close window

Web Exclusive

Print

VIOS 101: CPU and Processors

April 2015 | by Jaqui Lynch

IBM has provided virtualization technologies on its POWER servers since at least 2004 and now provides a variety of options such as Dynamic LPARs, PowerVM and Workload Partitions. Dynamic LPARs are the foundation for virtualization, allowing for multiple partitions on a server using dedicated I/O and whole CPU and memory increments. PowerVM takes this many steps further introducing virtual and/or dedicated I/O, micropartitions, shared processor pools and many other technologies. These virtualization techniques apply to AIX, IBM i and Linux. Although this type of virtualization has been around for a long time, a significant number of IBM i clients are now moving to a virtual environment so it seems like this is a good time to go over the basics of PowerVM virtualization.

What Is PowerVM?

PowerVM is the software from IBM that provides the capability to divide up the resources in a server among multiple LPARs. Standard Edition provides for shared processor pools, shared storage pools, virtual storage and networks, NPIV and SR-IOV, shared dedicated capacity and multiple shared processor pools. Enterprise Edition adds on Active Memory Sharing (AMS), PowerVP and live partition mobility (LPM). Many of these functions require the use of a custom LPAR called a VIOS (Virtual I/O server) and we'll look at each of these functions in detail.

IBM's POWER virtualization is built on the basics of LPAR technology. POWER servers (since POWER5) always run under a Hypervisor and the Hypervisor takes care of all the time slicing and dispatching between processors. PowerVM allows those LPARs to allocate fractions of cores (increments of 1/10 or 1/20 depending on the server) rather than having to use dedicated cores. Basically, all active cores in the server are in the default shared processor pool (SPP) until they get allocated to LPARs. Processors can be assigned as dedicated (which means no one else can use them), shared dedicated (which means other LPARs can use the cores if this LPAR is not) or shared.

Terminology

To truly understand how this works, we need to go over some important terminology.

Minimum, desired and maximum
Processor and memory settings have three values—a minimum, a desired and a maximum. For the most part, we ignore minimum and maximum. Minimum is used at boot time and is the minimum resource required to boot. It's also used during a DLPAR operation to ensure the minimums are being met. Maximum is the maximum the memory or CPU can grow to during a DLPAR operation. All of the dynamic changes are dependent on the desired setting and aren't impacted by maximums with the exception that desired can't be set larger than maximum. Minimum and maximum have nothing to do with resource allocation during normal operation—they only come into play when making a dynamic change later.

Processors and Processor Units
Dedicated LPARs allocate processors in whole physical core increments. Once the LPAR is booted, there's a one-to-one relationship between the physical core and the representation of that core in the LPAR. When the LPAR core gets dispatched, it will normally go to the same physical core each time. In a

dedicated LPAR, you provide minimum, desired and maximum cores.

Shared processor pools are different. In a shared processor pool LPAR, it's necessary to allocate processor units and virtual processors (VPs) as well as specifying weights and capping information. Processor units allow the physical processors to be allocated in fractional increments, and are allocated with a minimum of 1/10 or 1/20 depending on the server. As with dedicated cores, there's a minimum, desired and maximum setting for this. But unlike dedicated cores, these can be fractions rather than whole cores. The desired setting represents the amount of CPU the LPAR is guaranteed to run with. It's also referred to as the entitlement and sometimes called the guarantee. So if the LPAR has an entitlement of 0.5, that means it's entitled to half a core or 5ms of dispatch time. We then have to look at VPs to see how that gets broken up. When a VP for an SPP LPAR gets dispatched, it will get dispatched to any available core in the shared processor pool. At the most basic level, the shared processor pool will consist of any active core that isn't assigned to a dedicated LPAR. The dispatcher does try to send VPs back to the last physical core they ran on but that's not guaranteed.

It should also be noted that there's also the capability to have multiple shared processor pools. This is done primarily for licensing purposes. You may have 16 cores but only four DB2 licenses so you set up a DB2 pool that cannot exceed using four cores.

Virtual Processor

Every LPAR has VPs. Dedicated LPARs use them under the covers. These represent the actual assigned cores whether they're shared or dedicated. In a dedicated LPAR that has been allocated two cores, then the VPs would be two cores. VPs in the shared pool have a minimum, desired and maximum setting. In our shared processor pool above, we had assigned 0.5 cores as the entitlement or desired processor units. If our desired VPs are set to 1 then we would have one VP that's guaranteed 0.5 of a core but that can dynamically grow to a whole core if the LPAR is set to uncapped. The dispatch time for that VP is 5ms guaranteed. If VPs were set to 2 then we would have 2 VPs, each with a guarantee of 2.5ms (our half a core is divided between them).

