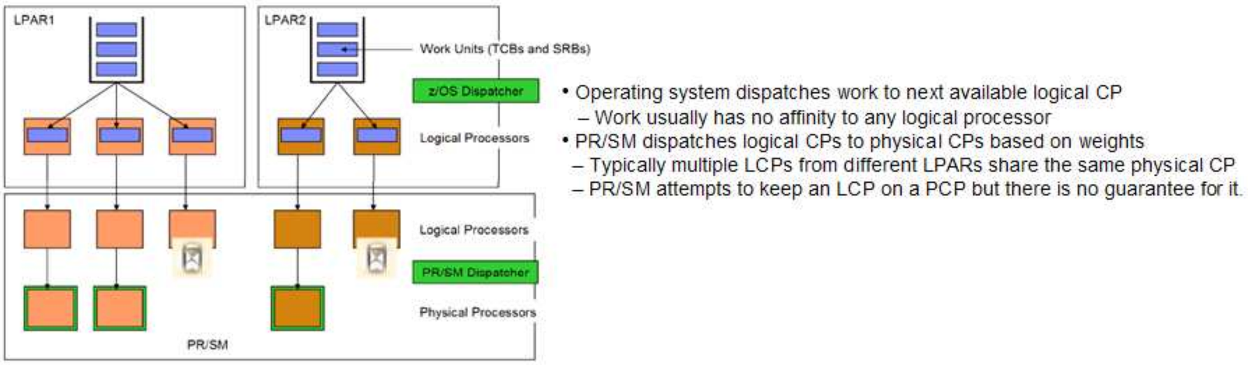
Prologue -

In addition to the performance improvements available with the IBM System z processors, z/OS workload management and dispatching are enhanced to take advantage of the System z hardware design. A mode of dispatching called HiperDispatch provides additional processing efficiencies.

The HiperDispatch mode aligns work to a smaller subset of processors to maximize the benefits of the processor cache structures, and thereby, reduce the amount of CPU time required to execute work. Access to processors has changed with this mode, and as a result, LPAR weights prioritization of workloads via WLM policy definitions becomes more important.

The Concept of HiperDispatch Mode

Without HiperDispatch, for all levels of z/OS, a TCB or SRB may be dispatched on any logical processor of the type required (standard, zAAP or zIIP). A unit of work starts on one logical processor and subsequently may be dispatched on any other logical processor. The logical processors for one LPAR image will receive an equal share for equal access to the physical processors under PR/SM™ LPAR control. For example, if the weight of a logical partition with four logical processors results in a share of two physical processors, or 200%, the LPAR hypervisor will manage each of the four logical processors with a 50% share of a physical processor. All logical processors will be used if there is work available, and they typically have similar processing utilizations.



- Operating system dispatches work to next available logical CP
  - Work usually has no affinity to any logical processor
- PR/SM dispatches logical CPs to physical CPs based on weights
  - Typically multiple LCPs from different LPARs share the same physical CP
  - PR/SM attempts to keep an LCP on a PCP but there is no guarantee for it.

With HiperDispatch mode, work can be managed across fewer logical processors. A concept of maintaining a working set of processors required to handle the workload is introduced. In the previous example of a logical partition with a 200% processor share and four logical processors, two logical processors are sufficient to obtain the two physical processors worth of capacity specified by the weight; the other two logical processors allow the partition to access capacity available from other partitions with insufficient workload to consume their share. z/OS limits the number of active logical processors to the number needed based on partition weight settings, workload demand and available capacity. z/OS also takes into account the processor topology when dispatching work, and it works with enhanced PR/SM microcode to build a strong affinity between logical processors and physical processors in the processor configuration.

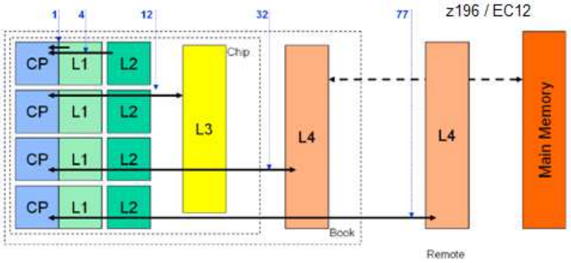
On today's z/OS environments there are always two dispatching (or work scheduling) processes:

1. On the z/OS system where the logical processors select the ready TCBs and SRBs from the dispatcher queue. There is in general 1 dispatcher queue for all regular (or logical) processors of the system (also additional dispatcher queues for offload processors (zIIPs and zAAPs). If a system contains many logical processors it is unpredictable which logical processor will select a ready TCB or SRB. Therefore a unit of work can be dispatched across all possible logical processors.
2. Within the z/OS Hipervisor. The Hipervisor or PR/SM dispatches logical processors of the partitions to physical processors. PR/SM always attempts to dispatch a logical

processor back to the same physical processor or if this is not possible at least to a physical processor of the same book. But that can't be guaranteed and therefore it is also possible that logical processors float across the physical processor configuration.

