recurring cost of $838.2 dollars per month. An additional advantage of using the Amazon EC2 services are tools such as the SpringSource Cloud Foundry, which can be used to migrate the Web application controller and views into the Amazon EC2 cloud.

B. Google App Engine

The Google App Engine is geared towards Web applications. App Engine recently began supporting Java based applications for hosting. Metadata extraction and conversion processes in the SeerSuite instance will need to be modified to work with the tool sets available in Google App Engine. The application would store document metadata as blobs with the DataStore API [23].

In the case of the Google App Engine, the application would need to be hosted with the billable quota enabled [13]. For case of the setup costs, this estimate assumes that a single instance would provide all the functionality required by CiteSeer^x. Estimates are provided in table IV. It is to be noted that I/O costs are not included as our calculations indicate that such costs fall within the Free Default Quota or the Billing Enabled Default Quota. In this case the setup costs are much lower at $172, as are the recurring monthly costs at $641 USD. An assumption of linear rate of growth in the collection size does not increase the recurring costs in a significant manner (<5%) over the year.

V. HOSTING SELECTED SERVICES IN THE CLOUD

The graph in Figure 3 is useful for identifying the right set of components to operate within a desired cost based on the

Google App Engine Estimates				
Setup Costs	Type	Units	Cost/Unit	Total($)
	Initial Transfer	1717 GB	0.1/GB	171.7
Recurring Costs	Type	Units	Cost/Unit	Total
	Data In	150 GB	0.1/GB	15
	Data Out	3 TB	0.12/GB	368.63
	Storage	1717 GB	0.15/GB	257.55
	Total			641.18

TABLE IV
GOOGLE APP ENGINE

data stored. We use the logs at the index and web applications to determine the data transfer needs of various components. The database status monitoring provides the amount of data transferred from the database. The size of the files modified by the maintenance system is obtained from estimates of the files created through this process. The data transfer between the ingestion, extraction, and focused crawler is based on the documents processed at these services. Figure 3 shows major

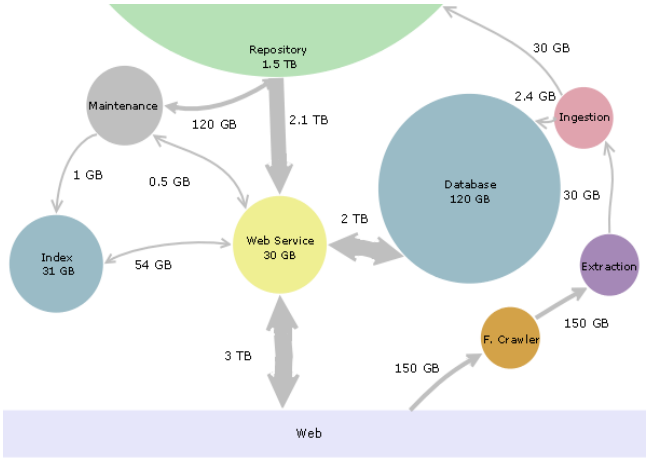

Fig. 3. Data Transfers within CiteSeer^x

components of CiteSeer^x and the data transferred between these components. This graph is useful for identifying an optimal set of nodes based on the data stored and the data transfers in and out of a component. For example, if a fixed budget is allocated for hosting, a subset of components can be hosted staying within budget. From the graph it is apparent that not hosting the repository, database, and web application together will lead to significant costs. However, the focused crawler, extraction and ingestion systems appear to be well suited to be hosted away from the other components. This configuration is also preferred as the hosting cost of data transfer into the system is less expensive than the cost of data transfer out of the system. Among the components, the index appears to be the least expensive to be hosted on the cloud.

An alternative approach would be to identify the peaks of traffic at CiteSeer^x and use cloud based services for peak load scenarios. This would involve hosting parts of the web services, repository services in the cloud. When a peak load is detected by the load balancers, the traffic could be directed

to cloud based services. Further detailed analysis is required for such an approach.

