

Hyperic Whitepaper:

Hyperic HQ

and

MySQL

Backend Performance Study

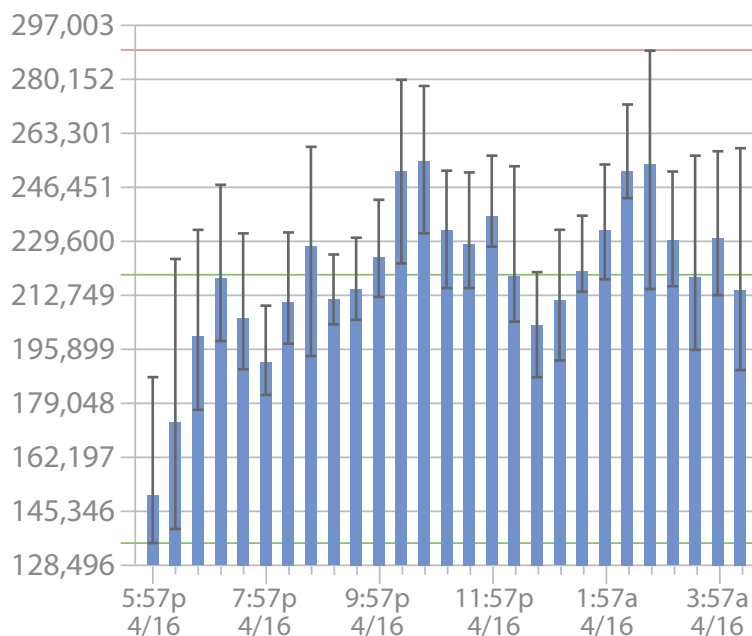


Table of Contents

- Executive Summary page 2
- Background page 2
- Terminology page 3
- Test Environment page 4
- Test Results page 5
- Test Interpretations page 6
- Conclusions page 7
- Appendix page 8



Executive Summary

In this series of backend performance studies, the performance engineers at Hyperic ran high load performance tests and proved that Hyperic HQ with MySQL as a backend database provided outstanding performance for gathering large volumes of metrics and storing data. In this test, during which we measured the performance of 675 nodes with above average metric collection load to determine baseline statistics and then simulated a 5-hour outage to record peak database usage, we found the following:

- Average metrics collected per minute – 220,000
- Peak metrics collected per minute (after simulated outage) – 2.3 million
- Server load and CPU utilization remained relatively low, suggesting that system resources were not fully utilized

The results demonstrate Hyperic HQ's capacity for monitoring and processing rules at scale. Additionally, the results provide additional insight for derived performance expectations when monitoring and managing larger scale. In fact, the additional resource capacity on both the Hyperic HQ Server and the MySQL database server prove that the combined solution could theoretically provide a capacity of over 40,000 nodes per single server instance combination.



Background

In the fall of 2006, demand for Hyperic software was accelerating, and so was the customer demand for backend MySQL database support. At the time, Hyperic supported PostgreSQL and Oracle. When adding a third option to the mix, Hyperic Engineering determined some significant obstacles that commonly restrain software development efforts from supporting additional databases:

- Lack of database abstraction layers
- Hard-coded pieces to account for different databases

In order to add support and isolate complexity, building database independence into Hyperic HQ was a critical first step. For Hyperic HQ 3.0, the team added the database abstraction layer by adding Hibernate into the core of the Hyperic HQ server.

Next, for Hyperic HQ 3.1, the team added beta support for MySQL as an optional database backend. This was no simple task, as queries and inserts into MySQL behave quite differently than in the other previously supported databases. And, being the first commercial, data-intensive, enterprise application to support MySQL, the art of tuning the application to leverage the full potential of MySQL required the expertise of both MySQL and Hyperic engineering, and sustained significant, large scale customer beta testing.

In Hyperic HQ, the server collects millions of metrics sent by HQ agents across networks. These metrics are, in turn, stored in a relational database management system (RDBMS), in this case, MySQL. Thus, the Hyperic HQ systems management suite is a robust, data intensive platform with heavy data throughput requirements. Hyperic also requires heavy data analysis and the ability to conduct fast queries and data joins.

