



A Technical Whitepaper
On

MySQL Performance Analysis and Tuning Tips on HP-UX

Prepared By
ViSolve Database Performance Team

1. EXECUTIVE SUMMARY	2
2. BENCHMARKING DATABASE PERFORMANCE	2
2.1 MEASURING PERFORMANCE IN A DATABASE.....	2
2.2 WHY SYSBENCH?.....	3
3. PERFORMANCE TUNING TIPS	3
3.1 FILE SYSTEM BUFFER SIZE.....	3
3.2 HOT FUNCTIONS.....	3
3.3 THREAD SCHEDULING	4
3.4 CPU UTILIZATION.....	5
3.5 COMPILING OPTIONS.....	5
3.6 BLOCKING ALARM SIGNALING	5
3.7 EXPORTING ENVIRONMENT VARIABLES	6
4. CONCLUSION	7

1. Executive Summary

In general, the performance of a database is based on several critical factors. The factors include, but not limited to, hardware, operating system, data schema, concurrency, volume of data, type of application, data access pattern and database configuration. To meet the business demands, your database application must address three vital constraints – Performance, Scalability and Reliability – each of which needs customization of different but correlated elements.

In this whitepaper, we are going to explore different strategies to considerably optimize the performance of MySQL in HP-UX 11.23 IPF.

2. Benchmarking Database Performance

2.1 Measuring Performance in a Database

The most common technique for measuring performance is to take a black box approach that measures the Transactions Per Second (TPS) an application is able to execute against a database. In this scenario a “Transaction” is a unit of execution that a client application invokes against a database. This could be a simple read query or a grouping of updates done in benchmark applications.

SysBench benchmark tool was used to measure MySQL’s performance in HP-UX 11.23 IPF.

2.2 Why SysBench?

SysBench is a modular, cross-platform and multi-threaded benchmark tool for evaluating OS parameters that are important for a system running a database under intensive load. SysBench is a flexible testing utility that allows a variety of different test modes to be plugged-in for use with the utility.

SysBench was originally designed to test parameters such as file I/O performance, scheduler performance, memory allocation and transfer speed, and POSIX thread implementation performance. SysBench allows a tester to configure the number of threads, the amount of data in the database, the access pattern, and whether the database is read-only, read-mostly, or read-write.

3. Performance Tuning Tips

3.1 File System Buffer Size

MySQL uses file system buffer cache for file I/Os. On Analysis, for SysBench File I/O Sequential Read tests, the performance numbers increase significantly when the buffer size is increased. Ideally, the maximum buffer size should be set as 50% of total memory size. However, in HP-UX, one can set a min of 1% and max of 90% of main memory for buffer caching. So, it is recommended to change the default file buffer size when running MySQL in HP-UX for better I/O performance.

3.2 HOT functions

It is very essential to understand how HP-UX 11.23 IPF synchronizes with MySQL, which could only be achieved by scrutinizing how well HP-UX's system and library functions perform in MySQL and the vice versa.

With the help of HP Caliper - a general-purpose performance analysis tool for understanding the performance and execution of your application in HP-UX and to identify ways to improve its run-time performance - one can derive the frequently call system, library and mysqld functions.

3.3 Thread Scheduling

Note that **not all platforms** are suited equally well for **running MySQL**. How well a certain platform is suited for a high-load mission-critical MySQL server is determined by the following factors:

- General stability of the thread library.
- The ability of the kernel and/or thread library to take advantage of **SMP** on multi-processor systems.
- The ability of the kernel and/or the thread library to run many threads which acquire/release a mutex over a short critical region frequently without excessive context switches.
- General file system stability/performance.
- Ability of the file system to deal with large files at all and deal with them efficiently, if your tables are big.

In general, if MySQL AB knows a platform well, it introduces platform-specific optimizations/fixes enabled at compile time. Also, the amount of testing of similar configurations MySQL AB has done internally. The number of users that have successfully run MySQL on that platform in similar configurations. If this number is high, the chances of hitting some platform-specific surprises are much smaller.

