A Real-Time Benchmark for Java™

Brian Doherty
Senior Software Engineer
Sun Microsystems, Inc.

Motivation

- Soft real-time Java applications abound
  - Financial trading
  - Telecommunications
- Existing benchmarks focus on throughput
- Existing benchmarks rely only on Java SE APIs
- Java RTS can run Java SE soft real-time apps
  - Small application modifications can leverage the better response time distributions delivered by Java RTS
- Lack of standardized benchmarks leaves product evaluators at the mercy of marketing proaganda.

Selecting a benchmark

- Applicability
  - Response time must be a primary performance factor
  - Benchmark must reveal the throughput trade-off
  - Hard real time not deemed necessary for first benchmark
- Time to Market
  - Response time applications already deployed
  - Response time characteristics of Java implementations already being marketed
  - Opportunity to improve all Java implementations
- Familiarity
  - Allow both users and Java implementors to understand the benchmark and results quickly

SPECjbb®2005

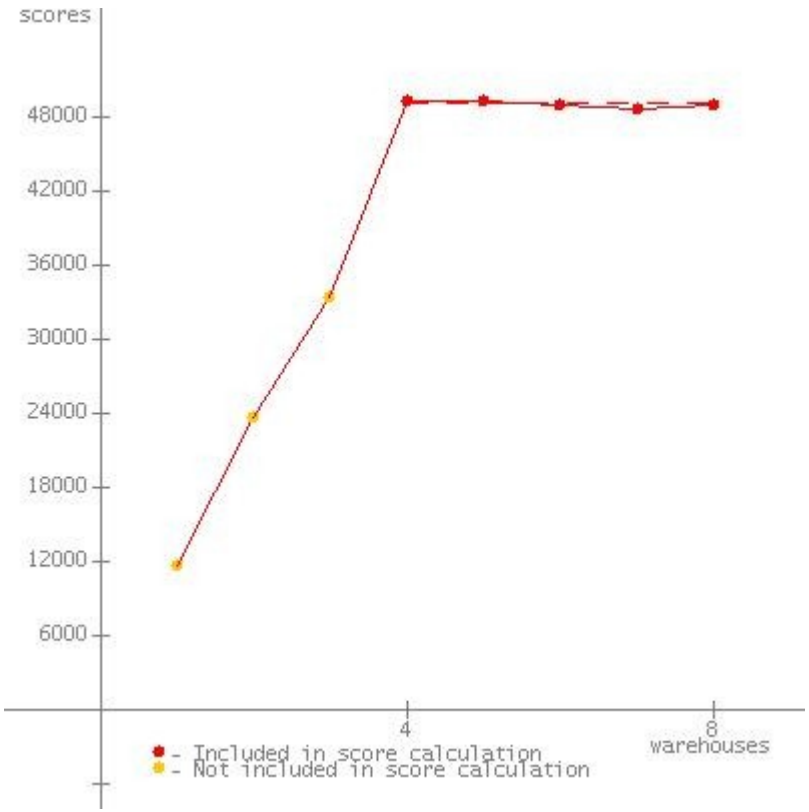
- Applicability
  - Measures throughput, but also tracks response times
  - Responsible for many Java performance improvements
- Time to Market
  - Existing source base
  - Well defined SPEC development and review processes
  - Minimal modifications to utilize response times
  - Adaptable to Java RTS RuntimeThread threads
- Familiarity
  - Well known in the general purpose industry
  - Well known by Java implementors

SPECjbb2005 Overview

- Java Business Benchmark
- Simulates a 3-tier business system
  - Presentation/Business Logic/Database
- Primary entity is the Warehouse
  - One thread per warehouse
- Two Phases
  - Warmup – one warehouse per CPU
  - Measurement – two warehouses per CPU
- Result - JBB Ops/sec (JBOPS)
  - Average throughput from N to 2N warehouses

SPECjbb2005 Example Output

Warehouses	SPECjbb2005 Bops	Incl. In Metric
1	11716	
2	23688	
3	33393	
4	49384	*
5	49376	*
6	48998	*
7	48734	*
8	48984	*
SPECjbb2005	(from 4 to 8)	49097 SPECjbb2005 bops