What does it mean that the LPAR can dynamically grow? An LPAR is set to be either capped or uncapped. If it's capped then it can never grow beyond its entitlement. So if we were to cap this LPAR, it could never use more than 0.5 of a core. However, if the LPAR was set to uncapped, then it can grow to use as many cores as there are VPs as long as no one else is using those cores. So, if we have an entitlement of 0.5 cores and two VPs and are uncapped then we could potentially use two cores if we needed them and if they are available.

The number of virtual processors specified for an LPAR represents the maximum number of physical cores that the LPAR can consume. Never configure more VPs for an LPAR than the number of physical cores in the shared pool as this will cause serious dispatching issues.

Weights

Additionally, the capability to grow beyond entitlement is impacted by the setting of the weight value for an SPP LPAR. The default weight is 128. Let's look at two LPARs, each with an entitlement of 0.5 and VPs set to two and both are uncapped with a weight of 128. The server has three cores. Each LPAR is guaranteed 0.5 of a core and can grow to use two cores if there's no contention. But let's say both LPARs need to grow to two cores at the same time—there are only three cores and combined they need four. This is where weights come into play. Each LPAR gets its entitlement so one core is now spoken for. The final two cores (each LPAR wants 1.5 more) will be allocated out based on the weights in a shares-based system. In our case, both LPARs have the same weight so each LPAR will get 50 percent of the available core. Hence, each LPAR will get a total of 1.5 cores. This is why we recommend giving VIOs the highest weight, production the next highest weight and leave the other LPARs as defaults or something smaller.

It should also be noted that setting the weight on an LPAR to 0 caps the LPAR. LPARs can be capped in three ways—setting them to capped, setting the weight to 0 or by the setting for desired VPs.

Since VPs represent cores, they're always allocated in whole increments. Desired VPs can also not exceed 10X the PUs (or 20X depending on the server), however having a high VP to entitlement ratio can cause affinity issues so don't automatically allocate 1/10 of a core and one VP.

SMT

This stands for simultaneous multithreading and is set within the LPAR. Different LPARs can have different SMT settings. Over time, various options have been available ranging from SMT1 (basically no SMT), SMT2, SMT4 and now SMT8. The easiest way to understand SMT is to think of it as a form of multithreading through a single core/VP. Effectively, several of the registers have been duplicated so multiple threads can run on the core at the same time as long as there are enough registers to do so. SMT threads are represented by logical processors and show as LCPU in commands such as vmstat. In SMT1 each processor (dedicated or virtual) appears as one logical processor to the operating system. SMT4 would show as four logical CPUs and so on. Each logical processor can run a thread.

So for a typical shared LPAR we will see something like:

```
                PUs     VPs     LCPU (SMT4)
Minimum         0.1     1
Desired 0.5     2       8
Maximum 4       8
```

POWER8 and POWER7/7+ LPARs default to SMT4, however in POWER8 the LPAR can run in SMT8 if it's running AIX v7.1. SMT is about providing more throughput (multithreading), not increasing individual core performance. It will improve throughput on a heavily loaded system but typically doesn't make a single thread run faster. Changes were made to the algorithms in POWER7 and POWER8 to try to provide the best balance between individual core performance and throughput.

SMT provides the greatest benefit when there are numerous concurrently executing threads with a mixed workload (e.g., Web servers or database servers). At dispatch time, the unit of dispatch is the VP. All of the SMT threads associated with the VP will be dispatched (if they have work assigned to them) when the VP is dispatched.

More to Come

In Part 1, we have covered information about virtualization of processors. In the next article, we'll look at memory and virtualization of I/O (network and disk). Part 3 will cover more advanced topics around the VIO server and PowerVM virtualization.

IBM Systems Magazine is a trademark of International Business Machines Corporation. The editorial content of IBM Systems Magazine is placed on this website by MSP TechMedia under license from International Business Machines Corporation.

©2019 MSP Communications, Inc. All rights reserved.

References

< Return to main article

ADVERTISEMENT

Print Email

PowerVM Enhancements in 2013

http://publib-b.boulder.ibm.com/abstracts/sg248198.html?Open

PowerVM Virtualization Introduction and Configuration

http://publib-b.boulder.ibm.com/abstracts/sg247940.html?Open

PowerVM Virtualization Managing and Monitoring

http://publib-b.boulder.ibm.com/abstracts/sg247590.html?Open

PowerVM Best Practices

http://publib-b.boulder.ibm.com/abstracts/sg248062.html?Open

< Return to main article

IBM Systems Magazine is a trademark of International Business Machines Corporation. The editorial content of IBM Systems Magazine is placed on this website by MSP TechMedia under license from International Business Machines Corporation.

©2019 MSP Communications, Inc. All rights reserved