Achieving CPU (& MLC) Savings through Optimizing Processor Cache

Todd Havekost, IntelliMagic

Customer experiences with z13 processors have confirmed that delivered capacity is more dependent than ever before on effective utilization of processor cache. After providing the framework for understanding and interpreting the enlightening metrics available from the SMF 113 records, this paper identifies potential ways to leverage those metrics to improve processor cache efficiency, thereby reducing CPU consumption and MLC software expense. Insights into the potential impact of various tuning actions are demonstrated using data from several real-life case studies.

1. Introduction

Processor cache utilization plays a significant role in CPU consumption for all z processors, but that role is more prominent than ever on z13 models. Achieving the rated 10% capacity increase on a z13 processor vs. its zEC12 predecessor (despite a clock speed that is 10% slower) is very dependent on effective utilization of processor cache.

This paper will introduce key processor cache concepts and metrics, laying the foundation for understanding the vital role processor cache plays in CPU consumption. Those metrics will be leveraged to identify specific ways to improve processor cache efficiency by optimizing LPAR topology and maximizing work executing on Vertical High CPs.

Throughout the paper several examples from real-life situations will be used to illustrate both the opportunities and the potential impacts of tuning actions.

2. Key Metrics (CPI and RNI)

The first key metric to consider is Cycles Per Instruction (CPI). CPI represents the average number of processor cycles spent per completed instruction. Conceptually, processor cycles are spent either productively, executing instructions and referencing data present in Level 1 (L1) cache, or unproductively, waiting to stage data due to L1 cache misses.[1] Keeping the processor productively busy executing instructions is highly dependent on effective utilization of processor cache.

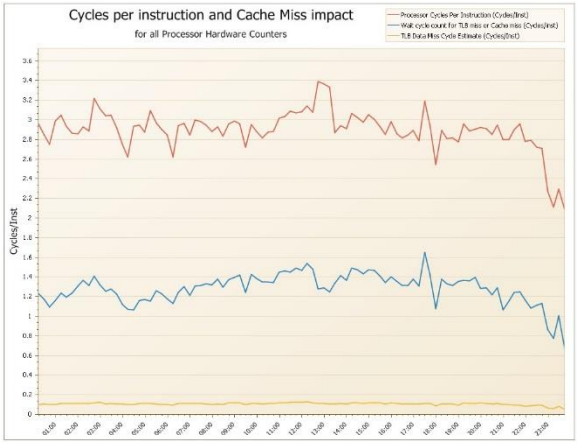

Figure 1. Cycles per instruction (CPI).

Figure 1 breaks out CPI into the "productive" and "unproductive" components just referenced.[2] The space above the blue and below the red line

[1] While "waiting to stage data" z processors leverage Out of Order execution and other types of pipeline optimizations behind the scenes seeking to minimize unproductive waiting.

[2] Another small component of CPI, Translation Lookaside Buffer (TLB) misses while performing Dynamic Address Translation, is represented by the yellow line on Figure 1.

reflects the cycles productively spent executing instructions for a workload. This would be the CPI value if all required data and instructions were always present in L1 cache, and it reflects the mix of simple and complex machine instructions in a business workload.

But since the amount of L1 cache is very limited,[3] many machine cycles are spent waiting for data and instructions to be retrieved from processor cache or memory into L1 cache. Note that these "waiting on cache" cycles, represented by the area under the blue line, represent a significant portion of the total, typically 35-55% of total CPI.[4] This highlights the magnitude of the potential opportunity if improvements in processor cache efficiency can be achieved and the importance of having clear visibility into key cache metrics.

A second key metric and one that correlates closely with the unproductive cycles represented by that blue line is Relative Nest Intensity (RNI). RNI quantifies how deep into the shared processor cache and memory hierarchy (called the nest) the processor needs to go to retrieve data and instructions when they are not present in L1 cache (see Figure 2). This is very important, because access time increases significantly for each subsequent level of cache and thus results in more waiting by the processor.

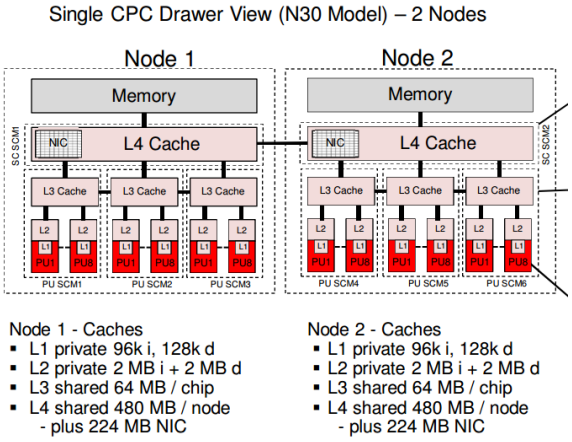

Figure 2. Architecture of the z13 cache [Sinram2015].

