Published in the 2nd USENIX Workshop on Hot Topics in Cloud Computing 2010

CiteSeer^x in the Cloud

Pradeep B. Teregowda
Pennsylvania State University

Bhuvan Urgaonkar
Pennsylvania State University

C. Lee Giles
Pennsylvania State University

Abstract

Information retrieval applications are are good candidates for hosting in a cloud infrastructure. CiteSeer^x, a digital library and search engine, was built with the goal of efficiently disseminating scientific information and literature over the web. The framework for CiteSeer^x, an application of the SeerSuite software, was designed with a focus on extensibility and scalability. Its loosely coupled architecture with service oriented interfaces allows the whole or parts of SeerSuite to readily be placed in the cloud.

1 Introduction

Digital library search engines have been a topic of research and development for the past several years [**?**]. The growth in information available both on the Web and from rapid growth in electronic resources make information retrieval systems like CiteSeer^x [**?**] invaluable. At time of this publication the CiteSeer^x collection indexes more than 1.6 million documents and receives several hundred thousand unique visits per day.

The rate of growth of digital information is always a challenge to the effective design of information retrieval systems. Particularly, Web based digital library search engines such as CiteSeer^x can readily take advantage of the reduced maintenance, elasticity, and availability of infrastructure on demand provided by a cloud infrastructure [**?**].

SeerSuite includes components common to other information retrieval applications. Inspired by services provided by CiteSeer [**?**], SeerSuite provides among others full text indexing, autonomous citation indexing ,and a personal portal in the form of MyCiteSeer. It extracts and publishes extensive metadata for documents, authors and citations. Its design takes advantage of open source applications such as Tomcat, Solr/Lucene, Java Spring Framework and open source RDBM systems. Advances in and development of automatic metadata extraction for parsing header and citation information have also been important.

SeerSuite based applications share a common set of infrastructure challenges to support an a growing set of documents. Although the CiteSeer^x architecture allows hosting of all components and services in the cloud, the size of the CiteSeer^x collection and the amount of data transferred make cloud hosting of CiteSeer^x not straightforward. There are several cost-effective approaches for solving this problem. We discuss some of these approaches in detail and identify the lessons learnt from this analysis. The rest of the paper is arranged in the following manner. Background architecture and services of SeerSuite are discussed in Section 2. The issues of hosting are identified in Section 3. Various strategies for hosting services are discussed in Sections 4, 5 and 6 with future work in Section 7 and conclusions in Section 8

2 Background and SeerSuite Architecture

Recent research on the role of cloud infrastructure in information retrieval systems has focused primarily on its use for information extraction [**?**]. Furthermore, the focus has been on the computational costs with little attention to data storage costs and applications pertinent to grid and distributed computing [**?**]. In contrast, we focus on the use of cloud infrastructure hosted on infrastructure already offered by various vendors.

2.1 Architecture

We give a brief introduction to SeerSuite architecture and services supporting SeerSuite. A brief commentary on the feasibility and refactoring required to host these services or components is included. Figure 1 shows components and services of SeerSuite. Service oriented interfaces allow components and services to be distributed across physical systems. Components and services can

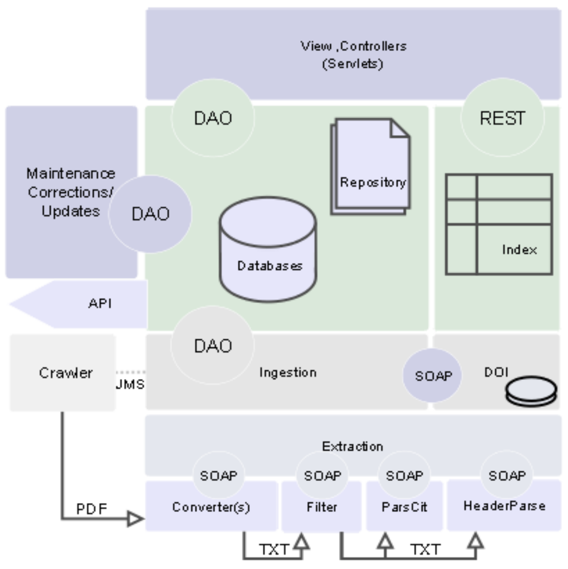

Figure 1: SeerSuite Architecture

be broadly grouped into those responsible for handling user requests and those handling acquisition and ingestion of documents. Among those handling user requests are the Web application which provides presentation and personalization services. The focused crawler, document conversion and metadata extraction, ingestion, and maintenance services are responsible for acquiring and ingesting documents which are then stored in the data storage components for user access.

