Running IBM System z at High CPU Utilization

Gary King
IBM
System z Performance

Version Date: November 2, 2007

This document can be found on the web, www.ibm.com/support/techdocs
Under the category of “White Papers.”

Overview

One of the strengths of the System z platform is the ability to run processors at high utilization. By design, many System z customers consistently run their processors at 90+% busy for extended periods, and some will literally run at 100%. Several questions have been asked about this unique trait and strength of System z including "Can any single workload be run at 100% CPU busy?" and "Is there a growth in CPU time per transaction at higher utilization?" This paper will discuss running System z at high utilization and address these questions.

System z has invested significantly in the operating system (z/OS), the firmware (PR/SM) and the hardware in order to provide a platform that scales well as engines are added and is capable of fully utilizing the capacity offered by a processor. The workload management functions provided by z/OS's Workload Manager (WLM) allow customer applications with divergent characteristics and importance levels to coexist within a single z/OS image. The Intelligent Resource Director (IRD) function provided through support in z/OS and PR/SM extends this management to multiple z/OS images running on a single processor. The System z clustering technology known as Parallel Sysplex encompasses a wide range of hardware and software technologies to further extend workload management across multiple z/OS images spread across multiple processor footprints. The net is that System z provides the technology to simultaneously run a diverse mix of customer applications and exploit the full capability of the underlying hardware while meeting customer service levels.

Can Any Single Workload Be Run At 100% CPU Busy?

The simple answer is no. Queuing theory tells us that under many conditions described by arrival rate and service time distribution, as a server approaches 100% the queue of work waiting to be processed grows exponentially as does response time. Generally speaking, homogeneous, transaction processing workloads behave like this. If thousands of users are submitting equal priority, short transactions the queuing theory conditions are met and as a processor approaches 100% busy the internal work queues will grow - causing numerous points of contention within the operating system and subsystem leading to poor response time and potentially a reduction in transaction rate. On the other hand if there are a limited number of heavy, compute intensive transactions or jobs that arrive at a system, then as the processor reaches 100% busy, job elapsed times may elongate but internal work queues will not grow exponentially – they are effectively capped by the limited number of transactions or jobs that arrive (for example, limited by the number of batch initiators that are started).

System z customers that wish to run their processors near or at 100% busy can do so effectively by taking advantage of the ability of System z to manage diverse workloads. A high priority transactional workload (that tends to match the typical queuing theory characteristics of arrival rate and service time distribution) should be kept below the utilization at which queuing theory would predict exponential growth in queues and response time. For a large multi-processor system this would typically mean to capacity plan so that the transactional workload peak reaches no higher than 90% of the capacity of the processor (for small multi-processor systems, queuing theory would suggest lowering this threshold as low as 70%). To this transactional workload, lower priority work may be added that has the characteristics of limited, compute

intensive work. Thus, the processor may run up to 100% busy while the workload management functions of System z insure the important work receives its share before giving the leftovers to the lower priority work.

Is there a growth in CPU time per transaction at higher utilization?

The simple answer is yes. Transactions being processed on a system share the physical hardware resources (CPs, caches, memory buses) and software resources (z/OS, subsystems) of that system. One might presume that if a transaction rate of X drives a processor at 50% busy, then at 100% the processor would be able to support 2X the transaction rate. However, this will not be the case. At transaction rate X, each transaction has access to on average 1/X the physical resources (cache and memory buses) and causes the software to have to manage X-relative control blocks and work queue depths. At transaction rate 2X, each transaction now has access to 1/(2X) (50% less) the physical resources and the software must now manage 2X (double) the number of control blocks and work queue depths. Thus, the CPU time cost to process a transaction at the 2X rate is increased as the hardware has to retrieve instructions and data further out in the cache/memory hierarchy and the software expends more instructions managing the additional control blocks and longer work queues. Essentially, although the application instruction path length of a transaction may stay constant, the hardware is slowing down (running at a lower MIPS rate) and the operating system and subsystem path length is increasing.

A system with N engines running at transaction rate X at 50% CPU busy is somewhat analogous to running a system with N/2 engines at transaction rate X at 100% CPU busy. Much of the hardware is often the same in these two cases as the level-2 cache and memory buses are shared amongst many engines, and, of course, in each case there is just one copy of the software that is being shared. It is well understood that there are "multi-processor(MP)-effects" when one compares processors with different number of engines (and at the same utilization) and that these MP-effects result in a growth in CPU time per transaction as the number of engines increases. For example, due to MP-effects, a 16way processor may yield 12x the capacity of a 1way processor rather than the theoretical linear scaling of 16x. Thus, the CPU time cost per transaction will have grown by 33% to be consistent with the 12x capacity yield (16/1.33 = 12). Many of the reasons for the MP-effects are the same as those that cause the growth in CPU time per transaction as the workload is increased on a system with a constant number of engines (thus resulting in high utilization) – namely, the increased sharing of common hardware and increased management of more work units in the software.

From the above discussion, it should not be surprising that the magnitude of the growth in CPU per transaction as utilization increases is similar to the MP-effects across a change in Nway (number of engines) within a processor family. Just as workloads with different characteristics scale differently across the Nway curve, workloads will see varying growth in CPU per transaction across different utilization on different Nways. For example, if we use the System z Large System Performance Reference (LSPR) workloads to compare the CPU time per transaction of a z9 20way to a z9 10way processor we would see a range of +11% to +23%. This same range would roughly approximate what we could expect to see comparing a z9 20way processor at 50% busy to the same processor at 100% busy.

This is an important factor to consider in capacity planning. How much capacity remains in an existing processor? If the existing processor is running at 50%, one should not presume there is room to contain a doubling of the workload. When consolidating footprints, if a "before" footprint is running at 50% busy and it is moving into a processor that is planned to be 90% busy, the 50%-busy workload might not fit unless an adjustment is made for the low-utilization effect (that is, the 50%-busy workload must be adjusted upward to the CPU cost per transaction it would have had at 90% on the "before" footprint).

It should also be noted that the magnitude of impact may vary based on the access a workload has to the shared resources. For example, a high priority workload that is consuming a relatively small portion of the overall processor may be dispatched frequently enough to maintain more of its use of shared resource than will the medium and lower priority workloads, thus the impact may be less. Conversely on a processor with multiple LPAR partitions, a small, low-weighted partition may be dispatched infrequently leading to difficulty in maintaining its fair share of the resources, thus suffering a larger impact.

Finally, it is for the reasons discussed above that the LSPR measurements on which the processor ratings (MIPS and MSUs) are based are conducted at 90% busy across all processor models. This provides a consistent view amongst the processors being evaluated and at a maximum utilization for homogeneous workloads. In this manner, the effect on CPU cost per transaction at high utilization is taken into account in the ratings. Note that periodically multiple LSPR workloads are run simultaneously to study workload interactions and workload management and these measurements are conducted at near 100% busy.