VI. Virtualization of Components

In the previous sections we discussed hosting SeerSuite services in the cloud. Such a hosting provides several advantages to SeerSuite instances. In this section we explore virtualization solutions, which can be part of SeerSuite itself. Such features can enable large scale SeerSuite instances, without the use of expensive hardware solutions. From our earlier discussions, we recognize the repository as the single largest data storage component of SeerSuite instances. The

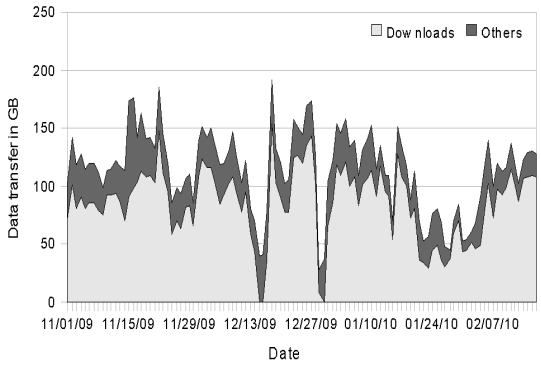

Fig. 4. Download Traffic

graph in Figure 4 indicates that a significant portion (60-80%) of the user traffic to CiteSeer^x is a result of downloads or access to cached copies stored in the repository. Cost efficient, scalable solutions to handle this traffic become crucial for large deployments. Thus, addressing virtualization of the repository is important.

The repository serves as a file storage service, which allows documents, metadata and corrections to be stored and retrieved. Graphs embedded in the document summary page are also stored in the repository. By function, the repository contains sets of document related files, a majority of which are never altered. The repository grows with an increase in the size of the collection. From statistics collected, the repository size grows by approximately one megabyte for the addition of one document. This measure includes a copy of the document, the text, sections of the document, document metadata, charts, individual metadata files, and version files in case of corrections. The repository is exposed in the current setup as a shared resource between web servers. This is due to the fact that changes to the repository which includes corrections, updates to charts, etc. have to be reflected across all services. Writes to the repository occur as a result of ingestion, corrections, or when charts are generated by the maintenance service for each document. These operations are less frequent and thus not constrained by performance issues.

Of these, corrections are initiated at the web application by the user. An XML file with a snapshot of the document metadata is generated and stored in the repository. This process ensures that database and index can be recreated from metadata if necessary. Currently, citation charts are generated periodically by the administrator.

Scalability of the repository presents a challenge for SeerSuite applications, especially CiteSeer^x. This is further exacerbated, by future plans to store and offer video, images and presentations linked to the document. In order to grow beyond the present collection size, the repository services have to be hosted on cost-efficient scalable infrastructure. Solutions involving storage systems, particularly SAN or NAS can be expensive to adopt and maintain. In this regard, virtualized hardware or cloud infrastructure services offer a desirable solution. In this context, we examine the user document

