POWER7 Virtualization

Best Practice Guide

Version 3.0

Sergio Reyes, Mala Anand and Pete Heyrman
STG Power System Performance and Development

Preface

This document is intended to address POWER7 PowerVM best practices to attain best LPAR performance. This document by no means covers all the PowerVM best practices so this guide should be used in conjunction with other PowerVM documents.

The following is a list of IBM reference and documents that are good references:

- AIX on POWER – Performance FAQ
  http://www.ibm.com/common/ssi/cgi-bin/ssialias?subtype=WH&infotype=SA&appname=STGE_PO_PO_USEN&htmlfid=POW03049USEN&attachment=POW03049USEN.PDF
- IBM System p Advanced POWER Virtualization Best Practices Redbook:
  http://www.redbooks.ibm.com/redpapers/pdfs/redp4194.pdf
- Virtualization Best Practice:
  http://www.ibm.com/developerworks/wikis/display/virtualization/Virtualization+Best+Practice
- Configuring Processor Resources for System p5 Shared-Processor Pool Micro-Partitions:
  http://www.ibmsystemsmag.com/aix/administrator/systemsmanagement/Configuring-Processor-Resources-for-System-p5-Shar/
- An LPAR Review:
  http://www.ibmsystemsmag.com/aix/administrator/lpar/An-LPAR-Review/
- Virtualization Tricks:
  http://www.ibmsystemsmag.com/aix/trends/whatsnew/Virtualization-Tricks/
- A Comparison of PowerVM and x86-Based Virtualization Performance:
  http://www-03.ibm.com/systems/power/software/virtualization/whitepapers/powervm_x86.html
- IBM Integrated Virtualization Manager:
  http://www-03.ibm.com/systems/power/hardware/whitepapers/ivm.html
- Achieving Technical and Business Benefits through Processor Virtualization:
  http://www.ibm.com/common/ssi/fcgi-bin/ssialias?infotype=SA&subtype=WH&appname=STGE_PO_PO_USEN&htmlfid=POL03027USEN&attachment=POL03027USEN.PDF

1.1 Introduction

PowerVM hypervisor as well as AIX operating system (AIX 6.1 TL 5 and above versions) on Power7 have implemented enhanced affinity in a number of areas to achieve optimized performance for workloads running in a virtualized shared processor logical partition (SPLPAR) environment. Customers by leveraging the best practice guidance described in this document can attain optimum application performance in a shared resource environment. This document covers best practices in the context of Power7 systems therefore this section can be used as an addendum to other PowerVM best practice documents.

1.2 Virtual Processors

A virtual processor is a unit of virtual processor resource that is allocated to a partition or virtual machine. PowerVM™ hypervisor can map a whole physical processor core or can time slice a physical processor core

PowerVM Hypervisor time slices Micro partitions on the physical CPU's by dispatching and un-dispatching the various virtual processors for the partitions running in the shared pool.

If a partition has multiple virtual processors, they may or may not be scheduled to run simultaneously on the physical processors

Partition entitlement is the guaranteed resource available to a partition. A partition that is defined as capped, can only consume the processors units explicitly assigned as its entitled capacity. An uncapped partition can consume more than its entitlement but is limited by a number of factors:

- Uncapped partitions can exceed their entitlement if there is unused capacity in the shared pool, dedicated partitions that share their physical processors while active or inactive, unassigned physical processors, COD utility processors and such.
- If the partition is assigned to a virtual shared processor pool, the capacity for all the partitions in the virtual shared processor pool may be limited
- The number of virtual processors in an uncapped partition throttles on how much CPU it can consume. For example:
    - An uncapped partition with 1 virtual CPU can only consume 1 physical processor of CPU resource under any circumstances
    - An uncapped partition with 4 virtual CPUs can only consume 4 physical processors of CPU
- Virtual processors can be added or removed from a partition using HMC actions. (Virtual processors can be added up to Maximum virtual processors of a LPAR and virtual processors can be removed up to Minimum virtual processors of a LPAR)

1.2.1 Sizing/configuring virtual processors

The number of virtual processors in each LPAR in the system should not "exceed" the number of cores available in the system (CEC/framework) or if the partition is defined to run in specific virtual shared processor pool, the number of virtual processors should not exceed the maximum defined for the specific virtual shared processor pool. Having more virtual processors configured than can be running at a single point in time does not provide any additional performance benefit and can actually cause additional context switches of the virtual processors reducing performance.

If there are sustained periods of time where there is sufficient demand for all the shared processing resources in the system or a virtual shared processor pool, it is prudent to configure the number of virtual processors to match the capacity of the system or virtual shared processor pool.

A single virtual processor can consume a whole physical core under two conditions:

1. SPLPAR had given an entitlement of 1.0 or more processor
2. This is an uncapped partition and there is idle capacity in the system.

Therefore there is no need to configure more than one virtual processor to get one physical core.

For example: A shared pool is configured with 16 physical cores. Four SPLPARs are configured each with entitlement 4.0 cores. To configure virtual processors, we need to consider the workload's sustained peak demand's capacity. If two of the four SPLPARs would peak to use 16 cores (max available in the pool), then those two SPLPARs would need 16 virtual CPUs. The other two peaks only up to 8 cores, those two would be configured with 8 virtual CPUs

The maximum virtual processors of a LPAR should not exceed the maximum CPU capacity available in the shared processor pool even though it is not restricted. There is no need to configure greater number virtual processors than the physical processors in the pool.

