

Performance tuning for Java applications

George Barnett, Atlassian

Topics

- ▶ How to make your apps run faster
- ▶ A few process tips
- ▶ Tuning ideas
- ▶ What doesn't work

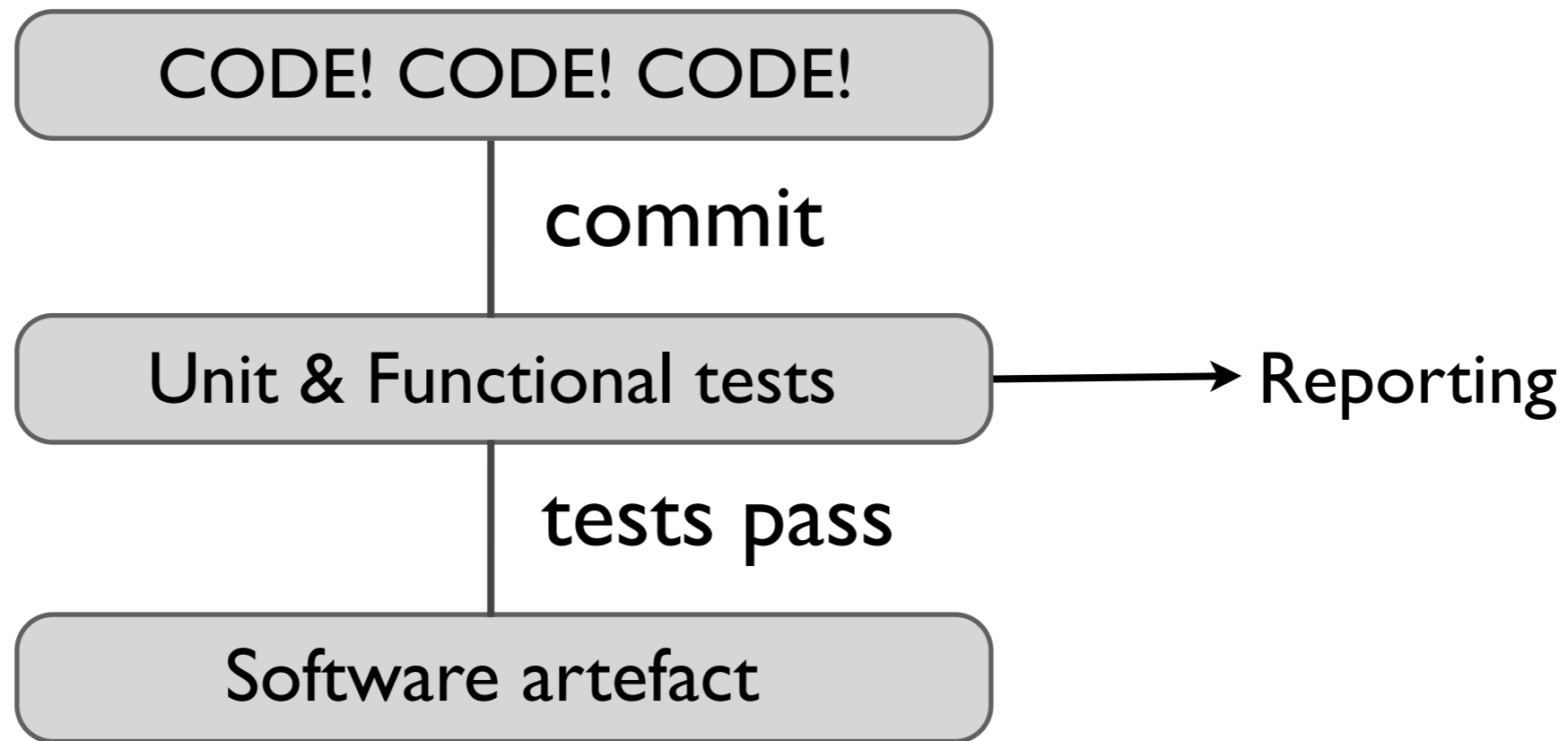
A Note

- ▶ X vs Y will always depend on the context
- ▶ Test with your application!

Performance Engineering

- ▶ Isolate performance issues before they become headaches in production.
- ▶ Make sure these issues get fixed.

Simple code lifecycle



Add to your testing



- Catch regressions on commit
- Create a standard benchmark
- Performance testing will save your Ops team time

Getting started

- ▶ Define test cases and traffic levels
 - ▶ Examine logs and monitoring data
- ▶ Use real world data
 - ▶ Eg, Public Atlassian instances such as <http://jira.atlassian.com>

Some useful tools

- ▶ Apache JMeter

- ▶ Howto:

- ▶ http://blogs.atlassian.com/developer/2008/10/performance_testing_with_jmete.html

- ▶ Example

- ▶ <http://confluence.atlassian.com/display/DOC/Performance+Testing+Scripts>

Some useful tools

- ▶ Automation
 - ▶ Maven
 - ▶ Bamboo
- ▶ Getting a repeatable build is critical!
- ▶ Automated Performance Testing will save your Dev team time!

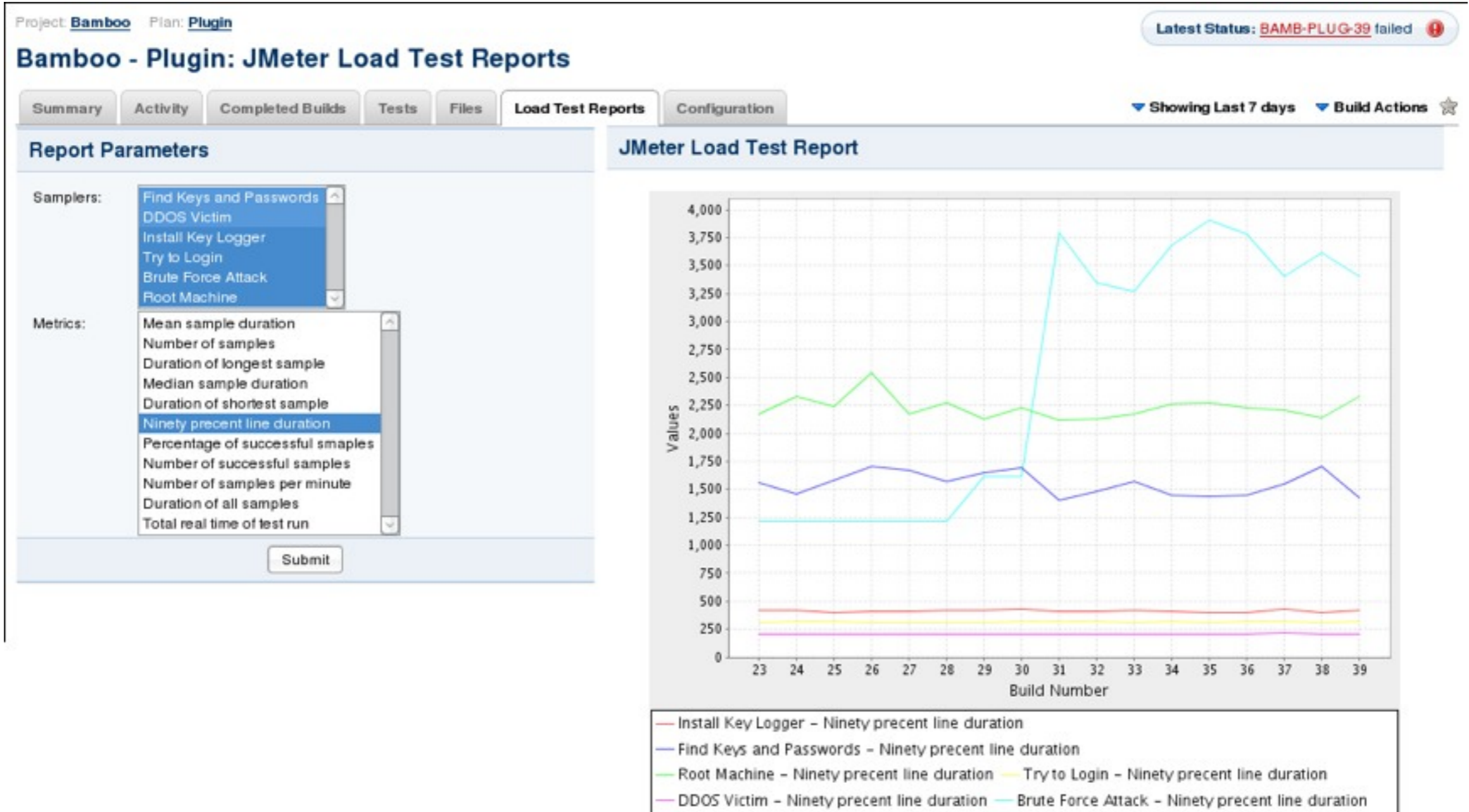
How often?

- ▶ Daily performance tests
 - ▶ Most products & libraries
- ▶ Weekly soak tests
 - ▶ Run for much longer than daily tests

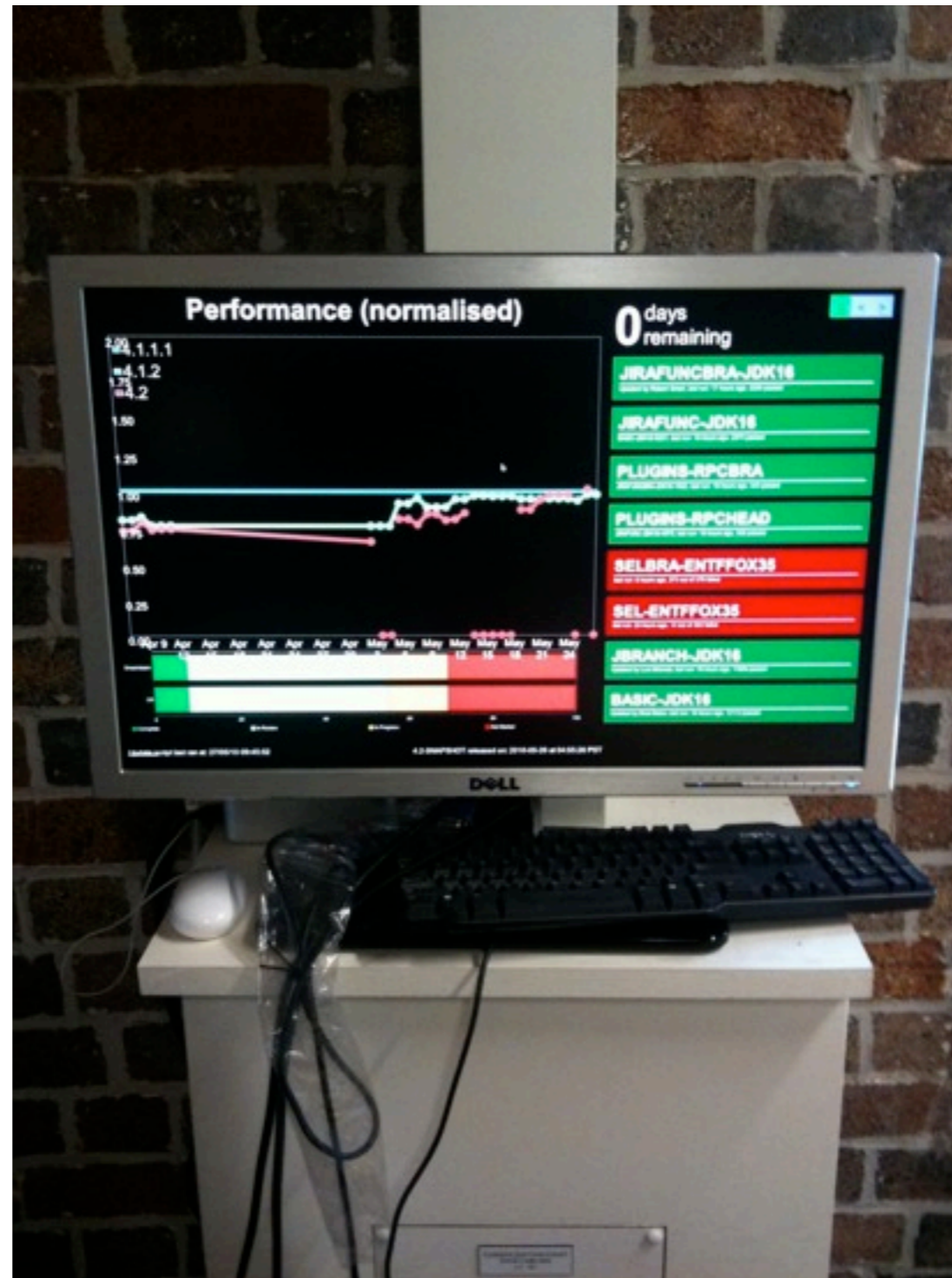
Reporting

- ▶ Put there data where you cant miss it
 - ▶ Build screens, dashboards, notifications
 - ▶ JMeter Aggregator Plugin


Reporting



Reporting



You are here

- ▶ How I did the testing 
- ▶ Hardware & Software
- ▶ Must have improvements
 - ▶ Java, Hardware, Virtualization and Garbage Collection
- ▶ Worth doing
 - ▶ Gigabit vs 100mbit, Heap size tuning, use an SSD
- ▶ Sadly, not useful
 - ▶ Tune MySQL, Replace MySQL