Sun Fire X4200 (2 chips, 4 cores, 4 threads, 2.6 GHz) 49097 SPECjbb2005 bops, 49097 SPECjbb2005 bops/JVM. SPEC and the benchmark name SPECjbb2005 are trademarks of the Standard Performance Evaluation Corporation. Results as of 12/05 on www.spec.org. For the latest SPECjbb2005 benchmark results, visit http://www.spec.org/osg/jbb2005

SPECjbb®2005rt[1]

- Based on SPECjbb2005
  - A well known Java server benchmark from SPEC
  - Active results submissions
  - Well understood by JVM implementors
  - Short time to market
- Enhanced to measure response times
  - Primary results metric coupled with throughput
  - Response time distributions
- Restructured to not overload the system
  - Fewer concurrent threads

1. SPECjbb2005rt is not an official SPEC® benchmark. It's simply a name for this benchmark that reflects both the lineage of the benchmark, SPECjbb2005, and how it differs from that lineage. Should this modified benchmark be accepted by the SPEC organization, it may not be given this name.

SPECjbb®2005rt Overview

- Reuses overall logic of SPECjbb2005
- Adds Injector threads
  - Asynchronously injects transactions at a prescribed rate
  - Injection rate automatically calibrated during warm-up or specified in the configuration file
- Adds a shared transaction queue between Injector threads and Warehouse threads
  - Response time metric computed from expected enqueue time to transaction completion time
  - Expected enqueue time != actual enqueue time
    - Injector thread scheduling delays will not reduce transaction rate and scheduling delays are not ignored

SPECjbb2005rt Overview (continued)

- Adds Java RTS RealtimeThread support
  - Injector and Warehouse threads
  - Priority levels
  - Configured declaratively
- Restructured to step up load level incrementally rather than adding threads incrementally
  - Uses a fixed number of warehouses, typically fewer than the number of CPUs.
- Adds per-transaction type response time histograms

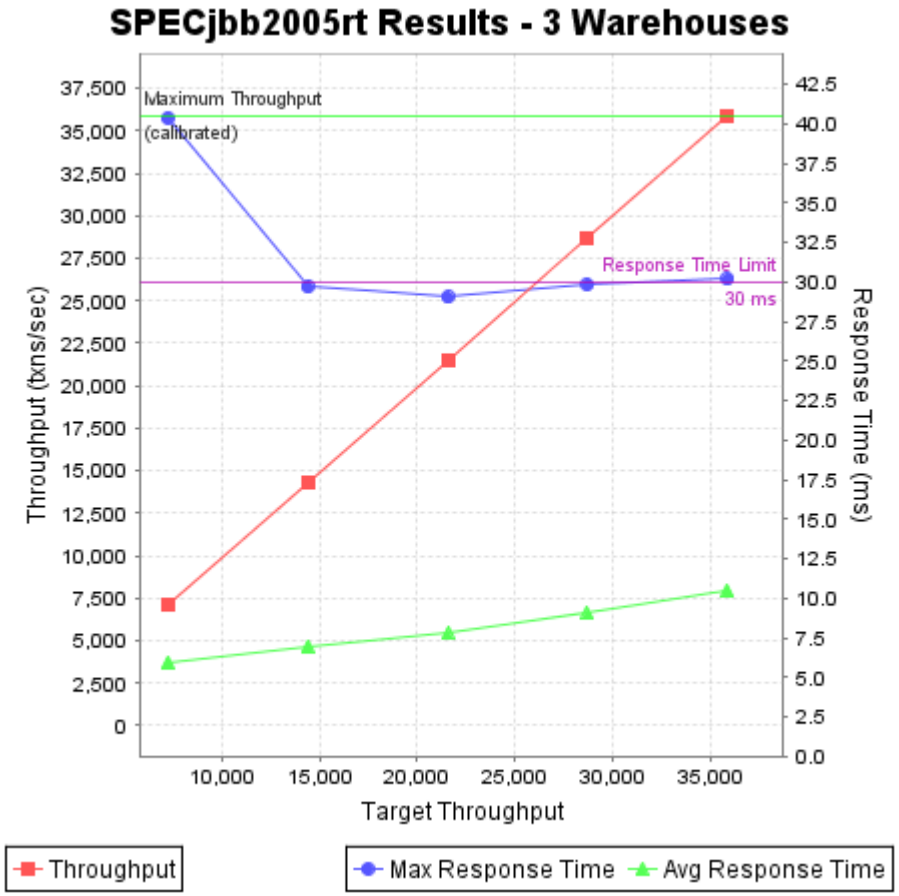
SPECjbb2005rt Overview (continued)