For example: A shared pool is configured with 32 physical cores. Eight SPLPARs are configured in this case each LPAR can have up to 32 virtual processors. This allows a LPAR that has 32 virtual processors to get 32 cpus if all the other lpars are not using their entitlement. Setting > 32 virtual processors is not necessary as there are only 32 CPUs in the pool.

1.2.2 Entitlement vs. Virtual processors

Entitlement is the capacity that a SPLPAR is guaranteed to get as its share from the shared pool. Uncapped mode allows a partition to receive excess cycles when there are free (unused) cycles in the system.

Entitlement also determines the number of SPLPARs that can be configured for a shared processor pool. That is, the sum of the entitlement of all the SPLPARs cannot exceed the number of physical cores configured in a shared pool.

For example: Shared pool has 8 cores, 16 SPLPARs are created each with 0.1 core entitlement and 1 virtual CPU. We configured the partitions with 0.1 core entitlement since these partitions are not running that frequently. In this example, the sum of the entitlement of all the 16 SPLPARs comes to 1.6 cores. The rest of 6.4 cores and any unused cycles from the 1.6 entitlement can be dispatched as uncapped cycles.

At the same time keeping entitlement low when there is capacity in the shared pool is not always a good practice. Unless the partitions are frequently idle or there is plan to add more partitions, the best practice is that the sum of the entitlement of all the SPLPARs configured should be close to the capacity in the shared pool. Entitlement cycles are guaranteed, so while a partition is using its entitlement cycles, the partition is not pre-empted, whereas a partition can be preempted when it is dispatched to use excess cycles. Following this practice allows the hypervisor to optimize the affinity of the partition's memory and processors and also reduces unnecessary preemptions of the virtual processors.

1.2.3 Matching entitlement of a LPAR close to its average utilization for better performance

The aggregate entitlement (min/desired processor) capacity of all LPARs in a system is a factor in the number of LPARs that can be allocated. The minimum entitlement is what is needed to boot the LPARs, however the desired is what an LPAR will get if there are enough resources available in the system. The best practice for LPAR entitlement would be to match the entitlement capacity to average utilization and let the peak addressed by additional uncapped capacity.

The rule of thumb would be setting entitlement close to average utilization for each of the LPAR in a system, however there are cases where a LPAR has to be given higher priority compared to other LPARs in a system, this rule can be relaxed. For example if the production and non-production workloads are consolidated on the same system, production LPARs would be preferred to have higher priority over non-production LPARs. In which case, in addition to setting higher weights for uncapped capacity, the entitlement of the production LPARs can be raised while reducing the entitlement of non-production LPARs. This allows these important production LPARs to have better partition placement (affinity) and these LPARs will have additional entitled capacity so not to rely solely on uncapped processing. At the same time if production SPLPAR is not using their entitled capacity, then that capacity can be used by non-production SPLPAR and the non-production SPLPAR will be pre-empted if production SPLPAR needs its capacity.

1.2.4 When to add additional virtual processors

When there is sustained need for a shared LPAR to use additional resources in the system in uncapped mode, increasing virtual processors are recommended.

1.2.5 How to estimate the number of virtual processors per uncapped shared LPAR

The first step would be to monitor the utilization of each partition and for any partition where the average utilization is ~100%, and then add one virtual processors. That is, use the capacity of the already configured virtual processors before adding more. Additional virtual processors are going to run concurrently if there are enough free processors available in the shared pool.

If the peak utilization is well below 50% mark, then there is no need for additional virtual processors. In this case, look at the ratio of virtual processors to configured entitlement and if the ratio is > 1, then consider reducing the ratio. In any case if there are too many virtual processors configured, AIX can "fold" those processors so that the workload would run on fewer virtual processors to optimize virtual processor performance.

For example if a SPLPARs has given a CPU entitlement of 2.0 cores and 4 virtual processors in an uncapped mode then the hypervisor could dispatch the virtual processors to 4 physical cores concurrently if there are free cores available in the system. The SPLPARs leverages unused cores and the applications can scale up to 4 cores. However if the system does not have free cores then hypervisor would have to dispatch 4 virtual processors on 2 cores so the concurrency is limited to 2 cores. In this situation, each virtual processor is dispatched for reduced time slice as 2 cores are shared across 4 virtual processors. This situation could impact performance; therefore AIX operating system processor folding support may be able to reduce to number of virtual processors being dispatch such that only 2 or 3 virtual processors are dispatched across the 2 physical.

1.2.6 Virtual Processor Management - Processor Folding

AIX operating system monitors the utilization of each virtual processor and aggregate utilization of a SPLPAR, if the aggregate utilization goes below 49%, AIX will start folding down the virtual CPUs so that fewer virtual CPUs will be dispatched. This has the benefit of a virtual CPUs running longer before getting pre-empted which helps to improve performance. If a virtual CPU gets lower dispatch time slice, then more workloads are time-sliced on to the processor which can cause higher cache misses.

If the aggregate utilization of a SPLPAR goes beyond 49%, AIX will start unfolding virtual CPUs so that additional processor capacity can be given to the SPLPAR. Virtual processor management dynamically adopts number of virtual processors to match the load on a SPLPAR. This threshold (vpm_fold_threshold) of 49% represents the SMT thread utilization starting AIX 6.1 TL6, prior to that vpm_fold_threshold (which was set to 70%) represented the core utilization.

With a vpm_fold_threshold value of 49%, the primary thread of a core is fully utilized before unfolding another virtual processor to consume another core from the shared pool on POWER7

systems. If free cores are available in the shared processor pool, then unfolding another virtual processor would result in the LPAR getting another core along with its associated caches. As a result now the SPLPAR can run on two primary threads of two cores instead of two threads (primary and secondary) on the same core. Workload running on two primary threads of two cores would have higher performance than the workload running on primary and secondary threads of a single core. AIX virtual processor management default policy aims at achieving higher performance so it unfolds virtual processors using only the primary thread until all the virtual processors are unfolded then it starts leveraging SMT threads.

