When Average is Not Average: **Large Response Time Fluctuations** in n-Tier Applications

Qingyang Wang, Yasuhiko Kanemasa,

Calton Pu, Motoyuki Kawaba

Outline

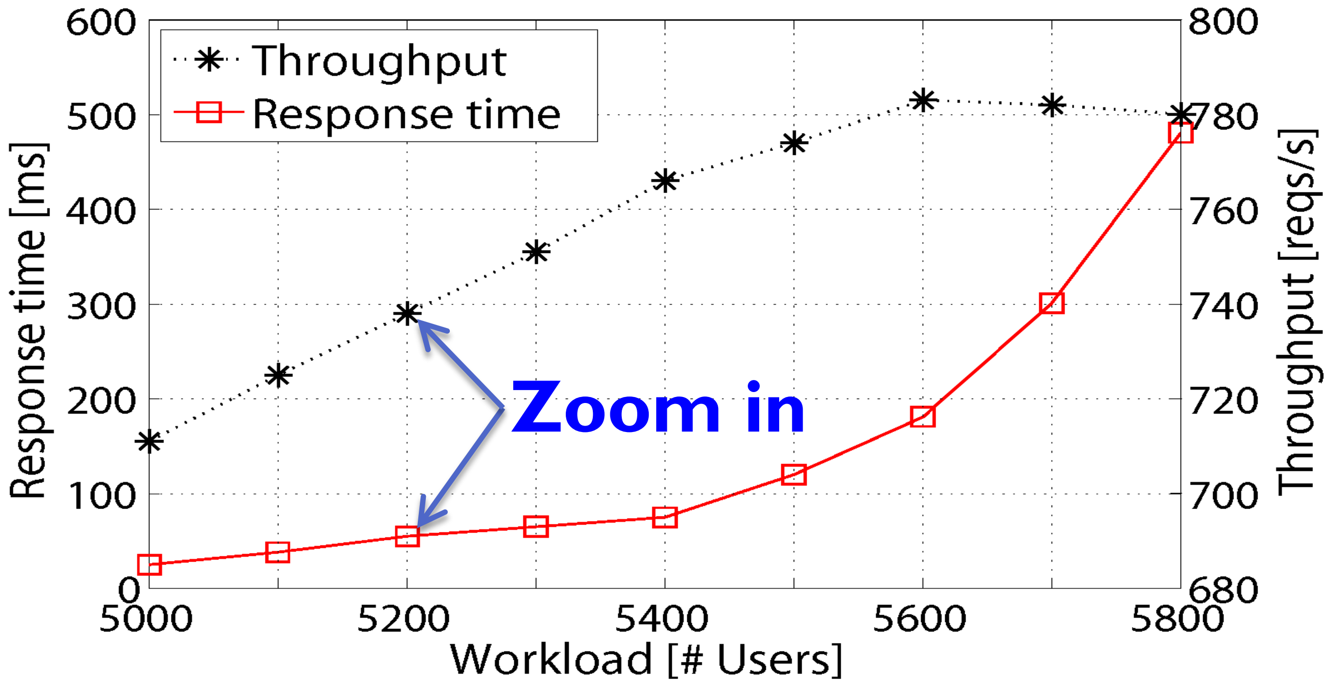
- Background & Motivation
- Analysis of the Large Response Time Fluctuations
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Solution
  - Transaction level scheduling
  - Limiting concurrency in the bottleneck tier
- Conclusion

Response Time is Important

- Response time is an important performance factor for Quality of Service (e.g., SLA for web-facing e-commerce applications).
  - Experiments at Amazon show that every 100ms increase in the page load decreases sales by 1%.
- Average response time may not be representative
  - We will show concrete instances of this phenomenon

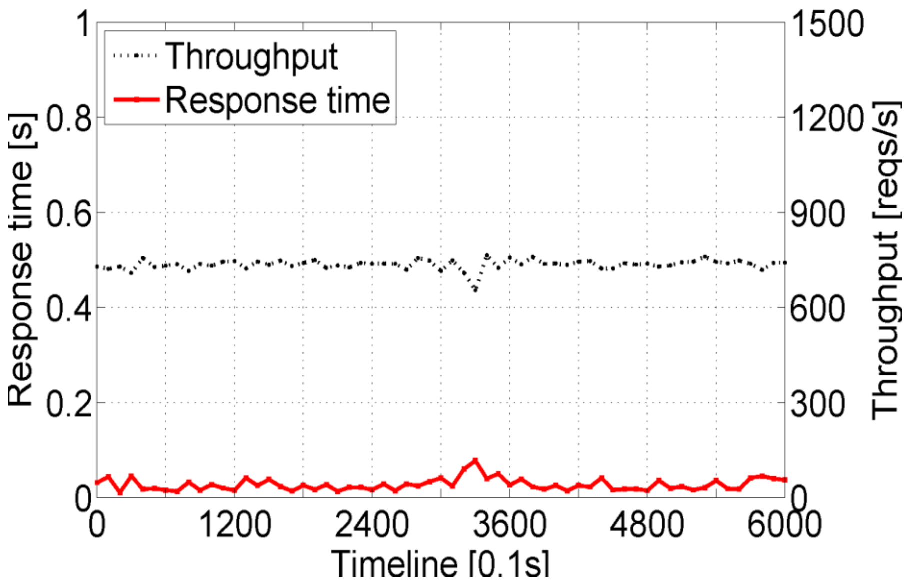
Motivational Example

- Response time and throughput of ten minutes benchmark on a 3-tier application with increasing workloads.
- What does the timeline graph look like?

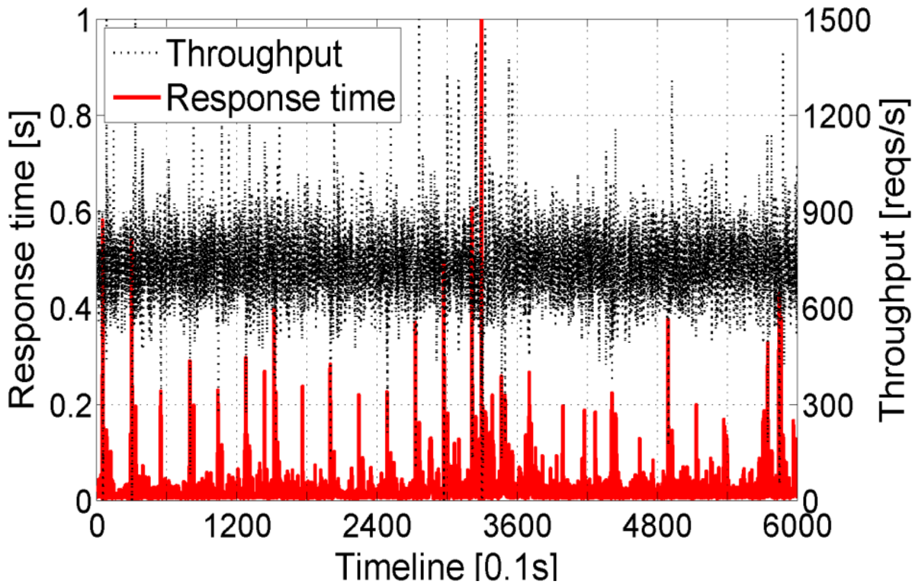


Motivational Example

Average at every 10s time interval

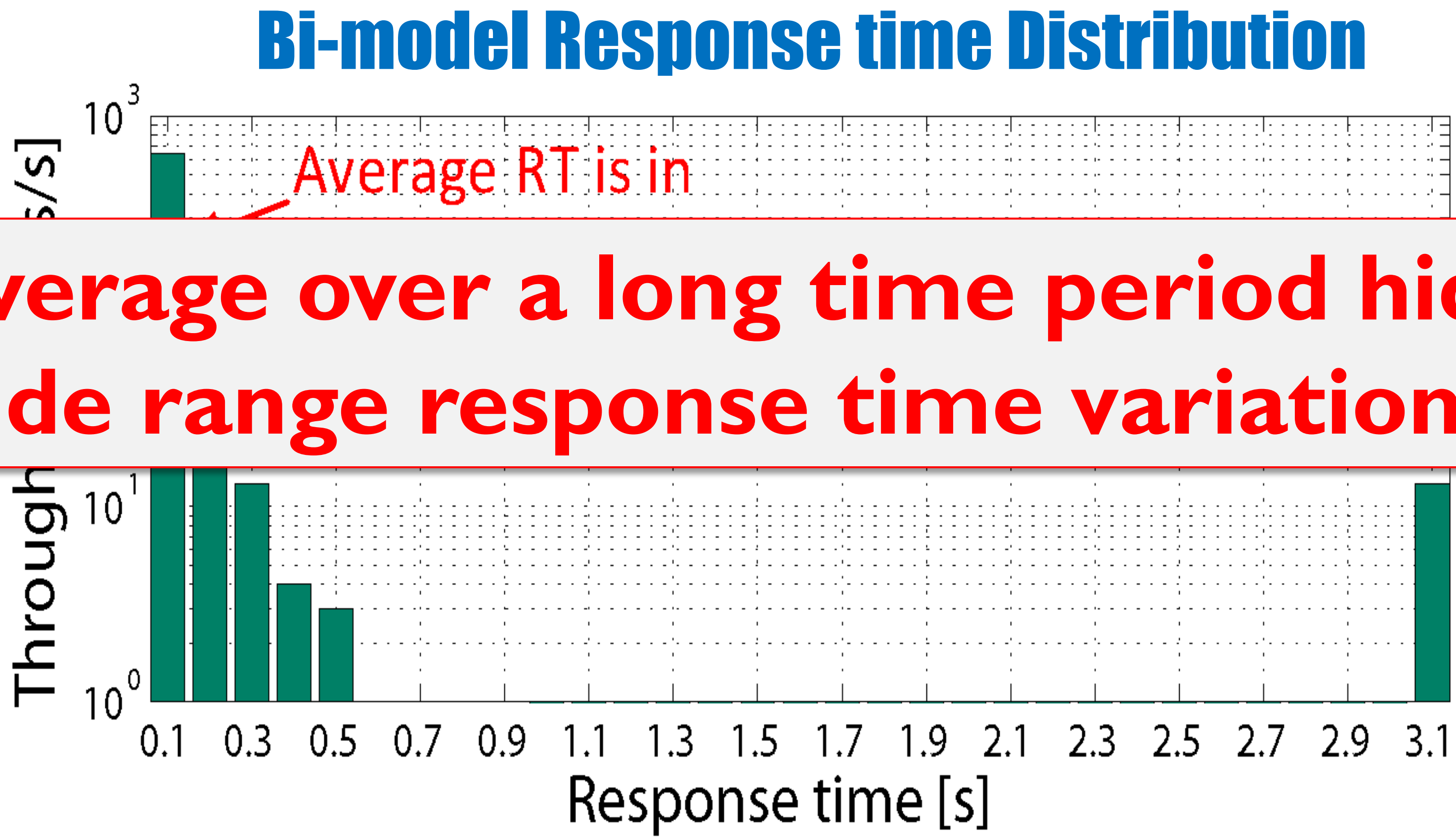


Average at every 100ms time interval



Motivational Example

- Statistic analysis of response time distribution



Average over a long time period hides wide range response time variations.