The contenders

- ▶ Dell PE 850 – “WALL-E”
- ▶ Xeon X3220 2.4Ghz Quad Core
 - ▶ Quad Core, 2 x 4Mb L3
- ▶ 8Gb DDR2 RAM
- ▶ 2 x 7.2K SAS Disks
- ▶ 1 x Intel X25-E 32GB SSD
- ▶ ~\$1500



The contenders

- ▶ Dell PE 2950 – “Johnny 5”
- ▶ 2 x Xeon E5405 2.0Ghz
- ▶ Quad Core, 2 x 6M L3
- ▶ 32Gb RAM
- ▶ 2 x 72Gb 15K Drives
- ▶ ~\$4000



The contenders

- ▶ Dell PE R610 – “EVE”
- ▶ 2 x Xeon E5520 2.2Ghz
 - ▶ Quad Core, 8M “SmartCache”
- ▶ 32Gb RAM
- ▶ 2 x 146G 15K drives
- ▶ ~\$4000




The Workload

- ▶ JIRA 4, MySQL 5, RHEL 5
- ▶ Java 1.6, 3G heap, ParNew, ParOld GC
- ▶ ~70,000 Issues, ~80 projects
- ▶ ~30 reqs/sec, JMeter 2.3.4 (patched, JSR-223)
- ▶ Traffic modelled on <http://jira.atlassian.com> with higher writes

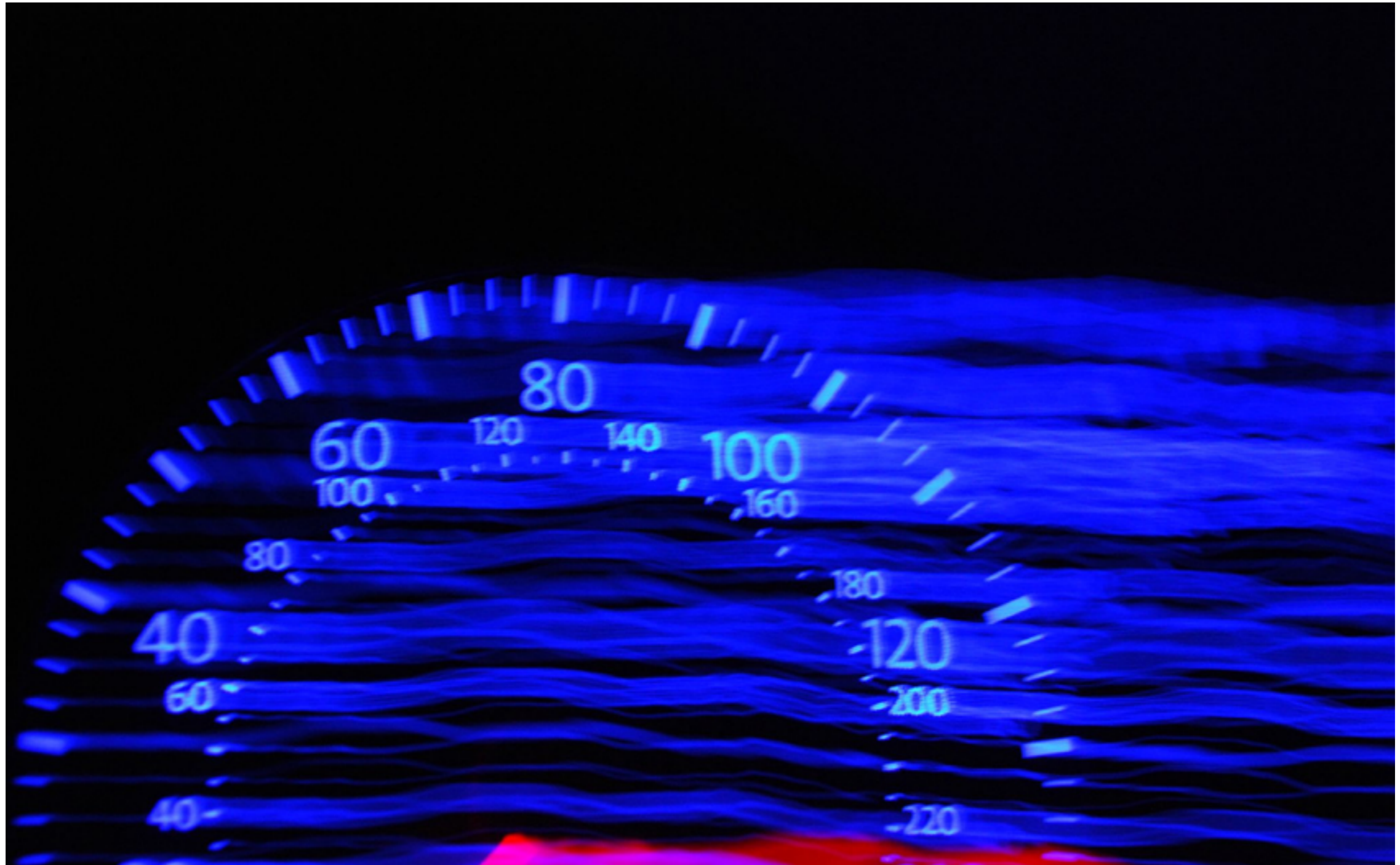
About the graphs

- ▶ Most show **Average Response Time in Milliseconds**
 - ▶ Includes time to generate AND deliver the response
 - ▶ Does not include any browser time
- ▶ **Arrows show if lower or higher is better!**

You are here

- ▶ How I did the testing
 - ▶ Hardware & Software
 - ▶ **Must have improvements** 
 - ▶ Java, Hardware, Virtualization and Garbage Collection
- ▶ Worth doing
 - ▶ Gigabit vs 100mbit, Heap size tuning, use an SSD
- ▶ Sadly, not useful
 - ▶ Tune MySQL, Replace MySQL

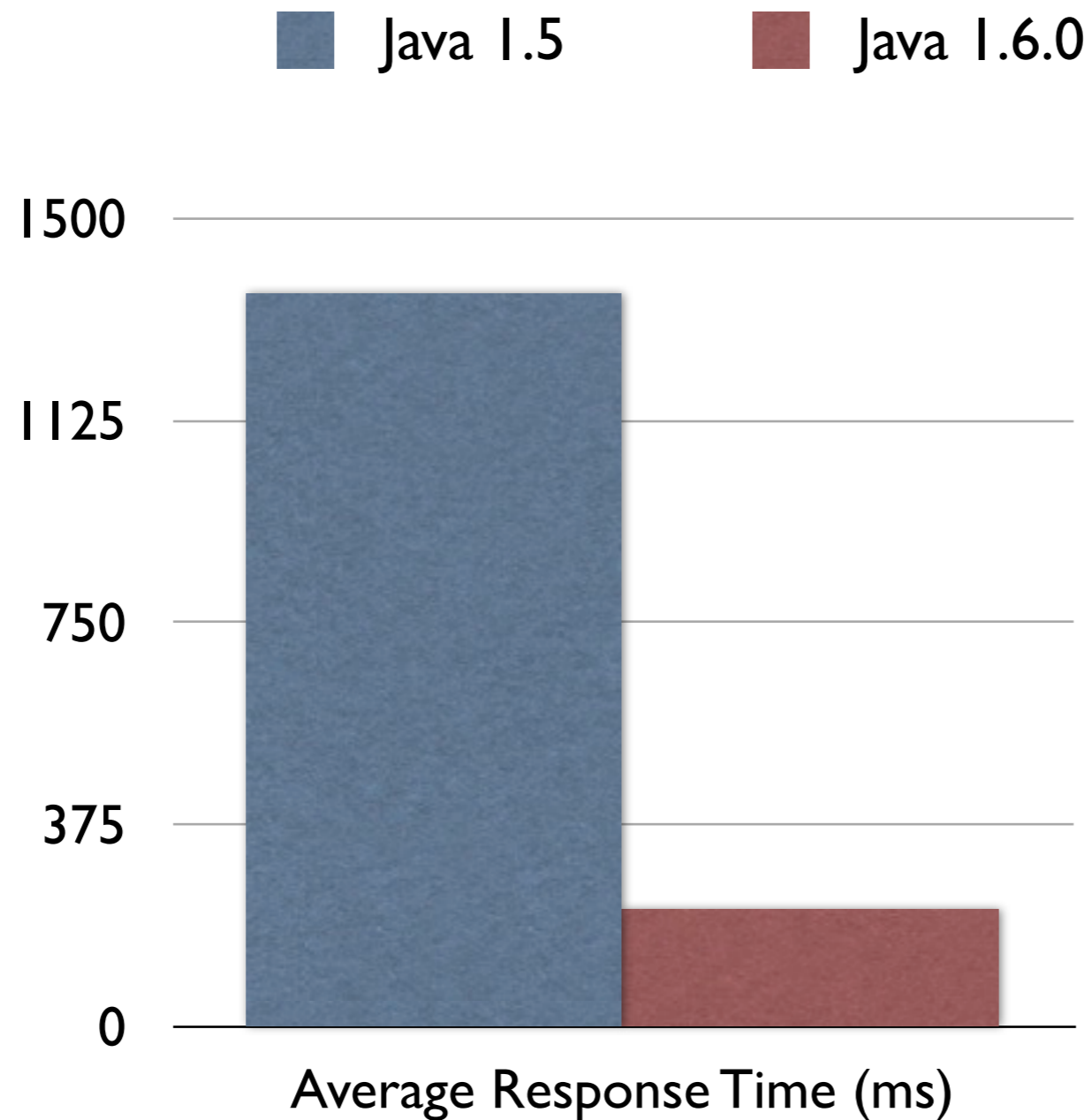
Must. Have.



Upgrade Java 1.5 to 1.6

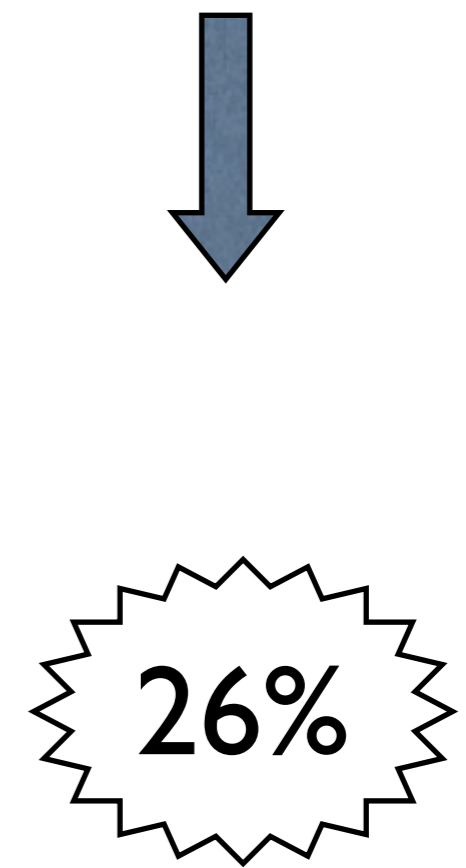
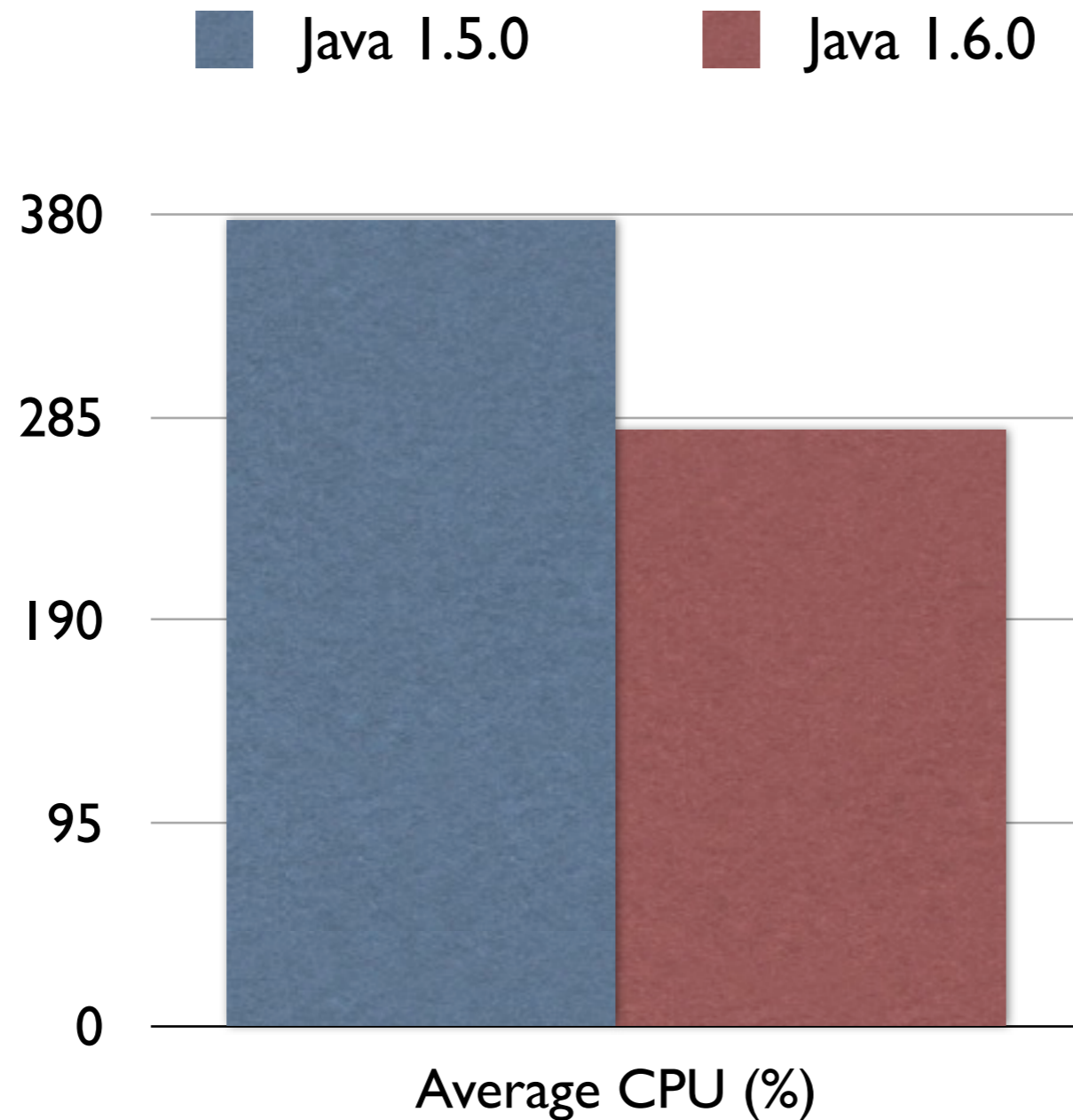
- ▶ Advances in compiler / VM technology
 - ▶ Produces better runtime code
 - ▶ Able to use modern instructions
 - ▶ Able to use modern hardware

