An Experimental Study of Rapidly Alternating Bottleneck in n-Tier Applications

Qingyang Wang, Yasuhiko Kanemasa, Jack Li, Deepal Jayasinghe, Toshihiro Shimizu, Masazumi Matsubara, Motoyuki Kawaba, Calton Pu
Scaling Web Applications On-Demand in Cloud

Good performance + Cost efficiency

High throughput + low response time

Workload

Web Server

Application Server

Bottleneck

DB clustering middleware

High resource utilization

DB Server

Bottleneck
What If No Bottleneck Was Detected?

How to scale a web application while no bottleneck is identified?

Bad performance

Workload

Web Server

Application Server

CPU util. 79.2%
Disk I/O 0.5%
Memory 30%

CPU util. 78.1%
Disk I/O 0.5%
Memory 50%

CPU util. 34.6%
Disk I/O 0.5%
Memory 50%

CPU util. 26.7%
Disk I/O 0.5%
Memory 50%

DB clustering middleware

DB Server

……

……
1. Throughput is limited with no saturated resources
2. Duration of each bottleneck is short (e.g., < 100ms)
Experimental Setup

- RUBBoS benchmark: a bulletin board system like Slashdot
- 24 web interactions
  - CPU intensive
- Workload consists of emulated clients

- Intel Xeon E5607
  - 2 quad-core 2.26 GHz
  - 16 GB memory
Motivational Example

- Response time & throughput of a 3-minute benchmark on the 4-tier application with increasing workloads.
No Obvious Bottleneck is Detected at WL 14,000

- **Workload is CPU intensive**
  - Disk I/O utilization (<5%), network I/O utilization (<20%), Memory usage (<40%);
Rapidly Alternating Bottleneck: Sources and Detection

- **Sources:** We find that other than bursty workload, system environmental conditions:
  - JVM garbage collection
  - VM collocation

- **Detection and Visualization:** We implement a fine-grained monitoring method based on passive network tracing.
  - Negligible monitoring overhead for running applications
Outline

- Introduction & Motivation
- Detection and Visualization
  - Fine-grained load/throughput analysis
- Two Observations of Rapidly Alternating Bottlenecks
  - JVM garbage collection (JVM GC)
  - VM collocation
- Conclusion & Future Works
Two Steps for Detecting Rapidly Alternating bottlenecks

1. Find the participating servers that present transient bottlenecks (e.g., 50ms)
2. Check whether the transient bottlenecks of each participating server occur in an alternating pattern
Passive Network Tracing Infrastructure

- Collect interaction messages in the system using SysViz to measure fine-grained **active load** and **throughput** on each server.
  - **Active load**: The # of concurrent requests in a server
  - **Throughput**: The # of completed requests of a server
Fine-Grained Active Load Calculation in a Server

SysViz monitoring for MySQL

![Diagram showing active load calculation with time, arrival timestamp, and departure timestamp.](image)
Active-Load/Throughput Correlation Analysis

The diagram illustrates the relationship between Server throughput and Active load in a server, with the following key points:

- **TP_{max}**
- **Non-Saturation area**
- **Saturation area**
- **Saturation point N^***

These areas and points help in understanding the performance characteristics of a server under varying loads.
Active-Load/Throughput Analysis for MySQL at WL 14,000

MySQL active load (every 50ms)

MySQL throughput (every 50ms)
Outline

- Introduction & Motivation
- Detection and Visualization
  - Fine-grained load/throughput analysis
- Two Observations of Rapidly Alternating Bottlenecks
  - JVM garbage collection (JVM GC)
  - VM collocation
- Conclusion & Future Works
Active-Load/Throughput Analysis at Workload 7,000

Tomcat throughput [req/s] vs Tomcat active load [#] (Point of Interest (POI))

MySQL throughput [req/s] vs MySQL active load [#]
Active-Load/Throughput Analysis at Workload 14,000

System achieves the maximum throughput at WL 14,000.
Timeline Analysis at Workload 14,000

Tomcat scatter graph

Active load [#]

Throughput [req/s]

Active load [#]

Tomcat GC running ratio [%]
Timeline Analysis at Workload 14,000 (Cont.)

**Tomcat scatter graph**

**MySQL scatter graph**

Throughput [req/s]

Active load [#]

Time (s)
Correlation Analysis of Rapidly Alternating Bottlenecks

Correlation coefficient: -0.42, negative correlation suggests rapidly alternating bottleneck.
Introduction & Motivation

Detection and Visualization
   - Fine-grained load/throughput analysis

Two Observations of Rapidly Alternating Bottlenecks
   - JVM garbage collection (JVM GC)
   - VM collocation

Conclusion & Future Works
Rapidly alternating bottlenecks can cause non-trivial performance loss in an n-tier system.

We proposed a rapidly alternating bottleneck detection and visualization method through fine-grained active-load/throughput analysis.

Ongoing work: more analysis of different types of workloads and more system factors that cause rapidly alternating bottlenecks.
Thank You. Any Questions?

Qingyang Wang
qywang@cc.gatech.edu
Backup slides
Resolving Rapidly Alternating Bottlenecks

- MySQL active load [#]
- MySQL throughput [req/s]
- Tomcat active load [#]
- Tomcat throughput [req/s]
Performance Gain After Resolving Rapidly Alternating Bottlenecks

![Graph showing response time and throughput for different workloads and Java Development Kit (JDK) versions.](image-url)
Active-Load/Throughput Analysis at Workload 14,000

- System achieves the maximum throughput at WL 14,000

Graph showing the relationship between workload and system throughput.