Capacity Planning and Performance Management on IBM PowerVM Virtualized Environment

Neeraj Bhatia
Contents

Introduction .................................................................................................................................................. 4
Section-I Concepts ......................................................................................................................................... 4
  – PowerVM and Power Hypervisor .............................................................................................................. 4
  – Partitioning Methods: Logical Partitions (LPARs) and Micro-Partitions .................................................. 4
  – Processing mode – Dedicated, Shared-dedicated and Shared .................................................................... 5
  – Processor Terminology: Physical, Virtual, Logical Processor ................................................................. 7
  – Processing Capacity: Desired, Minimum, and Maximum ........................................................................... 10
  – Virtual Processor Dispatch ..................................................................................................................... 11
  – Processor Sharing Mode: Capped v/s Uncapped ....................................................................................... 12
  – Virtual Processor Folding ....................................................................................................................... 14
  – Multiple Shared-Processor Pools ........................................................................................................... 15
  – Default Shared Processor Pool (SPP<sub>0</sub>) Attributes ..................................................................... 16
  – Processor Capacity Resolution ............................................................................................................... 17
Section-II Monitoring ................................................................................................................................... 20
  – What to Monitor? .................................................................................................................................. 20
  – Setting Thresholds .................................................................................................................................. 21
Section-III Sizing and Good Practices ......................................................................................................... 23
  – A Brief look at Hardware Configuration .................................................................................................. 23
  – Desired, Minimum and Maximum Values .................................................................................................. 25
  – Number of Virtual Processors .................................................................................................................. 27
  – Uncapped Weight .................................................................................................................................... 28
  – Dedicated to Shared .................................................................................................................................. 28
  – Choosing a Mix of Partitions .................................................................................................................... 29
Section-IV Capacity Planning Case Study ..................................................................................................... 29
Conclusion ..................................................................................................................................................... 33
Bibliography ................................................................................................................................................... 34
About the Author ........................................................................................................................................... 34
Introduction

In order to remain competitive, organizations continually exploit the IT technology. Traditionally organizations used to deploy only one service per server. Performance conflicts with other services, security and availability requirements are few rationales behind this approach. In addition, for every production server there exists development, testing and disaster recovery servers. With the increasing demand on the IT this practice resulted in sprawling infrastructure which is now not only difficult to manage but also consuming significant IT budget. In today’s tough business conditions businesses are scrutinizing about the inflationary spiral operational expenditures.

Server virtualization comes to the rescue. It transforms one physical into a host for many virtual machines, each of which can run an independent application. It means more services can run on a single physical server which makes the data center operations not only efficient but also cost effective. Most of the surveys show that around three out of four businesses had already used or planning to deploy virtualization.

A number of technologies are available for server virtualization and vendor market is also crowded. I have limited my discussion to one such offering from IBM. Usually processor, memory and I/O are three major areas come under server virtualization, but the scope of my discussion is restricted to processor virtualization. This paper starts with basic concepts and provides configuration guidelines and capacity planning examples.

At the end I’ve provided links to the documents I found most useful and referred during the writing. Some examples and diagrams are inspired from these texts and credit for them should be given to the authors of the texts and not to me. I recommend these documents to gain further understanding of micro-partitioning. Also, this is my personal work and opinions expressed here are my personal opinions and does not necessarily reflect the views and opinions of my employer.

Section-I Concepts

PowerVM and Power Hypervisor

PowerVM is the virtualization solution for AIX, IBM i, and Linux environments on IBM Power technology. As the foundation of PowerVM is the Power hypervisor. It is a firmware layer sitting between the hosted operating systems and the server hardware. Combined with features designed into the IBM power processors, the power hypervisor delivers the virtualized functions.

Partitioning Methods: Logical Partitions (LPARs) and Micro-Partitions

When it comes to partitioning in Power servers there are two methods: Logical Partitioning and Micro-Partitioning. Logical partitioning (LPAR) was introduced for Power servers with the IBM Power4 processor-based product line. This technology offered the capability to create separate logical systems, allowing each LPAR to run an operating environment on dedicated resources like CPUs and memory.

Micro-Partitioning technology allows allocating fractions of processors to a logical partition. A logical partition using fractions of processors is also known as a Micro-Partition. Micro-Partitions run over a set of processors called a Physical Shared-processor pool or simply Shared Processor Pool. Thus processors are shared among partitions. The benefit of Micro-Partitioning is that it allows significantly increased overall utilization of processor resources within the system. To assign a fraction of processor to a micro-partition, we use a concept of Virtual Processor
In a PowerVM environment, physical processors are abstracted into virtual processors that are available to partitions. Later we’ll discuss the concept of virtual processor in detail.

Throughout this paper we will use the term physical processor to represent a processor core as it would be easy to explain the concepts.

**Processing mode – Dedicated, Shared-dedicated and Shared**

You can assign processing capacity to a partition in either of two modes – assign entire processors for dedicated use or assign processor which can be shared with other micro-partitions.

**Dedicated Processing Mode**

In dedicated mode, physical processors cores are assigned as a whole to partitions. Thus the processing capacity in this case is fixed and can’t go beyond that. This is similar to processor configuration on SMP systems. For example, if you have assigned 2 processors to a partition in a dedicated processing mode, that’s it. It can’t access beyond two processors if required.

In dedicated processing mode, as only whole number of processors can be allocated to a partition, so the maximum number of partitions in dedicated processing mode in a system can be equal to the number of physical activated processors (each partition configured with 1 physical processor).

Partitions whose utilization profile is constant, predictable and bounded are candidates for dedicated processing mode. Careful planning is required to create a partition in dedicated processing mode, because underestimating the processor requirement can result in poor application performance as the partition will not get additional processing capacity during high workload timings.

Now since in the case of simultaneous multithreading, each core can execute instructions from two or four independent software threads simultaneously (2 in Power 5, Power 6 and 4 in Power 7), each dedicated CPU core will be visible as two or four logical processors to the operating system. With simultaneous multithreading turning off, each physical processor is presented as one logical processor only. We will come back to the concept of logical processor in a moment. Stay tuned!

**Shared Dedicated Mode**

The problem with dedicated processing mode is that processing capacity is fixed and during low workload timings processing capacity get wasted. To make better use of dedicated processing capacity started with Power6 technology, optionally dedicated partitions can be configured as processor donors. In this mode the unused cycles are harvested and then donated to the physical shared-processor pool associated with Micro-Partitioning. This ensures the opportunity for maximum processor utilization throughout the system.

The Power hypervisor ensures that only spare CPU cycles are donated, hence enabling this feature can help to increase system utilization, without affecting the critical partitions configured in a dedicated processor mode.

When the CPU utilization of the core goes below a threshold, and all the SMT threads of the CPU are idle from a hypervisor perspective, the CPU will be donated to the shared processor pool. The donated processor is returned instantaneously (within micro-seconds) to the dedicated processor partition when the timer of one of the SMT threads on the donated CPU expires, which means thread got instructions to execute.
Following is a HMC (Hardware Management Console) page to create a dedicated partition.

When processing mode is selected as “Dedicated”, you are asked for dedicated processors and optional processor sharing options. Value of Total managed system processors in the screenshot means we have 16 processors configured in the system. Desired processors are the number of processors you want to allocate to this partition, which are 8 in this case. Minimum and Maximum processors define the range for the dynamic operation. In other words, we can dynamically change the desired processors for this partition in the range of 4 to 12 processors.

We can further control which dedicated-processor partition can donate unused processing cycles (Shared-dedicated mode) by selecting processor sharing mode on the HMC while creating a partition. There are 2 check boxes under the heading “Processor Sharing”:

- “Allow when partition is inactive” - if checked, will allow the processors assigned for the dedicated processor partition to be included in the shared processor pool when the partition is powered off (that’s why inactive).
- “Allow when partition is active” - if checked, will allow the processors assigned for the dedicated processor partition to be included in the shared processor pool when the partition is active, but not making full use of the processors (Shared dedicated processing mode).

Shared Processing Mode

In shared processing mode, partitions get fractions of physical processor. The processing capacity assigned to a partition is known as its entitled capacity and expressed in processing units (PUs). One processing unit is equal to one physical processor’s worth of processing power. At minimum, a partition can get 0.1 processing units (in other words 10% of a processor’s capacity) and after that any value with an increment of 0.01. For example, 7.74, 3.33 are some examples of legitimate settings of partitions’ entitled capacity.