3.4 CPU Utilization

Based on the benchmark results, we focused on the CPU as one of our assumptions was if the system is busy with I/O there may be other transactions that can utilize the CPU. Our early analysis has shown that a good percentage of the CPU time is spent on MySQL stack and hence we focused on MySQL user space.

3.5 Compiling Options

This is an easy yet effective way to increase a database performance. The build and the environment (make files) contribute a significant performance difference in MySQL (Compiler flags, execution path, etc). Compiling MySQL (standard Edition) with HP C and setting the optimization level as +O2 provides around ~21% increase in performance in HP-UX.

3.6 Blocking ALARM Signaling

Alarms are events that happen at a given time, either once or periodically. A thread is associated with an alarm handling so that the function will be invoked every time the alarm “goes off”. The signal thread handles all signals. This thread also normally handles alarms and calls *process_alarm()* to force timeouts on connections that have been idle too long. Either a dedicated thread handler can be created or *thr_alarm()* can be used to code the application. If dedicated thread handler is used, then there will be an overhead due to *sigwait()*.

In a multithread environment ALARM signals will interrupt any system calls that may happen for whole instance thereby interrupting the proper workflow of clients and the server. So killing the ALARM signals during any system call will obviously increase the performance of any application

MySQL Performance Analysis & Tuning Tips on HP-UX

On analysis, it is found that HP-UX spends a significant amount time on *thr_alarm()* and *thr_end_alarm()* whose routines such as *sigprocmask()*, *sigismember()*, *sigaddset()*, *sigdelset()* are called at run time. On further analysis, it is found that in MySQL source, HP-UX does shifts to a different code path where other OS doesn't due to *thr_alarm()*. On avoiding this overhead, you will note a significant increase (20%) in the performance of MySQL.

3.7 Exporting Environment variables

PTHREAD_FORCE_SCOPE_SYSTEM forces the system scope irrespective of the range specified in the attribute for thread creation i.e., the system scope thread improve the performance better than process scope threads. **PERF_ENABLE**, used with **PTHREAD_FORCE_SCOPE_SYSTEM**, informs the pthread library to only use 1x1 threads.

_M_CACHE_OPTS - a thread cache variable to control the size of pthread specific cache. This helps to tune *malloc()* performance in a kernel threaded application, which will configure a thread-private cache for malloced blocks. This will also improve the speed in applications by reducing mutex contention among threads and by deferring coalescence of blocks.

4. Conclusion

The ViSolve Database Performance Team has been able to achieve 73% increase in the performance of MySQL in HP-UX 11.23 IPF by adopting the following performance enhancement modifications.

- File System Buffer Size should be 50% of main memory for better I/O performance. (Linux uses dynamic allocation and hence this is NOT an issue on Linux)
- As MySQL performance is platform-specific, it should be re-compiled using HP C compiler with at least +O2 optimization level
- MySQL spends a significant amount time in thread signaling (ALARM) - *thr_alarm()* and *thr_end_alarm()*. This overhead could be avoided by making appropriate source code modifications and there would be a substantial increase in the performance of MySQL.
- HP-UX scheduling - Exporting Thread and Thread Cache based environment variables will help increase the performance.
- The frequently (HOT) invoked system, *mysqld* and library functions in MySQL for HP-UX should be optimized.

If you are interested in enhancing the performance of MySQL in HP-UX 11.23 IPF, in general, and database performance tuning for your business applications, please write to us at mysql_support@visolve.com

About ViSolve

ViSolve is a software consulting and support organization focused on providing value-added solutions largely through open source products and technologies. We deliver scalable and cost-effective solutions/services in Networking and Security, Internet Caching(Squid), Databases(MySQL) and Web Applications(JBoss) with immediate and measurable business values.

Our knowledge and experience in a variety of platforms and technologies combined with our strong understanding of business processes enables us to implement comprehensive solutions/services you need to achieve your goals. ViSolve is committed to helping you get the most from your IT investments by offering 7x24 commercial support and services to your business.

Our Source initiative will be an advantage to boost up your market potential. For further information, please visit www.visolve.com