The formula to calculate RNI as provided by IBM is processor dependent and reflects the relative access times for the various levels of cache. Here is the z13 RNI formula [Kyne2017].

2.3 * (0.4*L3P + 1.6*L4LP + 3.5*L4RP + 7.5*MEMP) / 100

L3P = % of L1 misses sourced from the shared chip-level L3 cache
L4LP = % of L1 misses sourced from L3 or L4 cache in the same (local) node
L4RP = % of L1 misses sourced from L3 or L4 cache in a remote node or drawer
MEMP = % of L1 misses sourced from memory

Note that retrievals from memory (MEMP) have a weighting factor almost nineteen times higher than the factor for L3 cache (7.5 vs. 0.4), reflecting how many more machine cycles are lost waiting for data when it is not found anywhere in processor cache and must be retrieved from memory. IBM defines a threshold value for RNI of 1.0 above which a workload is considered to place a high demand on cache and thus may not achieve the rated capacity of a processor. But no matter the current RNI value, reducing it means less unproductive waiting cycles and thus less CPU consumption.

3. HiperDispatch

A key technology to understand when seeking to reduce RNI is HiperDispatch (HD). HD was first introduced in 2008 with z10 processors, but it plays an even more vital role on z13 models where cache performance has such a big impact.

With HD, the PR/SM and z/OS dispatchers interface to establish affinities between units of work and logical CPs, and between logical and physical CPs. This is important because it increases the likelihood of units of work being re-dispatched back on the same logical CP and executing on the same (or nearby) physical CP. This optimizes the effectiveness of processor cache at every level, by reducing the frequency of processor cache misses, and by reducing the distance (into the Nest) required to fetch data.

When HD is active, PR/SM assigns logical CPs as Vertical Highs (VH), Vertical Mediums (VM), or Vertical Lows (VL) based on LPAR weights and the number of physical CPs. VH logical CPs have a 1-1 relationship with a physical CP. VMs have at least a 50% share of a physical CP, while VLs have a low share of a physical CP (and are subject to being "parked" when not in use).

Work executing on VHs optimizes cache effectiveness, because its 1-1 relationship with a physical CP means it will consistently access the

[3] 96 KB for instructions and 128 KB for data per processor core on z13 models [Sinram2015].

[4] Based on the author's analysis of CPI data from 45 sites to date. This leads to the rule of thumb that a 10% reduction in RNI correlates to a 5% reduction in CPU consumption.

same processor cache. On the other hand, VMs and VLs may be dispatched on various physical CPs where they will be contending for processor cache with workloads from other LPARs, making the likelihood of their finding the data they need in cache significantly lower.

This intuitive understanding of the benefit of maximizing work executing on VHs is confirmed by multiple data sources. One such source is calculated estimates of the life of data in various levels of processor cache (derived from SMF 113 data). Commonly, cache working set data remains in L1 cache for less than 0.1 millisecond (ms), in L2 cache for less than 2 ms, and in L3 cache around 10 ms. This means that by the time an LPAR gets re-dispatched on a CP after another LPAR executed there for a typical PR/SM time-slice of 12.5 ms, its data in L1, L2 and L3 caches will all be gone. The good news is those working sets will be rebuilt quickly from L4 cache, but the bad news is that each access to L4 cache may take 100 or more cycles [for more detail on this analysis see Houtekamer2016].

This thesis that work executing on VHs experiences better cache performance is further substantiated by analyzing RNI data at the logical CP level. At the beginning of the case study presented in Figure 3, the Vertical CP configuration for this system consisted of three VHs and two VMs. Note that the RNI values for the two VM logical CPs (CPs 6 and 8) were higher than the RNIs for the VHs.[5]

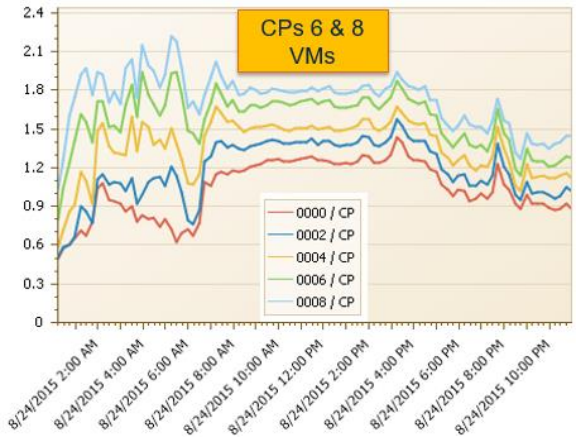

Figure 3. RNI by Logical CP Sys3 (3 VHs,2 VMs).