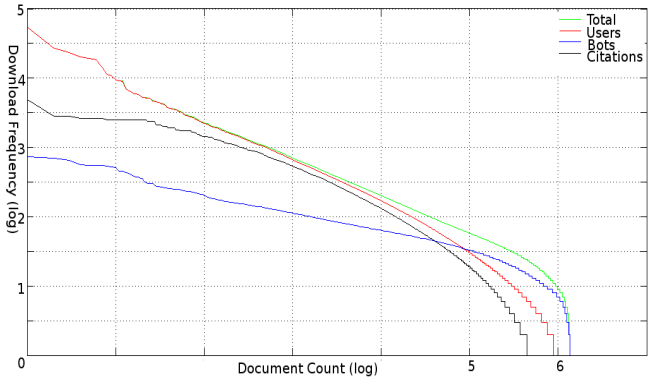

Fig. 5. Document Download Frequency

download pattern, which is one of the primary functions of the repository. The graph in Figure 5 identifies document download frequency on a log/log scale (for 6 months). Download frequency for user initiated downloads (red) and crawler initiated downloads (blue) are provided along with the citation ranking for documents in the collection. The frequency of documents downloaded in the graph follows a power law distribution. The behavior of crawlers and users can be clearly identified. The bots download documents less frequently, but download more documents overall. This information is useful in estimating the size of the collection accessed most by users. By identifying the most frequently accessed documents, we can determine a subset of documents which need to be placed either locally or in the cloud. This is relevant in the case where part of the collection is hosted in the cloud by placing the most frequently accessed documents there. From the graph we can infer which subset of the actual documents (1.5 Million) are actually downloaded either by users or bots. Another approach would be to pick only those documents cited in the collection (564,818), which is a conservative estimate. On the other end is the set of documents downloaded by crawlers - 1,394,669 documents.

The above figures can be misleading if overlap between individual sets is not considered. The overlap between the set of documents cited in the collection with the ones downloaded by users is 334,264. But documents downloaded by both

bots and users number 874,100. These figures provide an illustration of the complexities involved in placing resources and optimizing costs at the component level.

VII. An ad-hoc solution

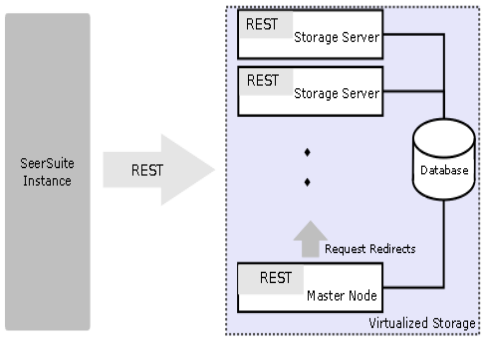

Fig. 6. Virtualized Storage

The SeerSuite architecture provides flexibility in adding new features. Replacing components of the system with other components is possible as long as the two interfaces are compliant. We take advantage of this fact, and experiment with a virtualized storage solution. This solution allows us to identify and estimate with greater accuracy parameters and bottlenecks for adopting a cloud infrastructure in SeerSuite.

The virtualized storage solution uses a REST based interface for the entire system. The storage allows GET, POST, and DELETE commands for downloading, adding, and deleting documents. Aspects of this system were inspired by distributed file systems [10], [6]. The architecture of the system is shown in figure 6. The master node serves as an index for finding documents in the storage system. Requests for documents are made to the master node, which redirects the client to the storage server containing the particular file. Writes or adds bypass the master. In case there are multiple copies of the document with the same version number available, the master redirects the user to the latest copy as per the database.

The storage node is the basic unit of the system, storing files on the underlying file system. Storage nodes allow users access to cached documents through GET requests. Storage nodes accept additions while logged into a central database. Additions log the location/storage server where the addition was made. The files are stored in the same hierarchical format as in the SeerSuite repository with the top directory and leaves labeled with the repository id and with the complete DOI, respectively.

Features such as version information, document checksums, and sizes are currently stored in the database. The system is primitive in that it provides no assurance on consistency or locking by itself. An important feature of such a system is that it can be deployed over already existing collections (with collaborators) without incurring significant costs by simply updating the database with document locations at the collaboration source.

A. Impact on other services

The introduction of virtualized storage has a major impact on the ingestion, maintenance, and web application processes and services. It has relatively little or no impact on the crawler and metadata extraction services. And the introduction is transparent to the user.

The ingestion system now uses POST calls to add documents to the repository. This allows the ingestion process to run on different systes. The impact on ingestion performance is minimal. The web application now points to the director for cached document downloads. For corrections applied to documents, a POST request is made to the repository containing the document to update metadata. Significant changes occur to the document index update process and to the citation chart generation. Indexing maintenance scripts now obtain body and full text of documents with GET requests to the storage servers. Citation charts are generated by javascript libraries, removing the need for static generation and storage in the repository. This approach adds costs to the application and clients. The Web application must obtain data for these charts from the database.