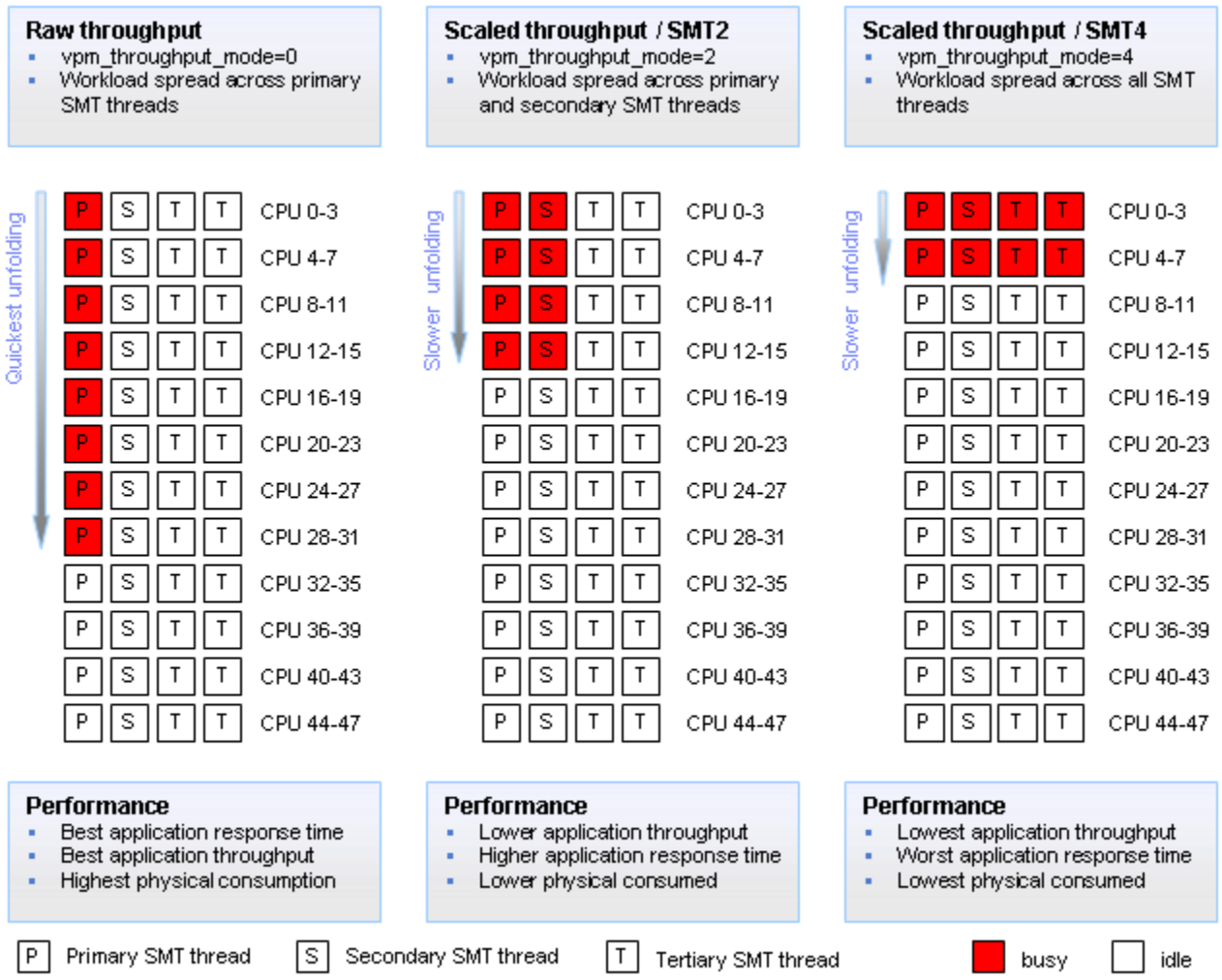
If the system is highly utilized and there are no free cycles in the shared pool, when all the SPLPARs in the system try to get more cores by unfolding additional virtual processors and use only the primary of thread of each core and leverage other three threads only when load increases, the fact that all the virtual processors are unfolded lead to hypervisor time slicing the physical cores across multiple virtual processors. This would impact performance of all the SPLPARs as time slicing a core across multiple virtual processors would increase cache misses, context switch cost etc., In such a situation unfolding fewer virtual processors so that a physical core is either not shared or shared across fewer virtual processors would improve overall system performance.

AIX *levels 61TL8 and 71TL2* provides a new dispatching feature through the schedo tuneable vpm_throughput_mode, which allows greater control over workload dispatching. There are four options, 0,1,2,4 that can be set dynamically. Mode0 and 1 cause the AIX partition to run in raw throughput mode, and modes2 and 4 switch the partition into scaled throughput mode.

The default behavior is raw throughput mode, same as legacy versions of AIX. In raw throughput mode (vpm_throughput_mode=0), the workload is spread across primary SMT threads.

Enhanced raw throughput mode (vpm_throughput_mode=1), behaves similar to mode0 by utilizing primary SMT threads; however, it attempts to lower CPU consumption by slightly increasing the unfold threshold policy. It typically results in a minor reduction in CPU utilization. Please note that this is different to changing the vpm_fold_threshold tunable.

