Variations in Performance and Scalability when Migrating n-Tier Applications to Different Clouds

Deepal Jayasinghe, Simon Malkowski, Qingyang Wang, Jack Li, Pengcheng Xiong, Calton Pu
Outline

- Motivation
- Experiment setup
- Results on Emulab/ Open Cirrus
- Results on EC2
  - Horizontal scalability
  - Vertical Scalability
- Issue 1: Multi-threading overhead
- Issue 2: Network driver overhead
- Conclusion
Motivation

- Clouds are much popular, but not a mature technology.
- More experimental studies are needed to better understand them.
- N-Tier applications are complex and migrating them to clouds is a non-trivial task.

We analyzed the performance and scalability when migrating n-tier applications from a traditional datacenter to an Infrastructure as a Service (IaaS) cloud.
Experiment Setup

- RUBBoS benchmark
  - Bulletin board system like Slashdot ([www.slashdot.org](http://www.slashdot.org))
  - Typical 3-tier or 4-tier architecture
  - Two types of workload
    - Browsing only
    - Read/Write mix
  - 24 web interactions

- MySQL Cluster
  - Middleware for database scale-out
    - Multi-master
    - In-memory

RUBBoS Deployment Topology with MySQL Cluster
Experiment Environment - Emulab

- Emulab (http://www.emulab.net)
  - Relatively modest testbed originally for network research
  - Virtual network & physical machines (not VM)

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server type</td>
<td>PC3000 in Emulab</td>
</tr>
<tr>
<td>Processor</td>
<td>Xeon 3GHz 64bit</td>
</tr>
<tr>
<td>Memory</td>
<td>2GB</td>
</tr>
<tr>
<td>Network</td>
<td>1Gbps</td>
</tr>
<tr>
<td>Disk</td>
<td>2 x 146GB 10,000rpm</td>
</tr>
</tbody>
</table>
Experiment Environment – Open Cirrus

- Open Cirrus (https://opencirrus.org/)
  - Open cloud-computing research testbed
  - Designed to support research into the design, provisioning, and management of services at a global, multi-datacenter scale.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server type</td>
<td>X3210</td>
</tr>
<tr>
<td>Processor</td>
<td>3.00GHz (Quad Core)</td>
</tr>
<tr>
<td>Memory</td>
<td>7.5 GB</td>
</tr>
<tr>
<td>Network</td>
<td>InfiniBand</td>
</tr>
</tbody>
</table>
**Experiment Environment – Amazon EC2**

- **Amazon EC2** – ([http://aws.amazon.com/ec2/](http://aws.amazon.com/ec2/))
  - An Elastic Compute Cloud that provides resizable compute capacity in the cloud.
  - It is designed to make web-scale computing easier for developers.

<table>
<thead>
<tr>
<th>Node Type</th>
<th>CPU</th>
<th>Memory</th>
<th>Platform</th>
<th>Price/Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1 EC2 Unit¹</td>
<td>1.7 GB</td>
<td>32 bit</td>
<td>$ 0.085</td>
</tr>
<tr>
<td>Large</td>
<td>4 EC2 Unit</td>
<td>7.5 GB</td>
<td>64 bit</td>
<td>$ 0.34</td>
</tr>
<tr>
<td>Ex-Large</td>
<td>8 EC2 Unit</td>
<td>15 GB</td>
<td>64 bit</td>
<td>$ 0.68</td>
</tr>
<tr>
<td>Cluster</td>
<td>33.5 EC2 Unit</td>
<td>23 GB</td>
<td>64 bit</td>
<td>$ 1.60</td>
</tr>
</tbody>
</table>

¹: EC2 Unit = 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor
# Software Settings

<table>
<thead>
<tr>
<th>Function</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web server</td>
<td>Apache 2.0.54</td>
</tr>
<tr>
<td>Application server</td>
<td>Apache Tomcat 5.5.17</td>
</tr>
<tr>
<td>DB clustering middleware</td>
<td>MySQL Cluster 5.0.51a</td>
</tr>
<tr>
<td>Java</td>
<td>Sun jdk1.6.0_14</td>
</tr>
<tr>
<td>Operating system</td>
<td>FC4 / FC8</td>
</tr>
<tr>
<td>System Monitor</td>
<td>dstat</td>
</tr>
</tbody>
</table>
## Summary of Experiments

- Automated experiment management approach (Elba¹)