The entitled capacity of a partition plays a crucial role. It determines two very important factors: the guaranteed capacity available to a partition within one dispatch cycle, which is 10ms. In other words, within one dispatch cycle every partition is guaranteed to receive its entitled capacity. The total entitled capacity of all the partitions configured on the system can never exceed the number of processor in that system. The second important aspect of entitled capacity is that it provides the basis for utilization statistics and performance monitoring. We will discuss this aspect in detail later in the monitoring section.

"The entitled capacity of a partition determines the guaranteed capacity available to a partition within one dispatch cycle and provides the basis for utilization statistics."
A concept related to shared-processing mode is virtual processor. In shared mode, partitions use virtual processors to access fractions of physical processors. The Power hypervisor abstracts the physical processors and presents a set of virtual processors to the operating system within the micro-partitions on the system. Virtual processors are further discussed in the next section.

HMC page to create a shared partition is presented below.

![HMC page](image)

As you can see in the HMC page, when we select processing mode as “Shared”, we need to define processing units (Min, Max and Desired), virtual processors (Min, Max and Desired) and sharing mode.

The desired processing units define the processing capacity you desire for this partition to have and Min/Max processing units decide a valid range in which you can change the processing units for this partition. If there is sufficient processing capacity available in the shared-processor pool, desired processing capacity will become its CPU entitlement. Next is the Shared processor pool, which defines the processor pool to which this partition should belong. Similarly desired virtual processors is the number of virtual processors you want this partition to have and Min/Max defines a valid range for the dynamic movement.

Under the heading “Sharing mode”, there is a check box titled “Uncapped”, when checked it will mark the partition as uncapped partition (otherwise it will be capped partition) and you need to mention its uncapped weight. Uncapped weight is irrelevant in case of capped partitions.

While micro-partitions can be created with a minimum of 0.1 processing units, you can create a maximum of 10 micro-partitions per activated processor. The maximum number of partitions depends on the server model. The limit is up to 254 partitions on Power 770, 780, and 795 models.

**Processor Terminology: Physical, Virtual, Logical Processor**

There is tremendous amount of confusion when it comes to CPU terminology in IBM Power systems environment. Hence it's worthwhile to discuss the related terminology in detail.
Processor, CPU and CPU core

In an IBM Power Systems environment, CPU or a processor means a single core. This isn’t a standard definition across other vendors but while discussing PowerVM virtualization, the safest and easiest term that can be used is a core. Hence for the sake of simplicity, throughout this paper, we will use CPU or physical processor which will represent the number of cores configured on the system.

On a dual core system, if there is a single physical CPU configured, then the number of cores would be 2 and it will be visible by lparstat or prstat command.

Simultaneous Multithreading (SMT)

Simultaneous multithreading, often abbreviated as SMT, is an IBM microprocessor technology for improving the overall efficiency of CPUs and system throughput by allowing multiple hardware instructions streams (threads) to run concurrently on the same physical processor core. The number of concurrent threads can be decided by the chip designers, but in most of the cases number of threads are limited to two or four, though there are examples with 8 threads per core, for example, the Oracle SPARC T3-1 is 128-threads server with 16-cores, thus 8 threads per CPU core and SPARC T3-4 is 512-threads server with 64-cores.

Simultaneous multithreading was first introduced on Power5 offerings, supporting two threads and has been further enhanced in Power7 offerings by allowing four separate threads to run concurrently on the same physical processor core.

With simultaneous multithreading ON, there is a concept of logical processor or LCPU. Each logical CPU represents one of the threads that can be run on the same core at the same time. It’s possible to use mpstat –s to see how the threads are being dispatched. In the vmstat, with SMT on, the lcpu=2 means you have one core. When a thread is dispatched, it’s dispatched to the logical CPU, which then maps to the real core.

Virtual Processors

A virtual processor (VP or VCPU) is a depiction of a physical processor that is presented to the operating system running in a micro-partition. One can view them as a logical entity that is backed up by the physical processor cycles. Virtual processors are allocated in whole numbers. Each virtual processor can represent between 0.1 and 1.0 CPUs, known as processing units (PU). So a virtual processor can never consume more than 1.0 physical processor. In other words, the capacity of a virtual processor will always be equal to or less than the processing capacity of a physical processor. A shared partition can define number of virtual processors up to 10 times the number of processing units assigned to the partition.

The number of online virtual processors decides the absolute maximum CPU consumption an uncapped partition can achieve. For example, if a partition has an entitlement of 2.0 processors and four virtual processors, this partition could be able to consume up to four physical processors, in which case, it will report 200 percent CPU utilization (doubled the capacity configured on it).
The entitlement capacity assigned to a micro-partition is equally distributed among the virtual processors within the micro-partition. For example, if a micro-partition has 2.40 processing units and four virtual processors, each virtual processor will have the capacity of 0.60 processing units.

Virtual processors serve several purposes in the shared world.

1) First, virtual processors determine how many cores a partition thinks it has. If 2 virtual processors configured on a partition, it will think it has two physical cores. That is the number you’ll see in the mpstat output.

2) Second, the number of virtual processors configured for a micro-partition establishes the usable range of processing units. For example, a partition with one virtual processor can operate with between 0.1 and 1.0 processing units and with two virtual processors can get between 0.2 and 2.0 processing units. The upper limit of processing capacity, up to which an uncapped micro-partition can grow, is determined by the number of virtual processors that it possesses. For example, if you have a partition with entitled capacity of 0.50 processing units and one virtual processor, the partition cannot exceed 1.00 processing units. However, if the same partition is assigned two virtual processors and processing resources are available in the shared processor pool, the partition can go up to 2.00 processing units (an additional 1.50 processing units) and 4.00 processing units in case of 4 virtual processors.

Selecting the optimal number of virtual processors depends on the workload in the partition. Later in the section on “Sizing and Good Practices” we will discuss the guidelines to set optimal number of virtual processors for a partition as there are implications of smaller and bigger values. By default, the number of processing units that you specify is rounded up to the minimum whole number of virtual processors needed to satisfy the assigned number of processing units. For example:

- If you specify 0.50 processing units, one virtual processor will be assigned.
- If you specify 2.25 processing units, three virtual processors will be assigned.

In order to make use of unused capacity available in the shared processor pool, you may like to change this default setting and assign more virtual processors to the partition. A micro-partition must have enough virtual processors to satisfy its assigned processing capacity. For example, if a micro-partition has an entitled capacity of 2.5 processing units then the minimum number of virtual processors would be 3.

Let’s take an example in the diagram given below. A shared-processor pool has two partitions each configured with entitled capacity of 1.0 and 0.5 respectively and two and one virtual processor is configured on each partition. The shared-processor pool has 4 physical processors. As partition 1 has two virtual processors, its entitled capacity is equally distributed on two processors; processors P1 and P2 in this case. Because partition 2 has only one virtual processor, it will be dispatched to a single processor, which in this case is P2. It is important to note that virtual processors can be dispatched on any physical processor at run time. Also, since partition 1 has 2 virtual processors, it can get up to two full physical processors provided shared-processor pool at spare capacity at that time. Similarly partition 2 can consume up to 1 physical processor.
Processing Capacity: Desired, Minimum, and Maximum

While setting up a partition, you have to define the resources for the partition, such as processor, memory and I/O resources. As we have seen earlier in the screenshots, following processor attributes are required while creating a micro-partition:

- Minimum, desired, and maximum processing units
- Minimum, desired, and maximum virtual processors

The desired processing units and desired virtual processor values can be changed dynamically without stopping or reactivating the partition and that’s where minimum and maximum values come into picture. The minimum and maximum settings for both processing units and virtual processors represent the extremes between which the desired values can be dynamically changed. Maximum value is only used as an upper limit for the dynamic operation and doesn’t play any role in processing capacity allotment for the partition.

“A desired value defines a value which you desire or like to have. This is not an ensured processing capacity because there might not be enough capacity in the shared-processor pool. In addition to define a lower boundary for dynamic operations, the minimum value defines a minimum value which must be available in the shared-processor pool for the partition to start, failing which partition will not start.