The new behavior choices are for scaled throughput mode SMT2(vpm_throughput_mode=2) and scaled throughput mode SMT4(vpm_throughput_mode=4). These new options allow for the workload to be spread across 2 or 4 SMT threads, accordingly. The throughput modes determine the desired level of SMT exploitation on each virtual processor core before unfolding another core. A higher value will result in fewer cores being unfolded for a given workload.



The AIX scheduler is optimized to provide best raw application throughput on POWER7 / POWER7+. Modifying the AIX dispatching behavior to run in scaled throughput mode will have a performance impact that varies based on the application.

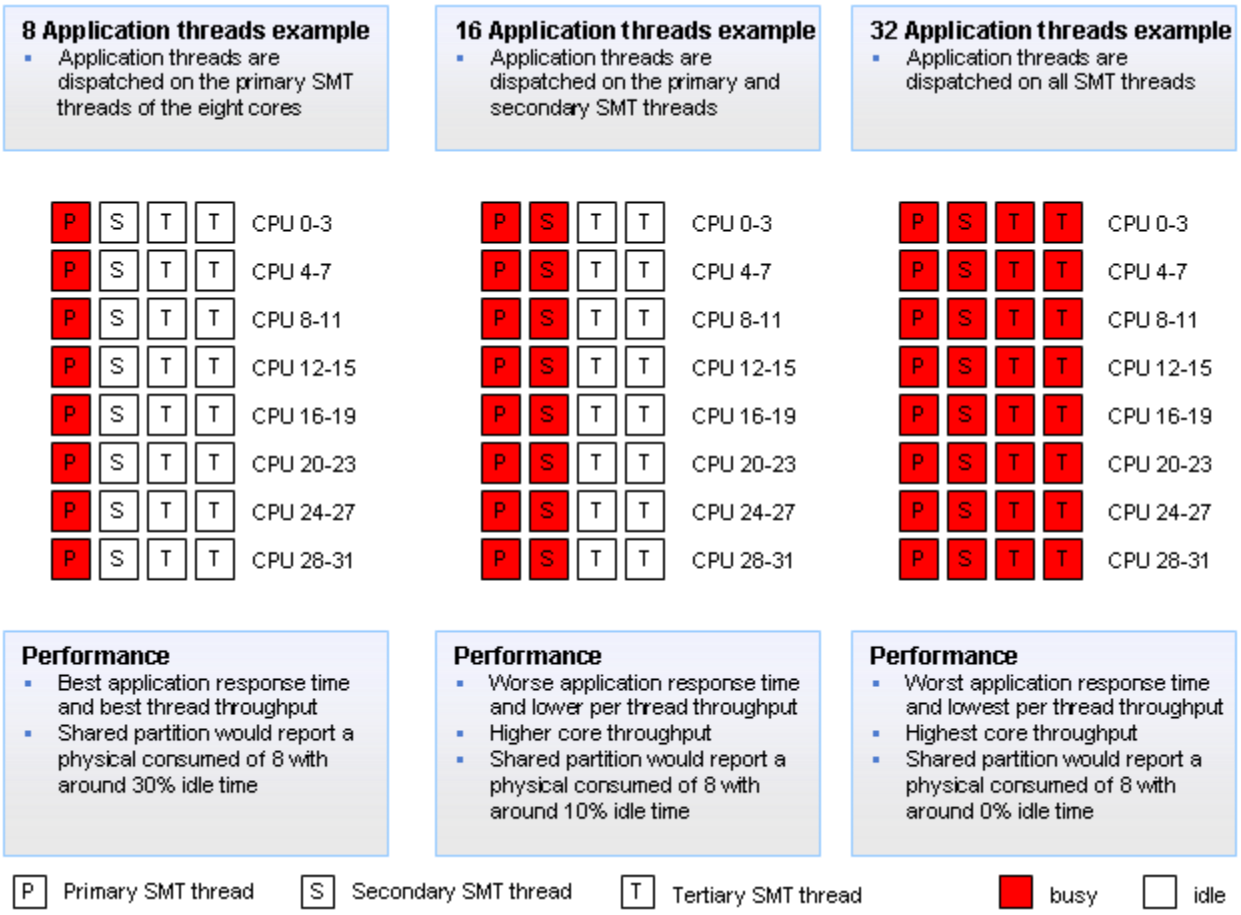
There is a correlation between the aggressiveness of the scaled throughput mode and the potential impact on performance; a higher value increases the probability of lowering application throughput and increasing application response times.

The scaled throughput modes; however, can have a positive impact on the overall system performance by allowing those partitions using the feature to utilize less CPU (unfolding fewer virtual processors), and thus reducing the overall demand on the shared processing pool in a over provisioned virtualized environment.

The tune-able is dynamic and does not require a reboot; which simplifies the ability to revert the setting if it does not result in the desired behavior. It is recommended that any experimentation of the tune-able be done on non critical partitions such as development or test before being considered for production environments. For example, a critical database SPLPAR would benefit most from running in the default raw throughput mode, utilizing more cores even in a highly contended situation to achieve the best performance; however, the development and test SPLPARs can sacrifice by running in a scaled throughput mode, and thus utilizing fewer virtual processors by leveraging more SMT threads per core.. However when a LPAR utilization

reaches maximum level, AIX dispatcher would use all the SMT4 threads of all the virtual processors irrespective of the mode settings.

The example below illustrates the behavior within a raw throughput mode environment. Notice how the application threads are dispatched on primary processor threads, and the secondary and tertiary processor threads are utilized only when the application thread count exceeds 8 and 16 accordingly.