Java 1.5 vs 1.6 (Johnny 5)



6.3X

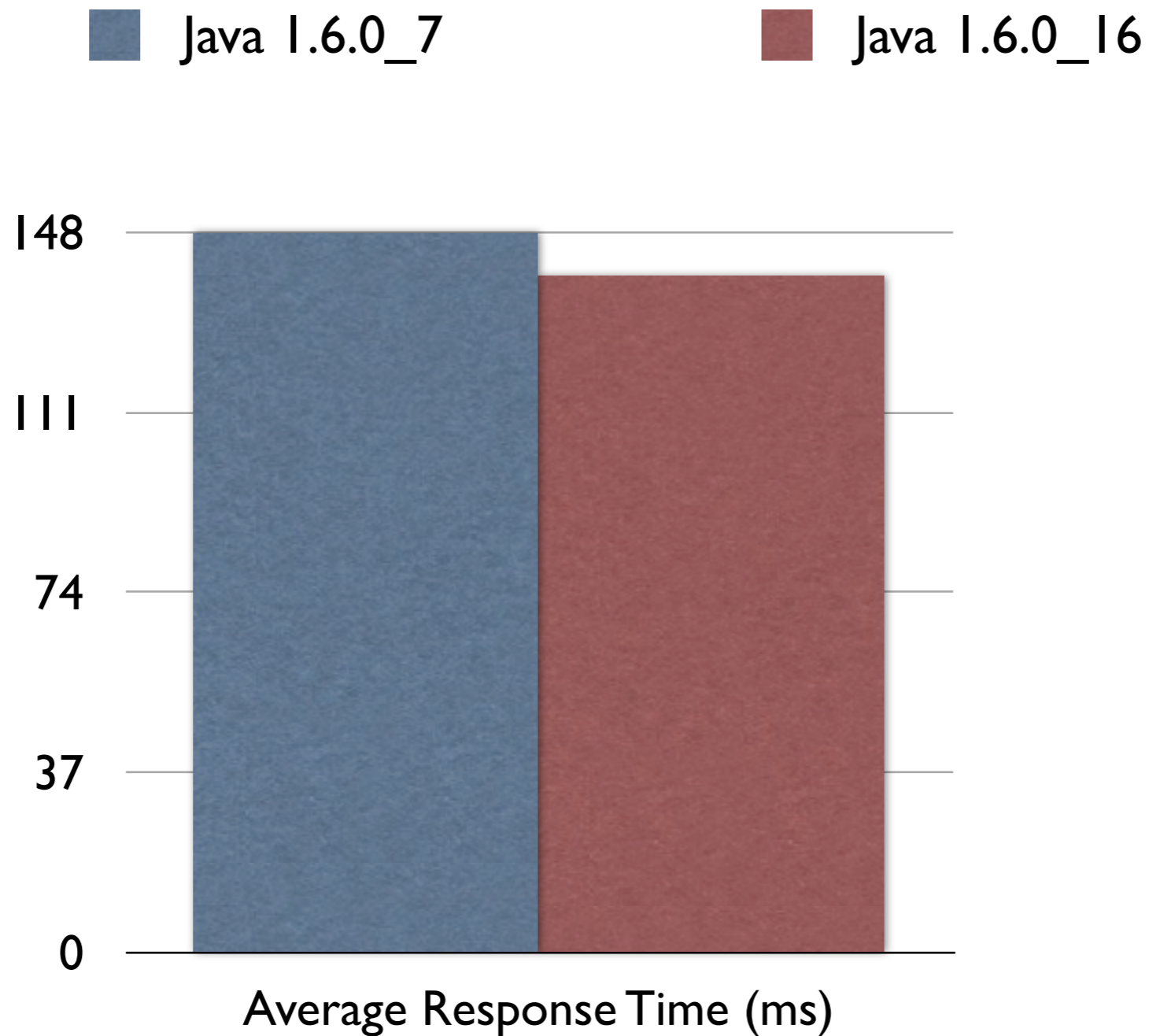
Java 1.5 vs 1.6 (Johnny 5)



Upgrade Java to 1.6

- ▶ HotSpot VM updates in 1.6
- ▶ Advancing technology
 - ▶ Compressed OOPS on 64bit
 - ▶ Escape Analysis
 - ▶ NUMA
- ▶ http://java.sun.com/performance/reference/whitepapers/6_performance.html

Java 1.6 vs 1.6 (Eve)



7%

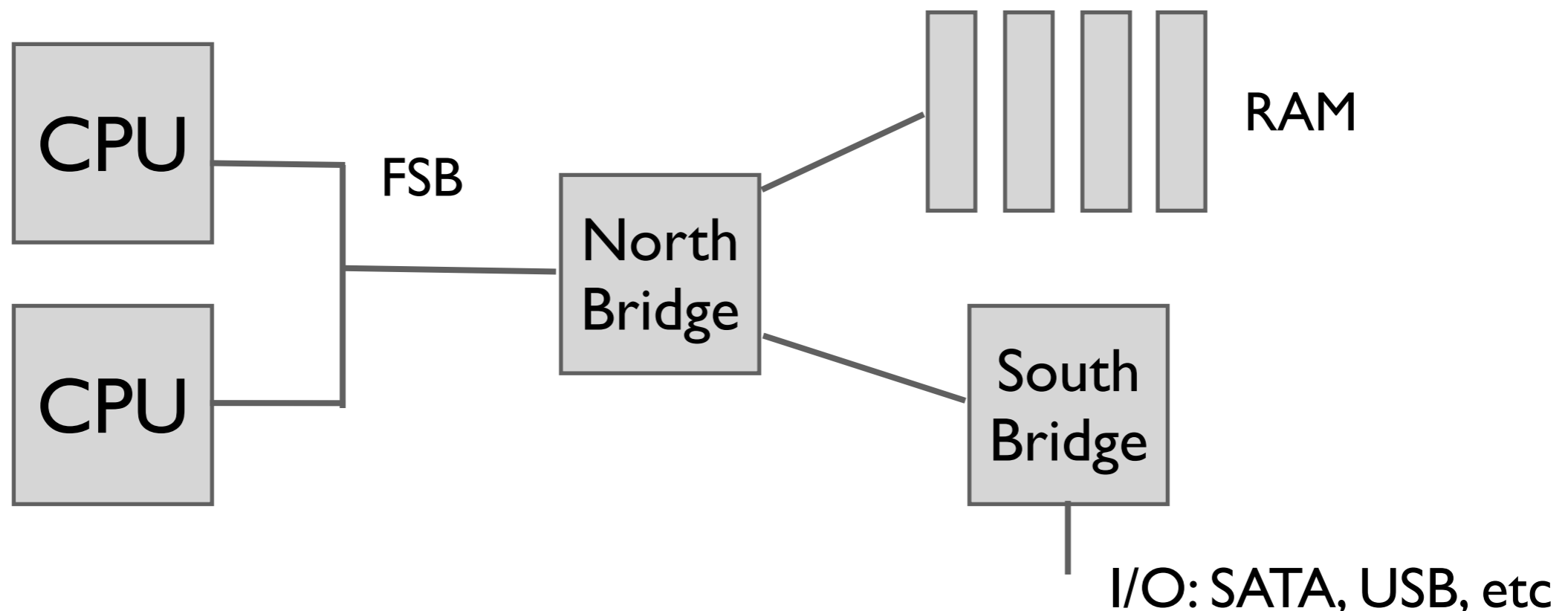
Hardware Upgrades

- ▶ Hyper-Threading
- ▶ QuickPath Interconnect
- ▶ Nehalem (Eve) have on die DDR3 Memory Controllers
- ▶ Memory throughput improvements – good for Java!

Technology	Speed
DDR2-800 (dual channel)	6.4 GB/sec
DDR3-1333 (dual channel)	21.3 GB/sec

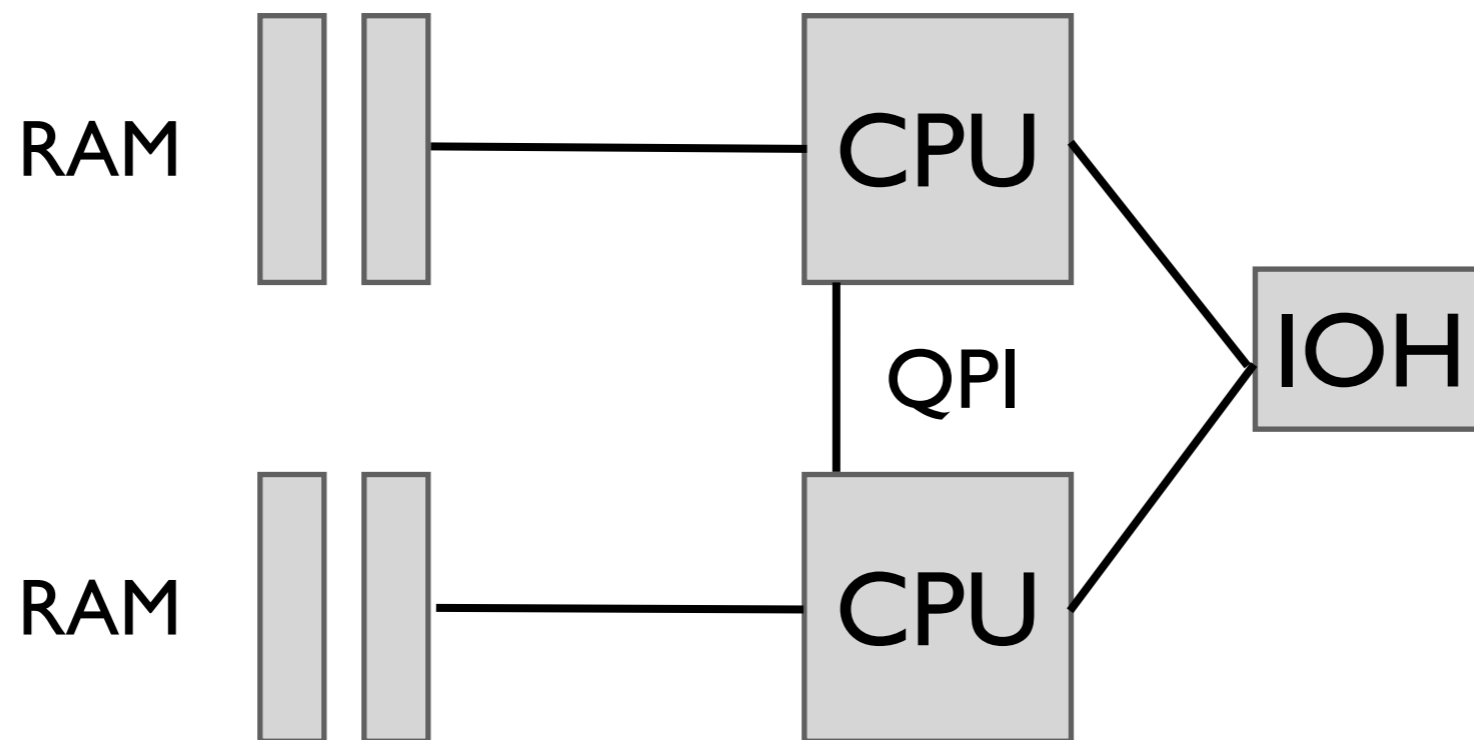
FSB architecture

- ▶ Shared bus
- ▶ $1333 \text{ MT/s} = 1333 \text{M Transfers / sec}$
- ▶ $4 \text{ transfers/clock} = 266 \text{Mhz.}$