When a partition is started, preference is given to the desired value you set. When there is enough capacity in the shared-processor pool, the allocated entitled capacity will be equal to the desired value. Otherwise, the entitled capacity will be lower than the desired value, a value greater than or equal to the minimum capacity attribute. If the minimum capacity requirement cannot be met, the micro-partition will not start.
The entitled processor capacity is allocated among the partitions in the sequence the partitions are started. Consider a scenario where a physical shared-processor pool has 2.0 processing units available and there are 3 partitions started in the sequence Partitions 1, 2, and 3 with following attributes:

- Partition 1 (Minimum: 0.5, Desired: 1.5, Maximum: 1.8)
- Partition 2 (Minimum: 1.0, Desired: 1.5, Maximum: 2.0)
- Partition 3 (Minimum: 0.5, Desired: 1.0, Maximum: 1.5)

Since Partition 1 is started first, it will get the entitled processing capacity of 1.5 because there is enough processing capacity of 2.0 processing units available in the physical-processor pool. Partition 2 will not be able to start because after allocating 1.5 processing units to partition 1, only 0.5 processing units are left in the physical-processing pool and Partition 2 minimum capacity requirement is greater than that. Partition 3 will get started with entitled capacity of 0.5, which is the capacity left in the physical-processing pool. This value is less than Partition 3’s desired capacity requirement. The allocated capacity entitlements are summarized below:

- Partition 1 will be activated with allocated capacity entitlement of 1.5, which is the desired value.
- Partition 2 will not start because minimum capacity requirements will not met.
- Partition 3 will be activated with allocated capacity entitlement of 0.5, which is lesser than the desired value but sufficient enough to start the partition.

It is important to note that the maximum settings don’t represent the amount of processing units that an uncapped partition can access. It is governed by the desired Virtual Processor setting. Let’s assume in previous example, Partition 1 and Partition 3 are configured with following virtual processor attributes:

- Partition 1 (Minimum: 2, Desired: 3, Maximum: 4)
- Partition 3 (Minimum: 2, Desired: 2, Maximum: 3)

The partitions 1 and 3 in this case can access up to 3 and 2 processing units respectively, which means 200% and 400% of their entitled capacity allotments.

A careful planning should be conducted to choose the values of minimum and desired attributes of a partition. Later in this paper we will see how to decide the optimal number of minimum, desired and maximum attributes for a partition.

**Virtual Processor Dispatch**

In PowerVM environment, hypervisor abstracts the physical processors and micro-partitions see processors in the form of virtual processors. Each micro-partition has a certain number of virtual processors configured in its profile. The number of virtual processors in a micro-partition and in all micro-partitions does not necessarily have to be fixed based on the number of physical processors in the physical shared-processor pool. However for every physical processor, you can define up to 10 virtual processors. Having understood the relation of physical processor and virtual processors, next important point to understand is the dispatch cycle of a physical processor and how processing capacity is distributed among virtual processors.

The Hypervisor manages the distribution of available physical processor cycles from the processors in the physical shared-processor pool. The Hypervisor uses a 10 ms dispatch cycle; each virtual processor is guaranteed to receive its entitled capacity of processor cycles during each 10 ms dispatch window.

Let’s take an example where a micro-partition is defined with entitled capacity of 0.8 and with two virtual processors. It means this micro-partition will get the equivalent of 80 percent of one physical processor in each 10
ms dispatch window from the physical shared-processor pool or 40 percent of two physical processors. Hence each virtual processor will have dispatch window of 4 ms instead of the 10 ms. Both virtual processors could be dispatched to the same or different cores. The Power Hypervisor is designed to dispatch virtual processors on the same physical processor across dispatch cycles where possible. In other words, the hypervisor tries to dispatch them to the same core on which they have dispatch previously, but there is no guarantee. This behavior is called Processor Affinity. This optimizes the use of cache memory because the working data set of a job can be read/written in the shortest time possible and also minimize context switching.

To further illustrate the dispatch mechanism, consider two partitions with two and three virtual processors and entitled capacity of 0.8 and 0.6 respectively. These five virtual processors are mapped to two physical processors, each having a dispatch cycle of 10 ms.

Micro-partition 1 is defined with an entitled capacity of 0.8 processing units and two virtual processors. This means each virtual processor could get a dispatch window of 4 ms on both physical processors. Two virtual processors means it can potentially get up to 2 physical processors, however it is very unlikely as it means that partition 2 is not using any CPU cycles (turned-off possibly). Figure given below shows two dispatch cycles. In cycle 1, partition 1 is dispatch to both physical processors and getting 4ms on each. In dispatch cycle 2, partition 1 is consuming more than its entitlement (that is 1.4 processors).

Micro-partition 2 is configured with an entitled capacity of 0.6 and three virtual processors. Since there are only two physical processors in the system, two virtual processors in this case would be dispatched to the same physical processor.

It is possible for a virtual processor to be dispatched more than once on the same physical processor during a dispatch interval, as with the case of partition 1 on Processor 1.

**Processor Sharing Mode: Capped v/s Uncapped**

Shared micro-partitions can be configured as capped or uncapped that determines the maximum processing capacity available to them from their Shared-Processor Pool.

In capped micro-partition mode, the processing capacity is bounded to its entitled capacity and cannot exceed beyond that. For example, if a micro-partition is configured as a capped partition with entitled capacity of 2.0 processing units, no matter how many spare capacity is available in the shared-processor pool, it can’t use beyond 2.0 processors.
In uncapped mode, the processing capacity can exceed the entitled capacity when resources are available in their Shared-Processor Pool. The upper limit up to which an uncapped micro-partition can grow is therefore determined by the virtual processors configured on the partition or the spare capacity available in the shared-processor pool.

If there is a competition for additional processing capacity among several eligible micro-partitions (uncapped micro-partitions with sufficient number of virtual processors), the extra capacity is distributed among them on the basis of uncapped weight. Each uncapped micro-partition has an uncapped weight which hypervisor uses to distribute spare capacity among the partitions. Uncapped weight is a whole number ranging from 0 to 255. The higher the uncapped weight of a micro-partition, the better the chances the partition will get spare capacity. The default value is 128 which mean that the additional capacity is distributed in evenly manner. The uncapped weight of 0 is as good as a capped micro-partition, which means micro-partition with an uncapped weight of 0 cannot get additional processing capacity above its entitled capacity.

To explain further, let’s take a simple example of two uncapped micro-partitions, both configured with default uncapped weight of 128, are competing for additional processing capacity. If there are 2.0 processing units available in their shared-processor pool, each will get \(2.0 \times \frac{128}{128 + 128}\) that is 1.0 processing unit, which equals to 1 full processor capacity.

A formula to calculate a particular micro-partition's share of the unused capacity can be estimated as:

\[
\text{Additional Capacity Share} = \frac{\text{Available Unused Capacity} \times \text{Uncapped weight of the partition}}{\text{Sum of uncapped weights of all the partitions}}
\]

Let’s consider that the uncapped weights for above partitions are now changed to 100 and 200 respectively. In that case, first partition will get \(2.0 \times \frac{100}{100+200}\) i.e. around 0.67 additional processing units and second partition will get 1.33 processing units.

Illustration given below is showing the CPU utilization in case of capped and uncapped partitions. Shared-processor pool in both cases has 4 physical processors and partitions have entitled capacity of 2 (shown by a horizontal line at 2.0 physical processors). As you can see uncapped partition has received additional processing capacity beyond its entitlement and in that case the idle capacity available in the shared-processor pool has come down (shown by a line chart).

On the other hand physical processors used in case of capped partition never went beyond the entitlement capacity of 2 and comparatively shared-processor pool has idle capacity available most of the time.

In both cases blue color region below the entitled capacity line, is showing available capacity which is ceded to the shared-processor pool.
Virtual Processor Folding

In order to take full advantage of unused processor capacity in the physical shared-processor pool, we create a micro-partition as uncapped and configure enough number of virtual processors. Physical processors are dispatched over these virtual processors and they decide the upper limit up to which an uncapped micro-partition’s processing capacity can grow. However setting optimal number of virtual processors is tricky because it requires careful planning and testing before reaching to an optimal setting for your environment. Unnecessary more number of virtual processors means inefficient use of processing capacity, and on the other hand lesser virtual processors means there is not sufficient CPU cycles when required.

While planning for the number of virtual processors for an uncapped micro-partition, it is important to note that there is processing associated with the virtual processors at the hypervisor level. Virtual Processor folding is an advance optimization technique which helps to manage idle virtual processors. The kernel scheduler dynamically increases and decreases the use of virtual processors in conjunction with the load of the partition. So with this functionality on, kernel puts idle virtual processors into a hibernation state so they do not consume any resources.