- Three Phases
  - Warmup
    - Single warehouse, synchronous injection
  - Calibration
    - Find the theoretical maximum throughput from one warehouse
    - Synchronous injection
    - Maximum computed as average over 3 measurements.
    - Can be skipped by specifying a fixed maximum.
  - Measurement
    - up to 1 warehouse per CPU, typically less
    - Throughput stepped up from 20% to 100% of maximum at 20% increments

SPECjbb2005rt Overview (continued)

- Result - response time @ JrtBB Ops/sec (JrtBOPS)
  - Maximum response time must be met for valid result
  - Can be used to compare throughput levels at a specific maximum response time of interest
  - Can be used to compare response time distributions at a specific throughput of interest
  - Can be used to compare real-time qualities of different deployment platforms
    - Hardware platforms
    - Operating Systems
    - Java implementations
  - Queuing statistics recorded, but not used in result metric

SPECjbb2005rt Example Output

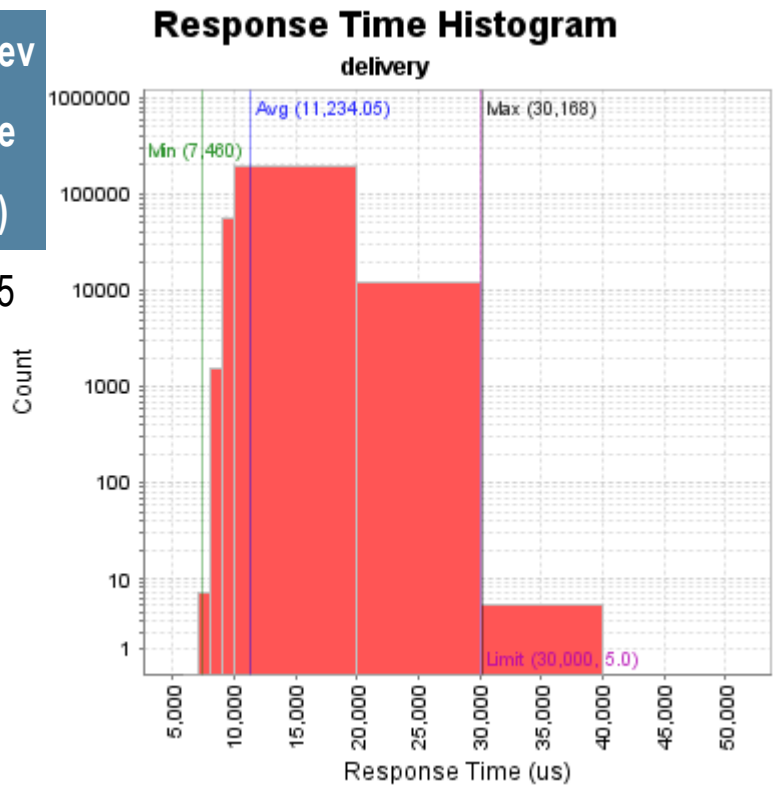


SPECjbb2005rt Example Output

Ware-Houses	Target Bops	Actual Bops	Avg Response (ms)	Max Response (ms)
3	7182	7182	5.98	10.50
3	14364	14363	6.99	29.70
3	21546	21545	7.75	29.00
3	28728	28727	9.04	29.80
3	35910	35909	10.50	30.20

SPECjbb2005rt Example Output

Transaction Type	Txn Count	Txn Exceeds 30000 (us)	Total Time (sec)	Min Time (us)	Max Time (us)	Avg Time (us)	StdDev Time (us)
Delivery	261155	5	2934	7460	30168	11234	3225