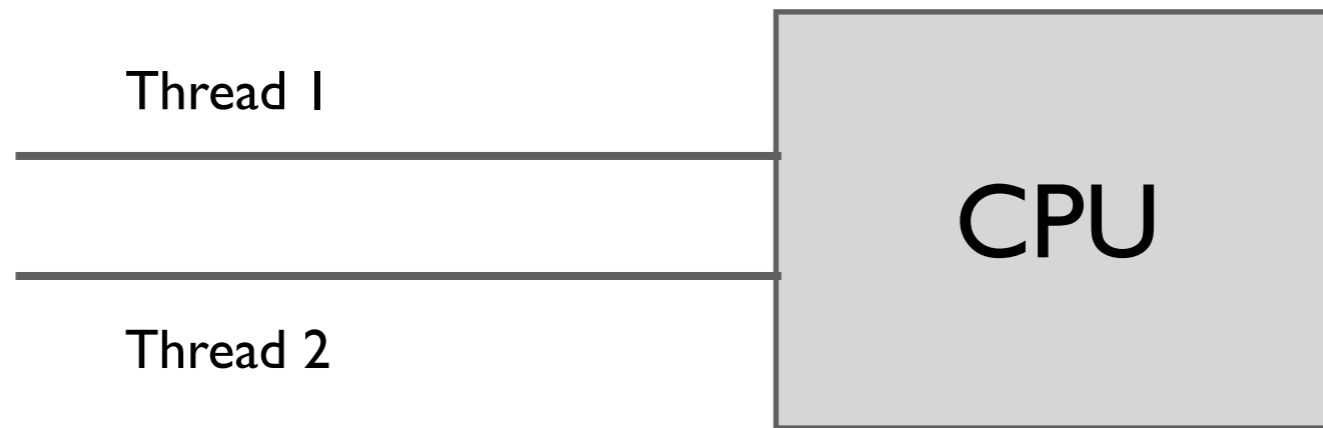


Quick Path Interconnect

- ▶ 3.2Ghz, 6.4 GT/sec
- ▶ 25.6 GB/sec



Hyper Threading

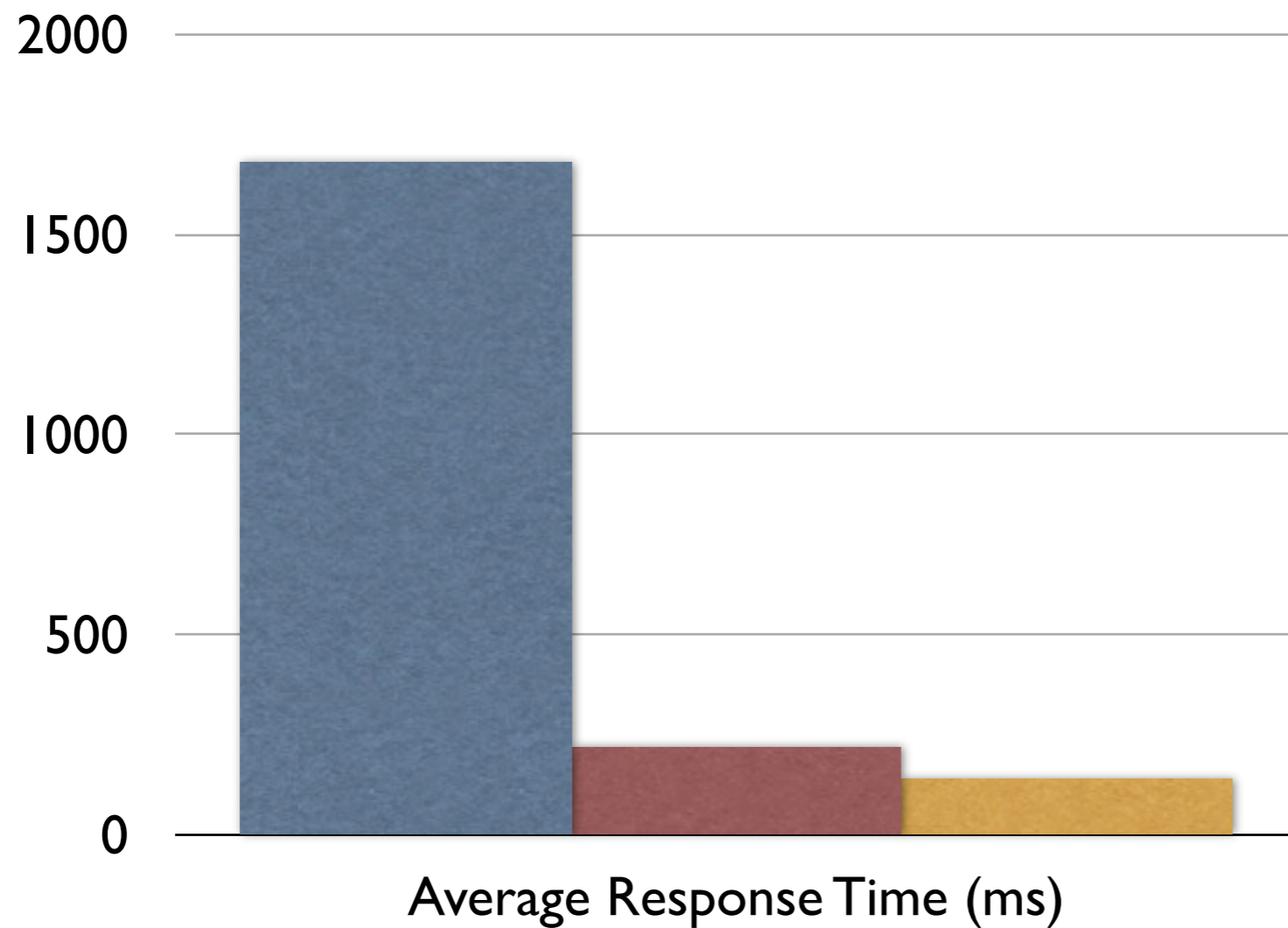


Thread 1 runs until it hits a cache miss.

Thread 2 runs on the same CPU while T1 is awaiting data

Hardware Upgrades

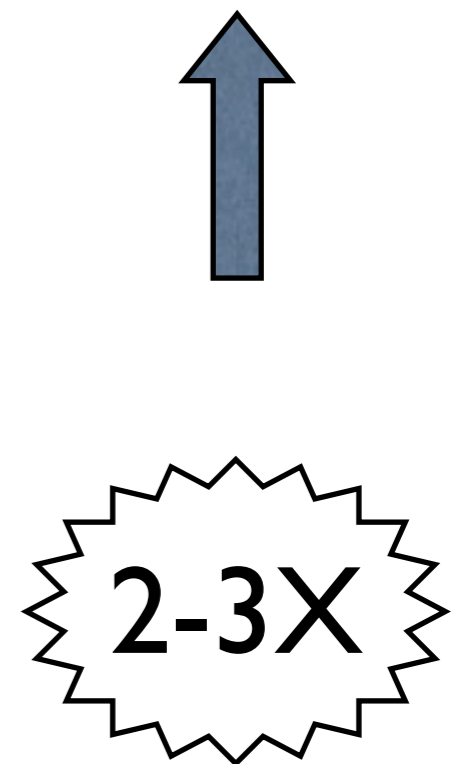
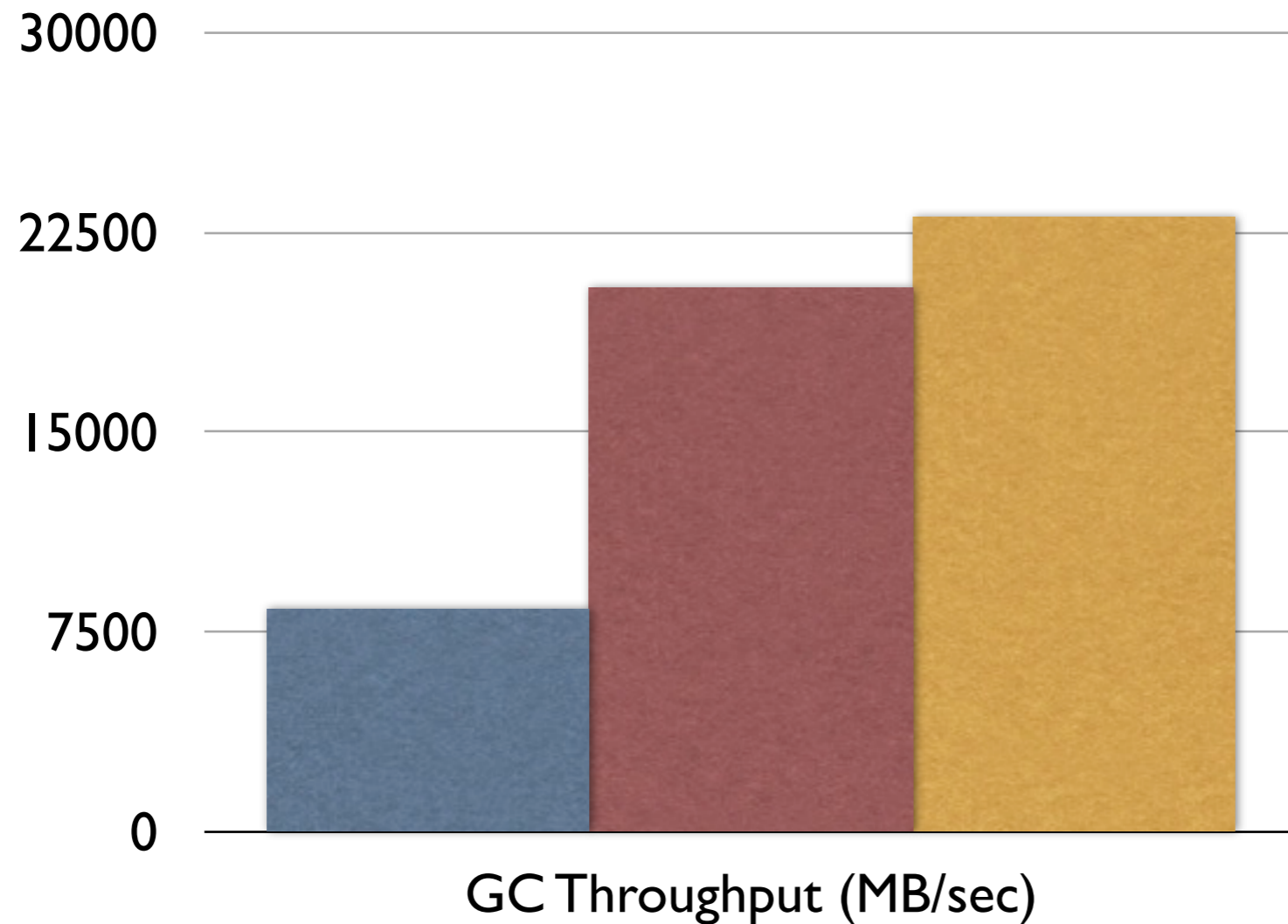
■ 2.4Ghz Xeon (Wall-E) ■ 2.0Ghz Xeon (Johnny 5) ■ 2.2Ghz Xeon (Eve)



8-12X

Improved GC Throughput

■ 2.4Ghz Xeon (Wall-E) ■ 2.0Ghz Xeon (Johnny 5) ■ 2.2Ghz Xeon (Eve)



Garbage Collection

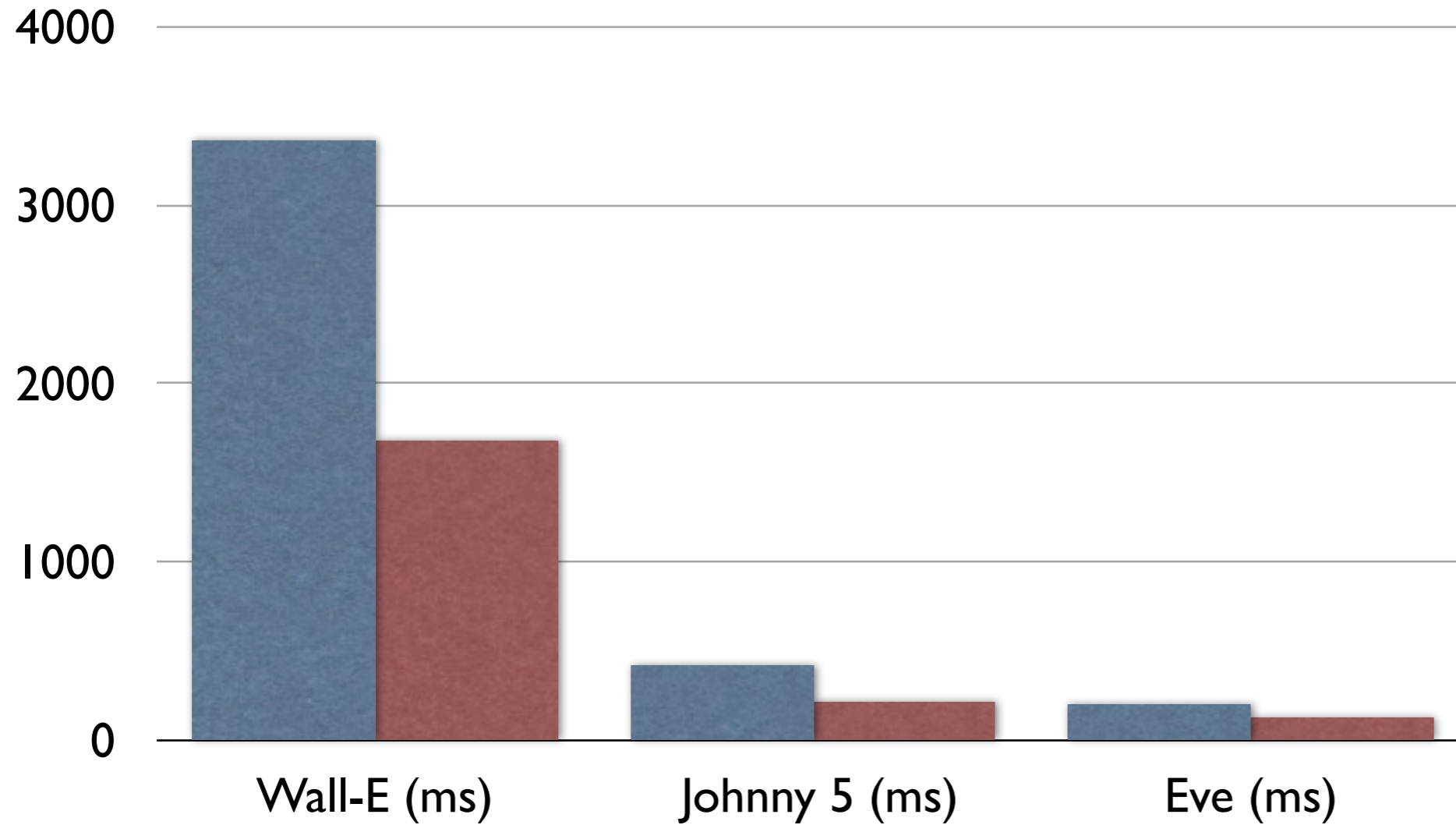
- ▶ Java's Virtual Machine manages memory
- ▶ New objects are continually created during runtime
- ▶ GC is the process of collecting dead objects to reclaim memory

Tune Garbage Collection

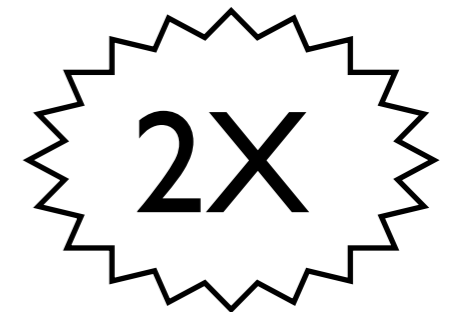
- ▶ Getting your GC tuning wrong is bad
- ▶ How bad?
- ▶ Are the defaults sane?