Take an example of an uncapped micro-partition with 2.8 entitlement capacity and 5 virtual processors. Without Virtual Processor folding, irrespective of the workload these 5 virtual processors used to remain idle and consume a few but valuable amount of resources. Virtual Processor folding puts idle virtual processors into a hibernation state so they do not consume any resources. So if at the moment system needs only 1.5 processors then it folds away 3 virtual processors and runs on just 2.

When virtual processors are deactivated, they are not dynamically removed from the partition. Even virtual processors are visible to the AIX. Processor folding means kernel no longer schedule processes to run on them. Also if SMT enabled, when a virtual processor is folded all the SMT threads (logical processors) are switched off together.

There are several important benefits of this feature including:

- Improved processor affinity and better use of processor memory.
- Reduced hypervisor workload because pointless context switches consume system resources.
- Increased average time a virtual processor executes on a physical processor because no more scheduling a virtual processor to a physical processor which does not have any work to do. It means system throughput will be improved.
However, there is no benefit one can expect from this feature when partitions are busy because it manages idle virtual processors. By default this feature is turned on, but it can be turned off. If the feature is turned off, all virtual processors defined for the partition are dispatched to physical processors.

The kernel measures the load on the system every second and uses that value to determine whether virtual processors need to be enabled or disabled. If the CPU utilization consumed in the last one-second interval is greater than 80 percent of the capacity provided by the number of virtual CPUs currently enabled in the partition, then additional virtual CPUs are enabled.

The advantage of setting large number of virtual processors is that it will accommodate spikes but on the downside during low-workload timings large number of idle virtual processors consumes processing cycles. Enabling this feature can be very useful for uncapped partitions because it allows the partition to be configured with a larger number of virtual processors without significant performance issues. The optimal setting of virtual processors is which accommodate immediate spikes or rapid short-term growth requirements.

In order to achieve optimal setting for your environment, long term trend consisting of peak business cycles and seasonal load (monthly or quarterly) should be analyzed. If we observe virtual processors folding happening and particularly during peak workload times it means we have the entitlement set too high for that partition. On the other hand, if we have a business-critical micro-partition with no processor folding taking place for a long period then we may like to consider raising the entitlement to make sure it has sufficient CPU cycles to accommodate sudden spikes. Later we will discuss further about optimal number of virtual processors.

**Multiple Shared-Processor Pools**

A physical shared-processor pool is a set of active physical processors that are not dedicated to any logical partition. For example, if a system has 10 active physical processors out of which 2 are dedicated to a logical partition, i.e. partition is in dedicated processing mode, Physical shared-processor pool would comprises of 8 physical processors. There can be only one Physical Shared-Processor pool per server. The size of the shared pool size can change dynamically as dedicated partitions are activated or deactivated. The understanding of Physical shared-processor pool is important because from there the actual processing capacity will be allocated to the partitions in the system.

In Power5 servers there can be only one shared-processor pool which is Physical-processor pool. All the activated partitions in the system get processing capacity from this processor pool according to their attributes like entitled capacity, number of virtual processor, uncapped weight, processing mode etc. In order to further control the processing capacity among micro-partitions in the system, Power6 and Power7 servers have a capability to support Multiple Shared-Processor Pools (MSPPs). This allows creating a set of micro-partitions as a member of a shared-processor pool and any spare capacity available in the shared-processor pool will be first distributed to its members.

A Shared-Processor Pool is a logical group of micro-partitions and their associated virtual processors that are grouped together for the purpose of controlling the processor capacity that the set of micro-partitions can consume from the physical shared-processor pool.

Power6 and Power7 servers which support Multiple Shared-Processor Pools capability thus have two types of Shared-Processor pools: the default Shared-Processor Pool and user-defined Shared-Processor Pool. Each Shared-Processor pool is identified by a unique number known as Processor Pool identifier. The default Shared-Processor Pool always has an identifier of 0 and is referred to as Shared-Processor Pool_0 \( (SPP_0) \). There can be up to a maximum of 64 Shared Processor Pools and referred to as Shared-Processor Pool_n \( (SPP_n) \).
In the context of Multiple Shared-Processor Pools, understanding of following concepts is important:

**Maximum Pool Capacity**

Each Shared-Processor Pool has a maximum capacity associated with it. The Maximum Pool Capacity defines the upper boundary of the processor capacity that can be utilized by the set of micro-partitions in the Shared-Processor Pool. The Maximum Pool Capacity must be represented by a whole number of processor units. The micro-partitions in a shared-processor pool can’t get additional capacity even if there is spare capacity available in other shared-processor pools if its maximum capacity is exhausted.

**Reserved Pool Capacity**

In addition to the total processor entitlements of the individual micro-partitions, you can optionally reserve some processor capacity from the physical shared-processor pool at the Shared-processor Pool level which can be used by the uncapped micro-partitions when they need additional processor capacity. The Reserved Pool Capacity is distributed among uncapped micro-partitions in the Shared-Processor Pool according to their uncapped weighting. Default value for the Reserved Pool Capacity is zero (0).

**Entitled Pool Capacity**

The Entitled Pool Capacity of a Shared-Processor Pool defines the guaranteed processor capacity that is available to the group of micro-partitions in the Shared processor Pool. The Entitled Pool Capacity is the sum of the entitlement capacities of the micro-partitions in the Shared-Processor Pool plus the Reserved Pool Capacity, if defined.

If the server is under heavy load, each micro-partition within a Shared-Processor Pool is guaranteed its processor entitlement plus any capacity that it might be allocated from the Reserved Pool Capacity if the micro-partition is uncapped.

**Default Shared Processor Pool (SPP_0) Attributes**

On all Power6 and Power7 systems a default Shared-Processor Pool is always automatically defined. It has a pool identifier of zero (which cannot be changed) and referred to as SPP_0. The attributes of default Shared-Processor Pool have fixed values and cannot be changed:

- Maximum Pool Capacity - For the default Shared-processor pool, this value is equal to the number of active physical processors in the physical shared-processor pool.
- Reserved Pool Capacity - For the default Shared-Processor Pool, this value is equal to 0 and cannot be changed.
- Entitled Pool Capacity - For the default Shared-processor pool, this value is equal to the sum of the entitled capacities of the micro-partitions in the default Shared-Processor as Reserved Pool Capacity is always zero in the default Shared-Processor Pool.

The diagram given below shows the configuration of Multiple Shared-Processor Pools and their relation with the physical processor pool. We have total 8 physical processors out of which 2 are in inactive state and 1 is configured in dedicated-processing mode, so the physical processor pool will comprise of total 5 physical processors from which the processing capacity will be assigned to shared-processor pools. We have three Shared-processor pools named SPP_0 (default), SPP_1 and SPP_2.
- The default shared-processor pool has two micro-partitions with entitlements 0.5 and 0.8. Since in the case of default shared-processor pool, reserved pool capacity would be 0 and the value of the maximum pool capacity is equal to the number of active physical processors in the physical shared-processor pool which is 5. Maximum Pool Capacity = 5, Reserved Pool Capacity = 0, Entitled Pool Capacity = 1.3

- The multiple shared-processor pool SPP\textsubscript{1} has two partitions with entitled capacity of 0.6 and 1.0 respectively. It has reserved capacity of 0.5 which means Entitled pool capacity in this case would be 0.6 + 1.0 + 0.5, that is 2.1. Also the total capacity the partitions can access at any time would be restricted to 3.0.

- The Shared-processor pool SPP\textsubscript{2} has three partitions with entitlement 0.5, 0.4 and 0.2 respectively. The reserved pool capacity of 0.5 will make the entitled pool capacity of the shared-processor pool to 1.6. Maximum pool capacity of 2.0 defines that the member partitions can get in total 2.0 processing capacity.

![Diagram of processor pools](image)

**Processor Capacity Resolution**

There are two levels of processor capacity resolution take place in Multiple Shared-Processor Pools structure:

- **Level-0:** In the first level, the resolution of capacity takes place within the same Shared-Processor Pool. Unused processor cycles from the micro-partitions in a Shared-Processor Pool are harvested and then redistributed to any eligible micro-partition within the same Shared-Processor Pool. If there is a competition for the processor capacity, uncapped weight decides the additional processor cycles a partition should receive. This level of capacity resolution is similar to capacity resolution in Power5 servers’ Shared-Processor Pool.
• Level-1: When all Level-0 capacity resolutions take place within the Multiple Shared-Processor Pools, there may be a chance of unused capacity available in some Shared-Processor pools because micro-partitions within such a processor pool are underutilized and possibly reserved pool capacity is not utilized. On the other hand, some Shared-Processor pools may be heavily loaded and need additional processor capacity above the entitled pool capacity. To further optimize the use of spare capacity at the system level, the hypervisor harvests unused processor cycles and redistributes them to eligible micro-partitions regardless of the Multiple Shared-Processor Pools structure. This is the second level of processor capacity resolution.