1.2.7 Processor Bindings in Shared LPAR

With AIX 6.1 TL5 and AIX 7.1 binding processors is available to application running in a shared LPAR. An application process can be bound to a virtual processor in a shared LPAR. In a shared LPAR a virtual processor is dispatched by PowerVM hypervisor. In firmware level 730 and later, the PowerVM hypervisor maintains three levels of affinity for dispatching, core, chip and node level. By maintaining affinity at hypervisor level as well as in AIX, applications may achieve higher level affinity through processor bindings.

1.2.8 Virtual CPU Dispatch Wheel Option

The eFW7.6 firmware introduced a new option on selected server models that allows the user to alter POWER Hypervisor's dispatch window. The default setting of a 10ms dispatch window has

not changed, so the partition dispatching behavior will be equal to that of previous firmware releases.

The tuneable allows the admistrator to increase the dispatch window to 50ms. This capability allows a virtual processor to attain its guaranteed entitled share over a larger window. A 50ms dispatch window provides virtual CPUs with the ability to utilize a physical core for larger period of time, before having a context switch. The potential downside is that a virtual CPU may have to wait longer periods of time before being dispatched to a physical CPU. Partitions will attain their guaranteed entitlement regardless of the dispatch window setting; these settings alter the frequency and responsiveness of vCPU dispatching.

The 10ms dispatch window is the default setting and was chosen to be so for its responsiveness; it is best suited for most environments. Some configurations may benefit for a larger dispatch window, primarily those with multiple partitions defined to have low guaranteed entitlements of less than 1. Network intensive applications may not be best suited for the 0.5ms dispatch window, considering that delays in dispatching can introduce additional latency, and as a worst case, cause network packets to be dropped.

It is highly recommended that testing is done in a non production environment if you wish to change the default dispatch window from 10ms to 50ms.

1.3 Recommendation on page table size for LPAR

The hardware page table of a LPAR is sized based on the maximum memory size of a LPAR not what is currently assigned (desired) to the LPAR. There are some performance considerations if the maximum size is set significantly higher than the desired memory.

1. Larger page table tends to help performance of the workload as the hardware page table can hold more pages. This will reduce translation page faults. Therefore if there is enough memory in the system and would like to improve translation page faults, set your max memory to a higher value than the LPAR desired memory.
2. On the downside, more memory is used for hardware page table, this not only wastes memory but also makes the table become sparse which results in a couple of things:
    a. dense page table tends to help better cache affinity due to reloads
    b. lesser memory consumed by hypervisor for hardware page table more memory is made available to the applications
    c. lesser page walk time as page tables are small

1.4 Recommendation for placing LPAR resources to attain higher memory affinity

POWER7 PowerVM has optimized allocation of resources for both dedicated and shared partitions as each LPAR is activated. Proper planning of LPAR configuration would enhance the possibility of getting both CPU and Memory in the same domain in relation to the topology of a system.

PowerVM Hypervisor selects the required processors and memory configured for a LPAR from the system free resource pool. During this selection process hypervisor takes the topology of the system into consideration and allocates processors and memory where both resources are in close proximity as best as it can. This ensures that the workload on a LPAR would have lower latency in accessing its memory.

1.4.1 What does the SPPL option do on Power 795 system

The Shared Partition Processor Limit (SPPL) attribute provides hints to hypervisor whether to contain partitions to minimum domains or spread partitions across multiple domains. On Power 795, a book can host 4 chips totaling up to 32 cores. If SPPL is set to 32, then the maximum size of a LPAR that can be supported is 32 cores. This hint enables hypervisor to allocate both physical cores and memory of a LPAR within a single domain as much as possible. For example in a three book configuration if the desired configuration is 4 LPARs each with 24 cores, 3 of those LPARs will be contained in each of the three books and the 4th LPAR would be spread across 3 books

If SPPL is set to MAX then a partition size can exceed 32 cores. This hint helps hypervisor to maximize the interconnect bandwidth allocation by spreading LPARs across more domains as possible for larger size LPARs.

Note that the SPPL value can only be set on 795 systems that contain 4 or more processing books where TurboCore mode is disabled. If there are 3 or less processor books, the SPPL setting is controlled by the system firmware.

On a Power 795 system where the SPPL value is set to max, there is a way to configure individual partitions to still be packed into minimum number of books. This is achieved by using the HMC command line interface through the lpar_placement profile attribute on the chsyscfg command. Specifying a value of lpar_placement=1 indicates that the hypervisor should try and minimize the number of domains assigned to the lpar. Setting it to 0 is the default setting and follows the existing rules when SPPL is set to max. In the 730 level of firmware, lpar_placement=1 was only recognized for dedicated processor partitions when SPPL=MAX. Starting with the 760 firmware level, lpar_placement=1 is also recognized for shared processor partition with SPPL=MAX and systems configured to run in TurboCore mode.

1.4.2 How to determine if a LPAR is contained within a drawer or book

From an AIX LPAR, lssrad command can be used to display the number of domains a LPAR is spread.

The lssrad syntax is:

```
lssrad -av
```

If all the cores and memory are located in a single book

```
REF1  SRAD     MEM      CPU
0     0      31806.31   0-31
      1      31553.75   32-63
```

REF1 is the second level domain. In the case of Power 795 this second level domain refers to book. For Power 770/780 this second level domain refers to the drawer . SRAD refers to processor socket numbers. However "lssrad" does not report the actual physical domains (namely book numbers and chip numbers). The output of lssrad indicates that the LPAR is allocated 16 cores from two chips in the same book/drawer.