Tune Garbage Collection

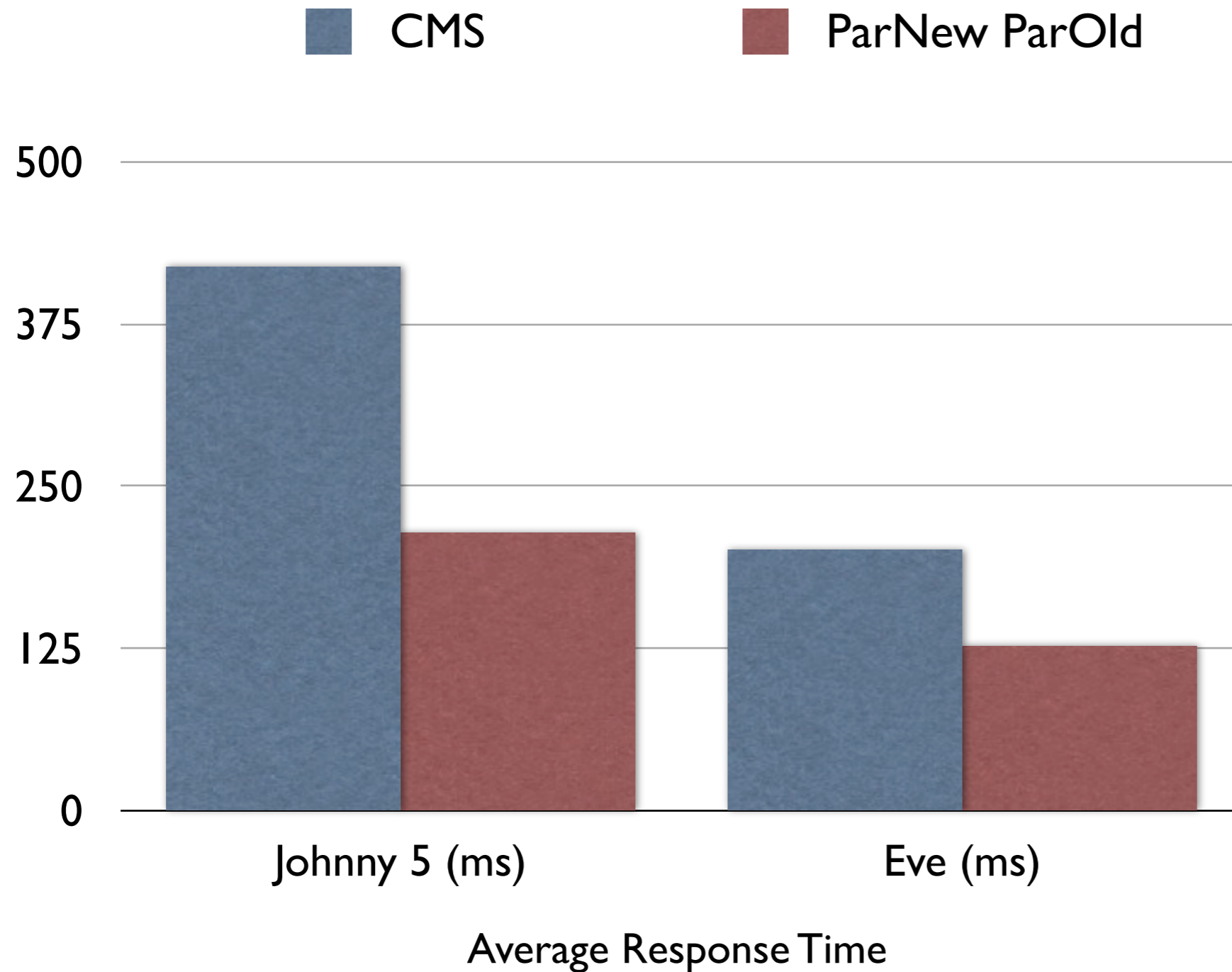
■ CMS ■ ParNew ParOld



Average Response Time



Tune Garbage Collection



2X

Tune Garbage Collection

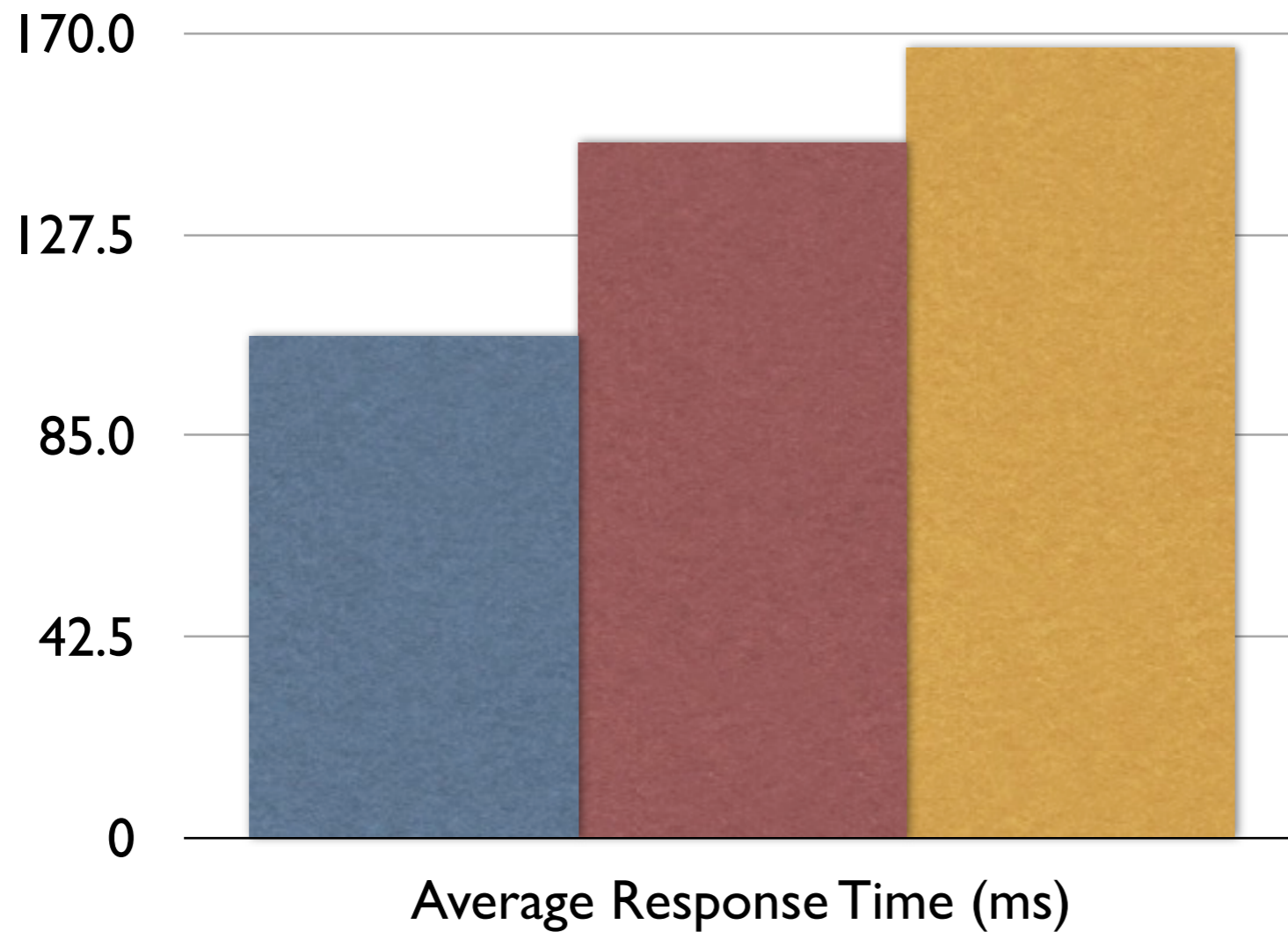
- ▶ Defaults are sane
- ▶ Parallel New + Parallel Old is fastest
- ▶ CMS prioritises low pauses over CPU usage
- ▶ CMS does not compact – can lead to heap fragmentation

Virtualisation

- ▶ Does being in a VM hurt?
- ▶ What if the risks are managed?

Virtualisation

■ Real ■ VMWare ESX 3.5 ■ VMWare ESX 4i




43%

Virtualisation

- ▶ On average ~30% slower
- ▶ Under extreme resource starvation, up to 100% slower
- ▶ Avoid virtualisation for high traffic instances
- ▶ See the Atlassian docs for recommendations
 - ▶ <http://confluence.atlassian.com/display/DOC/Running+Confluence+in+a+Virtualised+Environment>
 - ▶ <http://confluence.atlassian.com/display/JIRA/Running+JIRA+in+a+Virtualised+Environment>

You are here

- ▶ How I did the testing
 - ▶ Hardware & Software
- ▶ Must have improvements
 - ▶ Java, Hardware, Virtualization and Garbage Collection
- ▶ **Worth doing** 
 - ▶ Gigabit vs 100mbit, Heap size tuning, use an SSD
- ▶ Sadly, not useful
 - ▶ Tune MySQL, Replace MySQL

Maybe.

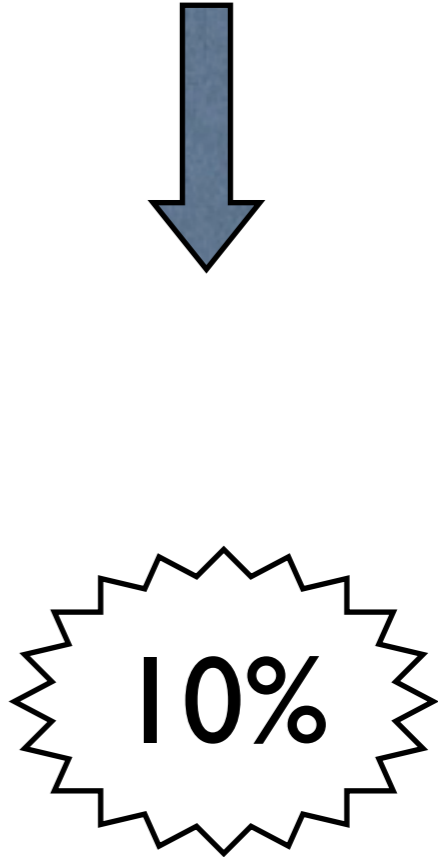
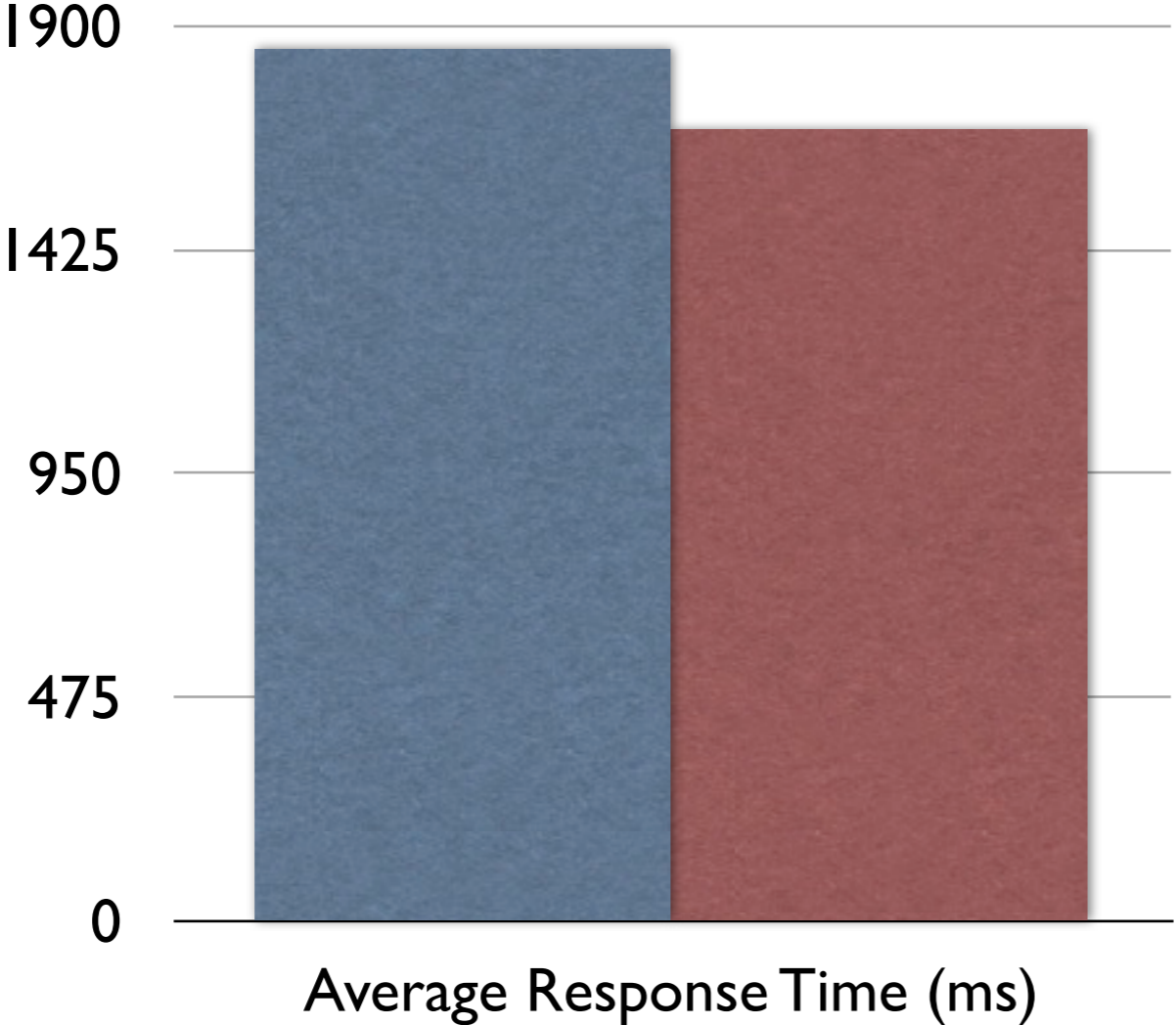