In summary, the system is more scalable as dependence on shared storage systems is reduced. Another positive feature is metadata about files in the repository can easily be exposed to users and researchers. Queries on dimensions such as size and checksum and can be made to the storage master.

This approach allows CiteSeer^x to distribute load across collaborative engines. By identifying the closest repository containing the document across collaborating engines, faster downloads can be achieved.

B. Performance

A deployment of the ad-hoc system discussed in the earlier sections, containing documents available in the CiteSeer^x collection was built. The deployment employs 3 systems, each with dual core processors, 3 GB of RAM, and 1.5 TB of storage, as two storage nodes and one master node (the master node also hosts the database).

To identify bottlenecks, we subject this setup to a JMeter [16] stress test. Test case data was obtained from log analysis presented earlier. Requests are replayed to emulate scenarios. Tests with different number of users (virtual users) and user introduction rate (ramp up time) with five repeats in each test were performed. Results for one set of ramp up times are shown in figure 7. The number of virtual users were varied, with time in which all the users arrive at the server constant at ten seconds. Each request was for documents choosen at random. From the graph as the number of virtual users increase, the error rate rises in an significant manner with 70 virtual users introduced in 10 s. Test results indicate errors occur when the number of active users is > 97. An examination of the master error logs indicates that database connection errors were the cause for server errors. When compared to the CiteSeer^x system, the ad-hoc system has a significantly higher throughput. These results indicate that particular care must be taken when building or deploying

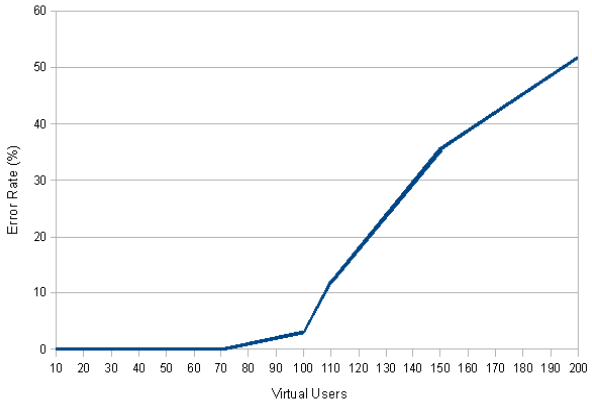

Fig. 7. Ad-Hoc Cloud Performance

CiteSeer^x on virtualized infrastructure. Both CiteSeer^x and the underlying system must support the expected load.

VIII. Future Work

Migrating an application such as CiteSeer^x to the cloud requires a detailed examination of the services, processes, and effectiveness. However, complete breadth of cloud based infrastructure offerings was not explored, i.e., open source cloud systems such as Eucalyptus [20]. Also not discussed were software as a service platforms for deploying SeerSuite. In future we propose to examine such systems, the deployment and optimization of SeerSuite for cloud operations.

Several features still in development for SeerSuite particularly services now provided with federation, could also take advantage of, and be built using cloud infrastructure. By breaking down components of the web application and designing for stateless instances, we can further improve the scalability and possibility that more components can be placed in the cloud.

In addition the ad-hoc system offers opportunities for research and fine tuning in SeerSuite instances. For example, an accurate estimation of the size of the repository to be based in the cloud could be determined. Placement of documents in multiple locations, use of file and server caches to take advantage of any existing temporal locality, and more sophisticated master nodes and schedulers need to be studied. While the system proposed has worked well in a research environment, translating such an effort into a full fledged system on production systems has yet to be realized.

IX. Conclusion

We discussed CiteSeer^x and an approach for establishing an information retrieval and digital library system in the cloud. We analyze the problem of placing services in the cloud and were able to identify from the internal data flows an optimal placement of services for optimizing costs. Analysis of logs allows us to understand the complexity of placing parts of the repository in the cloud.