During Level-1 Capacity resolution the additional capacity is allocated directly to the individual micro-partitions on the basis of uncapped weights within the system as a whole. In other words, when allocating additional processor capacity in excess of the Entitled Pool Capacity of the Shared-Processor Pool, the hypervisor takes the uncapped weights of individual micro-partitions competing for additional processor capacity regardless of the Multiple Shared-Processor Pool structure. The total additional capacity during level-1 resolution allocated to micro-partitions within a particular Shared-Processor Pool cannot be greater than the Maximum Pool Capacity for that Shared-Processor Pool.

If there is unused processor capacity available in the underutilized shared-processor pools, it will be harvested and ceded to the hypervisor. The hypervisor then allocate these unused processor cycles to the other micro-partitions if:
• Micro-partitions have already used all the processor cycles up to entitled pool capacity. In other words, level-0 capacity resolution already happened.
• The Maximum Pool Capacity of the shared-processor pool has not been met. It means there is a room for further receiving the processor capacity.
• The Micro-partition seeking additional capacity is uncapped.
• The Micro-partition has enough virtual processors to exploit the additional capacity.

"The scope for reallocation of unused processor capacity in level-0 is within a Shared-processor pool, while it is all the micro-partitions on the system regardless of the shared-processor pools structure in level-1 capacity resolution."

To further explain the capacity resolution in multiple shared-processor pools, let’s take an example. Figure given below shows the configuration of three shared-processor pools. Let’s assume that at one point of time, below are the capacity demands of individual partitions expressed in terms of physical processors:

- SPP₀: P₁ (0.2), P₂ (0.8)
- SPP₁: P₁ (2.0), P₂ (1.0), P₃ (0.5)
- SPP₂: P₁ (1.5), P₂ (1.5), P₃(0.8)

Partitions which are capacity donors are in green color whereas names in red color are capacity receivers. Let’s assume all the partitions in SPP1 have uncapped weight of 100 and partitions in SPP2 have uncapped weight of 200. Let’s discuss each shared-processor pool one-by-one.

The default shared-processor pool SPP₀ has access to all the physical processors in the physical pool; hence its maximum pool capacity is 8. Both the partitions in the pool are under-utilized, so there is no need of level-0
capacity resolution in the pool. The unused capacity will be harvested and then will be available for level-1 resolution.

Shared-processor pool SPP₁ is heavily utilized where partitions P1 and P2 need excess capacity beyond their entitled capacities and partition P3 is under-utilized. First the level-0 capacity resolution will take place where 1.0 processing units of partition P3 will be evenly distributed to partitions P1 and P2 as both have same uncapped weight of 100. Even after level-0 capacity resolution, partitions P1 and P2 will need additional capacity of 0.5 processing units. The shared-processor pool has maximum pool capacity of 4, which means it can get up to 4 physical processors by level-1 resolution.

The shared-processor pool SPP₂ is also over utilized and need excess capacity even after getting capacity from reserved pool capacity. The virtual processors setting and maximum pool capacity of the shared pool will enable the partitions P1 and P3 to get additional capacity by level-1 capacity resolution.

To summarize SPP₀ has excess capacity of 0.5 processing units which will be ceded to SPP₁ and SPP₂ during level-1 capacity resolution and SPP₁ and SPP₂ need 0.5 and 0.3 additional processing units respectively. Because during Level-1 Capacity resolution the additional capacity is allocated directly to the individual micro-partitions on the basis of uncapped weights,

- SPP₁ partitions additional capacity during level-1 resolution = 0.5 * (100/600) = 0.083
- SPP₂ partitions additional capacity during level-1 resolution = 0.5 * (200/600) = 0.166

In other words, during dispatch cycle of 10ms, partition P1, P2 in shared-processor pool SPP₁ will get around 0.08 processing units, whereas partitions P1 and P3 in the shared-processor pool SPP₂ will get approx 0.166 processing units. Clearly all the physical processors in the system, at this moment, are fully utilized and processing capacity requirements for some partitions will not be met.

A Note on Dynamic movement of Micro-partitions

A micro-partition can be moved dynamically from one Shared-Processor Pool to another. It is a simple reassignment of the Shared-Processor Pool-ID that a particular micro-partition is associated with. As the Entitled
Pool Capacity is made up of the sum of the entitled capacities of the micro-partitions plus the reserved pool capacity, removing a micro-partition from a Shared-Processor Pool will reduce the Entitled Pool Capacity for that Shared-Processor Pool. On the other hand the Entitled Pool Capacity of the destination Shared-Processor Pool will increase. The Reserved Pool Capacity and Maximum Pool Capacity however remain intact due to this movement. It means if the movement of a micro-partition to a target Shared-Processor Pool pushes the Entitled Pool Capacity beyond the Maximum Pool Capacity, then movement of the micro-partition will fail.

Section-II Monitoring

What to Monitor?

One of the challenging tasks in a virtualized environment is monitoring. Before setting up monitoring mechanism, the first question you need to answer is – what to monitor? Entitled Capacity utilization and Physical Processors consumption are two most important metrics to monitor. We will discuss these two in detail from monitoring perspective.

As discussed earlier the entitled capacity of a partition provides the basis for utilization statistics and monitoring. A partition consuming all of its entitled capacity will report 100% utilization. Depending on its virtual processor configuration an uncapped partition can consume unused processing capacity from the shared processor pool and in that case it will report more than 100 percent utilization.

Another important metric to monitor is capacity consumption in terms of physical processors. This metric can be derived from partitions’ entitled capacity and their entitled capacity utilization. For example, if a partition with entitled capacity of 0.50 processing units (0.5 physical processor) is having an entitled capacity utilization of 150%, it implies that it is consuming 0.50 * 1.5 = 0.75 physical processor.

Monitoring and thresholds can be set using any in-house developed custom scripts based on the native tools and utilities or more sophisticatedly through tools available in the market like Ganglia or Teamquest. I have intentionally excluded the details of monitoring from my discussion. But interested readers can refer the manuals of native AIX utilities like sar, vmstat, mpstat, lparstat, topas for further details. To summarize all these utilities give two extra columns when executing in a shared-processor partition:

- Physical processor consumed by the partition, shown as pc or phyc.
- Percentage of entitled capacity consumed by the partition, shown as ec or %entc.

An additional metric is important from monitoring perspective which can be used to understand the overall CPU utilization at physical system level. This is shown under the column “app” and indicates the available physical processors in the shared pool. For example the following lparstat output showing the size of the shared pool is 2 CPU’s (psize). The app column shows 0.65 available processors in the shared pool. The CPU utilization in this interval is 67.5% ( (2.0 - 0.65) / 2.0 )
Setting Thresholds

Due to the fluctuation nature of CPU utilization in virtualized environment, it is seen as a good practice to set thresholds at the physical server level. For example - whenever CPU utilization crosses a predefined threshold, an alert should be raised. It is important to note that multiple thresholds make more sense because some of the threshold breaches may be legitimate, those especially occurred during peak workload hours.

While power server level utilization view is important to see how much capacity is being in use as a whole, monitoring at individual partition level will tell us how much each partition is contributing to the overall CPU utilization. Similar thresholds should be configured at the partition level based on the entitled-level CPU utilization. No need to mention that thresholds values may depend on underlying partition's behavior. For example, critical threshold for a steady workload partition like a batch database can be 120% (which means at 120% entitled capacity utilization); on the other hand for partitions which depict fluctuations, it can be set higher like 150%.

Let's take an example of a system with 64 Physical CPU's. Two thresholds are set: Warning threshold is set at 75% CPU (in other words 48 CPUs, rest 25% available CPUs are maintained for expected utilization fluctuation), critical threshold at around 85% (that is 55 CPUs). The threshold values may differ from environment to environment because of the very nature of the workloads.