After the deployment of additional hardware caused the Vertical CP configuration to change to five VHs (Figure 4), the greatest reductions in RNI occurred for CPs 6 and 8. Now that they had also become VHs and were no longer experiencing cross-LPAR contention for processor cache, their RNIs decreased to a level comparable to the other VHs [Havekost2016].

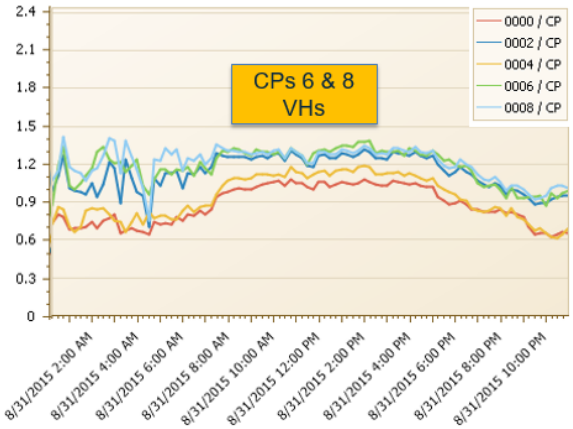

Figure 4. RNI by Logical CP Sys3 (5 VHs).

Up to this point we have seen that CPI decreases when there is a reduction in unproductive machine cycles waiting for data and instructions to be staged into L1 cache. Reducing CPI translates directly to decreased CPU consumption, which when occurring at peak periods results in reduced IBM Monthly License Charge (MLC) software expense.

Opportunities to optimize processor cache are worth investigating, because those unproductive waiting cycles typically represent at least one third of overall CPU, and often one half or more. Fortunately, the mainframe is a very metric rich environment, and the RNI metric is available that correlates to those unproductive waiting cycles.

Note that optimizing processor cache can have a particularly big payoff on z13 processors, which are more dependent than ever before on effective processor cache utilization. Two primary ways to reduce RNI will now be considered, by optimizing LPAR topology, and by maximizing work executing on VHs.

[5] RNIs for VMs and VLs are routinely higher than for VHs throughout the dozens of use cases reviewed by this author.

4. LPAR Topology

PR/SM dynamically assigns LPAR CPs and memory to hardware chips, nodes and drawers seeking to optimize cache efficiency. This topology can have a very significant impact on processor performance, because remote cache accesses can take hundreds of machine cycles.[6] The reader is encouraged to reference Figure 2 to provide a framework for the following discussion.

A z13 processor has one to four CPU drawers (along with some number of I/O drawers), and two nodes in each drawer. Each active physical CP (labelled "PU" on Figure 2) has its own dedicated L1 and L2 cache.

When a unit of work executing on a CP on a given Single Chip Module (SCM) accesses data in L3 cache, the first level that is shared (and thus is part of the "nest" and RNI metric), that access can be on its own SCM chip, "on node" (on a different SCM within this node), "on drawer" (on the other node in this drawer), or "off drawer". The more remote the access, the greater the number of machine cycles required, often hundreds of cycles. Similarly, access to L4 cache and to memory can be "on node", "on drawer", or "off drawer."

A first example showing the impact of LPAR topology on RNI is begins with Figure 5.

	Chip 2
Drawer 2 Node 1	SY22 VM00
	SY22 VM01
	SY22 VL02
	SY18 VH00
	SY18 VH01
	SY18 VH02

	Chip 2	Chip 3
Drawer 3 Node 1	SY18 VM03	**SY03 VH00**
	SY18 VM04	**SY03 VH01**
	SY16 VM00	**SY03 VH02**
	SY16 VM01	SY03 VM03
	SY16 VL02	SY03 VM04
	SY16 VL03	SY16 VL04
	SY20 VM00	
	SY20 VL01	

Figure 5. LPAR Topology – Example 1.