Gigabit Network vs 100Mbit

- ▶ Gigabit is faster, sure
- ▶ But 100Mb is OK if you're not doing big transfers, RIGHT?!

Use Gigabit

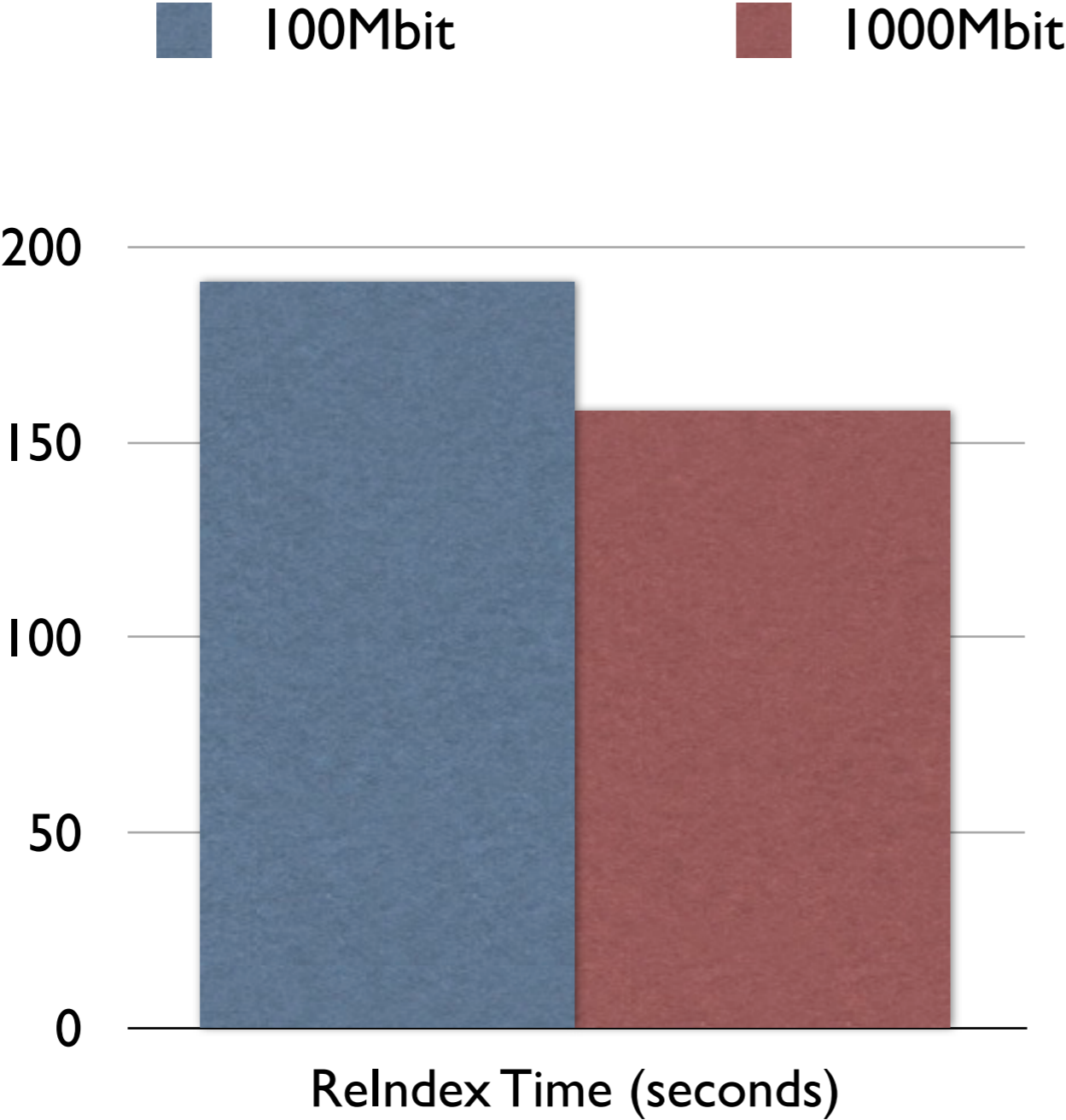
■ 100mb/sec ■ 1000mb/sec



Use Gigabit

- ▶ Lower RTT on Gigabit
- ▶ If you have enough CPU power, keep the DB on localhost
- ▶ Check your development environment

Use Gigabit

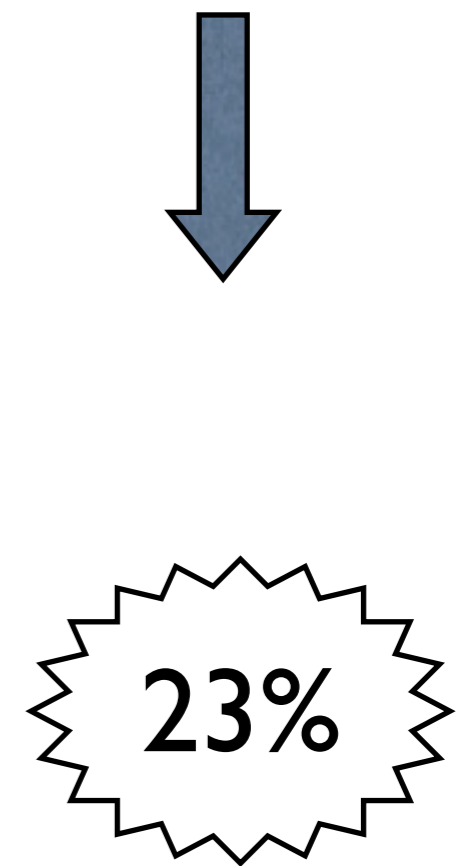
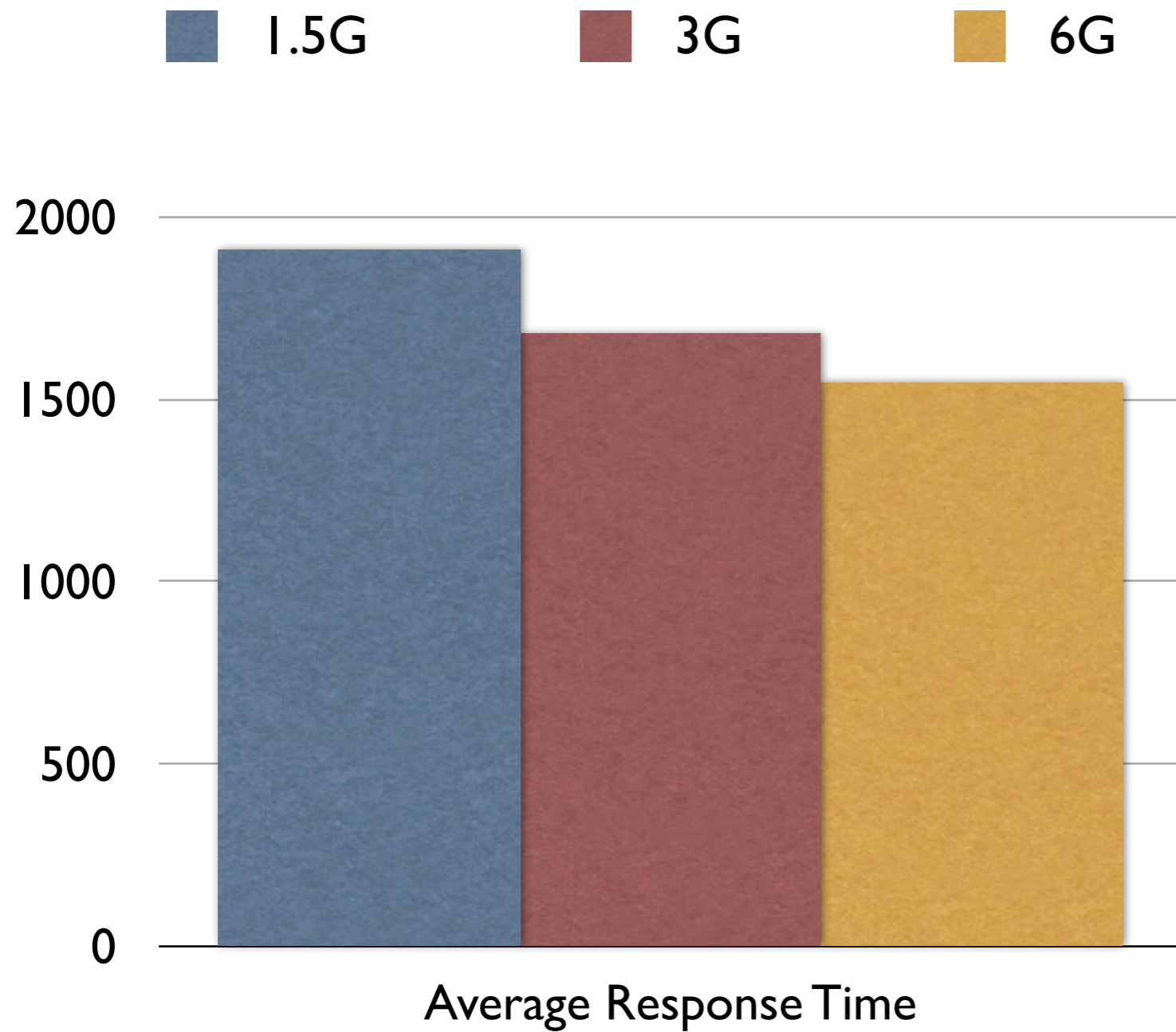