We discussed the repository system of SeerSuite in detail with regards to its function, usage and characteristics. We proposed an ad-hoc system to provide the same services provided by the repository by virtualizing the storage system. We discussed the impact of such virtualization on SeerSuite. Stress tests enabled us to understand which components of virtualization are susceptible to failure.

We believe we have made a strong case for adopting virtualization and cloud infrastructure services to serve the needs of information retrieval and digital library systems. We intend to explore and adopt such systems as part of building a more scalable, extensible and robust CiteSeer^x.

Acknowledgment

The authors acknowledge partial support from NSF CISE. Urgaonkar's research was funded in part by NSF grants CCF-0811670 and CNS-0720456.

References

[1] Amazon Elastic Block Storage Projecting Costs, http://aws.amazon.com/ebs.
[2] Apache Solr, http://lucene.apache.org/solr.
[3] E. A. Fox, M. A. Robert, R. K. Furuta, J. J. Leggett, *Digital libraries*, Commun. ACM, 38-4, 1995.
[4] M. Armbrust, A. Fox , R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, *Above the Clouds: A Berkeley View of Cloud Computing*, Technical Report No. UCB/EECS-2009-28, University of California at Berkley, 2009.
[5] W.Y. Arms, C. Blanchi, E.A. Overly, V. Reston,*An Architecture for Information in Digital Libraries*, D-Lib Magazine, 1997.
[6] D. Borthakur, *The hadoop distributed file system: Architecture and design*, Hadoop Project Website, 2007.
[7] S. Chakrabarti, M. Van den Berg, B. Dom, *Focused crawling: a new approach to topic-specific web resource discovery*, Computer Networks, Vol. 31 Number 11-16, 1999.
[8] I. G. Councill, C. L. Giles, E. Di Iorio, M. Gori, M. Maggini, A. Pucci, *Towards Next Generation CiteSeer: A Flexible Architecture for Digital Library Deployment*, ECDL, 2006.
[9] I. G. Councill, C. L. Giles, M-Yen. Kan, *Parscit: An open-source CRF reference string parsing package*, In Proceedings of LERC, 2008.
[10] S. Ghemawat, H. Gobioff, S.T. Leung, *The Google File System*, ACM SIGOPS Operating Systems Review, Vol. 37, Num. 5, 2003.
[11] C. L. Giles, K. Bollacker, S. Lawrence, *CiteSeer: An automatic citation indexing system*, ACM Conference on Digital Libraries, 1998.
[12] C. L. Giles, I. Councill, P. Teregowda, J. Fernandez, S. Zheng, http://citeseerx.ist.psu.edu, 2008.
[13] Google App Engine Billing and Budgeting, http://code.google.com/appengine/docs/billing.html.
[14] J. Gray, *Distributed Computing Economics*, Microsoft Research Tech Report, 2003.
[15] R. L. Grossman, Yunhong Gu, *Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere*, CoRR, 2008.
[16] E. H. Halili, *Apache JMeter*, Packt Publishing, 2008.
[17] H. Han, C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, E. A. Fox, *Automatic document metadata extraction using support vector machines*,In Proceedings of the 3rd ACM/IEEE-CS JCDL, 2003.
[18] C. D. Manning, P. Raghavan, H. Schütze, *Introduction to information retrieval*, Cambridge University Press, 2008.
[19] P. Mika, G. Tummarello, *Web Semantics in the Clouds*, IEEE Intelligent Systems, 2008.
[20] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff and D. Zagorodnov, *Eucalyptus : A technical report on an elastic utility computing architecture linking your programs to useful systems*, Technical Report, University of California at Santa Barbara, 1998.
[21] H.S. Pinto, S. Staab, C. Tempich, C. Y. Sure, *Semantic Web and Peer-to-Peer*, Semantic Web and Peer to Peer: Decentralized Management and Exchange of Knowledge and Information, Springer, 2006.
[22] SeerSuite Source, http://sourceforge.net/projects/citeseerx.
[23] Using the Datastore with JDO, http://code.google.com/appengine/docs/java/gettingstarted/usingdatastore.html.