Goal of This Research

- Reveal the causes of large response time fluctuations in n-tier applications under high hardware utilizations.
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Show heuristics to mitigate large response time fluctuations.

Aim for more precise usage of response time as an index of application performance

Outline

- Background & Motivation
- Analysis of the Large Response Time Fluctuations
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Solution
  - Transaction level scheduling
  - Limiting concurrency in the bottleneck tier
- Conclusion

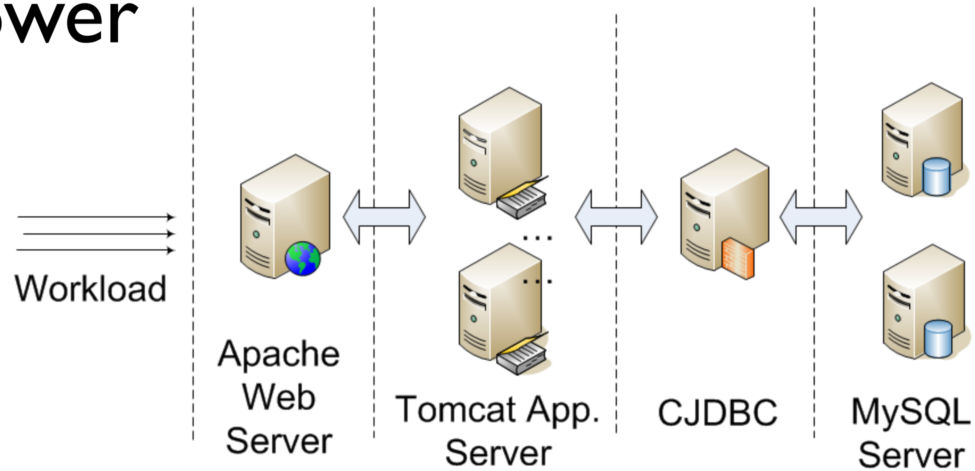
Experimental Setup (1): N-tier Application

Georgia Tech College of Computing

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

Experimental Setup (2): Hardware Configurations

- Commodity servers with different levels of processing power



Hardware	Processor # cores	Processor Freq.	Processor L2 Cache	Memory	Disk	Network
Large (L)	2	2.27GHz	2M	2GB	200GB	1Gbps
Medium (M)	1	2.4 GHz	4M	2GB	200GB	1Gbps
Small (S)	1	2.26GHz	512k	1GB	80GB	1Gbps

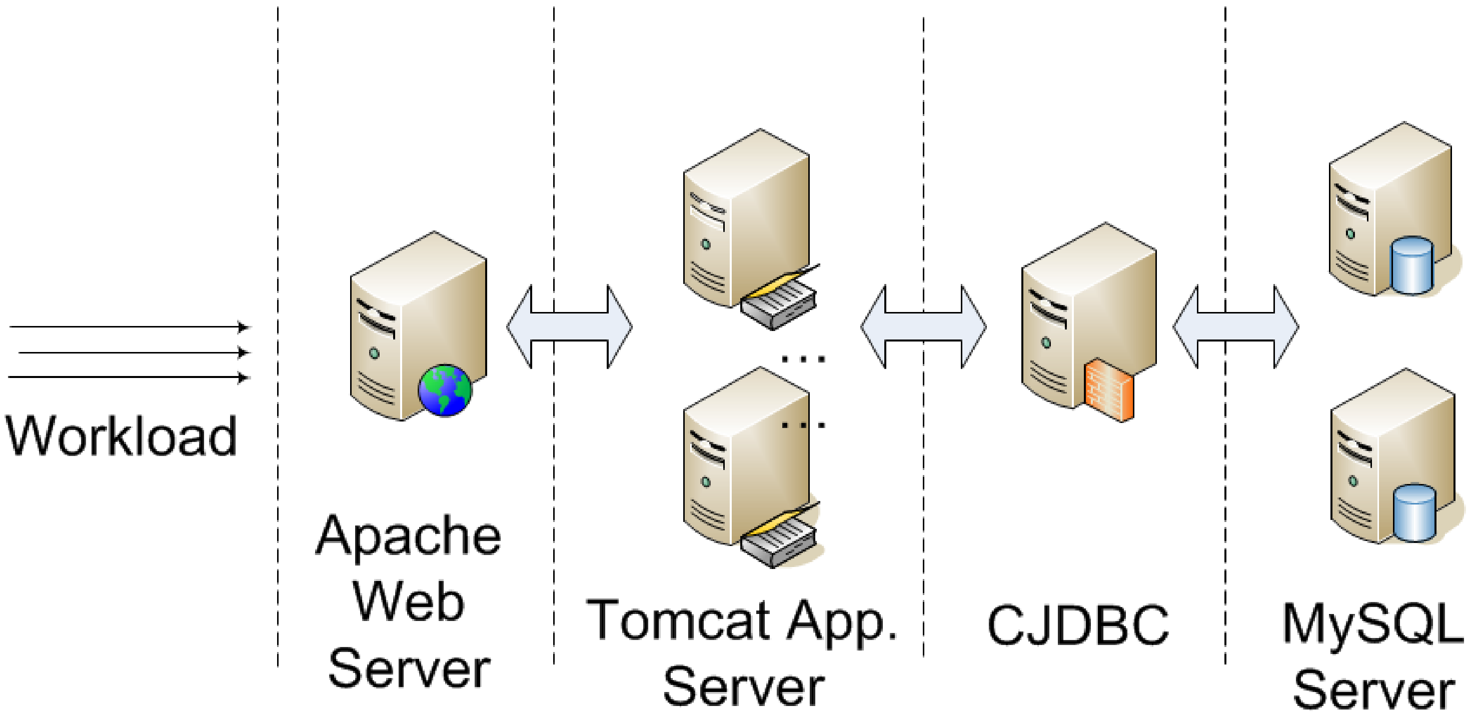
Experimental Setup (3): Software Configurations

Function	**Software**
Web server	Apache 2.0.54
Application server	Apache Tomcat 5.5.17
DB clustering middleware	C-JDBC 2.0.2
Database server	MySQL 5.0.51a
Java	Sun jdk1.6.0_23
Operating system	Redhat FC4
System Monitor	Sysstat 10.0.0.02, Collectl 3.5.1
Transaction monitor	Fujitsu SysViz

Experimental Setup (4): Sample Topology

- Notation

Sample topology (1/2/1/2)

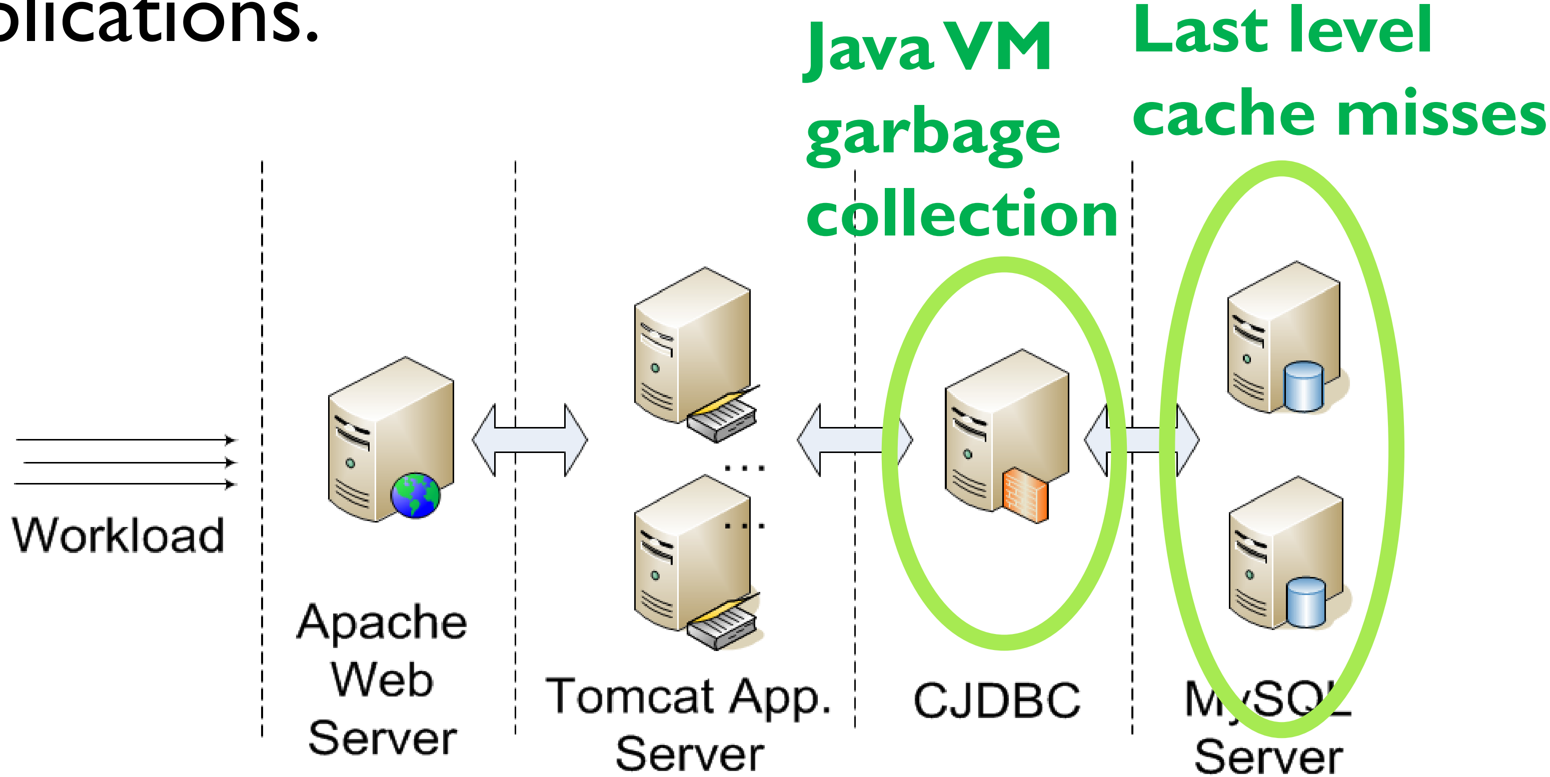


Outline

- Background & Motivation
- Analysis of the Large Response Time Fluctuations
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Solution
  - Transaction level scheduling
  - Limiting concurrency in the bottleneck tier
- Conclusion

