SPECjbb2013 1.0: An Overview

Charles Pogue
Hewlett-Packard Company
charles.pogue@hp.com

Anil Kumar
Intel Corporation
anil.kumar@intel.com

Douglas Tollefson
IBM Corporation
dougrt@us.ibm.com

Steve Realmuto
Oracle Corporation
steve.realmuto@oracle.com

ABSTRACT
SPECjbb2013 [1] is an entirely new version of the industry standard benchmark for evaluating Java server business performance from Standard Performance Evaluation Corporation (SPEC) [2]. It is designed with three categories which allow multiple configurations (Composite/single host, MultiJVMs/ single host, Distributed/single or multi hosts), enabling the user to systematically analyze their system. Additionally, the status of published results is summarized and a series of research project configurations are suggested.

Categories and Subject Descriptors
H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness)

General Terms
Design, Experimentation, Measurement, Performance, Reliability, Standardization

Keywords
SPEC, SPECjbb2013, Benchmark, Server, System Performance, Performance Engineering, Responsiveness, Java

1. SPECJBB2013 DESIGN
SPECjbb2013 is designed from the ground up to support distributed deployment using the latest data formats (e.g. XML), communiction using compression, messaging with security as well as the latest Java Development Kit (JDK) 7 features. No code was reused from earlier SPECjbb versions.

It models the infrastructure of a world-wide supermarket company, exercises point of sales in local supermarkets as well as online purchases, by processing related to user and supply chain management and by data mining operations in the company headquarters.

Reflecting the need of modern production environments where responsiveness is critical, this benchmark is designed to show response time details across gradually increasing load levels until the maximum capacity is reached, making the benchmark very useful for a comprehensive evaluation of system responsiveness. Utilization of the Java 7 fork / join framework exploits the parallelism utilized on modern multi-core processor architecture and serves as a good example to emulate in many applications. Other Java platform features like java.util.concurrent and NIO.2 are also exercised.

ICPE'14, March 22–26, 2014, Dublin, Ireland.
ACM 978-1-4503-2733-6/14/03.
http://dx.doi.org/10.1145/2568088.2576095

2. CONFIGURATION OVERVIEW
SPECjbb2013 has 3 benchmark components. The **Controller** (CTRL) directs the execution of the benchmark. The **Transaction Injector** (TxI) issues requests to the Backend and measures the end-to-end response time for each request. The business logic resides inside the **Backend** (BE) which is comprised of three main entities: the Supermarkets (SM), the Suppliers (SP) and the Headquarters (HQ), which mainly exercises intra-JVM communication. The BE processes these requests from the TxI and notifies the TxI after a request has been processed.

Multiple run configurations that allow diverse users to analyze and overcome bottlenecks at each layer of the system stack (e.g., hardware, OS, JVM, application) are supported.

The simplest configuration is shown in Figure 1, where the three benchmark components are running inside the same JVM (dotted line) on the same server (solid line).

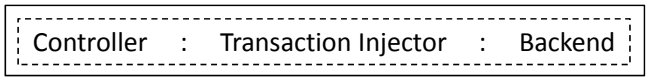

Figure 1. SPECjbb2013 Composite/Single Host Example

Figure 2 shows a configuration example where the CTRL, Txl and BE are running on the same server but each benchmark component resides in its own JVM.

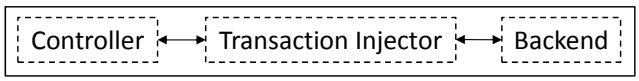

Figure 2. SPECjbb2013 MultiJVM / Single Host Example

SPECjbb2013 also allows the CTRL to reside on a separate unit, adding the complexity of network access between the CTRL and the host (TxL and BE) to the benchmark.

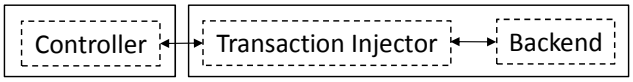

Figure 3. SPECjbb2013 Distributed / Single Host Example

Multiple hosts are also supported for even more complexity.

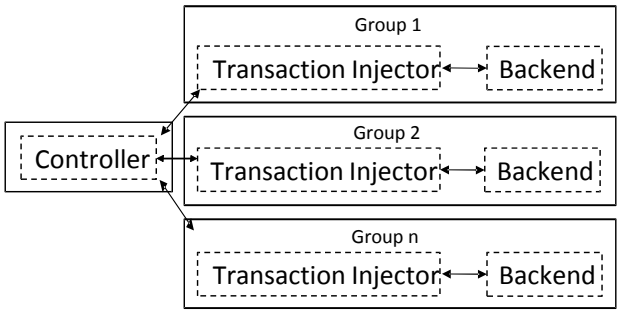

Figure 4. SPECjbb2013 Distributed / Multi Host Example

3. METRIC

SPECjbb2013 utilizes two metrics: a capacity throughput metric (max-jOPS) and a throughput under response time constraint metric (critical-jOPS). The max-jOPS metric indicates the sustainable full system throughput without any response time constraints. The higher the max-jOPS performance number, the better the system capacity throughput.