The first graph is showing physical CPUs used at system level and individual partitions level for a typical day. The green line in the first graph is showing warning threshold which is set at 48 CPUs and breached many times during 11AM – 8PM. The next two graphs are representing entitled CPU utilization for partitions P1 and partition P3. Partition P1's utilization is fluctuating and it is being heavily used specially during 11AM – 8PM, hence warning and critical thresholds in this case are set at 120% and 140% respectively. On the other hand, partition P3’s utilization is steady, so its thresholds are set at relatively low values of 100% and 120%. Though for illustration purpose, I have used a 24-hours profile, but one should consider comparatively bigger time period covering the typical workload cycle for analysis.

Some possible actions in case of threshold violations can be investigating the individual partition contributing to the server's utilization, workload management if possible or as a last resort stop/migrate least critical partition. Workload behavior of partitions is very important and configuration needs to be done in such a way that not many partitions should compete for the processor resources at the same time.
One gauge of system’s health is CPU run queue length. The run-queue length represents the number of processes that are currently running or waiting (queued) to run. Setting thresholds for run queue length is tricky in partitioned environment because uncapped partitioned can potentially consume more than their entitlement up to number of virtual processors. SMT introduced further complexity as it enable parallel execution: 2 simultaneous thread on Power5 and Power6 and 4 on Power7 environments.

The conventional rule of thumb of twice the number of CPUs (One process on CPU while second is waiting for some other factors like disk I/O, paging etc) is also applicable for dedicated-mode partitions with SMT off. With SMT turned on, you can simply multiple the number of CPUs by 2/4 depending on your system.

Things are little trickier in case of shared-partitions. For capped partitions the entitled capacity gives an upper bound of capacity which we can use for threshold settings. It means for capped partitions the two times of their entitled capacity represented by nearest whole number is a safe threshold for processor run queue. On the other hand, for uncapped partitions since virtual processors represent the maximum capacity limit, twice the number of virtual CPUs can be used for CPU run queue thresholds. In both cases, if simultaneous multi-threading is on, you should consider multiple the processors by 2 or 4 depending on your system.

Hence processor run queue length threshold guidelines can be summarized as follows:

- Dedicated-mode Partitions: Number of Processers X 2 (SMT =off)
  (If SMT=on, further multiple by 2 or 4 depending on your Power system version)
• Capped Shared Partitions: Entitled Capacity X 2 represented in nearest whole number (SMT=off)
  (If SMT=on, further multiple by 2 or 4 depending on your Power system version)

• Uncapped Shared Partitions: Virtual Processors X 2 (SMT=off)
  (If SMT=on, further multiple by 2 or 4 depending on your Power system version)

One point to note in case of uncapped partitions is that you don’t have wildly large number of virtual processors
defined. As a virtual processor represent from 0.1 to 1.0 physical processor, on a 64-way system if entitlement is 4
CPUs for an uncapped partition you can define virtual processors in the range of 4 to 40. Going with above logic of
thresholds, on a Power6 system with SMT=on if you define processor run-queue threshold at 40*2*2=160, which
assumes that at any time you can get up to 160 processor threads which is not likely to get on 64-way machine.

Section-III Sizing and Good Practices

Quite often it is seen in real world that application architect overestimate the processing capacity, which is even
50-200% more than what application actually needs. This results in underutilized partitions and simply beat the
very important purpose of virtualization which is to increase the utilization of installed capacity. In this section we
will discuss some important aspects of partitions sizing.

An important aspect of optimal partition configuration is that after initial sizing of the partition based on the
workload profile (for existing applications) or anticipated transaction volumes (new applications) you have proper
monitoring mechanism in place to collect right set of metrics at appropriate interval. By continuously monitoring
the environment and taking appropriate actions using dynamic logical partitioning, we can rightly size the
partitions and get a cost effective solution.

A Brief look at Hardware Configuration

Discussion of optimal settings for a partition needs a brief look at the hardware configuration of Power systems
which differs by a major change. Following diagrams are showing configurations for Power5 and Power6 systems.
The configuration for Power7 is same with Power6, just the cache size changed.
In Power5 systems, there is a single L2 cache on a processor chip which is shared by both the cores on the chip. In contrast with Power5, later servers (Power6 and Power7) have separate L2 for each core on a chip. L2 cache is a fast memory which stores copies of the data from the most frequently used main memory locations.

The key point here is that the partition’s performance depends on how efficiently it is using the processor cache. The minimal is the L2 cache interference with another partition, better the performance of the partition. It is because sharing the L2 cache with other partitions means there is a chance of processor’s most frequently accessed data will be flushed out of the cache and accessing from L3/main memory will take more time.

Following are major areas where configuration settings can affect partition’s performance:

- Wherever possible, using partitions in Dedicated processing mode instead of Sharing mode.
- Try to allocate processors in multiple of 2
- Try to allocate Shared-processor partitions using whole processors
- Set the minimal number of Virtual processors.

We’ll discuss these points one-by-one.

**Dedicated-Processing mode v/s Shared-Processing mode**

Partitions with dedicated-processor mode perform best. In dedicated mode, processors stick to partitions and no other partitions can use them. Also these partitions can only configured with whole partitions and not fractions of a processor as in the case of shared-processor partitions. This way the partition’s memory remains in the cache for longer duration and thus resulting in faster memory access. In other words lack of sharing gives them the best performance.

However as you must be thinking, there is a disadvantage of configuring a partition in dedicated-processor mode. If you have a partition in dedicated-processor mode which is under-utilized, you’re wasting processing capacity which could be otherwise utilized by other partitions if configured in sharing mode. In other words, you are optimizing an individual partition at the potential cost of overall system performance. Hence the choice between dedicated and shared partitions therefore falls more into business needs. You may like to consider configuring the business-critical partition in dedicated-mode if its utilization pattern is showing a plateau in long term.

Starting from Power6 however, you can configure a partition in shared dedicated mode and thus release processing capacity during idle times but that is also a compromise with the advantages of dedicated-mode because cache contents will be flushed.

**Processors in multiple of 2**

As mentioned earlier Power5 servers have a common L2 cache on a chip which is shared by two cores. If we assign processors as a multiple of 2, it will minimize the L2 cache contention with other partitions. Otherwise each partition has to essentially share its L2 cache with other partitions. Sizing partitions in multiple of two processors on Power6 and Power7 family of servers is not required as processors have private L2 caches.

When running shared processor partitions, the partitions can be scheduled on any processor in the shared processor pool. So while a 2-way dedicated partition can avoid cache conflicts from other partitions, 2-way shared
processor partitions will have some amount of cache sharing with other partitions. The hypervisor’s scheduling algorithms do reduce this by giving partitions affinity to certain processors, but the sharing cannot be eliminated as it can in dedicated environments.

**Whole Processors in sharing mode**

When configuring partitions in sharing-processor mode we should consider rounding up some partitions to the nearest whole processors (for example 1.00, 2.00 etc) as it will reduce the effect of partitions sharing the same processor. If a partition is not sized in whole processor it will be sharing the L2 cache with at least one partition and there will be chances of L2 cache contents getting flushed and subsequently reading the data will take time and resulting in performance degrade.

The effect of fractional processors is noticeable on small partitions because significant processing capacity will be consumed in warming up the cache every time partitions get a chance to run on the processor.

**Desired, Minimum and Maximum Values**

A careful planning should be conducted to choose the values of minimum and desired processing units of a partition. These values in association with other settings (number of virtual processors, capped or uncapped partition etc) are directly related to the performance of applications running on the partitions. Setting lower values will degrade the performance, and higher values on the other hand may result in a situation where partition cannot get started because there is not enough capacity available to meet the settings. For a partition, minimum value should be the value you think is a bare-minimum requirement for the applications run in the partition. Desired value should be the value which you think should be available most of the times or in other words value suitable for the anticipated typical workload. If the load and in turn processing requirements of a partition varies with the time (day/night behavior, peak/off-peak working hours), then to cater for it make it uncapped partition with minimum capacity attribute equal to the capacity required during lowest workload times and desired attribute equal to the capacity required during most of the time (possibly value between lowest and highest workload time). Also, have the enough number of virtual processors set so that uncapped partitions can get additional processing capacity during sudden spikes of workload.