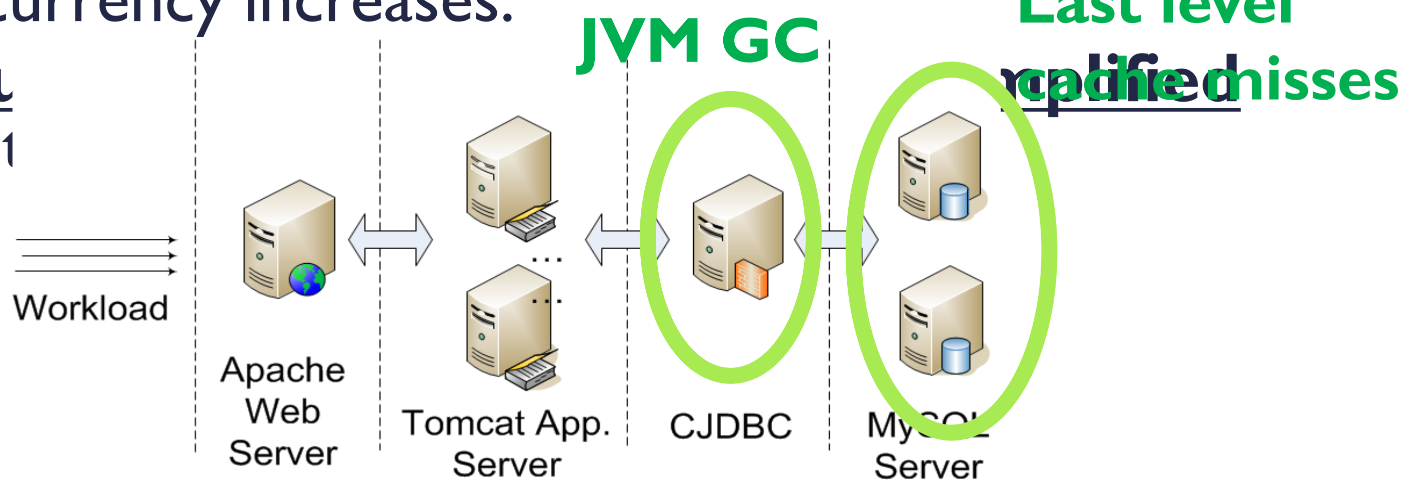
Transient Local Events

- Transient local events are pervasive in n-tier applications.



Negative Impact of Transient Local Events

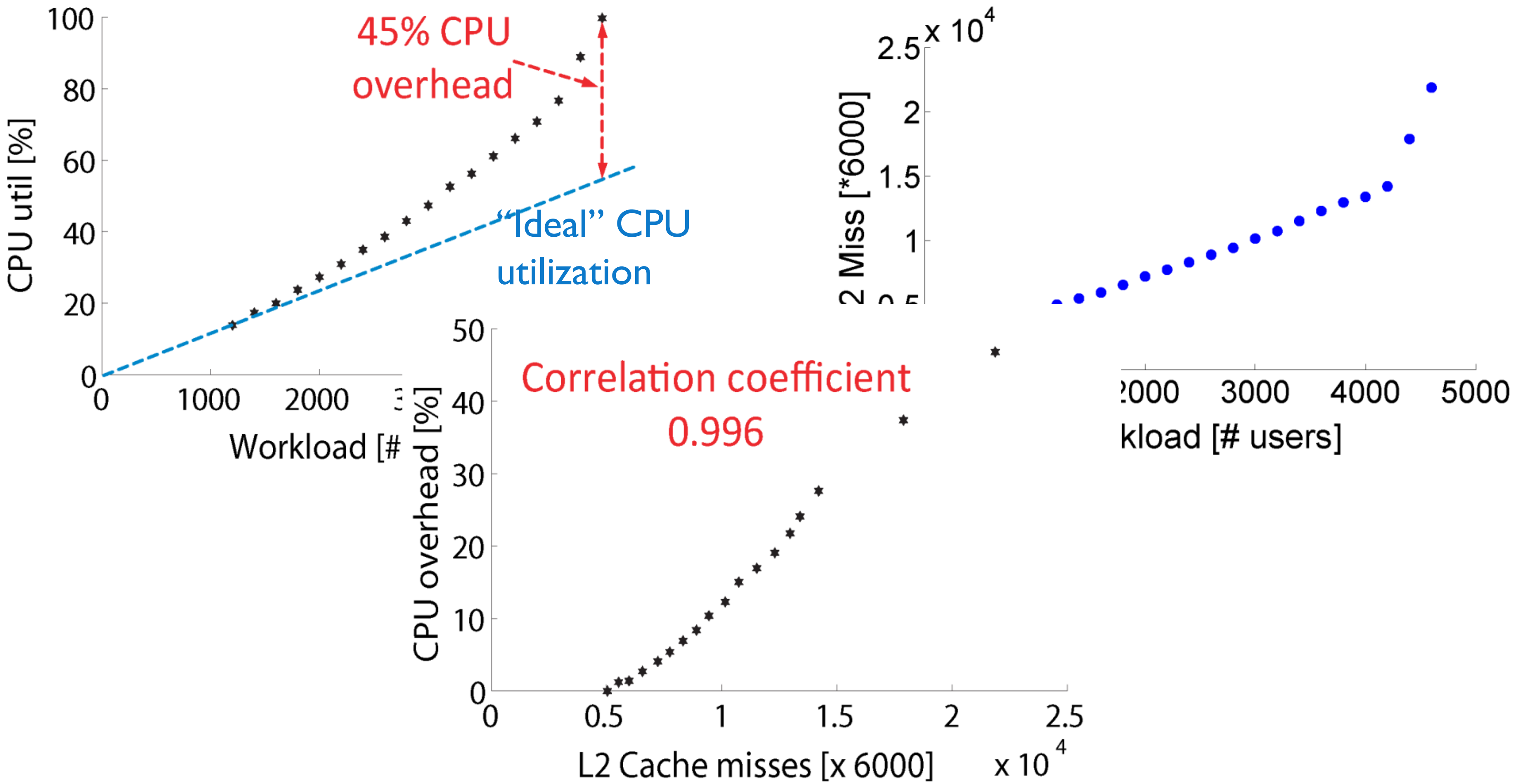
- High overhead caused by transient local events under high concurrency
  1. **Response time fluctuates slightly** in a tier under high workload.
  2. **Concurrency increases** as response time increases in the tier.
  3. **Overhead** caused by transient local events increases as concurrency increases.
  4. **The flu** ... mplified due to t



Non-Linear CPU Overhead Caused by Last Level Cache Misses

Non-linear increase of MySQL CPU utilization.

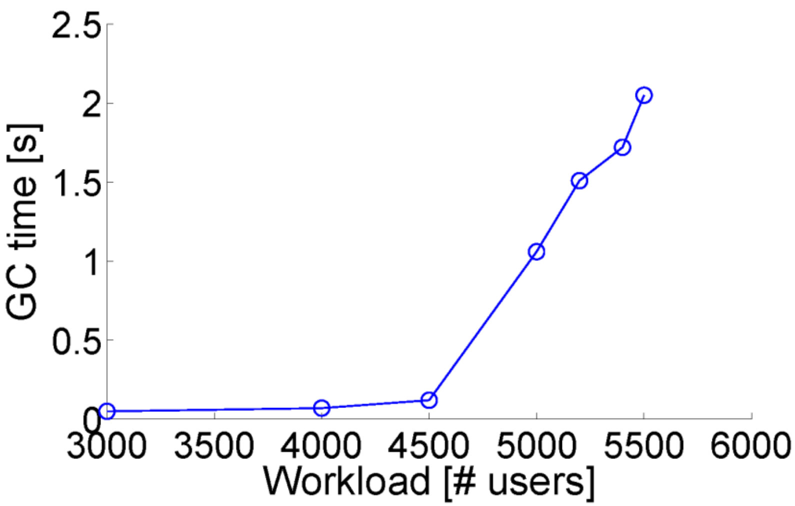
Non-linear increase of MySQL CPU Cache Miss



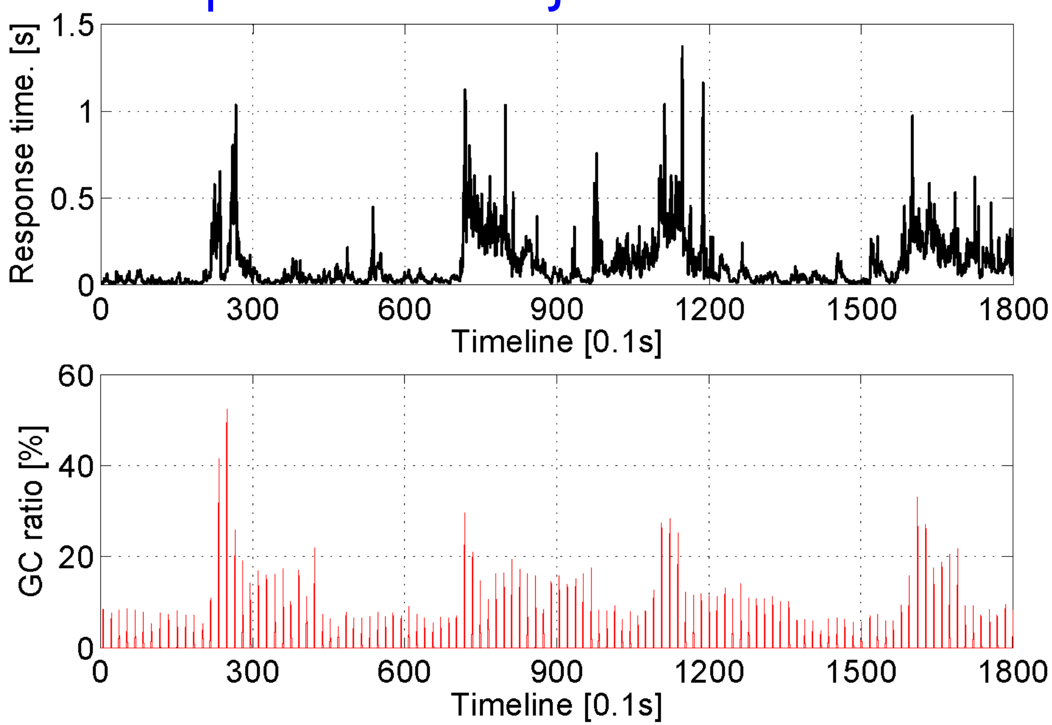
Non-Linear Increase of JVM GC as Workload Increases

- Negative impact of JVM GC
  - Consume CPU resources;
  - Increase the waiting time of pending requests.