26%

Heap size tuning

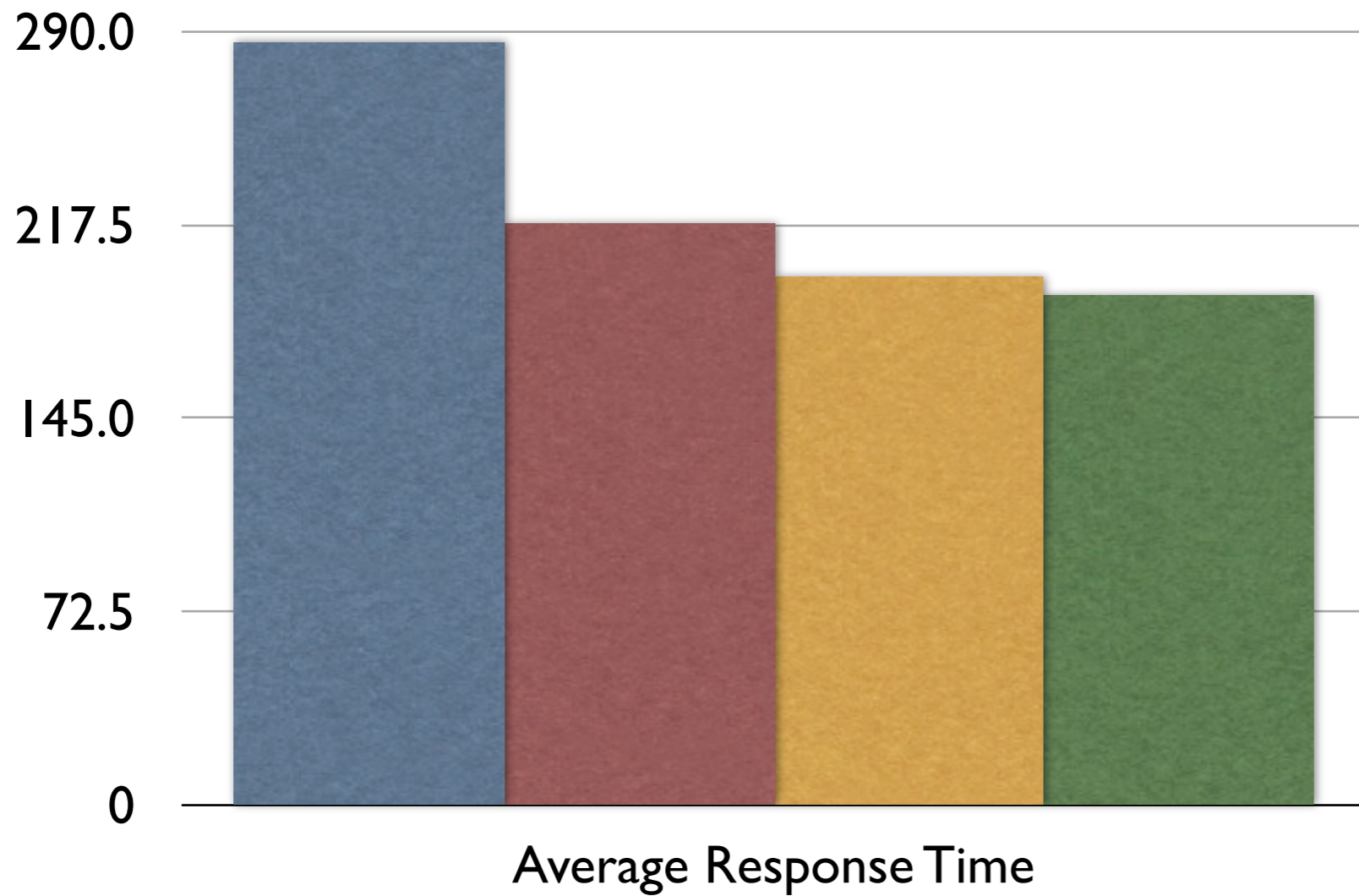
- ▶ Memory starvation is bad, but can we “drown” java with too much memory?

Heap size – Wall-E



Heap size – Johnny 5

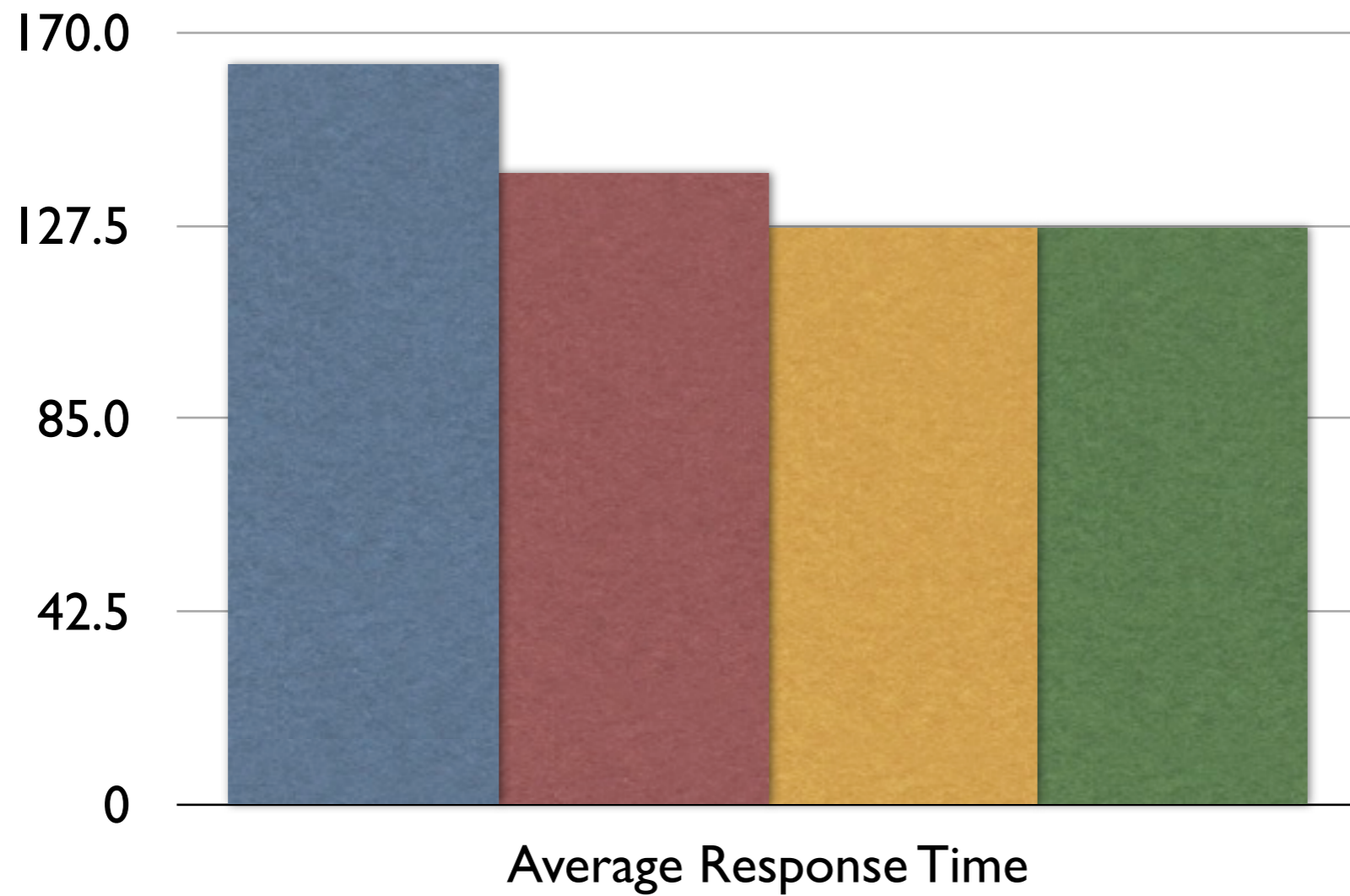
■ 1.5G ■ 3G ■ 6G ■ 15G



49%

Heap size – Eve

■ 1.5G ■ 3G ■ 6G ■ 15G



28%

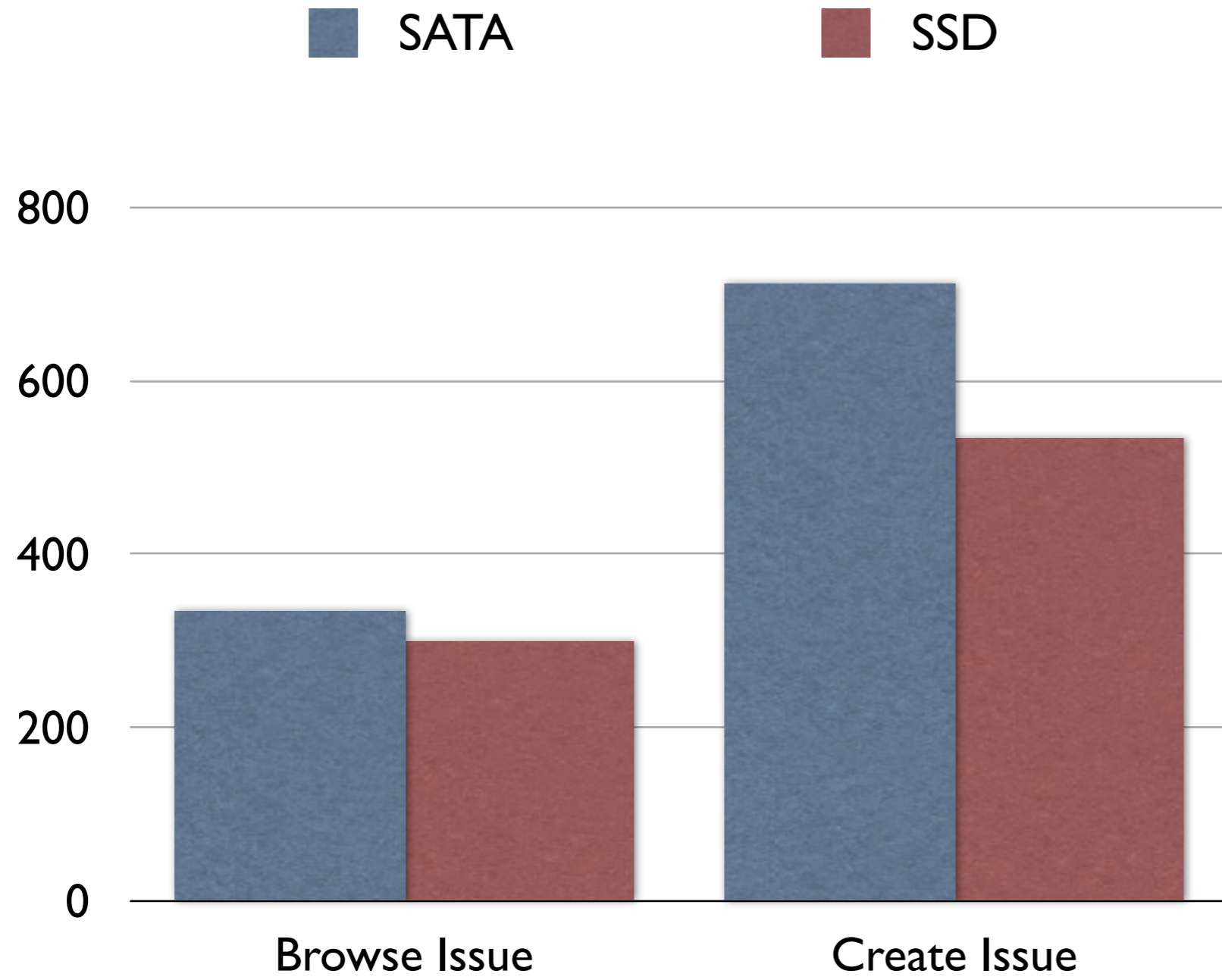
Heap size tuning

- ▶ Heap too small – bad
- ▶ Heap too big – not bad!
- ▶ Bigger heaps give the JVM more room to work
- ▶ Diminishing returns from really big heaps

Buy an SSD

- ▶ SSDs are FAST!
- ▶ SSDs are EXPENSIVE!
- ▶ SSDs are all the rage!
- ▶ DBAs are finally happy (almost)

Buy an SSD




33%

Buy an SSD

- ▶ Fast writes
 - ▶ 0.1ms Avg Service Time
- ▶ Most improvement comes from SSD on the DB
- ▶ Writing to the Lucene index means re-opening IndexReaders
 - ▶ Much faster on SSD
- ▶ Lucene Search Term Cache fits in OS buffers

You are here

- ▶ How I did the testing
 - ▶ Hardware & Software
- ▶ Must have improvements
 - ▶ Java, Hardware, Virtualization and Garbage Collection
- ▶ Worth doing
 - ▶ Gigabit vs 100mbit, Heap size tuning, use an SSD
- ▶ **Sadly, not useful** 
 - ▶ Tune MySQL, Replace MySQL

Meh.

