CiteSeer\textsuperscript{x}: A Cloud Perspective

Pradeep Terekowda, Bhuvan Urgaonkar, C. Lee Giles
Pennsylvania State University
Problem Definition

**Question:** How to effectively move a digital library, CiteSeerx, into the cloud

- Which sections, components, or subset of CiteSeerx could be most cost effective to move?

**Our contribution – analysis from an economic perspective.**

Solve by decomposing the application across

- Components
- Content
- Peak load hosting
SeerSuite - CiteSeer

SeerSuite

Framework for digital libraries

Flexible, Scalable, Robust, Portable, state of the art machine learning extractors, open source – use.

CiteSeer

Instance/Application of SeerSuite.

Collection of

> 1.6 million documents

> 30 million citations

Approximately 2 million hits per day
SeerSuite Architecture

- Web Application
- Focused Crawler
- Document Conversion and Extraction
- Document Ingestion
- Data Storage
- Maintenance Services
- Federated Services
Hosting models

- Component hosting
  - SeerSuite is modular by design and architecture, host individual components across available infrastructure.

- Content hosting
  - CiteSeerx provides access to document metadata, copies and application content
  - Host parts or complete set.

- Peak load loading
  - Support the application during peak loads
  - Support growth of traffic.
Component Hosting

- SeerSuite/CiteSeer is modular by design, composed of services which can be hosted in the cloud.
  - Expense of hosting the whole of CiteSeer is prohibitive.
  - **Solution:** Host a component or service i.e.,
  - Component/service code
  - Data on which the component acts
  - Interfaces, etc. associated with the component
  - **Goal:** Identify optimal subset/components.
Component Hosting - Costs

<table>
<thead>
<tr>
<th>Component</th>
<th>Amazon EC2</th>
<th>Google App Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial</td>
<td>Monthly Costs</td>
</tr>
<tr>
<td>Web Services</td>
<td>0</td>
<td>1448.18</td>
</tr>
<tr>
<td>Repository</td>
<td>0</td>
<td>1011.88</td>
</tr>
<tr>
<td>Database</td>
<td>0</td>
<td>858.89</td>
</tr>
<tr>
<td>Index</td>
<td>0</td>
<td>527.08</td>
</tr>
<tr>
<td>Extraction</td>
<td>0</td>
<td>499.02</td>
</tr>
<tr>
<td>Crawler</td>
<td>0</td>
<td>513.4</td>
</tr>
</tbody>
</table>

*Most expensive - host web services.*
Component Hosting – Lessons Learned

- Hosting components is reasonable
  - Having a service oriented architecture helps
- Amazon EC2
  - Computation costs dominate.
- Google App Engine
  - Refactoring costs?
  - Refactoring required not just for component, but other services.
- Storage and transfer costs maybe optimized
  - A study of data transfer in the application gives insights to costs.
- Approach suitable for meeting fixed budgets
  - How many components of an application can be hosted for a fixed budget.
Content Hosting

- **Approach:** Identify specific content
  - **Static Web Application content**
    - Javascript
    - Stylesheets
    - Images/Graphs.
  - **Repository content**
    - PDF files
      - Current Size: 1 terabyte
  - **Database content**
    - Partition database
      - Current size: 120 gigabytes
Analysis of Content Hosting

- Examining the traffic (requests) at peak loads.
  - Requests for stylesheets, images, javascript account for most of the requests.
- The size of these files is 2.2 MB
- Since these files are embedded in almost every web page, bandwidth consumed 390.3 GB.
- Costs < 142 dollars.
- Simpler to deploy
  - Move files to the cloud, update references to them in the presentation layer.
Content Hosting – Lessons Learned

- Hosting specific content relevant to peak load scenarios
  - Easy to do – minimal refactoring required, affects a minimal set of components (presentation layer).
- More complex scenarios need to be examined
  - Hosting papers from the repository
  - Hosting shards of the index
  - Database
Peak Load Hosting

Part of the load can be handled by an instance hosted in the cloud

Approach

- Look at various percentiles of the load (90%)
- Consider utilizing the cloud instance only at loads exceeding these percentiles.
## Peak Load Hosting - Costs

<table>
<thead>
<tr>
<th>Costs</th>
<th>Quantity</th>
<th>Amazon</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial Setup</strong></td>
<td><strong>Data In</strong></td>
<td>1820.4 GB</td>
<td>0</td>
</tr>
<tr>
<td><strong>Monthly</strong></td>
<td><strong>Stored</strong></td>
<td>1820.4 GB</td>
<td>182.4</td>
</tr>
<tr>
<td></td>
<td><strong>Data In</strong></td>
<td>14.78 GB</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td><strong>Data Out</strong></td>
<td>298.7 GB</td>
<td>44.8</td>
</tr>
<tr>
<td></td>
<td><strong>Transaction</strong></td>
<td>368 TPS</td>
<td>9.27</td>
</tr>
<tr>
<td></td>
<td><strong>CPU</strong></td>
<td>70 HRS</td>
<td>285.6</td>
</tr>
<tr>
<td><strong>Total (Monthly)</strong></td>
<td></td>
<td></td>
<td>521.7</td>
</tr>
</tbody>
</table>

As shown in the table above, CPU and Data Transfer costs dominate.
Peak Load – Lessons Learned

Hosting only during peak load conditions is economically feasible.