The MySQL database server has long been known for its speed and simplicity, as well as its ability to scale with web infrastructure. Traditionally, it was not often used as a backend data store for enterprise applications. However, as the benchmark results will show, MySQL has evolved beyond its traditional role and can perform exceptionally well in an enterprise setting.

This benchmark report is intended to demonstrate that MySQL can indeed fill a role in enterprise applications. This document is focused on the performance results and expectations both Hyperic and MySQL users can expect of large scale implementations. In this case, we will use two forms of performance testing to relate our results and interpretations:

1. Simulate an actual large scale deployment monitoring 32,000 services across 675 discreet managed platforms, or nodes.
2. Simulate a maximum scale deployment using the same setup and force a HQ Server outage for 5 hours to pool a backlog of monitoring metrics. Clearing the backlog queue will indirectly prove the maximum throughput the HQ Server and the MySQL database will allow.

Terminology

Before we begin, since the systems management industry offers us many different definitions for the same term, it's important to establish the Hyperic definitions of common terms so you can better understand the information provided in this report. The Hyperic definitions are as follows:

- **Platform** – A machine/operating system combination or any network or storage device, also referred to as **nodes**. Platforms are the lowest level of management, and include components such as CPU's, Network Interfaces, and File Systems. As the HQ Server depends on agents for metric collection and management operations, each platform has its own designated **agent**.
- **Server** – Any software that is installed on a platform under management. Databases, middleware, virtualization, application and web servers are all examples of servers. Servers run on platforms. Platforms host multiple Servers. Examples of servers include any installations of JBoss, Tomcat, or MySQL on a given platform.

- Service** – A component of a server dedicated to a specific purpose. Typically services are represented by the units of work of a given server. Different types of servers each define a list of one or more types of services they provide. Examples of services include Webapps deployed in Tomcat, or Virtual Hosts configured in Apache. Services can also be attached directly to a platform in the case of CPU, Network Interfaces and Filesystems.



Test Environment

The total setup includes 4 physical machines: one for the MySQL database, one for HQ Server, and two machines which have spawned 375 HQ agents each. All agents are monitoring the same filesystems and reporting their metrics to the HQ Server. Each machine is connected via gigabit ethernet.

Machine 1: HQ Server

CPU	2 Quad Core 2GHz
RAM	16 G
HQ Heap	4G
NIC	Gigabit, with 1 GB interconnect between HQ and MySQL machines

Machine 2: MySQL Database

CPU	2 Quad Core 1596 MHz
RAM	8 G
MySQL InnoDB buffer pool	4.5G
Thread concurrency	16
Flush method	O_DSYNC
Max tmp tables	48
Tmp table size	192M
Storage	RAID-1 146G SAS 3G HardDrives
NIC	1 Gigabit

Machine 3 and 4: Hyperic HQ Agents

CPU	2 Quad Core 1596 MHz, 64-bit
RAM	8G
OS	CentOS 5

The test environment was set up to monitor 675 agents and 32,000 services. The two machines hosting the agents spawned multiple agent processes to simulate a network with 675 nodes.

Additionally, to simulate real-life management activity across these nodes, the team created 54,600 active alert definitions, including 27,000 thousand multi-condition file service alerts, 27,000 file service value change alerts, and 600 platform availability alerts. The team also tracked 26,000 event logs.

Test setup

Number of platforms/agents = 675

Number of servers = 1,000

Number of services = 32,000

Number of active alert definitions = 54,600

- 600 Value Change Platform Availability
- 27,000 Value Change on file service
- 27,000 Multicondition on file service

Number of eventlogs tracked = 26,000



Test Results

Test Scenario #1: Simulate Actual Load

HQ averaged a metrics collection rate of 220,000 metric insertions per minute, with minimal load on the servers supporting the test.

HQ Server Statistics:

The average metric collection per agent was 325 / minute.