<table>
<thead>
<tr>
<th>Type</th>
<th>Emulab</th>
<th>Open Cirrus</th>
<th>Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiments</td>
<td>8124</td>
<td>430</td>
<td>1436</td>
</tr>
<tr>
<td>Node</td>
<td>95682</td>
<td>4480</td>
<td>25846</td>
</tr>
<tr>
<td>Configurations</td>
<td>342</td>
<td>23</td>
<td>86</td>
</tr>
<tr>
<td>Data points</td>
<td>3210.6 M</td>
<td>2.3 M</td>
<td>672 M</td>
</tr>
</tbody>
</table>

¹: http://www.cc.gatech.edu/systems/projects/Elba/
Notation

We use the notation \( #W-#A-#S-#D-p \)

- \( #W \) – Number of Web server (all experiments in this paper with one Web server)
- \( #A \) – Number of Application servers
- \( #S \) – Number of SQL nodes
- \( #D \) – Number of Data nodes
- \( p \) – Platforms (em- for Emulab, oc – for Open Cirrus, ec2- for EC2)

1-2-2-2-em

Emulab configuration with 1-Apache, 2-Tomcats, 2-SQL nodes and 2-Data nodes
Performance on Emulab

Throughput (Req/Sec) vs. # Users

Lines represent different configurations:
- 1-2-2-em
- 1-2-4-em
- 1-4-16-16-em

Improvement in throughput as the number of users increases for each configuration.
Performance on EC2

Throughput (Req/Sec) vs # Users on Large instances

- Degradation observed at specific # Users levels for different configurations.

Lines represent:
- 1-2-2-2-ec2
- 1-2-2-4-ec2
- 1-2-2-8-ec2
CPU Utilization (1-2-2-2)
Vertical Scalability on EC2

Throughput (Req/Sec) vs. # Users

Improvement

Lines represent different user configurations:
- I-2-2-2-small
- I-2-2-2-large
- I-2-2-2-exlarge
- I-2-2-2-cluster
Summary

- Emulab and Open Cirrus show better horizontal scalability
- EC2 show good vertical scalability, but not better horizontal scalability

Our analysis resulted in two findings:
- Multi-threading overhead
- Network driver overhead
Multi-Threading Overhead
End-to-End Response Time

Client End-to-End
Web Server End-to-End
Differences in Response Time

Response Time (s) vs. # Users for Web-Server and RUBBoS Client.

Time differences indicate a significant increase in response time as the number of users increases.

Context Switching (using LMBench)

![Graph showing context switching time per switch for Emulab and EC2. The x-axis represents the number of threads, and the y-axis represents the time per switch (microsecond). The graph shows a sharp increase in time per switch for Emulab as the number of threads increases, while EC2 maintains a relatively constant time per switch.]
Number of Context Switches

- **EC2**
- **Emulab**

![Graph showing the number of context switches for EC2 and Emulab against the number of users.](image-url)
Issues and Solution

- Overall throughput and number of context switches are related
- But, in EC2 we get less switches and similar overhead as Emulab
- When the number of threads is higher the application become unstable in EC2

Solutions:
- Rent many instances
- Re-write the application with less number of threads
With our Solution

![Graph showing improvement in response time with increasing number of users.](image-url)
Solution - Improvements

Throughput [Req/Sec] vs. # Users

- Original client
- Client with reduced threads

Improvement
Network Driver Overhead
Data Node Send Queue

![Graph showing the relationship between queue size (B) and number of users. The graph indicates two distinct lines representing different user scenarios: 1-2-2-4-em and 1-2-2-4-ec2.](image-url)
Ping-Pong Time without a Load

Round Trip time (ms)

Duration

emulab
EC2
Ping-Pong with a Load

Round Trip time (ms)

Duration

emulab
EC2
Solution

- Use network friendly database middleware
  - Reduce the pressure on the network
- Evaluated C-JDBC, an open source database middleware
  - Performed well and showed a very good scalability
  - Reduce the network traffic significantly
C-JDBC vs. MySQL Cluster Performance

Throughput [Req/Sec] vs. # Users

- C-JDBC
- MySQL
C-JDBC vs. MySQL Cluster Network Traffic

Data Transferred [MB/s]

# Users

C-JDBC
MySQL
Conclusion & Future works

- Studied the performance and scalability variations when migrating n-tier applications to clouds.
- Our results show, that cloud are new and need more experimental studies to better understand them.
- More specifically, application re-design is needed to handle cloud challenges.

Future works

- Extend our analysis into other clouds (e.g., Wipro)
- Micro level study to dig deep into observe phenomena
- With other database monument systems (e.g., Oracle, DB2)
Thank you.