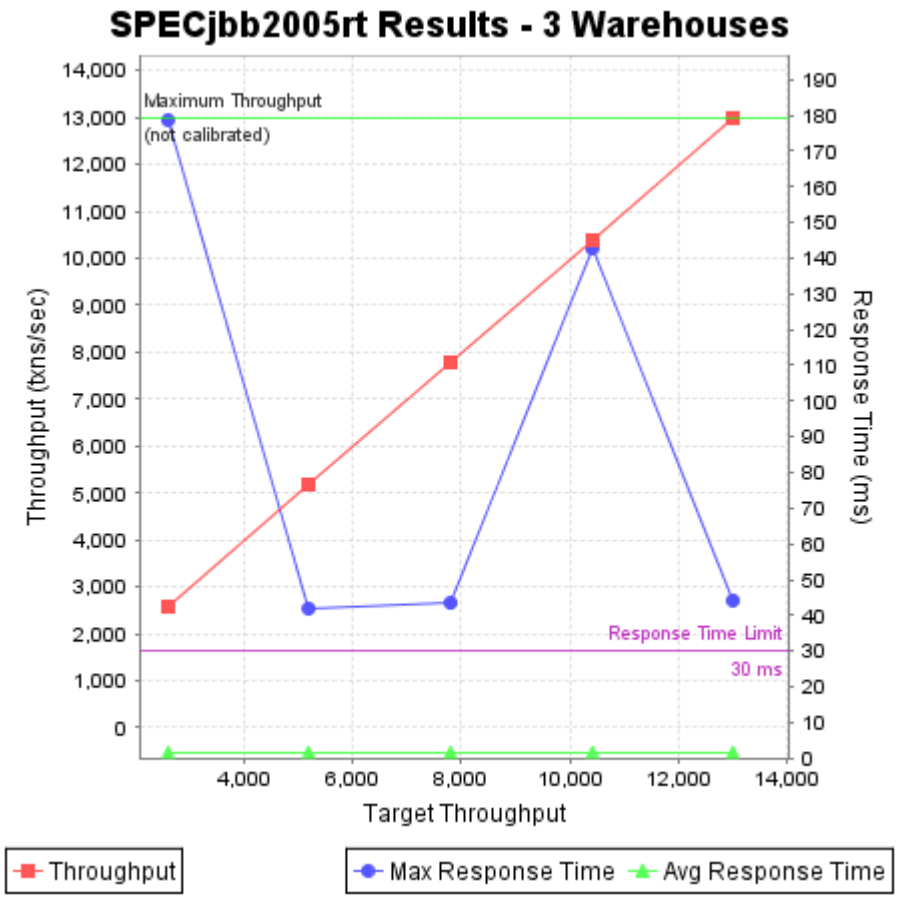
Example Results

- Three Java implementations
  - Different vendors
  - Different garbage collection policies
  - Different real-time characteristics
- Vendor names and command line options not revealed due to SPEC fair use restrictions
  - We're not out to point out competitive differences here, just to show the value and limitations of the benchmark
- Sun Fire X4200, 2x2.9Ghz AMD 220 SE (2 sockets, 4 cores) 8GB RAM, Dual Boot (Solaris 10 6/10, Windows XP 64-bit)

Brand X

- Java SE based runtime
- Throughput oriented garbage collector
- Fixed injection rate (13000 txns/sec)
- 60 minute measurement interval
- 30 ms maximum pause time
- Windows XP 64-bit (SP2)

Brand X Results Graph



Brand X Results Table

Ware-Houses	Target Bops	Actual Bops	Avg Response (ms)	Max Response (ms)
3	2600	2599	1.27	179.00
3	5200	5196	1.32	41.90
3	7800	7791	1.38	43.40
3	10400	10358	1.46	143.00
3	13000	12978	1.53	44.20

