Hyper-Threading: Be Sure You Know How to Correctly Measure Your Server's End-User Response Time

Stephen P. Smith
Core Software Division
Software and Solutions Group
November 19, 2003
Stephen.P.Smith@intel.com

Introduction

Recently, an Independent Software Vendor (ISV) approached Intel with a problem. It seemed that their test measurements showed that enabling Hyper-Threading (HT) [1] on an Intel® Xeon™ processor resulted in increased response time of their web server system. After some simple analysis, it was clear that their users were actually seeing better response time with HT enabled. The problem was actually how the ISV in their test setup measured response time.

This note will explain what happened to cause the confusion and point out how it can be avoided.

Background

Lets examine a simplified server system. It consists of a waiting queue and a work process per hardware CPU. Client transactions enter the server, wait in the queue, sojourn thru their required processing in the work process, and then return data to the client. Response time of the server, as experienced by the end user, is measured from when the server places the work in its waiting queue to when the work process completes the client's transaction.

When testing the work process in isolation, it was quite natural for the ISV to look at two items (a) CPU time of that process and (2) the elapsed wall-clock time (work process sojourn time) that the process took to complete a client's transaction.

The ISV assumed that any increase in the sojourn time of the work process would directly add to the response time experienced by the end user.

In general, both the above measures can be very misleading when trying to compare the performance of HT-enabled systems from those without HT. Hyper-Threading increases the total throughput of the system, but at the cost of making some of our normal performance measures more difficult to interpret.

Lets demonstrate the issues with a simple example.

Simple Example

Without loss of generality, lets analyze a single CPU and work process, which have a throughput of one transaction per second when the CPU is fully loaded. We will load this system with enough users to have an average response time of 4 seconds. Each user will have a think time of 10 seconds. Then, the number of users in the system must be 14, from the standard queuing theory/balance equation [2] for a closed loop system:

$$R = N / X - Z$$

where R is the response time, N the number of users, X is the throughput and Z is the user think time. We can further break server response time into its two components, R = Q + C, where Q is time in the wait queue and C is the time a job sojourns through the work process. Clearly, X <= 1/C, with strict equality occurring if and only if the work processes are never idle.

Lets examine the situation from the server's point of view. Clearly, to achieve one transaction per second, the CPU must be 100% busy. CPU time per transaction is one second. Sojourn time thru the work process is one second and average queue time is three second. At any time, we expect one user to be in the work process and another three to be waiting in the queue.

Now lets turn on HT. Our only change will be to add another work process so that both logical CPUs have a work process. Lets assume that we get **no** benefit from HT being enabled. That is, throughput will be the same as the non-HT case.

Now, again from the server point of view, CPU time per transaction is two seconds---CPU clock ticks in HT mode are double counted (or worth 1/2 of non-HT mode)[1]. Sojourn time thru the work process rises to 2 seconds, and average time in the work queue is 2 seconds. At any time, we expect two user jobs on the server to be in some state of processing in the two work processes and two to be in the waiting queue.

Ok, there you have it. CPU time in the non-HT case is 1 second while CPU time for the HT case is 2 seconds. Work process sojourn time in the non-HT case is 1 second; Work process sojourn time in the HT case is 2 seconds.

What has happened? Basically, nothing has, from the client point of view. No change in the client's view of server response time, server throughput, number of clients supported, etc. This is what we would have expected, since we assumed that HT provided no benefit. On the server, we have traded time in the wait queue for time running on the CPU.

---

[1] In general, measured CPU time per transaction under Hyper-Threading can be 1X to 2X more than the non Hyper-Threaded case. To illustrate the 1X case, assume in our example that the application, while it is now composed of two work processes under HT, must actually perform its work serially for each transaction. The application protects these critical sections with semaphores. Each transaction would then see one second of CPU time and one second of semaphore wait time for our load of 12 users.

So, if the code in your server's work process starts and stops a wall clock timer when the work process starts and finishes each job (i.e., you measure work process sojourn time), you are likely to see increases (up to 2X) in this "response time". However, this is not the response time experienced by the end users of your system.

A Hyper-Threaded Server Behaves as Two Servers, Each with One Half the Throughput (Sort Of..)

Another way to view the impact of enabling HT is that it turns a single server system into a system with two servers. If HT provides no benefit, then each of the two new servers has one half the throughput of the original single server. But this is too simplistic a view. Systems with HT enabled can adapt to the load placed on them.

It is not necessarily the case that either server CPU time or work process sojourn time increase with HT enabled. Only when parallel requests are made to the server system does HT step in to allow both logical processors to work on each request.

To illustrate, we reduce the load on our example system to the point where the CPU stays 100% busy but no queuing occurs on the server (by lowering the number of users from 14 to 11). Then, since only one job is available at any time, HT only uses one work process on one logical processor at a time. CPU time, work process sojourn time, and end-user response time will remain at one second, the same as the non-HT enabled case.

Hyper-Threading Enhances System Throughput and Reduces End-User Response Time

With Hyper-Threading, the processor core has the ability to simultaneously process instructions from two Operating System (OS) threads during the same OS time slice. This allows the processor to produce more throughput (transactions/sec) from the same system. Our balance equation shows that enhanced throughput will result in reduced end-user response time.

In our example, lets imagine that, instead of seeing no benefit from HT, each logical processor takes 1.66=5/3 CPU seconds to process a transaction when HT is enabled, rather than the baseline of 2 seconds.

The system is then able to do 2/1.66 = 6/5 transactions per second, assuming 14 users is enough to load the system. Checking using the balance equation, we find that R = 5/3, implying that Q=0, and that we have a valid transaction rate.

Thus, when HT has the 1.2X benefit of increasing throughput from 1 transaction a second to 6/5 transactions a second---which is well within the measured benefit of HT on many actual applications---our

example system has experienced a reduction in response time from 4 seconds to 1.66 seconds.[2]

However, if we continued to measure "response time" by work process sojourn time, we would think response time has increased under HT from 1.0 second to 1.66 seconds!

Conclusion

You should make sure your server's testing setup is **not** measuring response time from point of view of a server's work process, but rather by:

(a) Directly averaging true client response time or
(b) Measuring the true time each transaction spends on the server, including all queue time, or
(c) Calculate response time using the balance equation.

In practice, it is an excellent idea to measure each of these three items, if for no other reason than to sanity check your test results.

This note serves as a cautionary tale of what happened to one of our ISV's: Don't let inappropriate measurement techniques fool you into thinking that Hyper-Threading is providing no benefit to your end users.

References

[1] Hyper-Threading Technology Architecture and Microarchitecture, D. Marr et. al., Intel Technology Journal, Volume 06, Issue 01 February 14, 2002.
http://www.intel.com/technology/itj/2002/volume06issue01/art01_hyper/p01_abstract.htm

[2] P.J. Denning and J.P. Buzen, "The operational analysis of queueing network models," ACM Computing Surveys, 10:225--262, 1978.
http://cne.gmu.edu/pjd/PUBS/csurv-opanal.pdf

[2] Alternatively, with HT enabled, the system could support 16 users with total system response time less than 4 seconds