1.4.3 Optimizing Resource Placement

PowerVM when all the resources are free (an initial machine state or reboot of the CEC) will allocate memory and cores as optimal as possible. At the partition boot time, PowerVM is aware of all the LPAR configurations so, placement of processors and memory are irrespective of the order of activation of the LPARs.

However after the initial configuration, the setup may not stay static. There will be numerous operations that take place such as:

1. Reconfiguration of existing LPARs with new profiles
2. Reactivating existing LPARs and replacing them with new LPARs
3. Adding and removing resources to LPAR dynamically (DLPAR operations)

If the current placement is causing a performance issue, there are a couple procedures that can be used to improve placement of the partitions across the server.

1. If the Dynamic Platform Optimizer is available on your server, this is the most efficient and least disruptive method of improving the placement. Refer to 1.4.4 Optimizing Resource Placement – Dynamic Platform Optimizer for more information about this option.
2. When the Dynamic Platform Optimizer is not available, the most effective but also the most disruptive method to improve placement is a server reboot. The procedure to improve the placement through a server reboot is as follows:
    a. Make any profile updates that you need to make to the partitions.
    b. Power off all the defined partitions

c. Activate (power on) all the partitions specifying the profiles with the desired attributes. The partition need to be activated to at least the SMS prompt.

d. Once all partitions are simultaneously activated with the updated profiles (i.e. every partition must be active), start powering off all of the partitions.

e. Power off the server

f. Power on the server

g. Activate the partitions from the management console. Order of activation is not important.

3. If you are unable to perform a server reboot and all the processors in the system are licensed, you can improve the placement as follows:

   a. Power off all the defined partitions

   b. Use one of the following methods to free up all the resources assigned to all of the partitions:

      i. Use the HMC command chhwres to remove all processors and memory from all of the partitions. The command syntax to remove the memory is "chhwres –r mem –m <system name> -o r –q <number of Mbtyes> --id <lpar id>". The command syntax to remove processor units from a shared processor partition is "chhwres –r proc –m <system name> -o r –procunits <number> --id <lpar id>". The management console can display/report the resources assigned. Before proceeding to the next step, confirm that no resources are assigned to any partition.

      ii. Create a partition and indicate the partition profile should own all of the resources in the system. Activate this partition with the all resource profile to at least SMS menu. Power off this partition and then delete this partition.

   c. Activate the partitions in order of importance. The largest, most important partition should be activated first, next most important partition second and so on.

If you wish to try to improve the placement of a single partition, power off the partition, use the chhwres command as described above to remove all the processor and memory resources from the partition and then activate the partition with desired profile. The improvement (if any) in the placement will depend on a variety of factors such as which processors are free, which memory is free, what licensed resources are available, configuration of other partitions and so on.

Fragmentation due to frequent movement of memory or processors between partitions can be avoided through proper planning ahead of the time. DLPAR actions can be done in a controlled way so that the performance impact of resource addition/deletion will be minimal. Planning for growth would help to alleviate the fragmentation caused by DLPAR operations. Knowing the LPARs that would need to grow or shrink dynamically and placing them with LPARs that can tolerate nodal crossing latency (less critical LPARs) would be one approach to handle the changes of critical LPARs dynamically. In such a configuration, when growth is needed for the critical LPAR the resources assigned to the non-critical LPAR can be reduced so that critical LPAR can grow.

1.4.4 Optimizing Resource Placement – Dynamic Platform Optimizer

In firmware 760 level and later, on select Power servers, a feature is available called the Dynamic Platform Optimizer. This optimizer automates the previously described manual steps to improve placement. The following functions are available on the HMC command line:

"lsmemopt –m <system_name> -o currscore" will report the current affinity score for the server. The score is a number in the range of 0-100 with 0 being poor affinity and 100 being perfect affinity. Note that the affinity score is based on hardware characteristics and partition configurations so a score of 100 may not be achievable.

"lsmemopt –m <system_name> -o calcscore" will report the potential score that could be achieved by optimizing the system with the Dynamic Platform Optimizer. Again, a calculated score of 100 may not be possible. The scoring is meant to provide a gauge to determine if running the optimizer is likely to provide improved performance. For example, if the current score is 85 and the calculated score is 90, running the optimizer may not have a noticeable impact on overall systems performance. The amount of gain from doing an optimization is dependent on the applications running within the various partitions and the partition placement.

Also note that this is a system-wide score that reflects all of the resources assigned to all of the partitions. The placement of individual partitions contribute to the system wide score relative to the amount of resources assigned to the partition (large processor/memory partition contributes more to the system wide score than small partition). Making any configuration change (even activating a partition that wasn't assigned resources previously) can change the overall score.

"optmem –m <system_name> -o start –t affinity" will start the optimization for all partitions on the entire server. The time the optimization takes is dependent upon the current placement of partitions, the overall amount of processor and memory that need to be moved, the amount of CPU cycles available to run this optimization and so on.

"lsmemopt –m <system_name>" displays the status of the optimization as it progresses.

"optmem –m <system_name> -o stop" will end an optimization before it has completed all the movement of processor and memory. This can result in affinity being poor for some partitions that were not completed.

There are additional parameters available on these commands that are described in the help text on the HMC command line interface (CLI) commands. One option on the optmem command is the exclude parameter (-x or –xid) which is a list of partition that should not be optimized (left as is with regards to memory and processors). The include option (-p or –id) is not a list of partitions to optimizes as it may appear, it is a list of partitions that are optimized first followed by any unlisted partitions and ignoring any explicitly excluded partitions. For example, if you have partition ids 1-5 and issue "optmem –m myserver –o start –t affinity –xid 4 –id 2", the

optimizer would first optimize partition 2, then from most important to least important partitions 1, 3 and 5. Since partition 4 was excluded, its memory and processors remain intact.