Load average – peaked at about 8 and averaged about 2.5

CPU utilization – peaked at 20% and averaged 10%

JVM Free memory – peaked at 3GB with a low of 0.1GB and averaged 1.5GB

MySQL Server Statistics:

CPU utilization – peaked at 20% and averaged 5%

Note: See appendix for actual HQ metric charts monitoring the HQ Server performance during this test. Unfortunately, metric charts for the maximum throughput were not preserved for the purposes of this report. Hyperic will provide charts from a future test of similar nature shortly.

Test Scenario #2: Simulate Maximum Load

HQ achieved a peak of 2.3 million metric insertions per minute, with the maximum load being placed on the HQ Server itself, and moderate load on the MySQL Server.

HQ Server Statistics:

The average metric collection per agent was 2.3 million metric insertions per minute.

MySQL Server Statistics:

CPU utilization – peaked at 33% and averaged 10.6%



Test Interpretations

Interpretations for MySQL

As the test results show, MySQL handled the load in both tests easily. In fact, while the tests demonstrated the maximum load a single Hyperic HQ server can deliver to a single MySQL database instance, it did not prove out a theoretical maximum for multiple HQ Servers against a single MySQL database instance as is typically required in High Availability deployments or in large scale deployments.

Interpretations for Hyperic

Hyperic HQ is a data-intensive enterprise monitoring application, tailored for a web operations audience. For growing web-driven companies, scaling their web applications is of paramount importance to being in business. Determining scale can come in two varieties:

1. **Horizontal scalability** – monitoring and managing across tens of thousands of nodes.
2. **Vertical scalability** – concentrating monitoring and management capabilities by increasing the number of metrics and data being monitored on fewer nodes.

Given the reality of the restriction of the availability of physical hardware to manage, Hyperic chose to focus on the latter – managing the maximum amount of metrics on a finite

set of physical platforms. However, the results also translate to the former by spreading metrics across a theoretical number of platforms in a less concentrated form. We can do this by using the results of the metric inserts/minute to predict the number of managed platforms.

For instance, by using the actual value tests of 220,000 metrics per minute we can predict:

- 2,200 platforms monitoring 100 metrics/minute; or
- 4,400 platforms monitoring 50 metrics/minute

Or, using the maximum load of 2.3 million metrics per minute and a real-world average of 50 metrics collected per minute for Hyperic HQ Enterprise customers, we can create a theoretical maximum of:

- 10,000 platforms with 230 metrics/minute; or
- 46,000 platforms with 50 metrics/minute

Of course, each implementation is unique, with collection rates, additional data manipulation and deployment characteristics behaving differently. High rates of metric collections do incur small performance penalties for the systems that are targeted for collection, however, some metrics are so critical, the risk of not monitoring those processes carefully can have dramatic consequences.

Additionally, these tests do not demonstrate the effects that clustering of the HQ Servers would have. Linear scalability of the HQ Server is used by some of Hyperic's largest implementation. Given that the load on the HQ Server outweighed the load on the MySQL Server, theoretically there is substantial room for additional metric collection across a wider array of resources.

Conclusions

This test demonstrates both Hyperic HQ's ability to scale to meet the demand of even the most sophisticated web shops in the world, as well as MySQL's ability to handle enterprise-grade, transactional-based applications. While each individual implementation of Hyperic HQ and of MySQL is unique, this report sets a high baseline with which to predict your success in using either product in an enterprise setting. And at Hyperic, we believe these results provide significant support for why our customers are demonstrating an increasing demand to pair Hyperic HQ with MySQL as a backend database store.



Appendix

Metric Results during 24 hour actual load test.