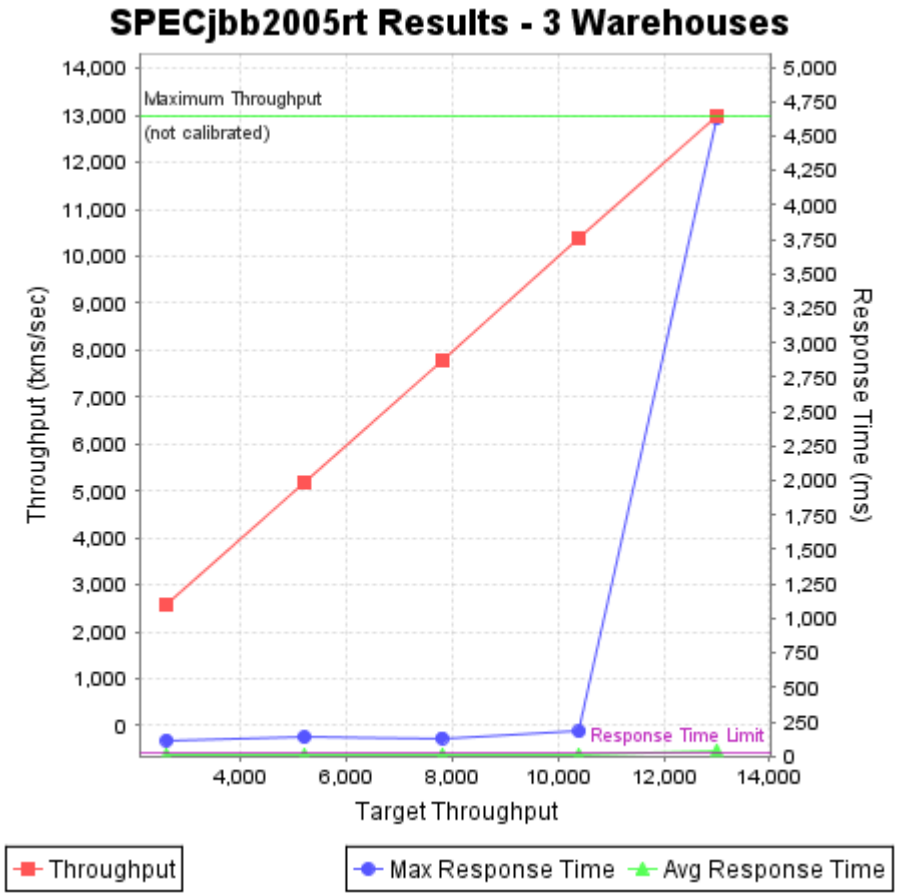
Brand X Results Summary

- Average response times are good
  - Generational garbage collection delivers reasonable results as long as the old generation doesn't fill up
- Maximum response times exceed the 30ms threshold at all throughput levels
- Some soft real-time applications might be satisfied with this level of performance, particularly if the old generation collections can be avoided.

Brand Y

- Java SE based runtime
- Pause time oriented garbage collector
- Fixed injection rate (13000 txns/sec)
- 60 minute measurement interval
- 30 ms maximum pause time
- Windows XP 64-bit (SP2)

Brand Y Results Graph



Brand Y Results Table

Ware-Houses	Target Bops	Actual Bops	Avg Response (ms)	Max Response (ms)
3	2600	2599	8.63	116.00
3	5200	5199	9.27	140.00
3	7800	7798	9.92	134.00
3	10400	10398	10.70	179.00
3	13000	12997	38.70	4631.00