2.1.1 Web Application

User requests at the Web application are processed with the support of the database, index or the repository. SeerSuite supports interfaces such as the OAI [?] API to allow programmatic access to data stored in the collection. The Web application allows users to search for authors, documents, citations and view document metadata. Some services provided by the Web application require state based interactions with the user, particularly MyCiteSeer. For cloud hosting minor refactoring will be required to support user interaction with MyCiteSeer. The Web application load depends on traffic, which varies throughout the day, making the Web application a strong candidate for hosting.

2.1.2 Focused Crawler

Document acquisition drives the growth of SeerSuite instances. In particular, focused crawlers [?] are used to efficiently harvest relevant documents. The focused crawler is a strong candidate for hosting, since it can take advantage of the elasticity and on demand provisioning by efficiently scheduling crawls.

2.1.3 Document Conversion and Information Extraction

Before the documents can be processed by the extraction system, the documents in PDF and PostScript format are converted into text and filtered to remove those not containing citations. Documents acquired from the Web are processed by multiple modules, which extract extensive document, citation, author metadata. These modules are based on state-of-the-art machine learning methods. Prominent among these is the header parser, which extracts document and author information. The ParsCit module is utilized to extract citation metadata. The metadata extraction system is not a strong candidate for a Platform-as-a-Service cloud offering, since extensive refactoring might be required.

2.1.4 Document Ingestion

The documents processed by the extraction and conversion service are then ingested into the system. This includes adding the document and related metadata to the database and to the repository. By comparing the checksum of the document to be ingested with others, the ingestion system avoids adding duplicates. Documents are assigned a unique document object identifier from the DOI service. The use of service oriented interfaces and the minimal code footprint allow the ingestion system to be easily hosted.

2.1.5 Data Storage

Persistence of data extracted is achieved by the use of index, databases and file storage components (repository). The database is utilized by the web application to provide document summaries and metadata. The index allows users to query the full text and citation information. The file storage caches documents crawled by the crawler and metadata extracted by the extractors as files.

2.1.6 Maintenance Service

Tasks not part of the ingestion system such as updates to the index, inference based metadata updates, and charts and generation of citation charts and statistics are performed by the maintenance system. The maintenance systems generate very little data and can be scheduled by the administrator. These services need to be closer to the data storage due to vast amount of information processed for each iteration of their operation. Their candidacy for the cloud is hence dependent on hosting of the data storage components.

2.1.7 Federated Services

SeerSuite provides several features not part of the main application. These features are supported by services which may not share the same framework or application components, but share infrastructure. Many of these services are still being developed. Such components are strong candidates for migration to the cloud, since they can take advantage of cloud offerings in their development.

2.2 Deployment

The current deployment of SeerSuite as CiteSeer^x is on a group of heterogeneous systems. Two Web application instances are hosted on the Apache Tomcat platform in a cluster. Each Web application instance is hosted on a system with two dual core CPUs and 16GB of RAM. The Web traffic is load balanced through a software based level four load balanced cluster. The database and repository are hosted on separate machines with large storage ($>$ 15 TB), dual core dual CPU's and 16GB of RAM. MySQL is the RDBMS for the system. Indices for document, tables, author names are hosted separately on machines with dual core dual CPU and 16GB of RAM. The repository is shared between the web servers using the Global File System over Global Network Block Device.

2.3 Terminology

A description of terms used in future discussions relevant in the context of SeerSuite are provided below.

Request Types: User requests can be grouped into search, document views, MyCiteSeer and others. The search request and document view requests involve the Web application, database and the index. MyCiteSeer requests involve the Web application and database. Others include requests for stylesheets, images, etc.

Peak Load: This represents a set of requests observed at the web server exceeding a set threshold of requests per second (90th percentile).

3 Problem Definition

SeerSuite as a whole can be hosted in Infrastructure as a Service platforms with minimal refactoring. Such a hosting, however, is expensive with current cloud offerings. This is due to the large collection size and the volume of data transfered between the application and the user. The key question we are interested in answering in this context is *moving which sections, components or subset of Citeseer^x to a cloud would be most cost-effective ?*.