When planning for partitions’ resource allocation it is better to represent partition utilization in physical processor units instead of entitled CPU utilization. Figure given below shows the hourly CPU utilization during a typical day for a partition expressed in physical processors consumed.
The partition has a plateau of around 1.9 processors but sometimes it spiked up to 3.5 processors. Now based on one-week’s data we can make following observations:

- On average 1.9 processors are adequate to run the application. If the application running on this partition is of less importance to the business and you don’t want it to interfere with other business-critical partitions running on the system, you may want to configure the desired processing units to 1.9 which are enough to run the application most of the times. Configuring the partition in capped mode or as uncapped with two virtual processors will ensure that partition will not take spare capacity from the shared pool and thus impact other partitions. Note that in uncapped mode with two virtual processors it may take up to 0.1 additional processing units (around 5.26% additional capacity).

- To ensure that partition should always get 1.9 processing units, you should set the minimum processing units to 1.9. If that is not the case partition will not start.

- If the application running on the partition is critical to your business and you want that sufficient capacity is always available to the partition, you should set the desired processing units to at least 3.5. However in this case, if the same pattern repeats in the future, most of the time the partition will behave as a donor of unused capacity (blue region in the graph). Setting the desired virtual processors to 4 or above will ensure that partition will be able to sustain any spike beyond 3.5 processing units.

- An alternate way is to configure the partition as uncapped with entitled capacity of 1.9 and 4 virtual processors. To ensure that it should get an additional capacity when needed, you define the uncapped weight as maximum (which is 255) compared to other uncapped partition running on the same shared-processor pool. However there is a side-effect to this approach. When the partition needs additional capacity it has to compete with other partitions to get its share of spare capacity. Depending on the competition with other partitions, the partition can be scheduled on processor multiple times. As the partition share the L1 and L2 cache with other partitions, it’s quite possible that whenever it will be scheduled back on a processor first it has to warm-up its cache (read the program data into the cache for later use). This reloading of cache data may take significant time and affect partition’s performance. This behavior is nicely explained by Nigel Griffiths [#8], so to avoid redundancy I am going to exclude it. Interested readers can refer the article for further explanation.

To summarize – entitlement should be defined in such a way that it represents “nearly right” capacity requirements for a partition. Thus on average each partition’s entitled capacity utilization would be close to 100 percent and there will be a balance between capacity donors and borrowers in the system. While reviewing a partition’s utilization it’s important to know that any capacity used beyond entitled capacity isn’t guaranteed (as it might be some other partition’s entitlement). Therefore, if a partition’s entitled CPU utilization is beyond 100 percent, it might be forced back down to 100 percent if another partition requires that borrowed capacity.

Processing units also decide the number of partitions that can run on a system. As the total processing units of all partitions running on a system cannot more than the number of physical processors, by assigning smaller processing units you can maximize the number of partitions on a system.
Number of Virtual Processors

Number of virtual processors for a partition decides the parallelism of its operation, in other words how many physical processors on which work will actually be spread across. That’s why it is also called spreading factor. While spreading the work across multiple processors allow more work to be done in parallel; it can potentially reduce the effectiveness of L2 cache.

In order to reach at an optimal number of virtual processors for a partition, you should consider application characteristics and workload behavior. Multi-threaded applications typically run faster when provided with more virtual processors, because the virtual processors enable the applications to run multiple threads simultaneously (i.e., exploit parallelism). Conversely, assigning several virtual processors to an application that isn’t well-threaded may use the resources inefficiently if the application can’t take advantage of the available parallelism. Assigning multiple virtual processors to a single-threaded application is clearly a waste of resources because the application can run only one thread at once, hence all but one processor will sit idle (provided virtual processor folding is turned off). Given a well-threaded application and 1.6 processing units total, four virtual processor each with 0.4 processing units is likely a better choice than two virtual processors each with 0.8 processing units; an application that isn’t well-threaded would likely operate faster with the two 0.8 sized processors.

From workload perspective, if a partition is consuming up to 0.7 processing units, one virtual processor is enough to efficiently run the partition. Partitions consuming 0.8 – 0.9 processing units should be configured with 2 virtual processors. Similarly partitions consuming up to 1.6 physical processors should work fine with 2 virtual processors and if consuming more than ~ 1.7 processors we should consider configuring an additional virtual processor. In short, low utilized partitions should be configured with minimal number of virtual processors.

Since the number of virtual processors limits the total amount of physical processor resource that the partition can potentially consume, the choice of virtual processors assigned to a partition should also consider anticipated resource requirements. Partitions which depict utilization spikes are candidates of higher virtual processors. The exact value of virtual processors you should configure for a partition is installation-specific, but 50 to 100 percent more is reasonable. Web-based applications which are well-threaded and show workload spikes are candidates of high virtual processors up to two times of their capacity entitlement.

One interesting argument I have seen is that when processor folding feature is there, administrators set the virtual processors too high with the argument that hypervisor will take care of it and decide how many of them it needs to run the workload. It will, but setting virtual processors too high will unnecessarily spread the virtual machine across the system.

It is important to note that setting the number of virtual processors higher than number of physical processors for either partition in the system doesn’t make any sense as work can’t be spread out more than the actual number of processors.

Number of virtual processors should also be considered from software licensing perspective. Because an application has the potential of using more processor resource than the partition’s entitled capacity, many software vendors that charge on a processor basis require additional processor licenses to be purchased simply based on the possibility that the application might consume more processor resource than it is entitled. To minimize the software licensing costs, desired number of virtual processors should be kept to minimum provided there is a room for additional capacity in case of workload spikes.
The Maximum Processing Units setting defines the number of processing units that can be dynamically assigned to a partition. For example, a partition with two processing units and a maximum of six will enable system administrators to dynamically increase its processing units by allocating an additional four processing units using DLPAR operation on the HMC. Similarly the Maximum Virtual Processors setting determines the number of virtual processors that could be dynamically assigned to a partition. The important point here is that the "Desired Processing Units" and "Desired Virtual Processors" settings are the key to controlling how much uncapped resource a micro-partition can access in the shared pool. The names are misleading as it seems like that the maximum settings represent the amount of processing units that an uncapped partition can access. However that is not the case. The fact is - the maximum amount of processing power that an uncapped partition can access is limited by the "Desired Virtual Processor" setting.

**Uncapped Weight**

While adding new partitions to an existing system or tuning the weight of existing partitions, it is important to consider the effects on other partitions on the system. If a partition belonging to an application consistently exceeds its guaranteed processor allocation, changing the weight of a different partition or adding another partition might reduce the amount of resource available for the application and affect performance.

You can start with an uncapped weight of 128 for a new partition added on the system. Later by analyzing the monitoring data if you find that the partition need additional processing capacity and understand the impact of it on another partitions in the same shared-processor pool, you can increase its uncapped weight. This will increase the chances of getting additional processing capacity. You should increase the weights in small increments as it will reduce the chances of major side-effects on other more-critical partitions.

Monitor system performance and note those partitions that consistently exceed their guaranteed CPU allocation. If any of those partitions host performance-sensitive applications, it might be necessary to increase their guaranteed allocation or their weights to avoid affects from other partitions.

As explained earlier, uncapped weights decide the spare capacity allotment in case of competition. It’s too risky to rely on uncapped weights to ensure service performance level of individual partitions. A better approach is to configure non-competing partitions on a server, optimal settings of capacity entitlement and virtual processors. Clearly this needs long terms trends and multiple iterations of configuration changes before reaching to an optimal setting for your environment. As long as workload behavior is constant, an optimal setting can stay optimal for your environment. In case of business demand changes, you may need to re-assess the configuration.

**Dedicated to Shared**

Partitions running in dedicated mode depict constant performance level because dedicated-mode partitions have fixed processing capacity. Uncapped partitions in shared-mode, on the other hand, may show different performance level because they may get additional capacity depending on the virtual processers configuration and of course the spare capacity available in their shared-processor pool.

While converting dedicated mode partitions to shared mode partitions stepwise approach should be used. At first step you can convert dedicated mode partition to shared uncapped partition and set virtual processors equal to its entitled capacity. This way you not only preserve its entitled capacity (like in dedicated mode) but also donate any unused capacity to the shared processor pool. Later when you observe that partition’s entitled utilization is
running close to 100 percent then you can increase number of virtual processor one at a time. This will allow partition to borrow unused processing cycles from the shared processor pool. Configuring the partition in shared mode will also allow you to assign processor in fractions which is not the case with dedicated mode partitions.

Also after converting to uncapped shared processing mode you might get varying application response times based on the available capacity in the processor pool. The important point to note that is any capacity a partition get beyond its entitled capacity is not guaranteed and it will get away to ensure other partition’s entitled capacity or if any other partition with higher uncapped weight need the processing capacity.