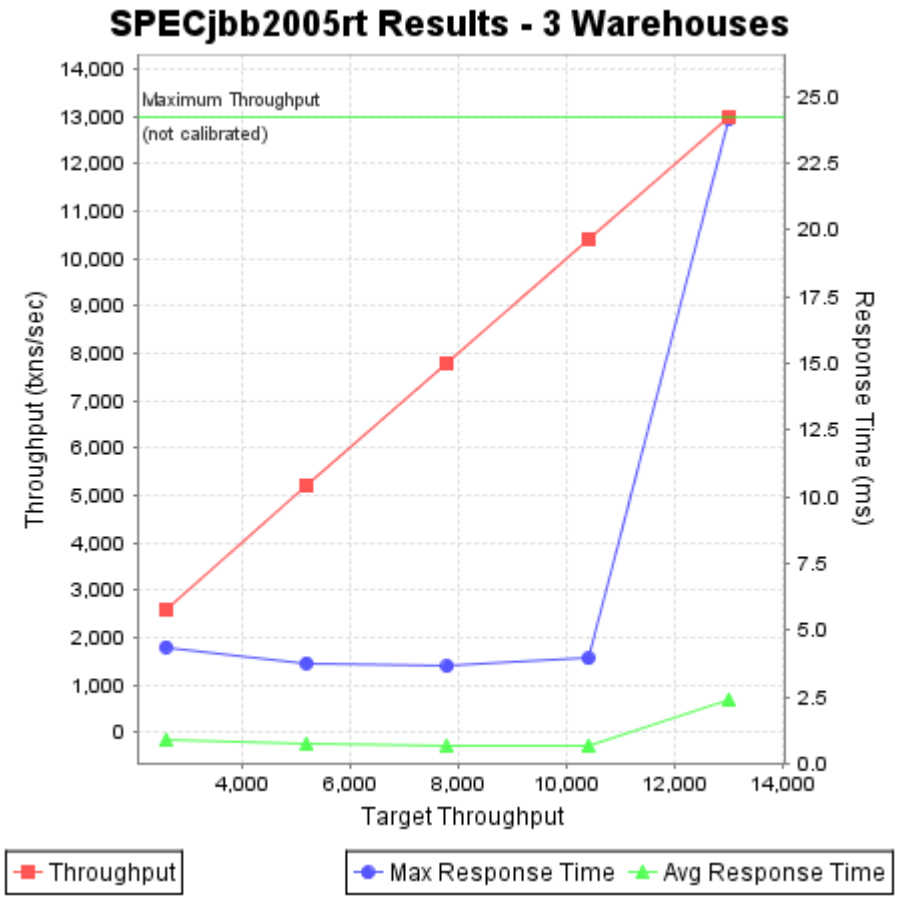
Brand Y Results Summary

- Average response times are good
  - Concurrent garbage collection keeps up well until the 100% throughput level is reached
- Maximum response times exceed the 30ms threshold at all throughput levels
- Some soft real-time applications might be satisfied with this level of performance; However, running at the 100% throughput level does not seem practical from a response time perspective.

Brand Z

- Java RTS based runtime
- Real-time garbage collector
- Fixed injection rate (13000 txns/sec)
- 60 minute measurement interval
- 30 ms maximum pause time
- Solaris 10, Real-time scheduling class

Brand Z Results Graph



Brand Z Results Table

Ware-Houses	Target Bops	Actual Bops	Avg Response (ms)	Max Response (ms)
3	2600	2600	0.86	4.38
3	5200	5200	0.72	3.76
3	7800	7800	0.63	3.65
3	10400	10400	0.67	3.93
3	13000	13000	2.41	24.10

Brand Z Results Summary

- Average response times are very good
  - Real-time garbage collection keeps up well
  - Shows signs of stress at the 100% throughput level
- Maximum response times are very good
  - Never exceeds the 30ms threshold
  - Shows signs of stress at the 100% throughput level
- Many soft real-time applications would be satisfied with this level of performance
  - Some soft real-time applications may not find the results at the 100% throughput acceptable

Limitations

- Scoring is complicated
  - Throughput, response time, response limit tuples
  - One tuple per injection step
- Short measurement interval allows non-deterministic behaviors to slip by
  - Long measurement intervals increase runtime cost
- Calibration mechanism is simplistic
- Calibration mechanism is subject to natural variation
- Shared transaction queue a potential scalability limitation

Limitations

- Benchmark geared toward soft real-time applications
- Benchmark not guaranteed to measure all aspects of real-time responsiveness
  - Does not use or measure NoHeapRealTime threads
  - Does not use or measure AsyncEventHandler threads
  - Does not use or measure network interfaces or packet related dispatch delays
  - ...
- Other real-time benchmarks needed and encouraged

Conclusion

- Established the need for a real-time benchmark
- Proposed a Java real-time benchmark
- Presented benchmark results
  - Three different Java runtimes
    - One hardware platform
    - Two different Operating Systems
  - Three different response time distributions with identical injection rates
- While not perfect, the proposed benchmark can reveal some real-time differences among various Java runtime systems and underlying platforms

A Real-Time Benchmark for Java™

Brian Doherty
brian.doherty@sun.com