CJDBC JVM GC time in 3 minutes



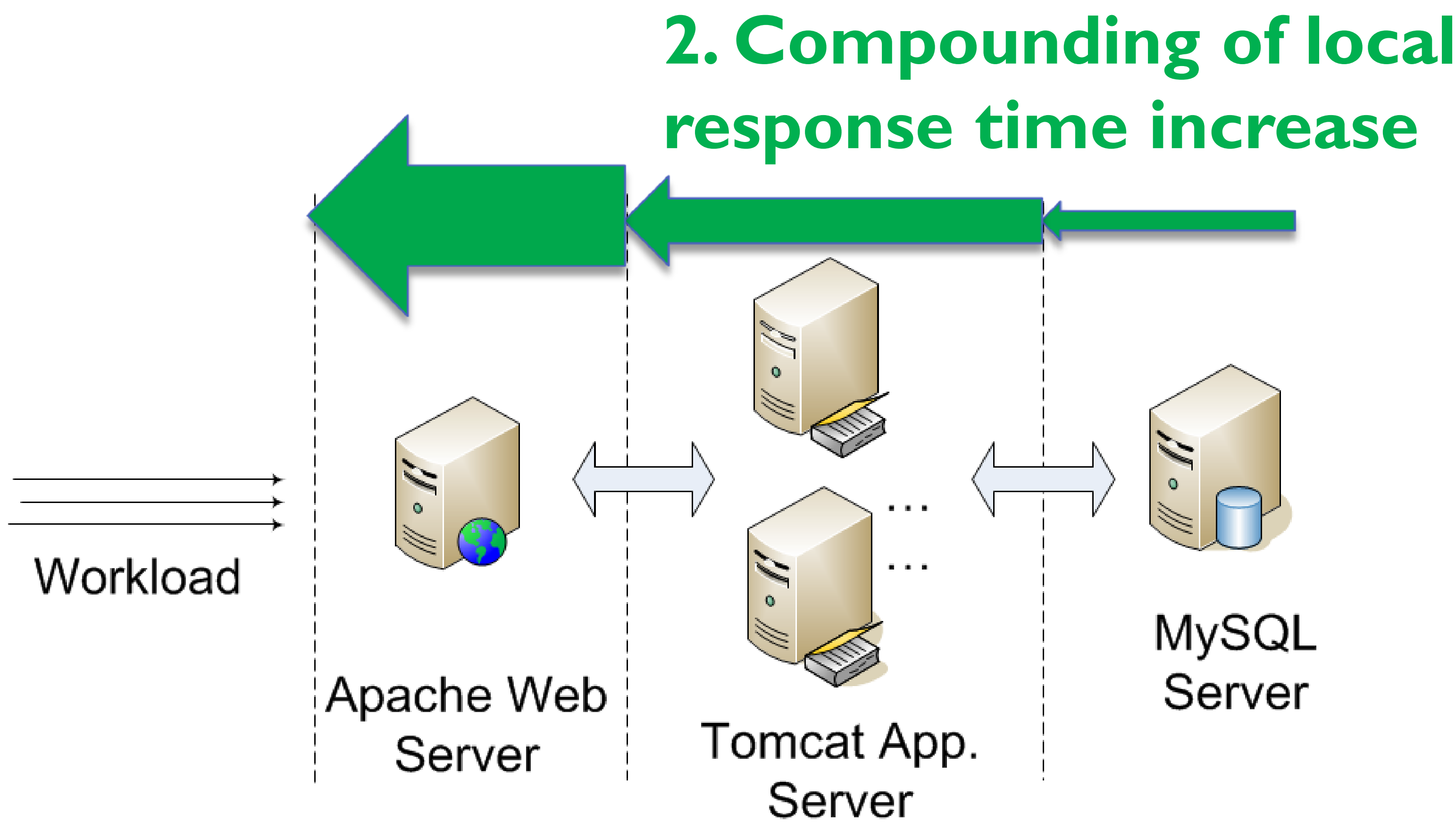
Response time and JVM GC in WL 5500



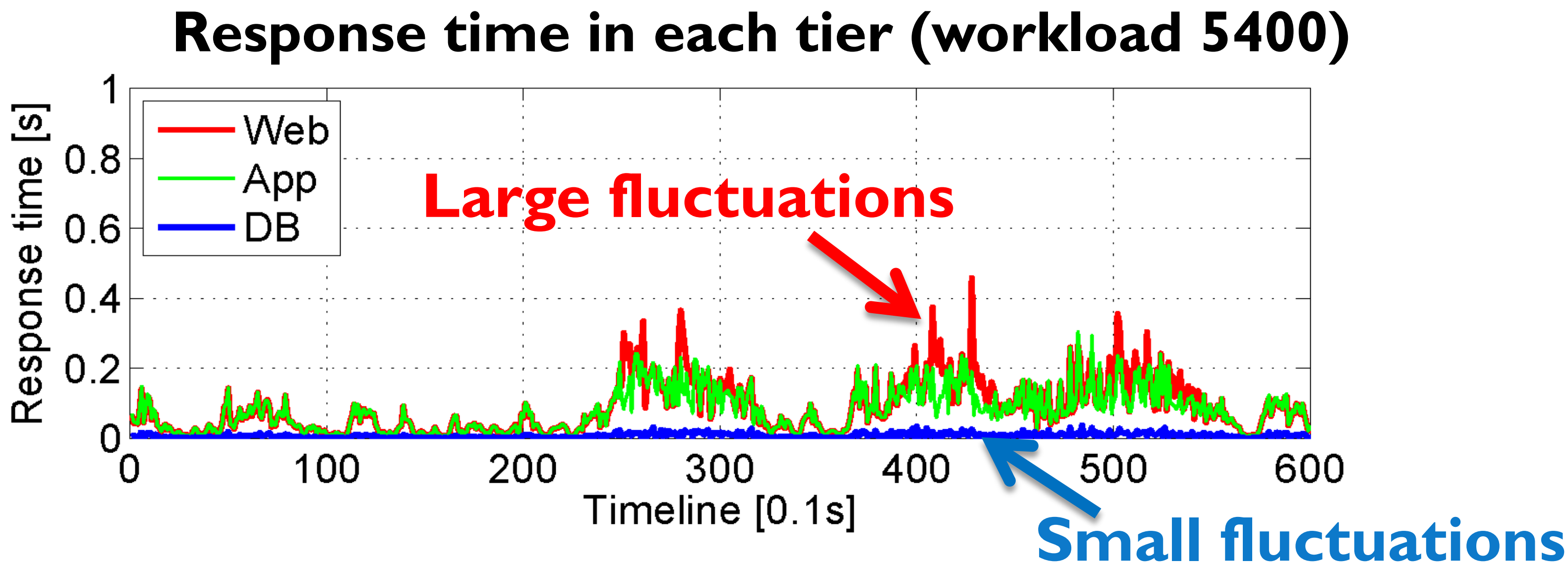
Outline

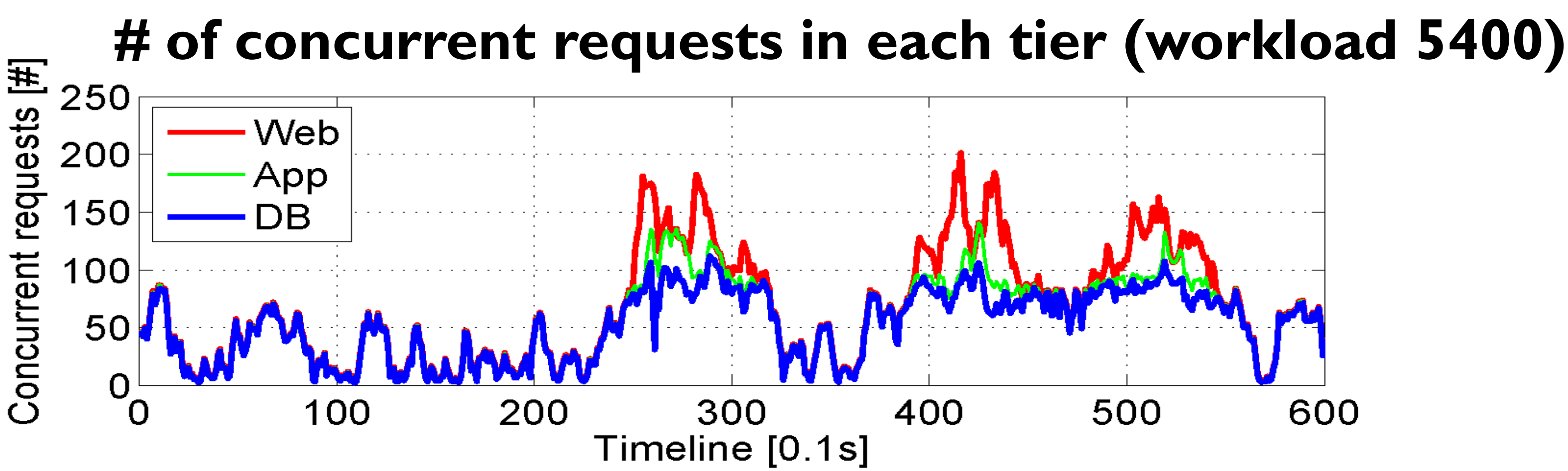
- Background & Motivation
- Analysis of the Large Response Time Fluctuations
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Solution
  - Transaction level scheduling
  - Limiting concurrency in the bottleneck tier
- Conclusion

Compounding of Local Response Time Increase



Bottom-Up Response Time Fluctuation Amplification





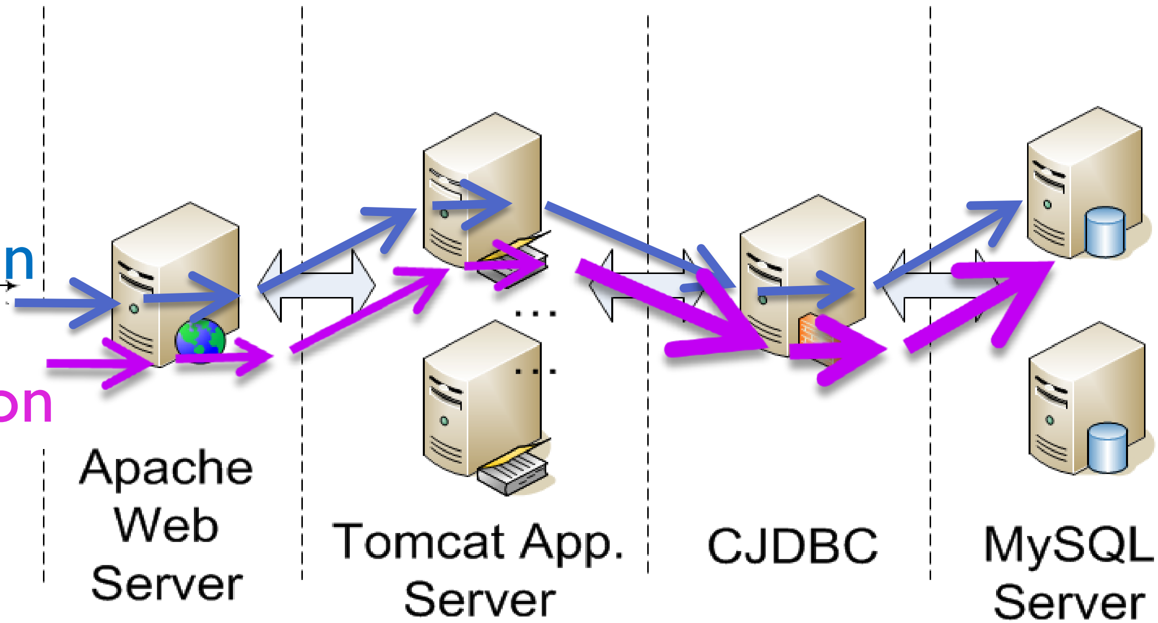
Outline

- Background & Motivation
- Analysis of the Large Response Time Fluctuations
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Solution
  - Transaction level scheduling
  - Limiting concurrency in the bottleneck tier
- Conclusion