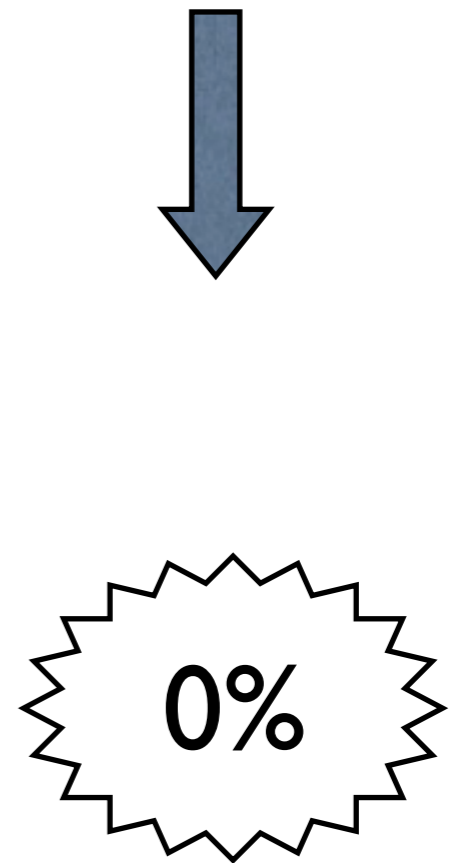
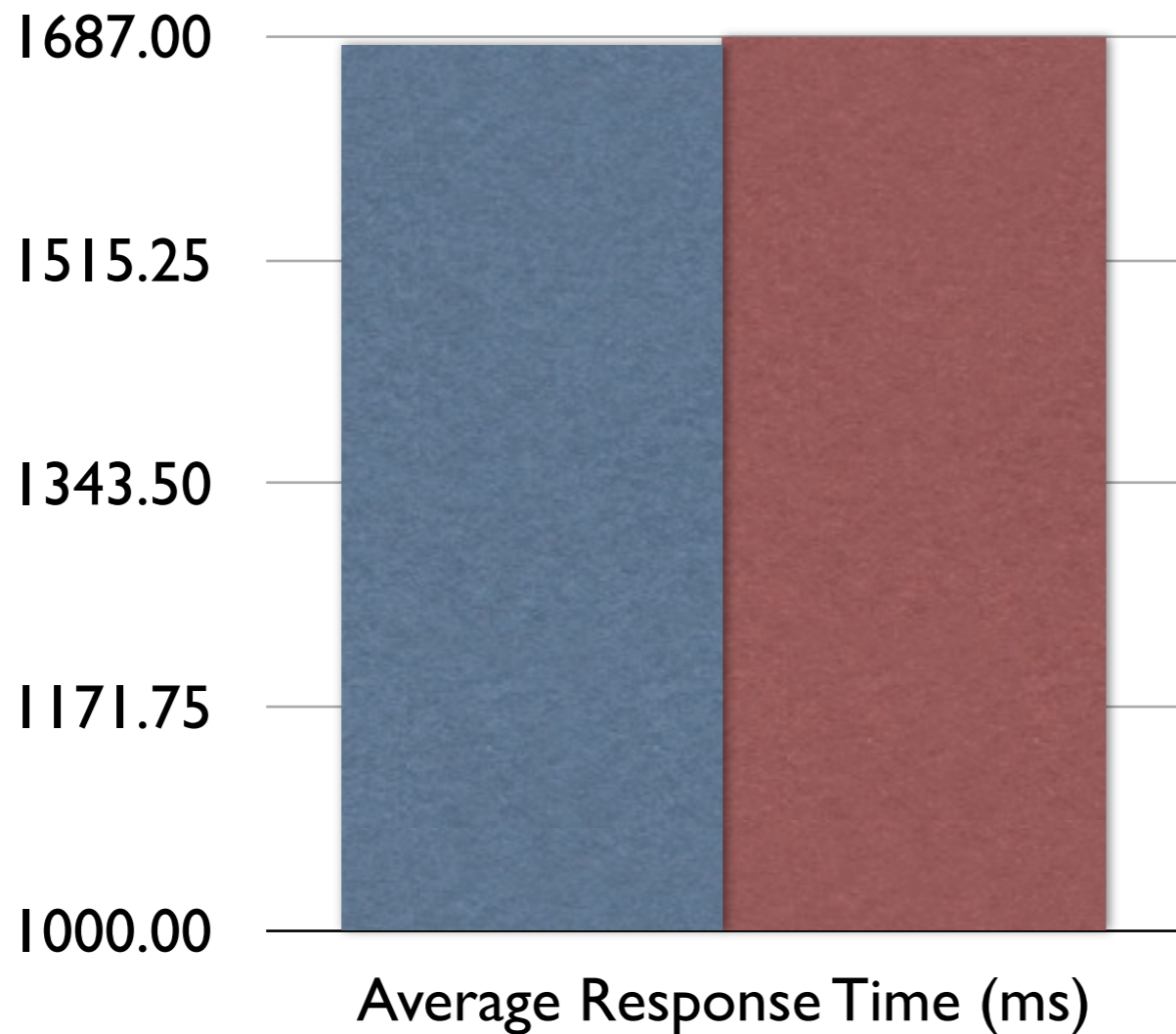


Replace MySQL with MySQL

- ▶ MySQL 5.0.45 – RHEL 5
- ▶ MySQL 5.0.84 – Percona HighPerf
- ▶ Patches to fix contention points
 - ▶ Improved I/O
 - ▶ Better scaling

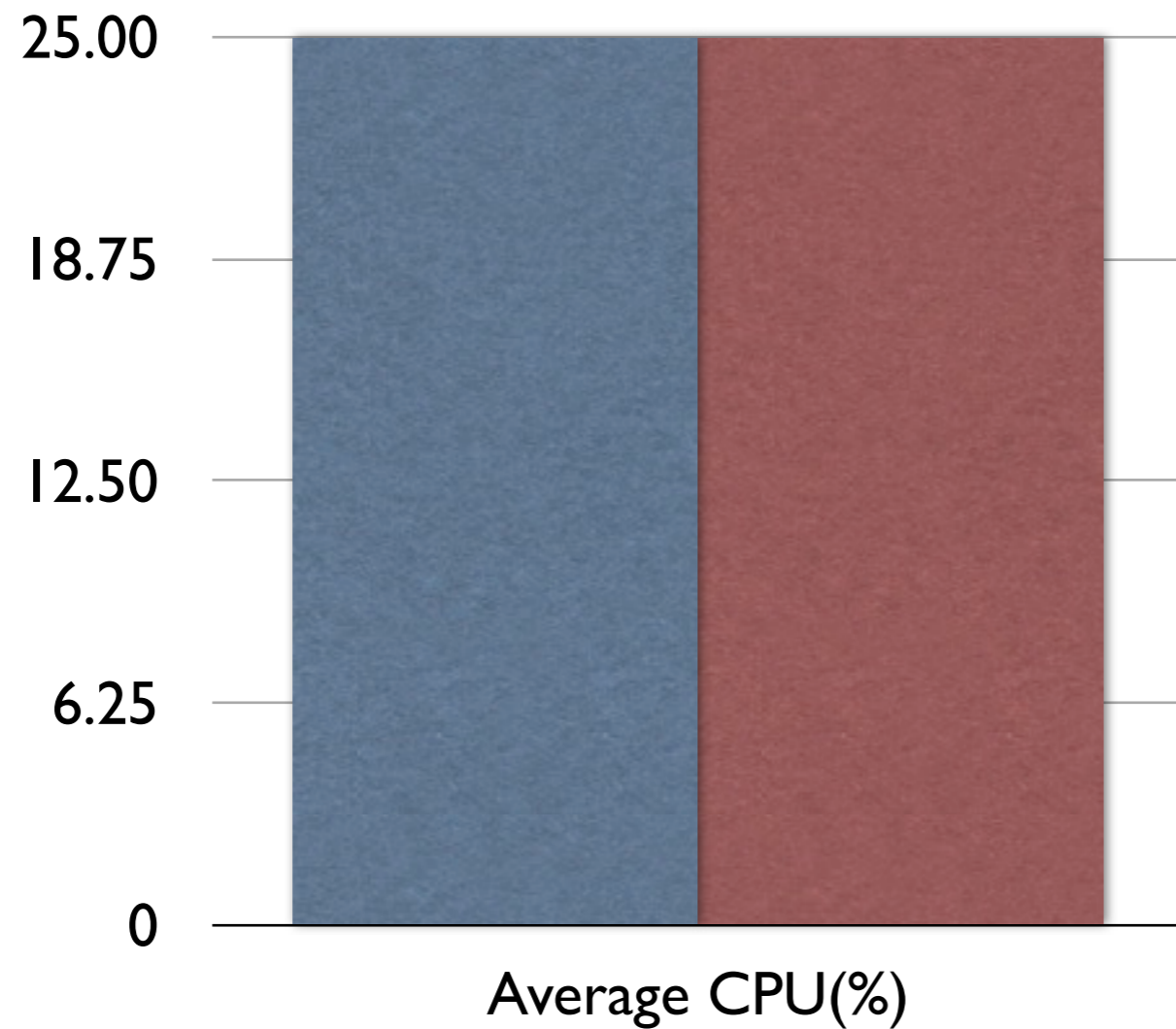
Replace MySQL with MySQL

■ MySQL 5.0.45 ■ Percona MySQL 5.0.85



Replace MySQL with MySQL

■ MySQL 5.0.45 ■ Percona MySQL 5.0.84



0%

Replace MySQL with MySQL

- ▶ JIRA doesn't do enough DB queries
- ▶ JIRA uses Lucene for many queries
- ▶ Percona Highperf MySQL really shines at high query rates
 - ▶ Try this if you share your DB server with other apps

Tune MySQL

- ▶ InnoDB-Heavy-4G.conf

sort_buffer_size = 8M

join_buffer_size = 8M

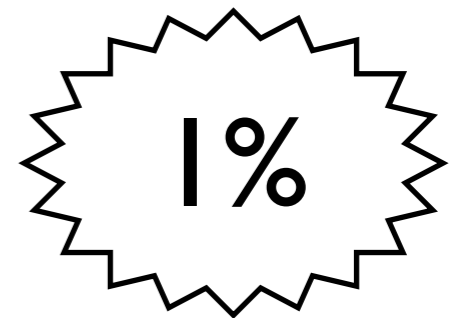
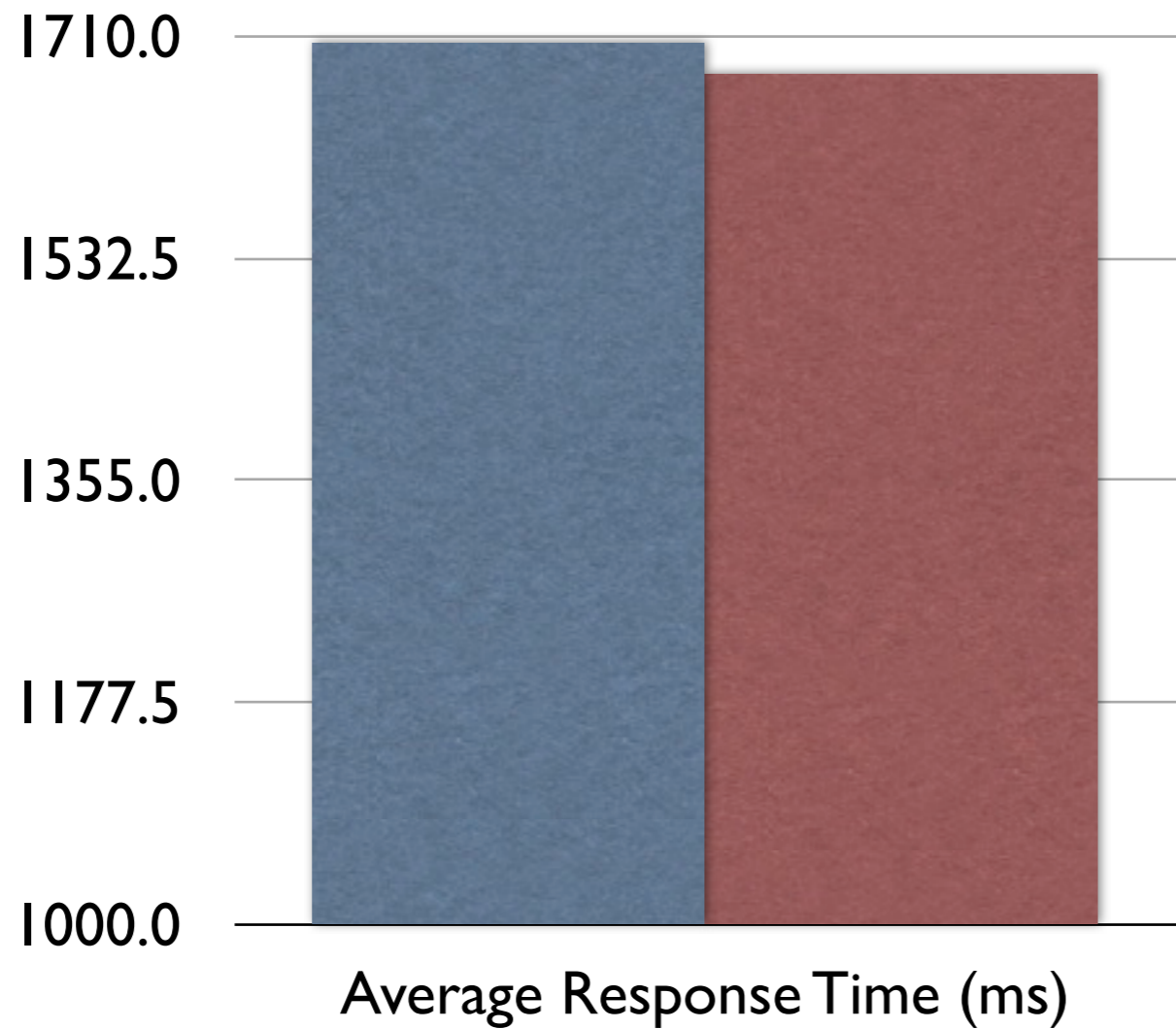
read_buffer_size = 2M

read_rnd_buffer_size = 16M

thread_concurrency = 16

Tune MySQL

■ Defaults ■ Tuned



Tune MySQL

MyISAM	ALL Engines
key_cache_age_threshold, key_cache_block_size, key_cache_division_limit, key_buffer_size read_buffer_size, read_rnd_buffer_size bulk_insert_buffer_size	join_buffer_size - Large joins without Indexes sort_buffer_size - Used by ORDER BY / GROUP BY query_cache_size query_cache_limit query_cache_min_res_unit

Tune MySQL

- ▶ Be sure to set `innodb_buffer_pool_size`
- ▶ Note! `innodb_log_file_size` will affect shutdown speed

Remember!

- ▶ Upgrade!
 - ▶ Java to 1.6
 - ▶ Your server
- ▶ Use Parallel Garbage Collection with a big heap
- ▶ Remove virtualisation if you can
- ▶ Use 1000mbit network
- ▶ Buy an SSD for write workloads

A final word

- ▶ **TEST EVERYTHING!**

Thanks! Questions?

- ▶ <http://blogs.atlassian.com/developer>

Flickr Attributions:

Siesta in Denmark - <http://www.flickr.com/photos/jakecaptive/27123533/>

Day 11 - <http://www.flickr.com/photos/ivanomak/4057632458/>

Blue Streak - <http://www.flickr.com/photos/jettajet/2061715292/>