Some functions such as dynamic lpar and partition mobility cannot run concurrently with the optimizer. If one of these functions is attempted, you will receive an error message on the MC indicating the request did not complete.

To run the optimizer there must be some unlicensed memory installed or available licensed memory. The more free memory (or unlicensed memory) available, the faster the optimization will complete. Also, the busier CPUs are on the system, the longer the optimization will take to complete as the background optimizer tries to minimize its effect on running LPARs. LPARs that are powered off can be optimized very quickly as the contents of their memory does not need to be maintained. The Dynamic Platform Optimizer will spend the majority of the time copying memory from the existing affinity domain to the new domain. When performing the optimization, the performance of the partition will be degraded and overall there will be a higher demand placed on the processors and memory bandwidth as data is being copied between domains.

When the optimizer completes the optimization, the optimizer can notify the operating systems in the partitions that physical memory and processors configuration have been changed. Partitions running on AIX 7.1 TL2 (or later), AIX 6.1 TL8 (or later), VIOS 2.2.2.0 and IBM i 7.1 PTF MF56058 have support for this notification. For older operating system versions that do not support the notification, the dispatching, memory management and tools that display the affinity can be incorrect. For dispatching and memory management, for some configurations, the performance may actually degrade after running the optimizer even though the partition has better affinity because the operating system is making decisions on stale information. For these older versions, a reboot of the partition will refresh the affinity. Another option would be to use the exclude option on the optmem command to not change the affinity of partitions with older operating system levels. Rebooting partitions is usually less disruptive than the alternative of rebooting the entire server.

1.4.5 Affinity Groups

In PowerVM Firmware level 730 and later, support was added that can be used to place multiple LPARs (allocate resources) within a single chip, drawer or book. The maximum size of the affinity group is tied directly to the hardware configuration. For example, on Power 795 with 32 cores in a book, the total physical core resources of an affinity group should not exceed 32 cores or the memory physical contained within a book.

This affinity group feature can be used in multiple situations:

1. LPARs that are dependent or related, such as server and client, application server and database server, could be grouped so that they can reside in the same book.

2. Affinity groups can be created large enough such that its forces the assignment of LPARs to be in different books. For example, if you have a 2 book system and the total resources (memory and processors) assigned to the two groups exceed the capacity of a single book, these two groups will be forced to be placed in separate book. A simple example of this is if there is a group of partitions that totals 14 cores and a second group that totals 20 cores, since these groups exceed the 32 cores in a 795 book the groups will be placed in different books.
3. If a pair of LPAR are created with the intent of one being failover to another partition, assuming that one partition fails, the other partition which is placed in the same node, if both are in the same affinity group, would use all the resources freed up from the failed LPAR.

The following HMC CLI command adds or removes a partition from an affinity group:

```
chsyscfg -r prof -m <system_name> -i name=<profile_name>
lpar_name=<partition_name>,affinity_group_id=<group_id>
```

where `group_id` is a number between 1 and 255 (255 groups can be

defined - `affinity_group_id=none` removes a partition from the group.

When defining affinity groups containing shared processors, the sum total of the processor units should be in full processor units. For example, if you configured two groups each with 8.50 processing units, the total processors in the shared pool would be the sum of the entitlement or 17 in this case. This is going to result in 8 processor in one affinity domain and 9 processors in the other affinity domain. In this situation, round up each affinity group. Change the entitlement of the partitions in the group such that the total entitlement adds up to 9.0 per group so the processors can be fully allocated from a single domain.

When the hypervisor places resources at frame reboot or when the dynamic platform optimizer is started, the hypervisor first places all the LPARs in group 255, then the LPARs in group 254 and so on. Because of this, the most important partitions with respect to affinity should be placed in the highest configured group.

1.4.6 Lpar_placement=2

In firmware level 760 and later, a new value was added to the existing lpar_placement option to provide additional control over partition placement. The value of '2' indicates that the partition memory and processors should be packed into the minimum number of domains. In most cases this is the default behavior of the hypervisor, but certain configurations may spread across multiple chips/drawers. For example, if the configuration is such that the memory would fit into a single chip/drawer/book but the processors don't fit into a single chip/drawer/book, the default behavior of the hypervisor is to spread both the memory and the cores across multiple chip/drawer/book. In the previous example, if lpar_placement=2 is specified in the partition profile, the memory would be contained in a single chip/drawer/book but the processors would

be spread into multiple chips/drawers/books. This configuration may provide better performance in a situation with shared processors where the majority of the time the full entitlement and virtual processors are not active. In this case the hypervisor would dispatch the virtual processor in the single chip/drawer/book where the memory resides and only if the CPU exceeds what is available in the domain would the virtual processors need to span domains. The lpar_placement=2 option is available on all server models and applies to both shared and dedicated processor partitions. For AMS partitions, the attribute will pack the processors but since the memory is supplied by the AMS pool, the lpar_placement attribute does not pack the AMS pool memory. Packing should not be used indiscriminately, the hypervisor may not be able to satisfy all lpar_placement=2 requests. Also, for some workloads, lpar_placement=2 can lead to degraded performance.

1.4.7 Lpar_placement considerations for failover/disaster recovery

Many customers use a pair or cluster of systems for recover purposes where in the case of failure of partition or server the work is offloaded to another server. The placement of these partitions upon physical boundaries (drawers/books) is something to consider to optimize the performance in a failover situation.