Growth potential

- Can be used to handle growth in traffic, instead of procuring new hardware.
- Hosting a specific component under stress; such as a database
  - In such a case it will cost 385 dollars to host the database in Amazon EC2.
Conclusions

SeerSuite/CiteSeer\textsuperscript{x} and different approaches were proposed for hosting CiteSeer\textsuperscript{x}.

Investigated cost of hosting for

- **Component**
  - Economically reasonable
  - Refactoring costs

- **Content**
  - Simplest approach
  - More complex scenarios require deeper study

- **Peak load**
  - Very reasonable
  - Support for growth and scalability.
Future Work

- Cost of refactoring – particularly for Google App Engine.
- Cost comparisons for other cloud offerings – Azure, Eucalyptus.
- Privacy and user issues – myCiteSeer and private clouds.
- Technical issues with cross hosting – load balancing, latency needed to be addressed.
- Virtualization in SeerSuite, components built with cloud hosting in mind (Federated Services).
Q & A
Appendix
Assumptions

Instance sizes are larger than expected load (15% average usage for current infrastructure).

Instances include libraries and/or allow these libraries to be included.

Maintenance traffic is not accounted (< %1).

Effort required to maintain – extra personnel costs are not included (Assumed to be the same as existing).

Naïve clustering and load balancing.
<table>
<thead>
<tr>
<th></th>
<th>Amazo</th>
<th>Google</th>
<th>Initial</th>
<th>REP</th>
<th>Amazo</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored</td>
<td>120</td>
<td>12</td>
<td>18</td>
<td>12</td>
<td>Stored</td>
<td>1638.4</td>
</tr>
<tr>
<td>Data In</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Data In</td>
<td>30</td>
</tr>
<tr>
<td>Data Out</td>
<td>2150.4</td>
<td>322.56</td>
<td>258.05</td>
<td>2270.4</td>
<td>340.56</td>
<td>272.45</td>
</tr>
<tr>
<td>Transactions</td>
<td>134</td>
<td>34.73</td>
<td>0</td>
<td>69</td>
<td>17.88</td>
<td>0</td>
</tr>
<tr>
<td>CPU</td>
<td>489.6</td>
<td>72</td>
<td>348.05</td>
<td>489.6</td>
<td>1011.8</td>
<td>593.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Amazo</th>
<th>Google</th>
<th>Initial</th>
<th>WS</th>
<th>Amazo</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored</td>
<td>32</td>
<td>3.2</td>
<td>4.8</td>
<td>3.2</td>
<td>Stored</td>
<td>30</td>
</tr>
<tr>
<td>Data In</td>
<td>2</td>
<td>0</td>
<td>0.2</td>
<td></td>
<td>Data In</td>
<td>4253.9</td>
</tr>
<tr>
<td>Data Out</td>
<td>54</td>
<td>8.1</td>
<td>6.48</td>
<td></td>
<td>Data Out</td>
<td>3072</td>
</tr>
<tr>
<td>Transactions</td>
<td>101</td>
<td>26.18</td>
<td>0</td>
<td></td>
<td>Transactions</td>
<td>20</td>
</tr>
<tr>
<td>CPU</td>
<td>489.6</td>
<td>72</td>
<td>83.48</td>
<td></td>
<td>CPU</td>
<td>489.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Amazo</th>
<th>Google</th>
<th>Initial</th>
<th>CR</th>
<th>Amazo</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Stored</td>
<td>0</td>
</tr>
<tr>
<td>Data In</td>
<td>150</td>
<td>0</td>
<td>15</td>
<td></td>
<td>Data In</td>
<td>150</td>
</tr>
<tr>
<td>Data Out</td>
<td>30</td>
<td>4.5</td>
<td>3.6</td>
<td></td>
<td>Data Out</td>
<td>150</td>
</tr>
<tr>
<td>Transactions</td>
<td>19</td>
<td>4.92</td>
<td>0</td>
<td></td>
<td>Transactions</td>
<td>5</td>
</tr>
<tr>
<td>CPU</td>
<td>489.6</td>
<td>72</td>
<td>90.6</td>
<td></td>
<td>CPU</td>
<td>489.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SeerSuite Architecture

- Web Application
  - User interaction, supports various interfaces.
  - Built using the java Spring framework.
- Focused Crawler
  - Acquire documents from the web specific to a particular topic
- Document Conversion and Extraction
  - Process acquired documents to enable ingestion into the collection.
- Document Ingestion
  - Add processed documents to the collection.
SeerSuite Architecture

● Data Storage
  ● Store acquired documents – persistence, faster access and use.

● Maintenance Services
  ● Processes, which help maintain freshness – statistics, index, graphs.

● Federated Services
  ● Services, not yet completely part of SeerSuite, but may share the same framework, infrastructure.
Appendix - Digital Libraries
Outline – HotCloud 2010

● Introduction
● Motivation/Our Contributions
● SeerSuite
● Component Hosting
● Content Hosting
● Peak Load Hosting
● Future Work
● Conclusions