Mix-Transaction Scheduling

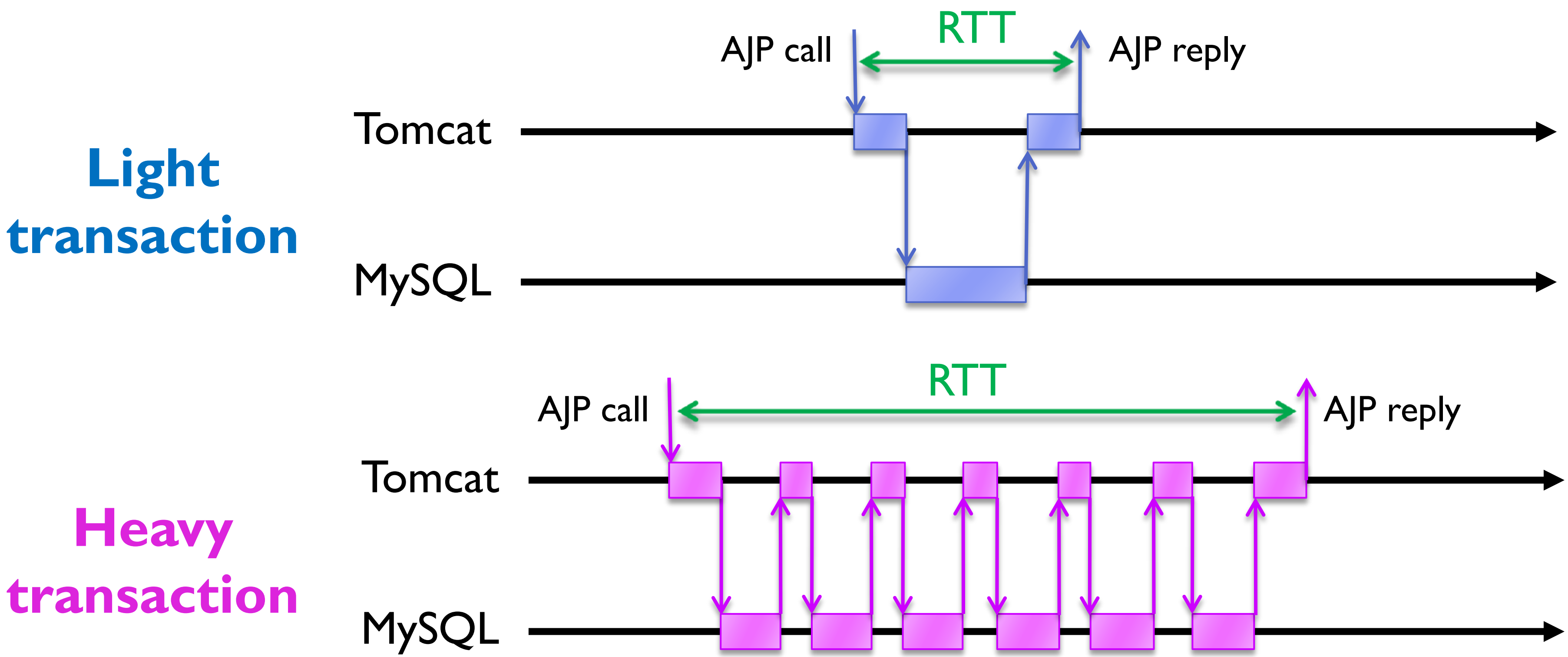
Light transaction

Heavy transaction



Limitations of Inner-Tier Job Scheduling in n-Tier Applications

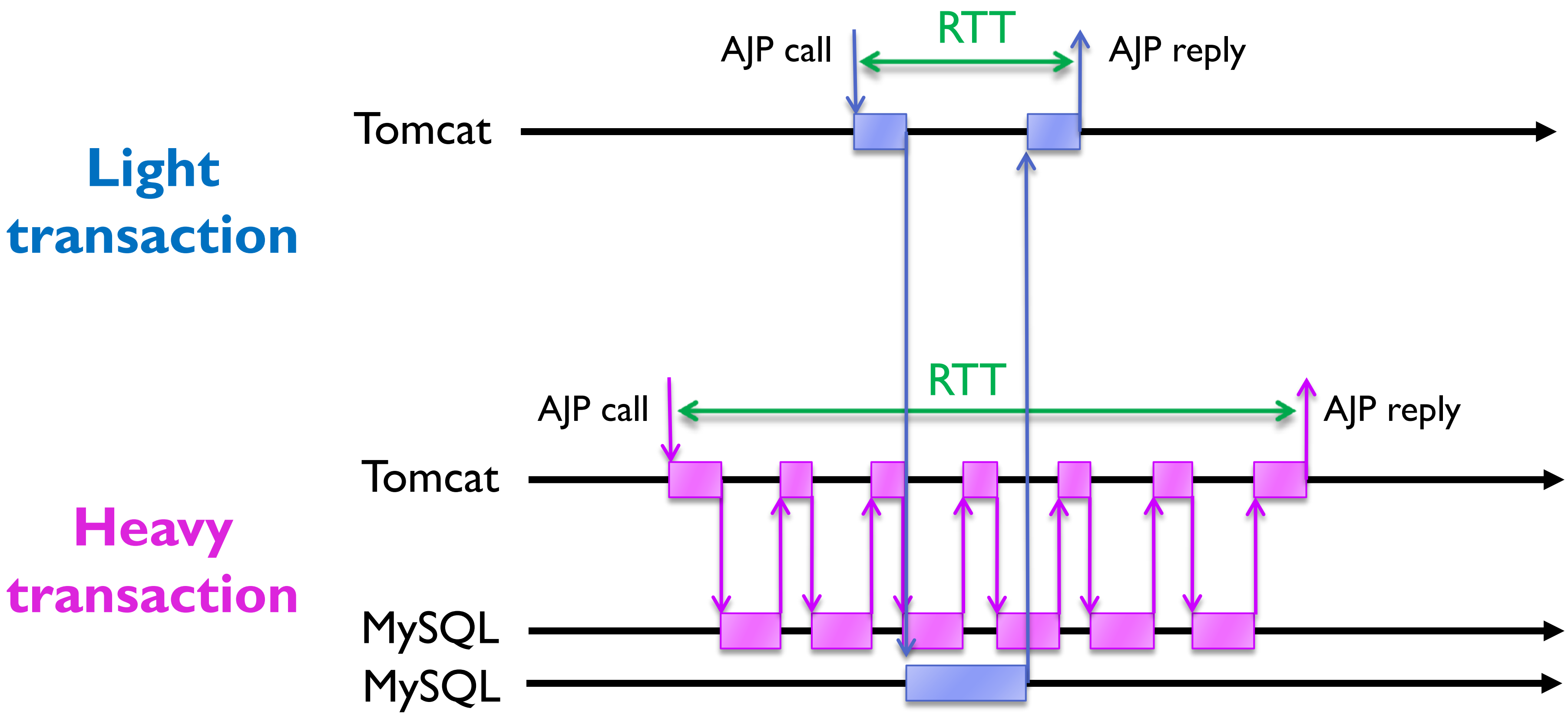
- Delay of light transaction processing due to interference of heavy transactions.



Limitations of Inner-Tier Job Scheduling in n-Tier Applications

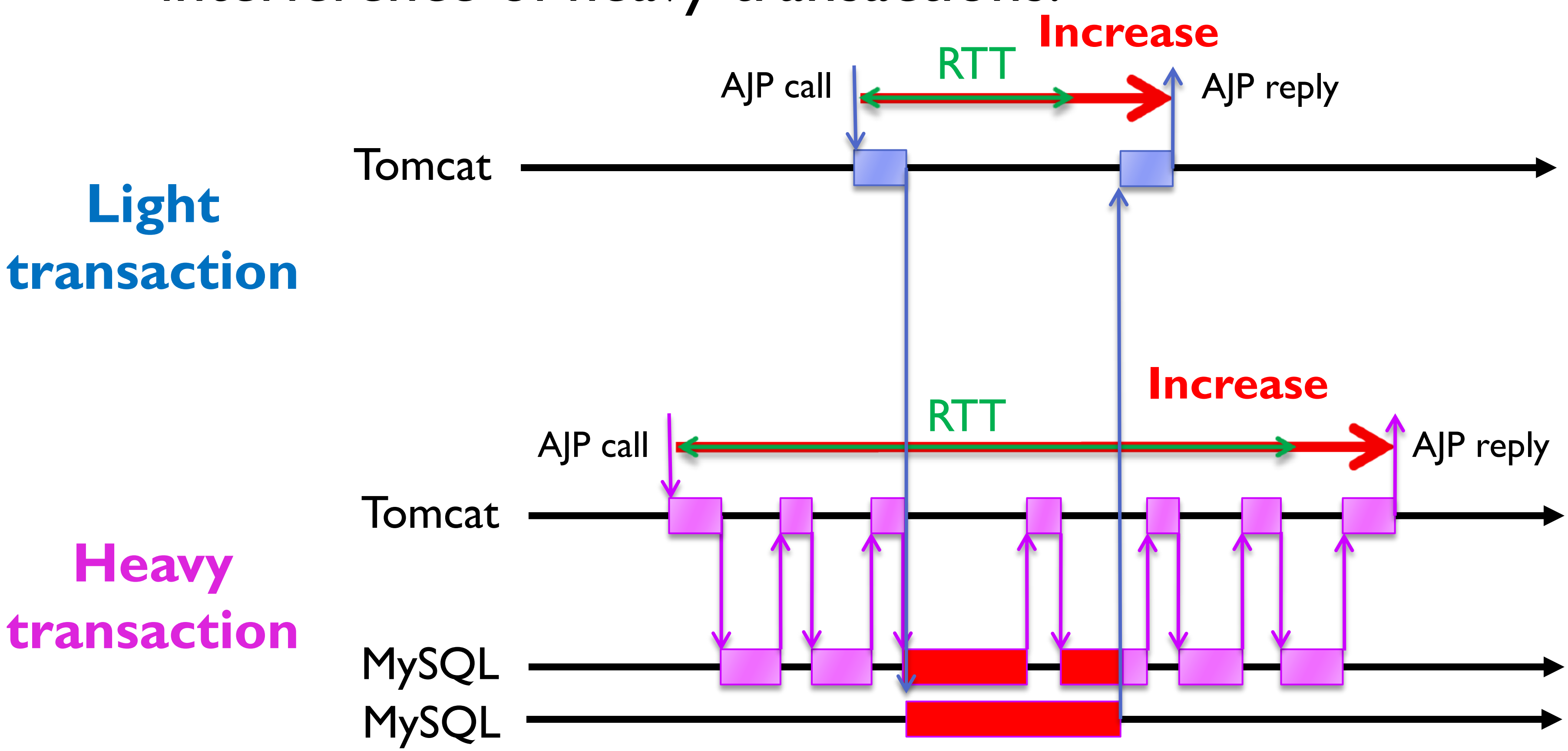
Georgia Tech College of Computing

- Delay of light transaction processing due to interference of heavy transactions.