The first step in achieving good placement is to follow the step in "1.4 Recommendation for placing LPAR resources to attain higher memory affinity". The entitlement of the partitions needs to accurately reflect the CPU consumption under normal load (ie. if the partition is consuming 4.5 processor units on average, partition should be configured with at least 5 VPs and 4.5 processor units and not 5 VPs with 0.5 processor units. Since the hypervisor uses the entitlement to determine how much CPU a partition will consume, the placement of partitions will not be optimal if the entitlement is undersized. After ensuring the entitlement is correct, the next step is to place partitions into hardware domains with spare capacity such that when a failover occurs, there is unused capacity available in the domain.

As an example, lets assume two 2-drawer 780 model with 16 cores per drawer. On server ONE is Partition A1 which is configured for 16 virtual processors (VPs) and 8.0 cores and Partition B1 also is configured for 16 VPs and 8.0 cores. Server TWO is similar configuration with partitions A2 and B2. Without any other directive, the hypervisor may place partitions A1 and B1 into the same drawer since the resources can be contained within a drawer. When running normally (non-failover), each server is only licensed for 16 of the 32 installed cores. In the event of a failure of server, the transactions that were being processed by A1 will failover to A2, same for B1 failing over the B2. Also, for failover, all cores in the server are activated such that the workload can be contained in a single server. What would be ideal for placement in the failover scenario would be that partition A2 is contained in a drawer with 16 cpus and similarly B2 is contained in a different drawer.

If all cores were licensed, one way to achieve this placement would be to configure A2 and B2 with 16.0 entitlement even though only 8.0 entitlement is required in the normal situation. Since each drawer has 16 cores, the hypervisor would be forced to place these partitions on different drawers. Another way to achieve the desired placement is though the configuration of memory. For example, if each drawer has 256GB of memory, you could set each partition to 8.0 cores, 16 VPs and 230 GB. In this situation when the hypervisor tries to place the partitions, the memory requirements of A2 and B2 force the hypervisor to place the partitions in separate drawers.

Allocation of memory beyond 50% of the capacity of the drawers would force this placement split.

Situations with multiple partitions per drawer can rely on the affinity group support to bundle up smaller partitions into a single affinity group. The affinity group, not individual partitions, would be the entity that is placed in the drawers/books. When creating the groups, the sum total of the memory and processor units should be such that it fits within a drawer/books and is large enough to consume at least 50% of the processor or memory such that multiple groups could not be placed in the same book/drawer. Also, if shared processors are being utilized, the total processing units for a group should be a multiple of full processor units.

Some users may consider using the Dynamic Platform Optimizer after a failover as a placement strategy but there is a performance cost involved in running the optimizer. The partitions, as the resources are moved between domains, run a bit slower and overall there is more CPU and memory bandwidth demands for the system. If there is sufficient unused capacity in the system or the optimization can be delayed to a time when there is unused capacity then the Dynamic Platform Optimizer may have a role to play in some failover scenarios.

1.4.8 Server evacuation using partition mobility

There may be need to move multiple partitions from one server to another server or multiple servers using partition mobility, for example in the case of hardware maintenance or firmware upgrade. In such a scenario, the order in which the partitions are migrated will have an effect on the placement of the partitions on the target server. For servers and partitions that support the dynamic platform optimizer, you can just run the optimizer after all partitions are migrated to the target server. In the case where the optimizer is not available or when partitions are not at the correct level to support optimization, the best practice is to move the partition in order from largest to smallest partition, where largest means the most memory and processors. Moving partitions in this order allow the hypervisor to optimize the allocation of the CPU and memory resources to the large partitions and then "fit" the smaller partitions around the bigger partitions. This effectively mimics the process that the hypervisor follows for allocation of CPU and memory resources to partitions after a server reboot.

1.5 PowerVM resource consumption for capacity planning considerations

PowerVM hypervisor consumes a portion of memory resources in the system; during planning stage take that into consideration to lay out LPARs. Factors that affect the amount of memory consumed in the size of the hardware page tables in the partitions, IVE resources, HCA resources, number of I/O devices, hypervisor memory mirroring and other factors. The IBM system planning tool (http://www.ibm.com/systems/support/tools/systemplanningtool/) should be used to estimate the amount of memory that will be reserved by the hypervisor.

1.6 Licensing resources (COD)

Power systems support capacity on-demand where customers can license capacity on-demand as the business needs for compute capacity grows. In addition to future growth COD resources can be used to provide performance improvement.

For COD memory, licensing is managed as an overall total of the memory consumed and there is no specific licensing of individual logical memory blocks (LMBs) or DIMMs. If there is COD memory installed on the system, this can provide performance benefits because the hypervisor

can use all the physical memory installed to improve partition placement. For example, on a two drawer system, each drawer has 256GB of installed memory but only 384GB of licensed memory, a 240GB partition can be created on one drawer and a 120GB partition on the second drawer. In this case, all 384GB is available and can be divided between the drawers in a manner that provides the best performance.

For COD processor, the licensing is on an individual core basis. When there are fewer licensed cores than installed cores, the hypervisor identifies cores to unlicensed and puts these cores into a low power state to save energy. In firmware level 730 and later, during a server reboot, the hypervisor places all defined partitions as optimally as possible and then unlicenses the unused processor cores. In firmware level 760 and later, when activating additional cores the hypervisor dynamically changes which cores are licensed and unlicensed to improve partition placement.

The dynamic platform optimizer can be utilized when configuration changes are made as a result of licensing additional cores and memory to optimize the system performance.