Each entry in this diagram represents a general purpose logical CP and indicates the system identifier, polarity (VH, VM or VL), and relative logical CP numbered starting from 0. VHs are highlighted in bold. zIIP CPs have been removed because this analysis is focused on general purpose CPs.

In this scenario, the logical CPs from several LPARs are competing for physical CPs on three chips. Green and orange shading has been added to compare two primary Production systems that execute similar data sharing workloads, each having three VHs and two VMs.

Note that the VHs and VMs for System 3 (shaded green) are collocated on the same chip. The "RNI by Logical CP" chart for this system appeared earlier in Figure 3. The RNIs for the two VM CPs on that system were higher than for the VHs, but not dramatically so.

Contrast that with the topology of System 18 (in orange), which has three VHs in Drawer 2, but its two VMs reside in a different drawer, Drawer 3. These VMs are located not on a different chip in the same node, not on a different node in the same drawer, but in the most remote possible location, in a different drawer.

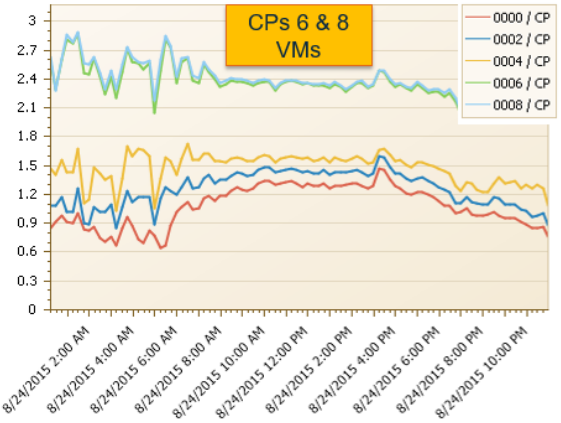

Figure 6. RNI by Logical CP Sys18 (3 VHs,2 VMs).

Figure 6 shows the extent of the negative impact to RNI that results from cache references for the VMs on system 18 routinely travelling across drawers. The gap in RNI between the two VMs (CPs 6 and 8) and the three VHs is much larger on this system with the adverse LPAR topology than it was on system 3 (from Figure 3), where the VHs and VMs were collocated on the same chip. And Figure 7 shows the improvement in RNI when CPs 6 and 8 changed from VMs to VHs was also greater, because it is very likely that all five of those VHs are now collocated on the same chip.

[6] LPAR topology data is provided by the SMF Type 99 Subtype 14 record. As opposed to some SMF 99 subtypes which can generate overwhelming volumes, the volume of subtype 14 data is very manageable, one record per logical CP every five minutes, making this another data source that warrants collection and analysis.

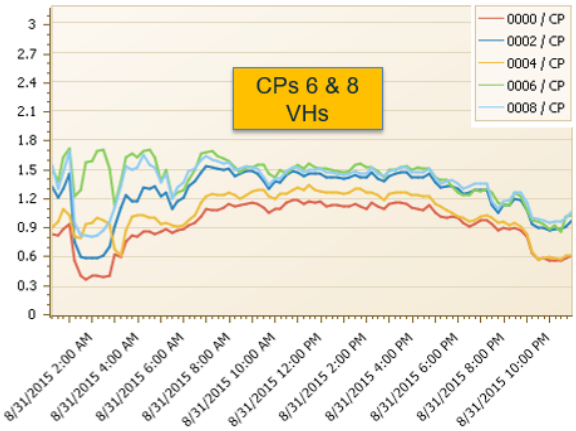

Figure 7. RNI by Logical CP Sys18 (5 VHs).

Additional detailed analysis in Figure 8 breaking out the waiting cycles of CPI by the various causes shows the impact of all that far slower off-drawer access on system 18's CPI. Note the significant increases in cycles spent on off-drawer accesses for system 18 for L3 cache (gray), L4 cache (light blue), and memory (dark blue). The cumulative impact of the L3 and L4 off-drawer accesses added more than an extra half a cycle (0.57) per instruction to system 18.

Figure 8. Estimated Impact Cache Misses.

Drilling further into this cache miss data on system 18 by logical CP (Figure 9) demonstrates how those off-drawer accesses predominantly occurred on the two VM logical CPs, CPs 6 and 8, as expected. Note that work executing on those two VMs required two full additional cycles per instruction compared to work executing on the VHs.