Limitations of Inner-Tier Job Scheduling in n-Tier Applications

- Delay of light transaction processing due to interference of heavy transactions.

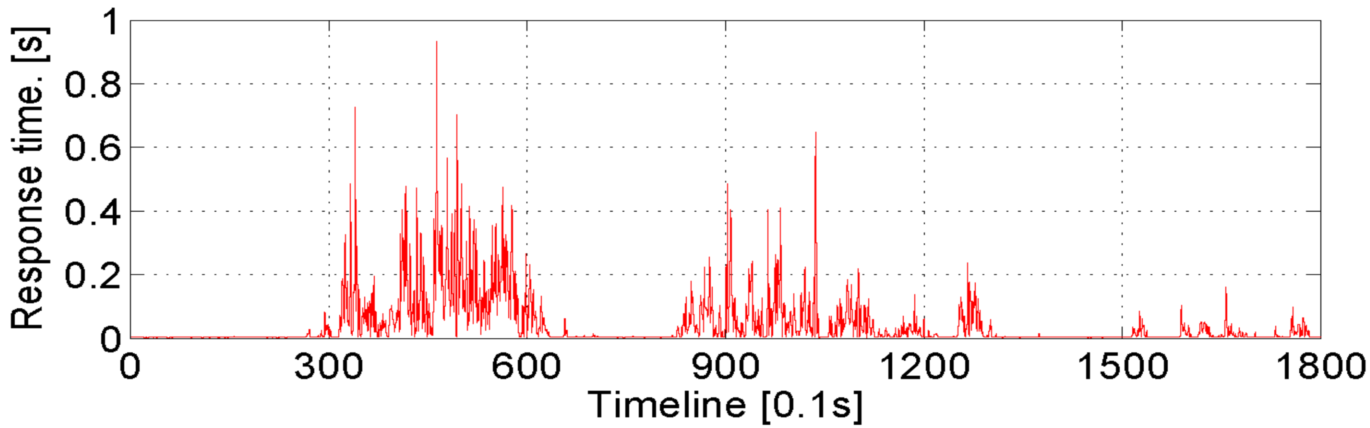


Limitations of Inner-Tier Job Scheduling in n-Tier Applications

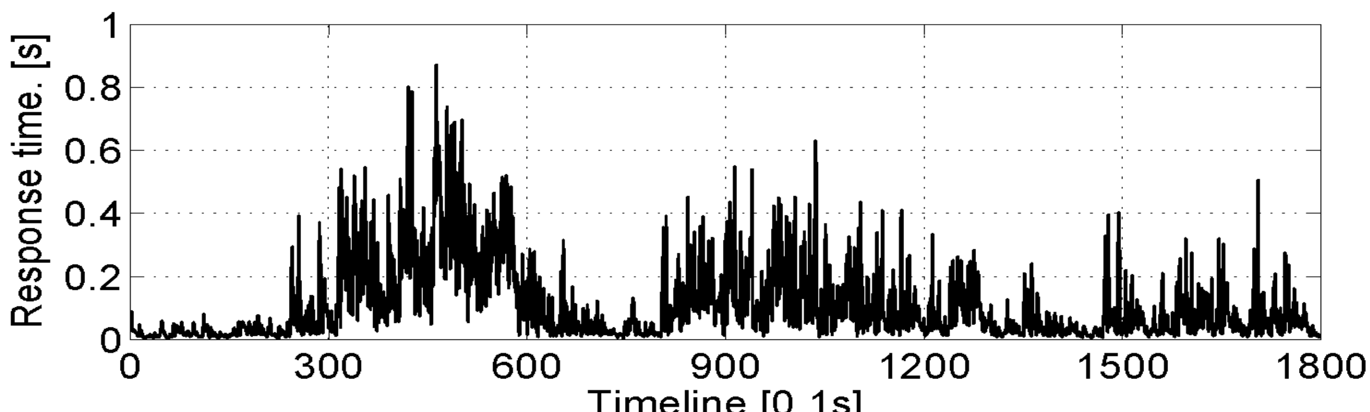
Georgia Tech College of Computing

- Delay of light transaction processing due to interference of heavy transactions.

Lightest transaction response time



Mix-transaction response time



Outline

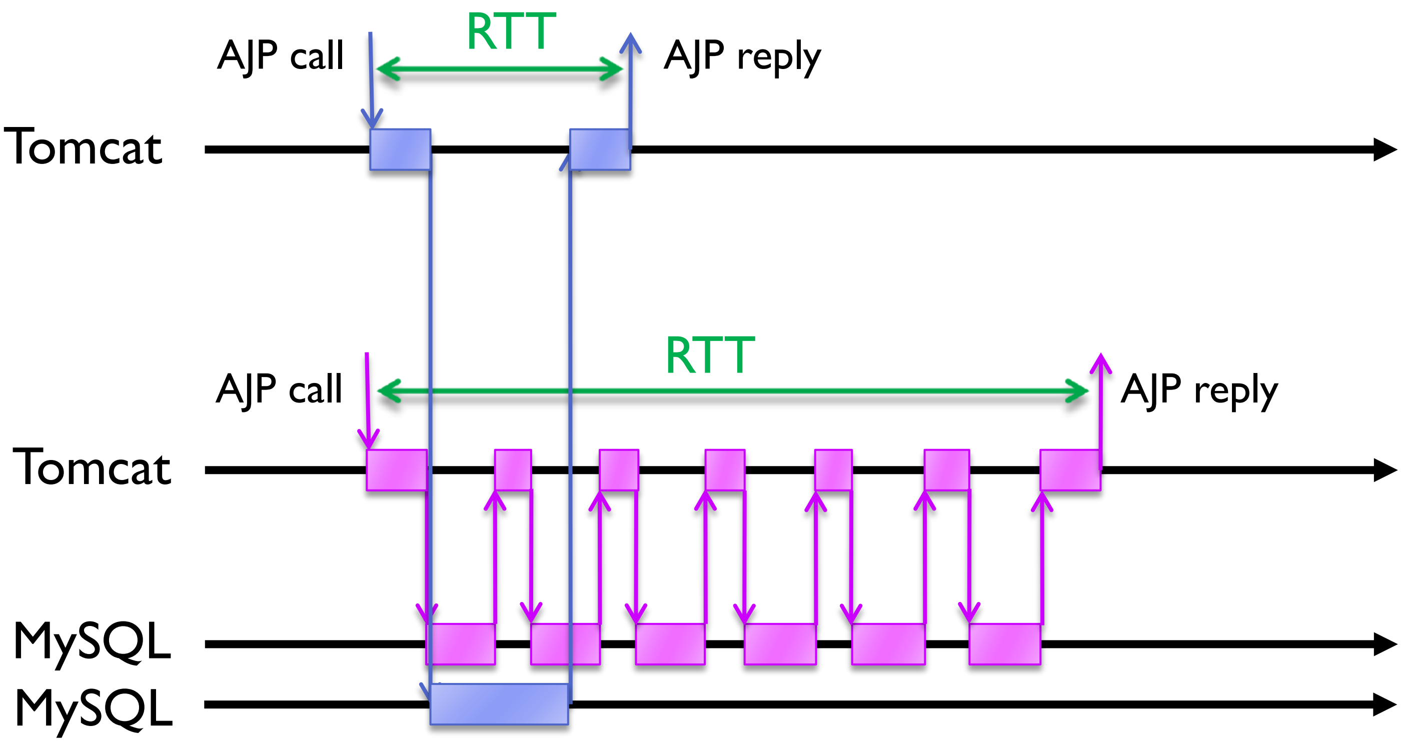
- Background & Motivation
- Analysis of the Large Response Time Fluctuations
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Solution
  - Transaction level scheduling
  - Limiting concurrency in the bottleneck tier
- Conclusion

Heuristic I: Transaction Level Scheduling

- Heuristic (i): We need to grant **higher priority** to **light transactions**; schedule transactions in an upper tier which can distinguish light from heavy.