Choosing a Mix of Partitions

In order to maximize the resource utilization and ensure sufficient capacity for critical partitions, many shops mix production environments and non-production environments on the same system. However a planning should be done to ensure that production and non-production peak workload times are not the same otherwise non-production partitions will consume the much-needed processing power during the peak workload time. This is seldom the case. In most of the case their peak workload timings overlap. There are few solutions to this issue:

- Have separate shared-processor pools for production partitions. But the scope of this solution is limited as multiple shared-processor pools capability is only available in Power6 and Power7 based systems.
- Configure the non-production partitions as capped. Capped partitions are restricted to consume additional processor cycles beyond their entitled capacity.
- A more flexible way is to configure the non-production partitions as uncapped and keep their uncapped weight to minimum. The number of virtual processors should be set to maximum physical CPUs which you think a partition should consume. This will effectively cap the partition at number of virtual processors. The benefits of this approach is that, non-production partitions can get additional resources up to their virtual processors but at the same time will remain harmless to production partitions with higher uncapped weights.

Section-IV Capacity Planning Case Study

The theory and techniques of capacity planning process is outside the scope of this paper. The important point to understand is that before sizing any new partition or capacity planning of an existing partition, you must consider following points:

- Determine the purpose and nature of the applications to be run on the partition, like web server supporting an online web store or batch database of a banking system.
- Understand the business workload profile.
- Identify any seasonal or periodic trends and its impact on the workload profile.
• Understanding of the busiest hour in the working day, the busiest day in the week, busiest month of the year.
• Calculate the processing requirements necessary to support workload profiles.

For illustration purpose, let’s take an example of a real-life case study where a new partition is to be added on an existing system. The new partition will support two business processes. As the functionality, profile and resource utilization characteristics are completely different for these processes, we will measure them separately from sizing and monitoring perspective. It is always better to measure and forecast the capacity in business metric terms because that’s what business understands and same units are used by business to perceive the performance, throughput and forecast the business demand. We will call our business metrics as metric1 and metric2.

The sizing exercise starts with getting the anticipated workload profile (or in other words volumetric) from the business or application teams. Table given below shows total anticipated transaction counts along with business/non-business transaction proportions:

<table>
<thead>
<tr>
<th>Business Metric</th>
<th>Daily Total</th>
<th>Business Hours</th>
<th>Non-business Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric 1</td>
<td>50,000 - 60,000</td>
<td>80%</td>
<td>20%</td>
</tr>
<tr>
<td>Metric 2</td>
<td>1,500 - 2,000</td>
<td>85%</td>
<td>15%</td>
</tr>
</tbody>
</table>

The transactions profile for metric1 is more like online transaction which happens mostly during day/evening. Some examples of such transactions are Internet banking transactions, eCommerce transactions. Metric2 is more like batch transactions, which are more controlled and mostly run during night hours.

As this is a new application deployment, we sized the partitions based on the performance testing results. Multiple iterations of results show that on average 1.1 – 1.5 physical CPUs are used whereas peak CPU usage is around 1.9 – 2.2

Based on these results we have configured a partition with the following parameters.

• Min Processing Units: 2.0
• Desired Processing Units (Entitled CPU): 2.0
• Max Processing Units: 4.0
• Desired Virtual CPUs: 3
• Max Virtual CPUs 4

Considering that the partition will support critical business functions, we have configured the partition in shared uncapped mode with 3 virtual CPUs so that it can sustain unexpected workload.

Now let’s suppose business want us to plan the capacity for additional workload due to the coming festival season and make sure cost-effective capacity is always available to support the services.
In order to forecast the capacity of the partition, we have collected 8-days data comprise of peak workload and complete business cycle (weekends, weekdays etc). It is important to note that the baseline data we collect to forecast the capacity is representative data which truly reflects the actual workload and contains all the cyclic/seasonal workloads.

Figures given below show the hourly transactions profile of both business processes. The business and non-business transaction clusters are clearly visible in the graphs. It is observed that peak transactions per hour (tpH) for metric1 is observed around 4,200 at 21:00 – 22:00 hours, on the other hand metric2 peaked around 245 at 2:00 - 3:00 hours.

The CPU utilization view is interesting. During peak business hours the physical CPUs used were in the range of 2.0 to 2.6 which means 30% more than its entitled capacity.
The next view is the correlation plot between CPU utilization and the number of transactions. With some time lag, CPU utilization is significantly correlated with the number of metric1 and metric2 transactions. The overlap of metric1 and metric2 workload during 21:00 – 22:00 hours can be accounted for comparatively high CPU utilization during these hours.

In order to see the impact of additional workload on CPU utilization by the partition, we will use a statistical technique called “Regression Analysis”. Those who are not familiar with it can refer my other paper titled “Linear Regression: An Approach for Forecasting” [#9].

To summarize, regression analysis is a statistical method to investigate relationships among variables and can potentially be used for forecasting. Without getting into much depth, I will summarize the important points. Based on the 8 days data explained earlier, the adjusted R-square between CPU utilization and metric1, metric2 is calculated as 0.908 which implies quite significantly metric1 and metric2 workload is driving CPU utilization of the partition.

The multiple regression equation in this case can be stated as:

\[
\text{Entitled CPU utilization} = 0.377680 + 0.000162 \times \text{Metric1} + 0.001547 \times \text{Metric2}
\]

Now this is our forecasting model. As long as the trend observed during 8 days of data collection is going to continue, this model will hold true and will give us satisfactory results.

Now considering the peak-hours workload as baseline, currently we have around 4,200 metric1 transactions and at the same time around 110 metric2 transactions. The milestone 1 which is going to hit in next 3 months will put 45% and 25% additional metric1 and metric2 workload respectively, which means 6,090 and 138 transactions. Replacing these value in our forecasting model,

\[
\text{Peak entitled CPU utilization} = 0.377680 + 0.000162 \times 6090 + 0.001547 \times 138
\]

\[
= 157.76\%
\]

Putting the business milestone 2 workload values into the equation,

\[
\text{Peak entitled CPU utilization} = 0.377680 + 0.000162 \times 7560 + 0.001547 \times 154
\]

\[
= 184.15\%
\]
Hence the estimated peak entitled CPU utilization for additional workload would be 157.76% and 184.15% respectively, which means 3.15 and 3.68 Physical processors respectively.

Clearly current value of entitled capacity of 2.0 processing units is not going to support additional workload. So based on this analysis, we should increase the entitled CPUs to 4 and to keep some margin for unexpected workload, set the virtual processors to 5 or 6. Another option which is worth considering for reducing the pressure on additional processing capacity is to shift metric2 workload by few hours, if possible. It will reduce the chances of running two business processes at the same time and result in CPU spikes. Such workload management options should be more important from the business perspective than their technical implications.

I have simplified the illustration a lot but the principle of capacity planning would be the same.

Conclusion

Processor virtualization is an integral part of today’s Infrastructure. Organizations realized several benefits, including increased utilization, reduced IT and facilities cost, flexibility. But processor virtualization is just the tip of the iceberg. Latest technologies took the virtualization to the whole new level and set the foundation for today’s much-spoken Cloud computing framework. Active Memory Sharing allows physical memory sharing to be shared among multiple partitions, Virtual I/O server allow sharing of physical I/O resources among multiple partitions.

As with any technology, you need to sharpen your pencils to figure out how processor virtualization can be used in your environment. Monitoring and alerting are the keys to understand - what you have, how much you are using and when you can run out of capacity.
Bibliography

7. Local, Nar & Far Memory part 6 - Too High a Virtual Processor number has a Bad Side Effect by Nigel Griffiths, September 2011, IBM developerWorks
8. Local, Nar & Far Memory part 5 - Low Entitlement has a Bad Side Effect by Nigel Griffiths, September 2011, IBM developerWorks

About the Author

Neeraj Bhatia is a researcher, writer, IT Service/Infrastructure Capacity Planner. He has 7 years of IT experience in the areas of Oracle Database Administration, Performance Optimization and Capacity Planning. He has around 2 years of experience in Optimizing and Capacity Planning of IBM PowerVM systems. Currently Neeraj is a Capacity Analyst in Barclays Bank PLC, based in Pune, India.

When not working and researching, he likes to watch movies, cook, play acoustic guitar, or just relax by listening to good music. He can be reached at neeraj.dba@gmail.com