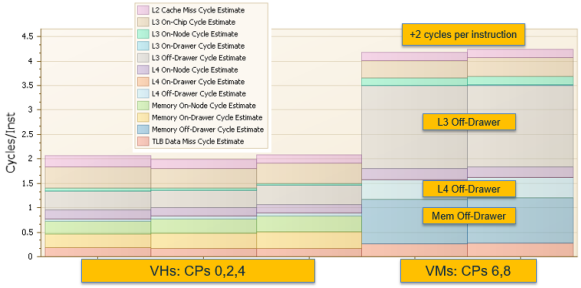

Figure 9. Cache Miss Impact by Logical CP Sys18.

When there is a significant disparity between the RNIs for VMs and VHs on the same system, an investigation of the LPAR topology is warranted. This analysis also highlights another benefit of having more work executing on VHs. In addition to avoiding cross-LPAR contention for cache from other LPARs (as identified earlier), since PR/SM is very likely to collocate VHs, the incremental work now executing on these VHs is unlikely to be subject to the cross-node or cross-drawer access times often experienced on VMs.

A second LPAR topology scenario involves an entirely different kind of opportunity. In the use case depicted in Figure 10, PR/SM configured all ten VH CPs from two Production LPARs in the same node on a single drawer. The outcome of this configuration was that the two LPARs were sharing the 480 MB L4 local cache of that single node between them.

Drawer 2 Node 1	Chip 1	Chip 2
Before: 1 node 480 MB L4L cache	SY07 VH00	SY17 VH00
	SY07 VH01	SY17 VH01
	SY07 VH02	SY17 VH02
	SY07 VH03	SY17 VH03
	SY07 VH04	SY17 VH04

Figure 10. LPAR Topology – Shared Node.

Increasing the memory allocation for both LPARs caused them to exceed the memory available in a single drawer and forced PR/SM to assign them to separate drawers and thus separate nodes. Now that these two LPARs had the L4 local cache of two nodes available to them (960 MB), we would expect more misses to be resolved from there. (The z13 RNI formula is repeated here to highlight how much lower the weighting is for L4 Local accesses than L4 Remote and Memory.)

2.3 * (0.4*L3P + 1.6*L4LP + 3.5*L4RP + 7.5*MEMP) / 100

	L4LP	L4RP	MEMP	RNI
Before	4.38%	0.91%	4.85%	1.48
After	5.84%	0.59%	3.82%	1.31

Figure 11. Impact of LPAR Topology Change.

Figure 11 shows that an additional 1.5% of L1 misses were now sourced from L3 or L4 local

node cache (L4LP in Green), reducing the frequency of accesses from L3 or L4 remote cache and especially from memory. This resulted in an 11.5% reduction in RNI and corresponding 6% increase in effective capacity. Note that this change did not require deploying any additional hardware, but instead involved utilizing the existing hardware more effectively.[7]

5. Maximizing Work Executing on VHs

A second way to reduce RNI is to maximize work executing on VHs. The two variables determining the Vertical CP configuration are (1) LPAR weights and (2) the number of Physical CPs. Each of these will be considered in detail.

5.1. Setting LPAR Weights

There are several ways to maximize work on VHs through setting LPAR weight values.[8] One is to adjust LPAR weights to increase the number of VHs for high CPU LPARs that currently have a significant workload executing on VMs and possibly even VLs. In Figure 12, a small weight change on a large LPAR changing the LPAR weight percentage from 70 to 71 percent increased the number of VHs from seven to eight.

12 CPs	% Shr	CP Shr	VHs	VMs	VLs	RNI
Before	70%	8.4	7	2	3	
After	71%	8.52	8	1	3	-2%

Figure 12. LPAR Weight Change on Large LPAR.

This resulted in a measured decrease in RNI of 2% for a given measurement interval, which correlated to a CPU reduction of 1%. 1% less CPU on a large LPAR can translate into a meaningful reduction in MLC software expense, especially when compared with the level of effort required to identify and implement this type of change. Benefits from tuning LPAR weights typically produce single digit percentage improvements as in this case, but there can be larger opportunities as we will observe.

A second way to increase work executing on VHs involves tailoring LPAR weights to increase the overall number of VHs assigned by PR/SM on a processor. The LPAR weight configuration of 30/30/20/20% in Figure 13 appears routine, but unfortunately on a z13 it results in zero VHs.[9]

6 CPs	% Shr	CP Shr	VHs	VMs	VLs
LP01	30%	1.8	0	2	2
LP02	30%	1.8	0	2	2
LP03	20%	1.2	0	2	1
LP04	20%	1.2	0	2	1

Figure 13. LPAR weight configuration with 0 VHs.