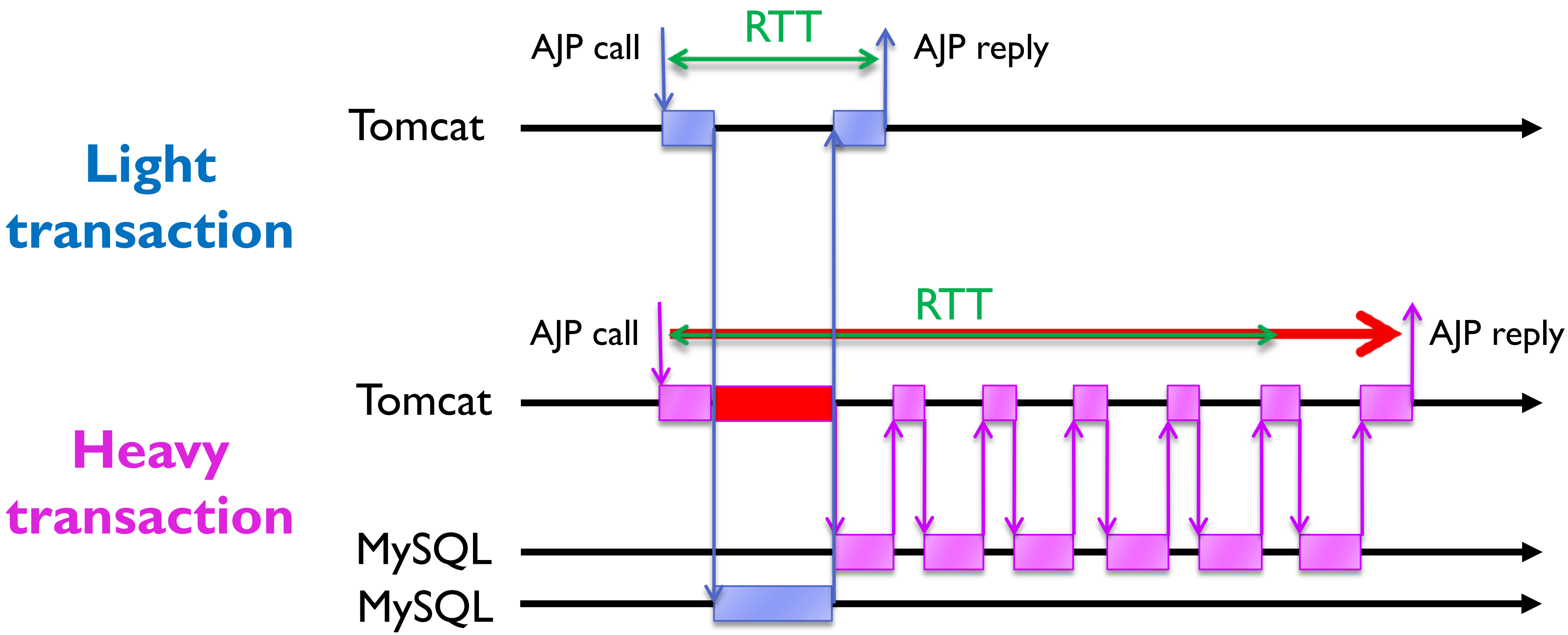
Light transaction

Heavy transaction



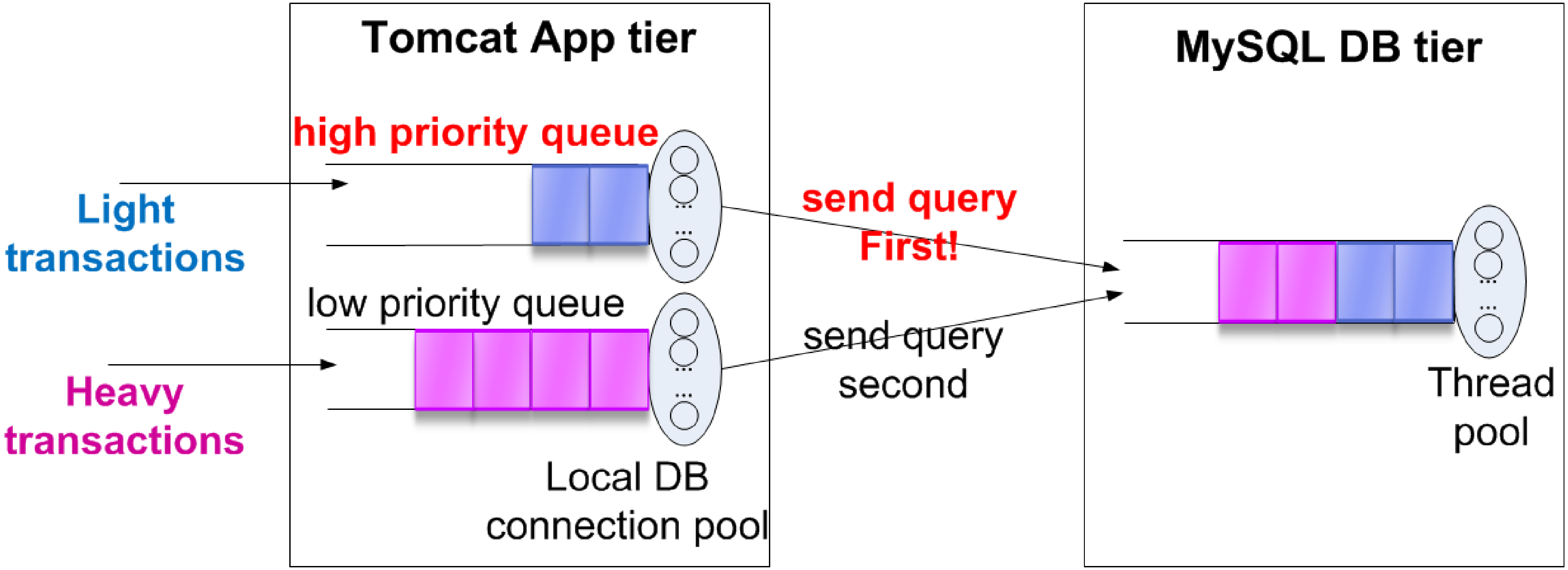
Heuristic I: Transaction Level Scheduling

- Heuristic (i): We need to grant **higher priority** to **light transactions**; schedule transactions in an upper tier which can distinguish light from heavy.



Heuristic I: Transaction Level Scheduling

- Heuristic (i): We need to grant **higher priority** to **light transactions**; schedule transactions in an upper tier which can distinguish light from heavy.