To answer this question, we consider the entire application with a particular focus on the Web application, its supporting components, the index, database and the repository. We present three different approaches by presenting first a hypothesis and discuss the cost and implications. We make use of the existing log monitoring data collected from the CiteSeer^x deployment. The Web application logs for a period of 15 days were analyzed to obtain data for the following sections. Figure 2 shows the number of requests made during this period along with the type of request.

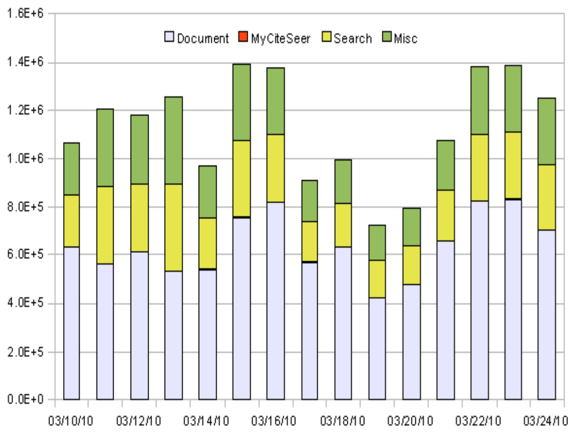

Figure 2: Traffic by Request Type

4 Component Hosting

Hypothesis: Based on the cost of hosting, we can choose one or several components to host in the cloud.

SeerSuite includes several components among which we would like to an optimal set. We consider components of the system, choosing components based on the size, data transfer and feasibility of migration. In our case, the cost of hosting is dominated by the amount of data stored in the cloud and the volume of data transferred in and out of the cloud. We consider the amount of data stored in these components and the volume of data transferred through the component.

Figure 3 shows the flow of data between components of CiteSeer^x. Data for creating this graph was obtained from log files, and application specific monitors. In the case of crawlers and information extraction system, the data flow was assumed proportional to the number of documents acquired and processed. This graph is useful for determining candidates for hosting in the cloud. For example, if CiteSeer^x were hosted entirely within the cloud, the amount of data stored in the cloud would be 1.7 TB, with 3.2 TB of data transferred between the user, web and the application. Clearly, the repository is the largest component in size, while the web application has

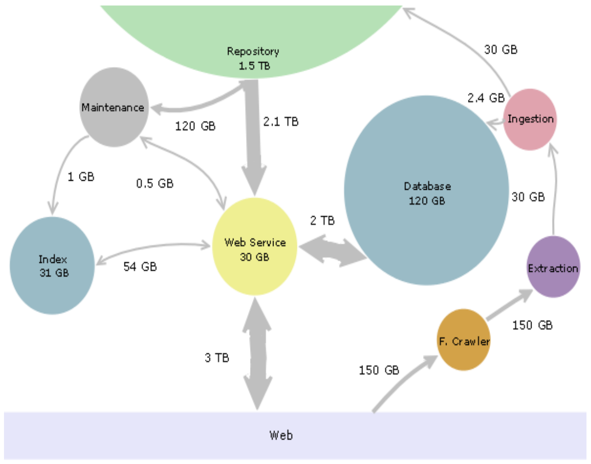

Figure 3: Data Flow - CiteSeer^x Components

the largest amount of data volume crossing it. We provide a cost estimate for the components based on cloud infrastructure services offered by Amazon EC2 [?] and Google App Engine [?]. Cost estimates are based on a 30 day month.

Choice of vendors is a result of support in terms of environment and code libraries offered or supported by these vendors. In the case of Amazon EC2, we consider a mapping of one to one to an extra large instance for hosting the database, application, index, repository, extraction and crawler services. We assume that additional instances are provided as required in Google App Engine with no additional cost.

Cost			Amazon	Google
Initial Setup	Data In	1820.4	0	182
Monthly	Stored	1820.4	182.04	273.06
	Data In	152	0	15.2
	Data Out	3072	460.8	368.64
	Trans.	368	190.77	0
	CPU	30*24	2937.6	144
Total Monthly			$3771.21	$800.9

Table 1: CiteSeer^x Hosting

Table 1 provides the cost of hosting CiteSeer^x for a month. We now examine the cost of hosting individual components, with all other services hosted locally. Estimates are provided in Table 2. CiteSeer^x suffers initial setup costs, as a substantial collection already exists; however, new services will not incur this cost.