As Figure 14 shows, relatively small LPAR weight changes could increase the number of VHs from zero to two, and increase the amount of work eligible to execute on VHs from 0% up to 33%. On a comparable system in this environment, the RNI for work executing on VHs was 20% lower than work on VMs, so it is likely this change would significantly reduce CPU consumption on this processor.

6 CPs	% Shr	CP Shr	VHs	VMs	VLs
LP01	34%	2.04	1	2	1
LP02	34%	2.04	1	2	1
LP03	16%	0.96	0	1	2
LP04	16%	0.96	0	1	2

Figure 14. Adjusted LPAR weights creating 2 VHs.

If the characteristics of the workloads on LPARs change between shifts, automating LPAR weight changes corresponding to those shifts may be another way to increase the workload executing on VHs. And finally, fewer, larger LPARs may be a configuration option to increase the size of the workload executing on VHs.

[7] IBM specialists assisting us at my former employer identified this opportunity and the workaround to create the desired topology, and measured the increase in effective capacity (which aligned very well with the RNI rule of thumb).

[8] The specifics of how PR/SM determines Vertical CP assignments based on LPAR weights and the number of physical CPs is beyond the scope of this paper [for details see Havekost2017].

[9] This is the configuration after an IBM June 2016 z13 microcode change [see Snyder2016 for details].

5.2. Increasing Number of Physical CPs

Along with LPAR weights, the second variable in the equation PR/SM uses to assign VHs is the number of physical CPs. We will now consider the options and considerations for maximizing work executing on VHs through increasing the number of physical CPs.

One way to increase the number of physical CPs is to utilize sub-capacity processor models. These models provide additional physical CPs for the same MSU capacity, which translates into more VHs without incurring additional hardware expense.

If single engine speed requirements or other considerations in your environment do not require full capacity models, sub-capacity models could be selected when upgrading to new generation technologies. This approach adds physical CPs, along with the associated L1 and L2 processor cache, without incurring additional hardware expense. This would result in more VHs and thus greater processor cache efficiency.

Total	zEC12-711	z13-710	z13-617	z13-525
MSUs	1593	1632	1610	1603

Figure 15. z13 models with similar capacity ratings.

Figure 15 shows various z13 models that have similar MSU capacity ratings as a zEC12-711. Opting for sub-capacity models in this example would result in seven to fifteen more physical CPs than the full capacity z13-710, significantly increasing the available processor cache and workload executing on VHs. Selecting a sub-capacity model would likely result in additional realized capacity due to the CPU savings resulting from increased processor cache efficiency, even though the rated capacity would be comparable to the z13-710.

A second way to increase the number of physical CPs is to install or deploy additional capacity, leveraging a one-time hardware expense to achieve greater, recurring MLC software expense savings.

Traditional mainframe capacity planning has been predicated on the assumption that running mainframes at high utilizations is the most cost-effective way to operate. The results of this processor cache analysis and the case study we will consider shortly challenge that approach. They indicate that in many cases significant reductions in MSU consumption and thus MLC software costs may be achieved by operating at lower utilizations due to the increased impact that cache effectiveness has on z13 processors.

Understandably, developing the business case to acquire additional hardware capacity may be challenging. But many sites would not require that business case because they already have a business practice of pre-purchasing additional capacity they do not immediately deploy.[10] But even these sites tend to deploy that previously acquired capacity in a "just in time" manner, rather than reaping the recurring benefits of MLC software savings by deploying surplus capacity as soon as it is acquired.

In either case, whether acquiring additional hardware capacity or deploying previously acquired capacity, the framework for reevaluating this approach is that in most mainframe environments software represents a much larger expense than hardware, with MLC software typically constituting the single largest expense line item. If a one-time hardware acquisition expense achieves software savings which are realized on a recurring basis year-after-year, there may be a strong business case for acquiring or deploying hardware capacity that significantly exceeds the business workload requirements.

One reservation frequently expressed about this approach is that Independent Software Vendor (ISV) licenses are a barrier to deploying surplus capacity, because many ISV contracts are capacity-based and not usage-based. Capacity-based ISV contracts can severely limit your operational flexibility until they are renegotiated to become usage-based. A proactive initiative to renegotiate any capacity-based ISV contracts can place you in the enviable position of having the flexibility to configure your environment in the most cost-effective manner going forward.[11]

[10] Reasons for this practice include negotiating a volume discount or long-term lease, avoiding the procurement effort involved with frequent acquisitions, or acquiring capacity required for seasonal peaks, which is then typically deactivated for the remainder of the year.