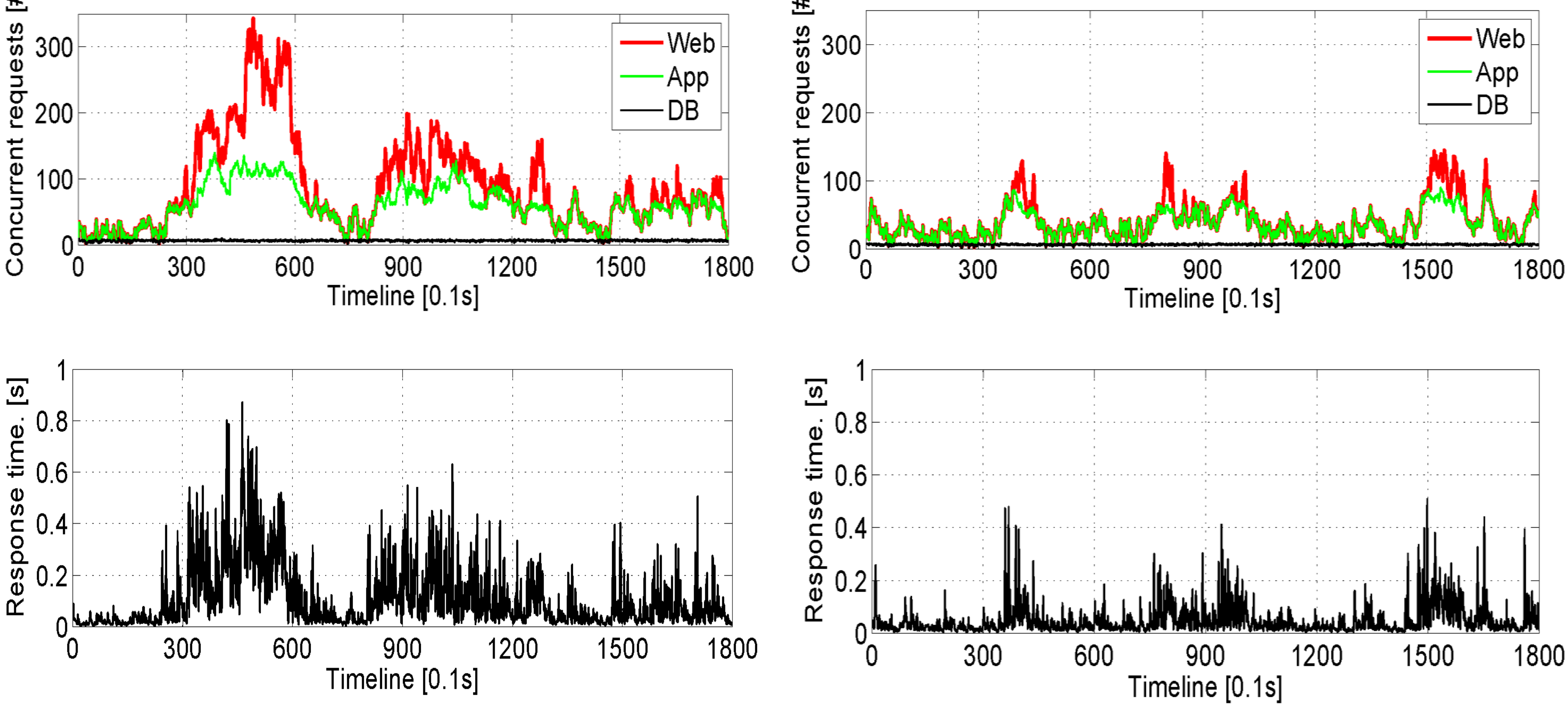


Cross-tier-priority based scheduling

Heuristic I: Transaction Level Scheduling

1/2/1 configuration, workload 5800

Original DBconn2 Case | **DBconn2 CTP Scheduling Case**

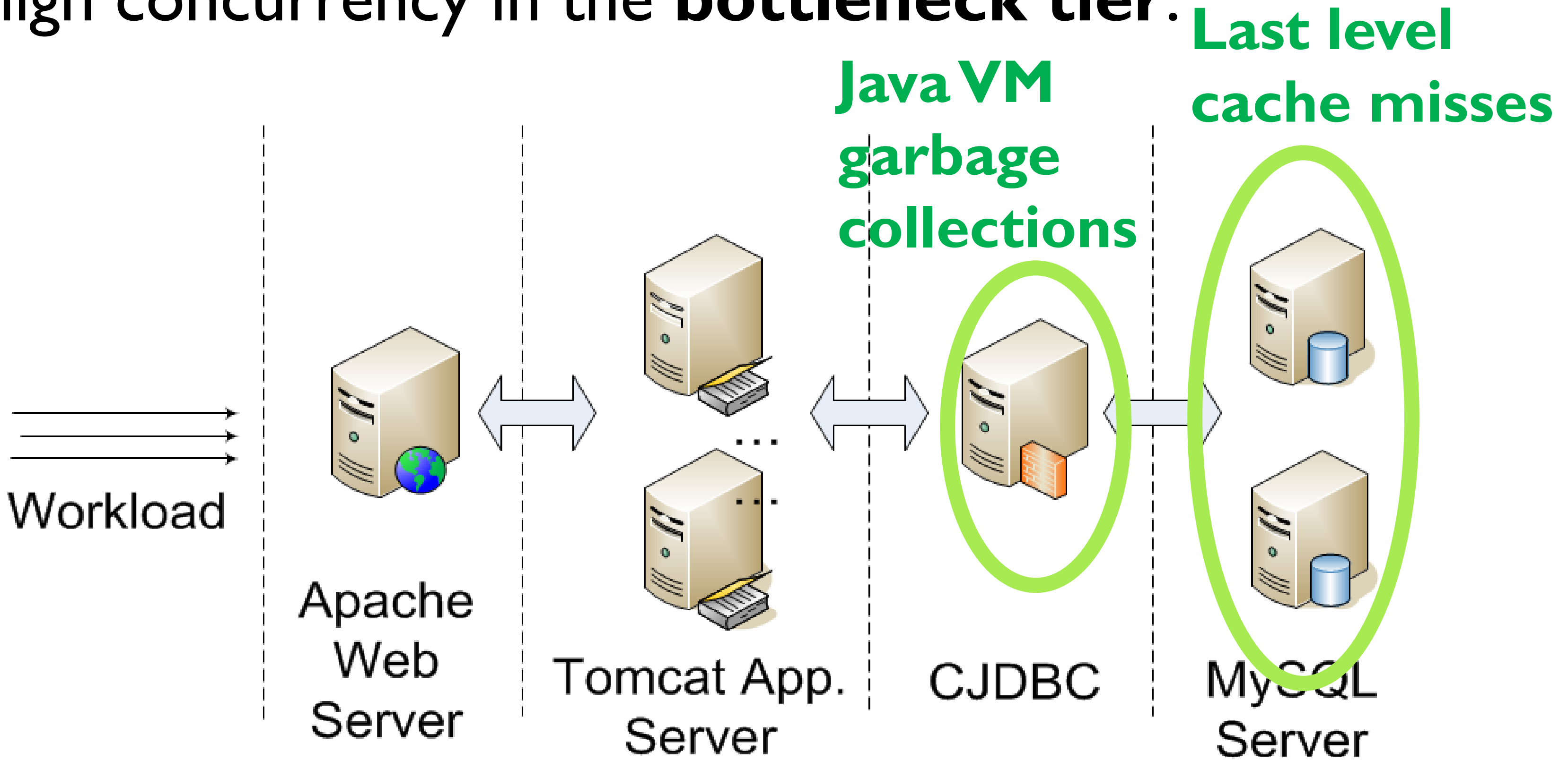


Outline

- Background & Motivation
- Analysis of the Large Response Time Fluctuations
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Solution
  - Transaction level scheduling
  - Limiting concurrency in the bottleneck tier
- Conclusion

Heuristic II: Limiting Concurrency in Bottleneck Tier

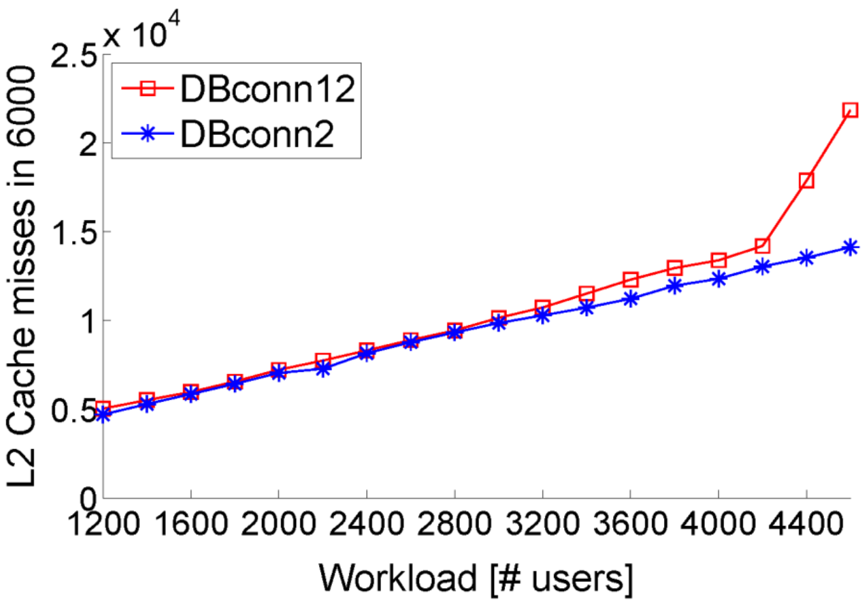
- Heuristic (ii): We need to **restrict** the number of **concurrent requests** to avoid overhead caused by high concurrency in the **bottleneck tier**.



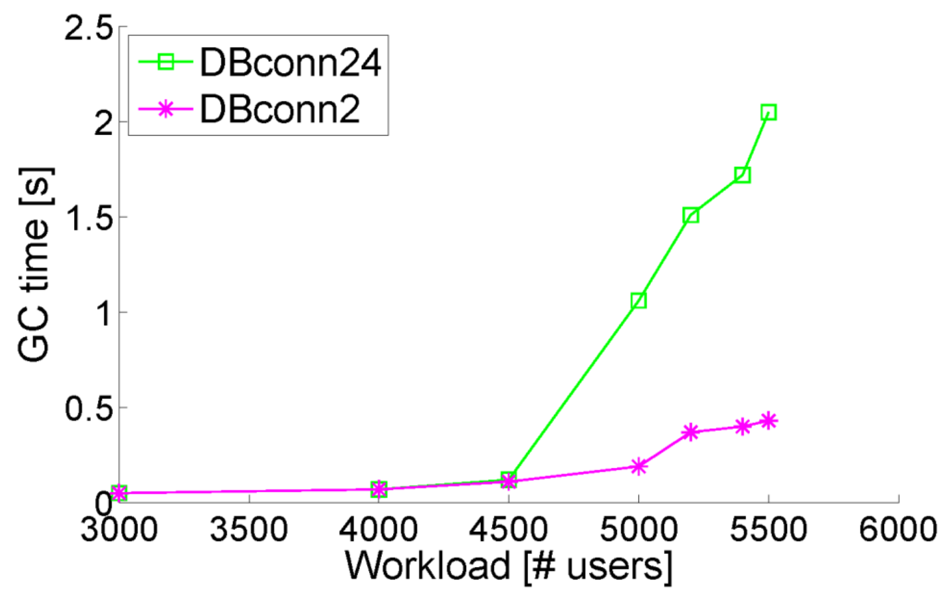
Heuristic II: Limiting Concurrency in Bottleneck Tier

- Heuristic (ii): We need to **restrict** the number of **concurrent requests** to avoid overhead caused by high concurrency in the **bottleneck tier**.

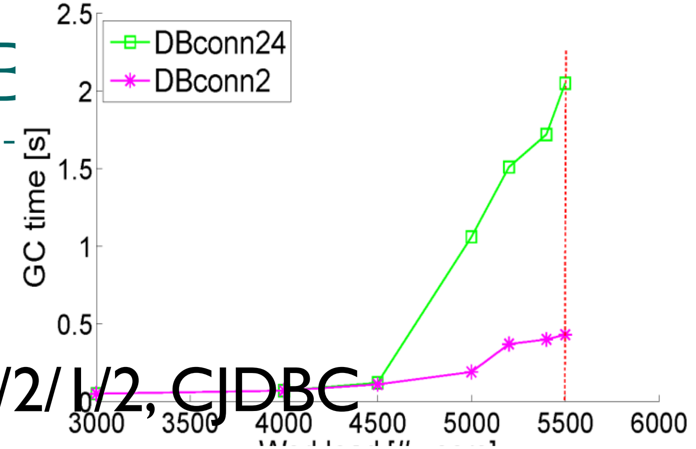
1/2/1, MySQL L2 cache miss



1/2/1/2, CJDBC JVM GC

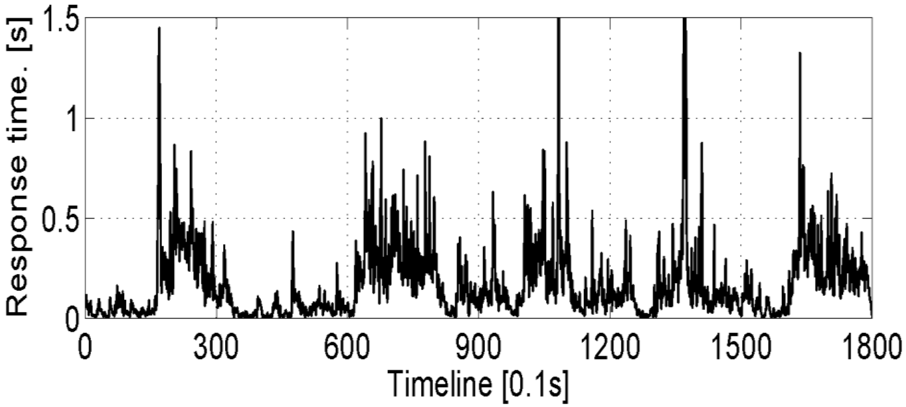


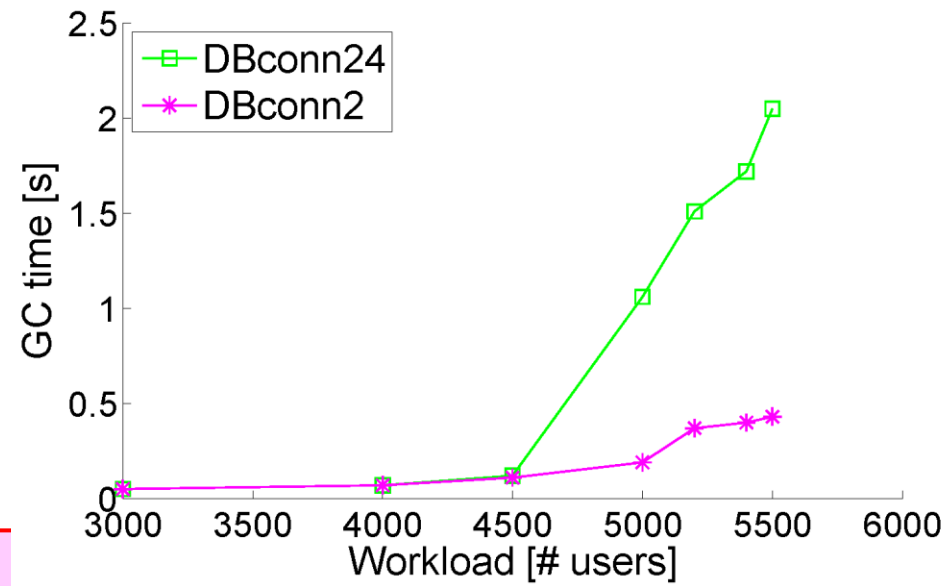
Heuristic II:
Limiting Concurrency in E



1/2/1/2, CJDBC

DBconn24 case





Limiting concurrency in bottleneck tiers can mitigate the large fluctuations of end-to-end response time.

Outline

- Background & Motivation
- Analysis of the Large Response Time Fluctuations
  - Transient local events
  - Compounding of local response time increase
  - Mix-transaction scheduling
- Solution
  - Transaction level scheduling
  - Limiting concurrency in the bottleneck tier
- Conclusion

Conclusion

- Under high resource utilization:
  - Average response time may not be representative to system performance.
  - Beyond bursty workload, many system environmental conditions cause large response time fluctuation.
- To reduce wide range response time variations:
  - **Transaction level scheduling** is useful.
  - **Concurrency settings** of an n-tier application needs to be optimized.
- Ongoing work: More analysis of system environmental conditions

Thank You. Any Questions?

Qingyang Wang
qywang@cc.gatech.edu