Note that Amazon currently provides free data transfers into their cloud. If this were not the case, hosting services on Amazon would be much more expensive and also incur initial setup costs. Calculating the cost of hosting the entire application leads to a figure of $3771 for Amazon EC2 and $800 for Google App Engine. The cost of hosting components also lends support to the conclusions drawn about data and access [?].

Component	A. EC2		G. App Engine	
	Initial	Month	Initial	Month
Web Service	0	1448.18	0	942.53
Repository	0	1011.88	163.8	593.21
Database	0	858.89	12	348.05
Index	0	527.08	3.1	83.48
Extraction	0	499.02	0	90.6
Crawler	0	513.4	0	105

Table 2: Component Costs in the Cloud (Costs in USD)

Individual components hosted in the cloud have implications beyond the cost of hosting them. Costs related to refactoring code for migration has not been accounted for in Table 2. In the case of the Google App Engine, existing code written in languages not supported by App Engine will require significant refactoring. Along with components hosted, components hosted locally may require refactoring. This refactoring is minimal if the service or component utilized a service oriented interface and significant when services are closely coupled.

Lessons Learned: If the cost of hosting an entire service is prohibitive, hosting components may be a reasonable approach to taking advantage of cloud infrastructure. The cost effectiveness of such an approach depends on data transferred through the service. Loosely coupled components are easier to migrate. For existing components and code, refactoring costs will provide closer estimates of costs. This approach is suitable when a fixed budget constrains the placement of services or components. By identifying components, data transfer, and refactoring costs, a hosting solution can be identified.

5 Content Hosting

Hypothesis: Content, particularly static content can be hosted in the cloud such that, traffic can be shared across local and cloud hosted content.

An analysis of peak traffic at the web services provides an insight on how this can be achieved. ¿From the figure 4, we see that most requests for peak traffic occur for Javascript and Stylesheets which can be hosted independent of the web application. In this case the amount of data stored on the cloud is small, so despite the high volume of traffic the cost of hosting is cost-effective. The size of the files to be placed on the cloud is 2.24 MB. By hosting these files in the cloud, the amount of data transferred for CiteSeer^x from the cloud is 390.26 GB costing less then $80 (on both Amazon EC2 and Google App Engine services) per month. While this is just part

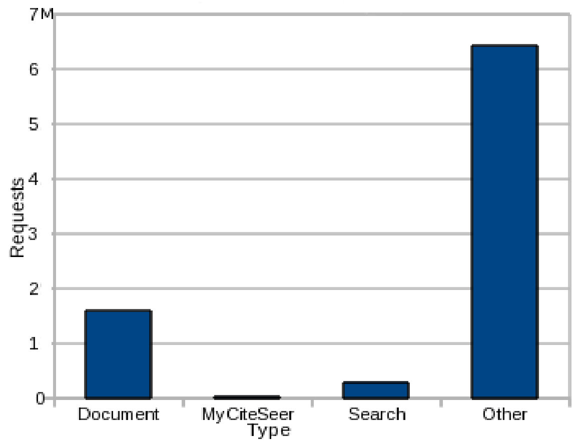

Figure 4: Request Types at Peak

of more than 3 TB of data volume between the application and the user, it helps the system satisfy a significant number of peak load requests.

The same approach can be used to identify elements like subset of the repository, into the cloud. Such an approach would involve identifying the most commonly accessed documents and placing them both locally and in the cloud. During peak loads, clients can be directed to the cloud for access.

Lesson Learned: Hosting specific content relevant to peak load scenarios in the cloud can be beneficial, and the simplest approach to hosting services in the cloud.

6 Load based partitioning

Hypothesis: A copy of the application or a component, supporting a locally hosted application during peak loads can be hosted in the clouds.

This approach is particularly important for supporting the growth in traffic, flash crowds providing users access to service.

Figure 5 shows the requests received at the web server. The total number of requests are shown in gray, with document requests shown in blue. Search requests are shown in brown. From the graph we identify that the 90th percentile is represented by 60 requests per second. Most of these requests are for elements associated with presentation (javascript and stylesheets). Assuming that the traffic growth continues at the same pace and as more features (Algorithm and Figure search) are added, There is a need for provisioning more systems. Instead of procuring these systems, infrastructure at the cloud can be considered to fulfill this need.