[11] This ISV contract renegotiation initiative was successfully achieved at my previous employer.

The following use case shows the potential magnitude of the impact of deploying hardware on processor cache efficiency. The case involves a high RNI workload and hardware capacity that had been previously acquired but not deployed.

Figure 16 is a weekly tracking chart that compared the Production business online workload (red line) versus CPU consumption (blue line). For the last several months in a zEC12 environment, these values tracked very closely. But after the four processors were upgraded to z13-711 models, CPU consumption far exceeded business transaction workload.

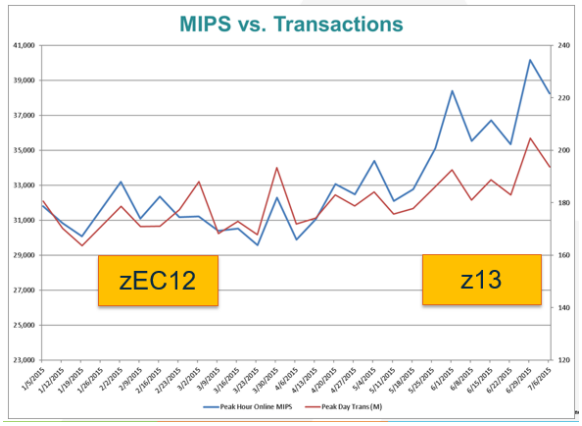

Figure 16. CPU impact of z13 implementations.

As stated earlier, the cycle speed on z13 processors is 10% slower than the zEC12, yet rated capacity is 10% higher. This high RNI (1.6) workload did not achieve the improvement in cache effectiveness the z13 relies upon to achieve this increased capacity, resulting in a capacity shortfall of 4000 MIPS versus the zEC12 configuration.

When additional capacity (z13-716 models) was deployed that enabled most of the workload to execute on VHs, MIPS consumption was reduced dramatically (see Figure 17). This resulted in 9000 MIPS savings versus the prior z13-711 configuration for equivalent business workloads.

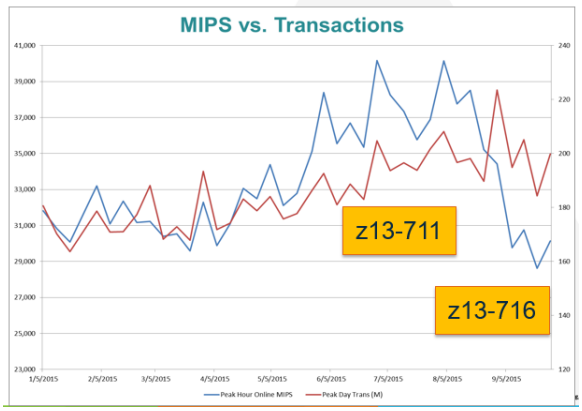

Figure 17. CPU impact of z13-716 upgrades.

Ultimately all the previously acquired capacity was deployed through upgrades to z13-726 models, resulting in a 13,000 MIPS reduction from the z13-711 configuration (see Figure 18). The primary factor driving this final round of savings was not processor cache efficiency but a reduction in the MSU/CP ratio, which we will now consider.

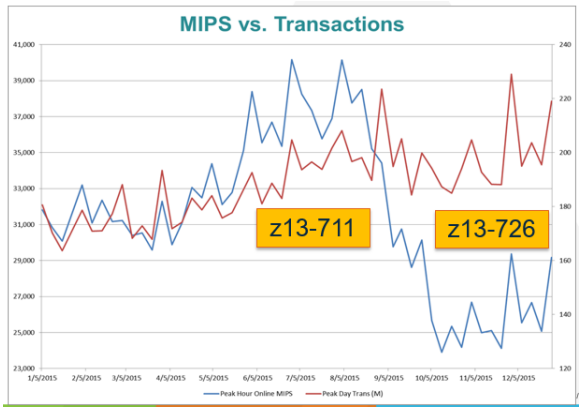

Figure 18. CPU impact of z13-726 upgrades.

6. MSU/CP Ratio

Another important factor impacting measured CPU consumption that adds to the business case for acquiring or deploying additional hardware relates to the multiprocessing (MP) effect. The MP effect applies to any hardware environment, and reflects the fact that adding cores (CPs) increases the overhead required to manage the interactions between physical processors and shared compute resources.