The critical-jOPS metric indicates the maximum level of performance that a system can achieve while meeting specific response time constraints. The higher the critical-jOPS performance number, the better system capacity throughput with minimum service level agreement (SLA) responsiveness requirements.

4. DIFFERENT FROM TYPICAL BENCHMARKS

A typical benchmark exercises a platform near 100% utilization and reports a throughput metric. This does not reflect usual production systems which operate below 50% utilization while ensuring graceful response at full utilization. To reflect production environment deployments, this benchmark exercises the system in gradual increments to max sustained capacity. In addition to metrics max-jOPS and critical-jOPS, benchmark report shows response time behavior across the gradual load increments in a graph which is very useful to evaluate the systems responsiveness from light load to heavy load [5].

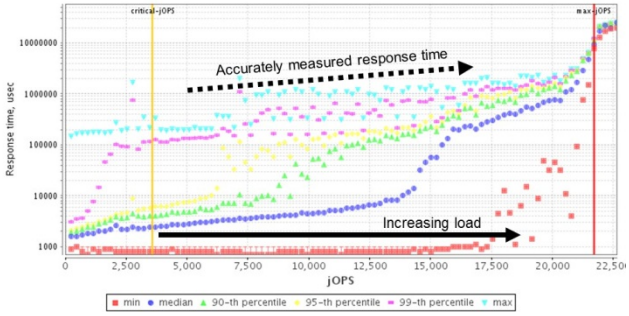

Figure 5. SPECjbb2013 Overall Throughput

5. CURRENT RESULT SUMMARY

Current results for the benchmark include submissions from many major manufacturers, indicating the benchmark's wide reaching appeal. The hardware used for the submissions covers an extensive range of systems including blade servers, rack mount servers and tower servers. A total of 20 different models have been used up to this point, ranging from simple one processor systems to complex systems using 16 processors.

6. RESEARCH OPTIONS

SPECjbb2013 has more than 300 control parameters that may be configured by researchers. While published compliant runs must follow the specified run rules, when used for research or testing, there is flexibility to set hundreds of control parameters to configure the benchmark to emulate different production environments.

Examples of such research options are `specjbb.group.count=1` which sets the number of Backends and `specjbb.forkjoin.workers=2` which sets the number of fork / join workers in the thread pool.

The benchmark can be run at a fixed injection rate for any desired period. This mode is intended for testing requiring a steady state, such as software stack testing as well as stressing the platform for longer durations. User defined load levels of a given injection rate could also be executed.

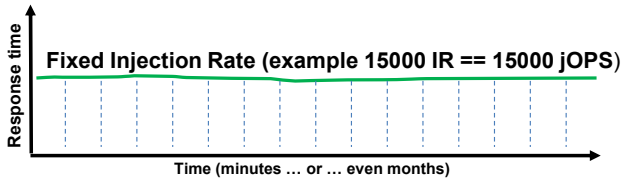

Figure 6. SPECjbb2013 Fixed Injection Rate Example

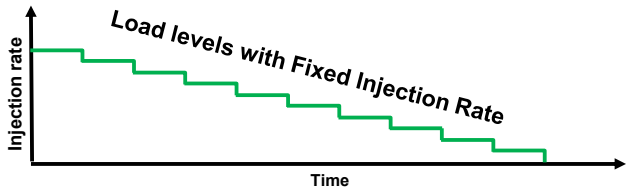

Figure 7. SPECjbb2013 Load Levels Example

Many of the above properties are listed in the SPECjbb2013.props files as part of the benchmark kit.

7. CONCLUSION

SPECjbb2013 is a highly configurable performance benchmark utilizing the latest Java technologies. The benchmark can be used from single server configuration, to virtualized and private cloud environments measuring response time accurately and therefore providing a challenging modern workload that motivates improvements in key components of the system (hardware, OS, JVM) to improve system response times.

8. ACKNOWLEDGMENTS

The authors want to acknowledge the additional members of the SPEC OSG Java Subcommittee who have contributed to the design, development, testing, and overall success of the SPECjbb2013 benchmark: Aleksey Shipilev, Daryl Maier, David Keenan, Elena Sayapina, Seiguei Katkov, Klaus-Dieter Lange, and Vasanth Venkatachalam

9. REFERENCES

[1] SPECjbb2013 web site: http://www.spec.org/jbb2013/

[2] Standard Performance Evaluation Corporation home page: http://www.spec.org

[3] SPECjbb2013 Design Document: http://www.spec.org/jbb2013/docs/designdocument.pdf

[4] Shipilev, A., and Keenan, D. 2012. SPECjbb2012: updated metrics for a business benchmark. In Proceedings of the ICPE '12 Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering. DOI= http://dx.doi.org/10.1145/2188286.2188340

[5] Maier, D. and Kumar, A. 2013. Java Interprocess Communication Challenges in Low-Latency Deployments [CON7370] https://oracleus.activeevents.com/2013/connect/sessionDetail.ww?SESSION_ID=7370