Two strategies are possible in partitioning based on load. Of these, one strategy would be to host a copy of the entire application in the cloud, using load balancers

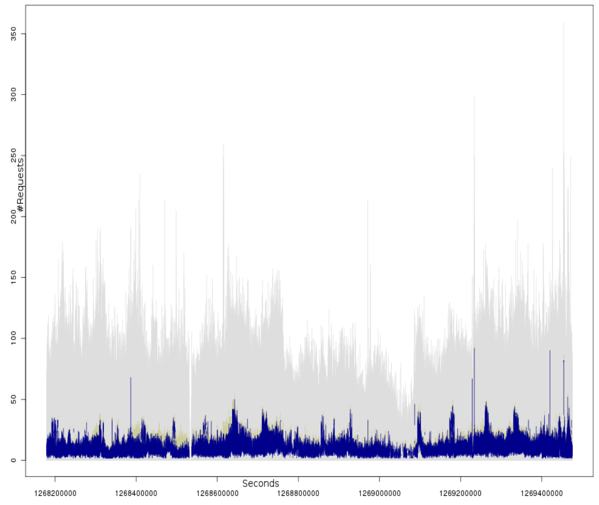

Figure 5: Number of Requests per Second

to identify and direct traffic during peak load conditions. Table 3 provides the costs of such a hosting solution for CiteSeer^x in Amazon EC2 and Google App Engine. All data measurements are in GB, and transaction measurements in transactions per second obtained via iostat.

Cost			Amazon	Google
Initial Setup	Data In	1820.4	0	182
Monthly	Stored	1820.4	182.04	273.06
	Data In	14.78	0	1.48
	Data Out	298.7	44.8	35.84
	Trans.	368	9.27	0
	CPU	70	285.6	7
Total Monthly			$521.71	$317.38

Table 3: CiteSeer^x Peak Load Hosting

These costs can be considered in comparison to the cost of procuring, maintaining systems. Savings by avoiding adoption of storage systems locally add to the attractiveness of cloud infrastructure.

An alternate approach would be to host only the component under stress in the cloud, For example, a database replica to support a locally hosted database could be deployed in the cloud. If this instance were used only during peak load conditions, the costs would decrease to $385, since the instance would be in use for 70 hours.

Lessons Learned: By utilizing a replica or subset of the application for handling only peak loads, we can take advantage of cloud infrastructure in a cost-effective manner. This can resolve issues stemming from the growth of the collection and user traffic.

7 Future Work

We were however not able to examine the temporal nature of traffic and user behavior. By identifying user patterns, the hosting solutions can be optimized to take advantage these patterns. While this discussion included the Amazon EC2 and Google App Engine for cost comparison, this work needs to be extended by examining in depth options offered by other cloud offerings, private clouds and virtualization solutions.

Products like private clouds offered Eucalyptus [**?**] can be utilized to take advantage of hardware already existing as part of the system. Components related to user interaction with CiteSeer^x hosting with services like Amazon Virtual Private Clouds, and local clouds can be considered for these services.

Impact of including cloud hosted services on other services has not be considered in the current discussion. Inclusion of cloud services could require significant refactoring and changes to maintenance cycles. Several opportunities exist within SeerSuite framework for adopting virtualization and cloud infrastructure. In particular, the repository can be restructured to take advantage of cloud based storage solutions in an effective manner. Hadoop [**?**] based metadata extraction and log analysis systems could enable faster document acquisition.

8 Conclusions

Preliminary costs on hosting SeerSuite instances such as CiteSeer^x in the cloud do not seem unreasonable. We develop different approaches that can be adopted either for their cost-efficiency, simplicity, or handling peak loads. Cost estimation for each approach, along with lessons learned from analysis provide a guideline for further exploration. Our future work would focus on adoption of virtualization and extraction systems suitable for hosting in the cloud. In addition to these goals, we would like to examine user behavior, issues in privacy, security for components hosted in the cloud that were not discussed in this work. As part of these discussions, we have presented a detailed examination of the existing deployment of SeerSuite in CiteSeer^x.

9 Acknowledgments

Authors acknowledge partial grant from NSF.