To account for the typical MP effect when running at relatively high utilizations, the MSU/CP ratios in IBM's processor ratings are not linear, but decrease significantly as more CPs are added as Figure 19 shows [Kyne2015].

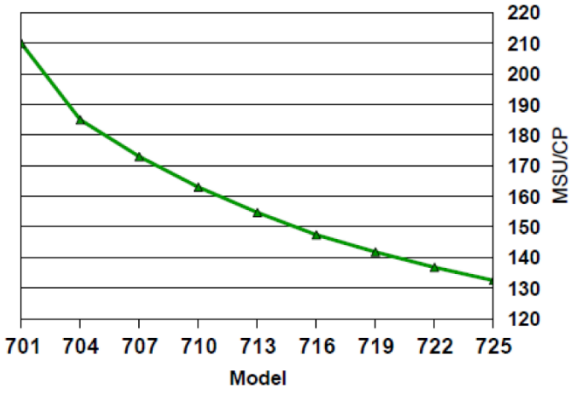

Figure 19. MSU/CP ratios for various z13 models.

But if the business workload remains the same when CPs are added, overall processor utilization will decrease, and additional overhead from the MP effect is likely to be negligible. In that case, a lower MSU/CP rating translates directly into lower MSU consumption for the same workload. Figure 20 compares the MSU/CP ratios for the four processor models in the previous use case.

CEC	MSUs/CP	vs zEC12-711	vs z13-711
zEC12-711	144.8		-9.7%
z13-711	160.4	10.7%	
z13-716	147.4	1.8%	-8.1%
z13-726	131.3	-9.3%	-18.1%

Figure 20. MSU/CP ratios from use case.

The Orange value shows the 10%+ increase associated with the original z13-711 implementation, one that the High RNI workload fell far short of achieving. The Green value shows the decrease of 18% in the MSU/CP rating for the z13-726 from the z13-711. Or expressed another way, a workload that would generate 1000 MSUs on a 711 processor would generate only 819 MSUs when run on a 726, solely due to the reduced MSU/CP rating. This is independent of the CPU savings from improved processor cache efficiency that would also accompany this change.

This means that there are two significant benefits of deploying additional hardware capacity while running at lower utilizations that combine to create even larger savings. First, less CPU is consumed due to operating efficiencies from more effective use of processor cache. And second, the CPU that is consumed translates into fewer MSUs, due to the decrease in the processor MSU/CP ratings.

7. Summary and conclusions

Processor cache performance plays a more prominent role than ever before in the capacity delivered by z13 processors. And this is a trend that is likely to continue with future processor models, due to the engineering challenges associated with increasing cycle speeds.

Since unproductive cycles spent waiting to stage data into L1 cache typically represent one third to one half of overall CPU, it is incumbent on the mainframe performance analyst to have clear visibility into the key metrics and to leverage those metrics to optimize processor cache. This paper presented methods to reduce RNI and CPU consumption, along with case studies confirming the effectiveness of these methods in real-life environments.

Another trend that is very likely to continue is that software expenses will consume a growing percentage of the overall mainframe budget, while hardware costs will represent an ever-smaller percentage. Considering these factors, it is indeed time for a thorough re-examination of the traditional assumption of mainframe capacity planning that running mainframes at high utilizations is the most cost-effective way to operate.

References

[Sinram2015] Horst Sinram, *z/OS Workload Management (WLM) Update for IBM z13, z/OS V2.2 and V2.1*, SHARE Session #16818, March 2015.

[Kyne2017] Frank Kyne, Todd Havekost, and David Hutton, *CPU MF Part 2 – Concepts*, Cheryl Watson's Tuning Letter 2017 No. 1.

[Houtekamer2016] Gilbert Houtekamer, Wim Oudshoorn, and Todd Havekost, *Getting More MIPS out of your processors*, CMG Proceedings, 2016.

[Havekost2016] Todd Havekost, *Achieving Significant Capacity Improvements on the IBM z13: User Experience*, Share Session #18345, March 2016.

[Havekost2017] Todd Havekost, *Beyond Capping: Reduce IBM z13 MLC with Processor Cache Optimization*, Share Session #20127, March 2017.

[Snyder2016] Bradley Snyder, *z13 HiperDispatch – New MCL Bundle Changes Vertical CP Assignment for Certain LPAR Configurations*, IBM TechDoc 106389, June 2016.

[Kyne2015] Frank Kyne, *A Holistic Approach to Capacity Planning*, Cheryl Watson's Tuning Letter 2015 No. 4.