Figure 1:
Hyperic HQ
Load Average

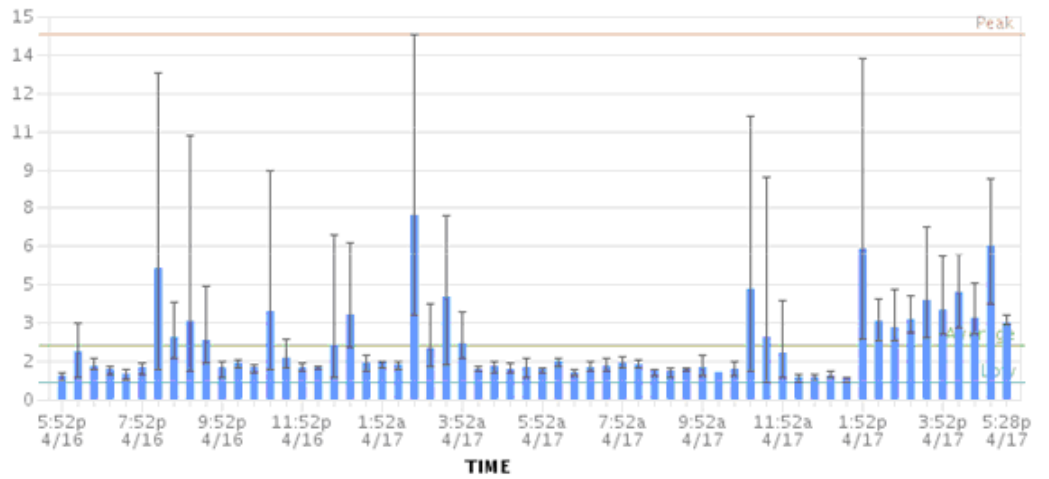
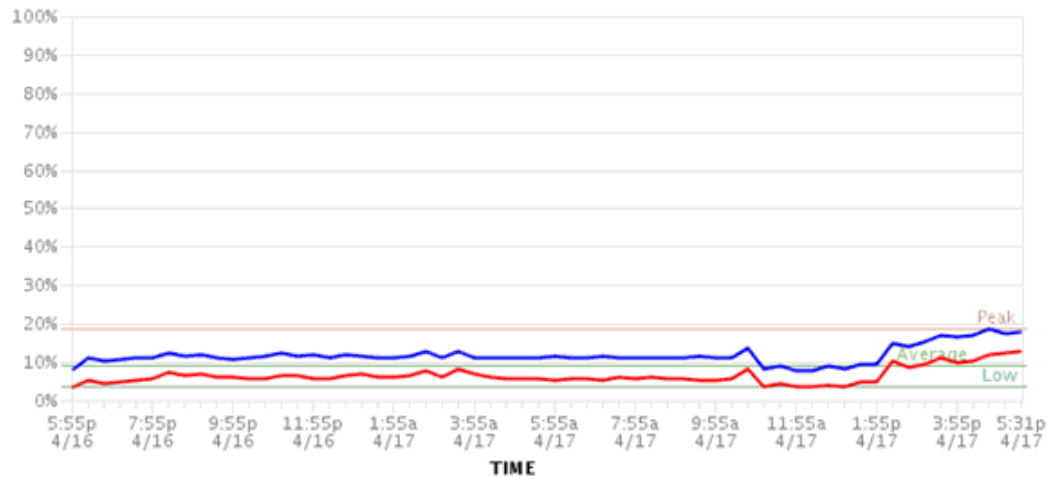


Figure 2:
Hyperic HQ CPU Usage





Appendix

Figure 3:
Hyperic HQ JBoss JVM
Free Memory

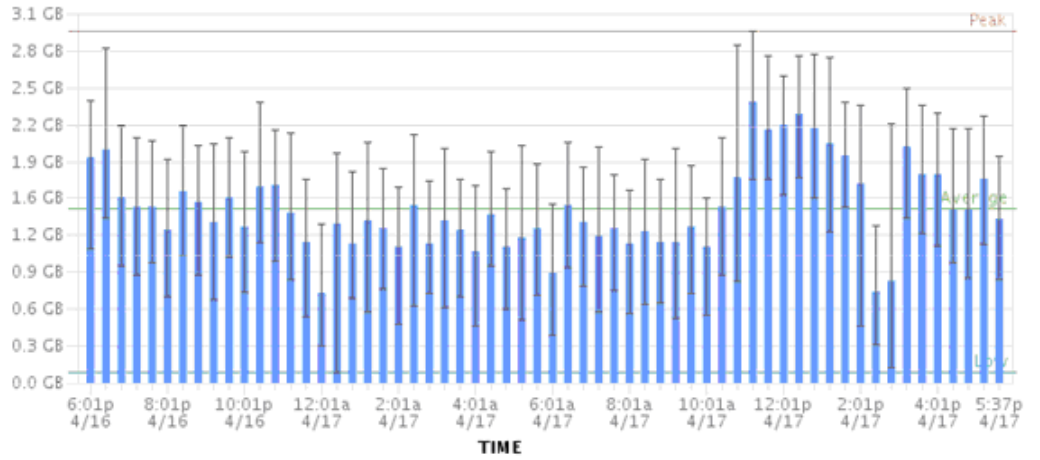
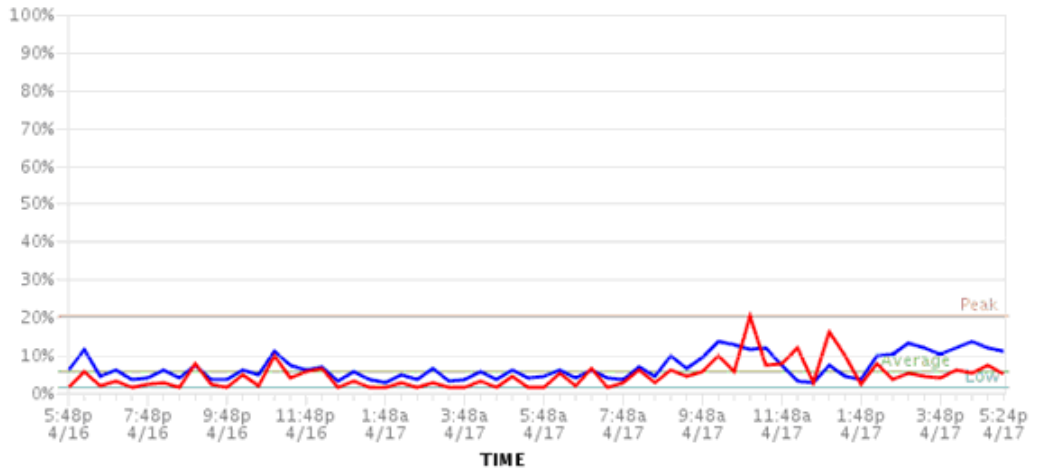


Figure 4:
MySQL CPU Usage





Appendix

Figure 5:
MySQL Metric Inserts
per Minute

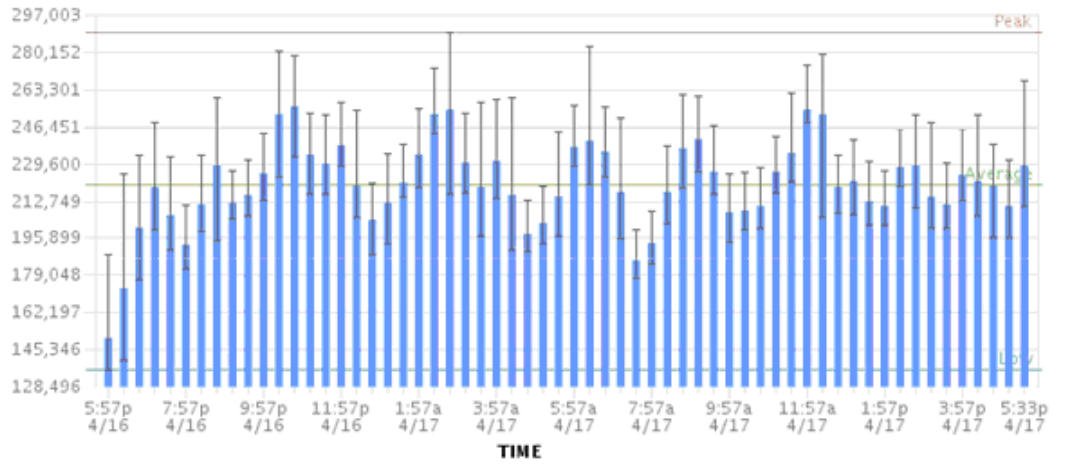


Figure 6:
MySQL CPU Utilization
at Maximum Load