The disadvantage of this type of dispatching is that a unit work which was first dispatched on a logical processor 1 could be dispatched next on a logical processor 15 on a larger partition. Even if PR/SM achieves that the logical processors will be redispatched on the same physical processor or book it is still possible that the unit work finds itself on a different physical processor or even different book just because of the z/OS dispatching process. So there is a high likelihood that it must regain its cache context either from memory or remote level 2 caches which has an impact on the execution time of the work and thus an impact on the throughput of the system.

HiperDispatch: Motivation

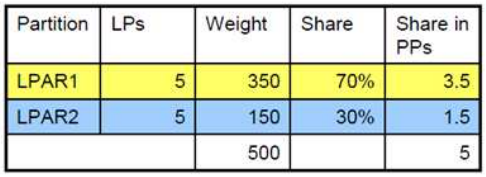


- Cache latency for a System z (1, 4, 12, 32, 77 are relative access times)
- On the latest System z there is an additional cache level
  - – Access to L3 which is shared by 4 LCPs is faster than the access to "old" L2 on previous System z machines.
  - – But access to L4 ("old" L2 on previous System z) is worse
- Result:
  - – Motivation increases to improve the performance of the system.

For the latest System z the motivation becomes even higher than for previous mainframe machines where an additional cache structure has been introduced. The processors of a book are now organized in chips with an additional cache on the chip. Also the cache structures have been renamed on new System z machines. It can be observed that the access to the local L4 structure which was the local L2 cache structure on previous z machines is now a little more expensive. At the same time the access to chip cache structure (L3) is more effective. So now there is even higher motivation to not only keep dispatches local to a book but also local to a chip.

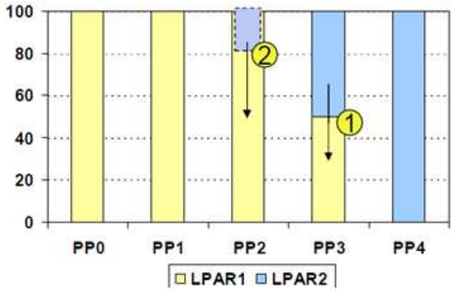
In the next step we map this result to the logical processors and we start to differentiate logical processors:

- Those logical processors which could fully use a physical processor are named **High** processors and we assign a physical processor share of 100 to them
- The remainder of the previous calculation is used to define a shared or now called **Medium** processor which can use a physical processor only for a limited time
- Finally all the logical processors which have been defined in excess to the partition share are named **Low** processors and they do not get a processor share initially.



Partition	LPs	Weight	Share	Share in PPs
LPAR1	5	350	70%	3.5
LPAR2	5	150	30%	1.5
		500		5

Example
- Assignment of logical processors to physical processors in Hiperdispatch mode
- LPAR1
  - 3 physical processors (**High** Processors)
  - Share of 50% of the 4th processor (**Medium** Processor)
- LPAR2
  - 1 physical processor
  - Share of 50% of the 4th processor



What about the "un-used" share of physical processors?
- 1.5 for LPAR1 and 3.5 for LPAR2
  - **Low** Processors (**parked** = not used)
- If demand exists AND the other partition does not need its share
  1. **Medium** processors can use up to all of their physical processors
  2. **Low** processors can be **un-parked** and start to use physical processors which are not needed by other partitions

The example shows a small system with two partitions LPAR1 and LPAR2. Based on the partition weights the share of LPAR1 results in 3 High and 1 Medium processor with a processor share of 50%. For LPAR2 the calculation results in 1 High and 1 Medium processor. Because there are 5 logical processors are defined for both partitions 1 logical processor for LPAR1 and 3 for LPAR2 are treated as low processors. They are not used initially and placed in a so called park state. As long as both partition have high demand the assigned processors for LPAR 1 and LPAR2 reflect the share and they are sufficient for processing. The benefit of the high processor is now that they get a physical processor assigned and that PR/SM will always re-dispatch them on the same physical processor.

We now assume that LPAR1 has low demand and LPAR2 has high demand. LPAR2 can now use more CPU capacity than it is entitled too because of its weight. So the low processors for LPAR2 must be used. This is done by un-parking the low processors and PR/SM will then try to dispatch them on physical processors which are not used by LPAR1. As we can see it is necessary to have a mechanism which parks and unparks the low processors and also which ensures that they can use physical processors efficiently.

Processor Categories

The logical processors for a partition in HiperDispatch mode fall into one of the following categories:

- Some of the logical processors for a partition may receive a 100% processor share, meaning this logical processor receives an LPAR target for 100% share of a physical processor. This is viewed as having a high processor share. Typically, if a partition is large enough, most of the logical partition’s share will be allocated among logical processors with a 100% share. PR/SM LPAR establishes a strong affinity between the logical processor and a physical processor, and these processors provide optimal efficiencies in HiperDispatch mode.
- Other logical processors may have a medium amount of physical processor share. The logical processors would have a processor share greater than 0% and up to 100%. These medium logical processors have the remainder of the partition’s shares after the allocation of the logical processors with the high share. LPAR reserves at least a 50% physical processor share for the medium processor assignments, assuming the logical partition is entitled to at least that amount of service.
- Some logical processors may have a low amount, or 0%, of physical processor share. These “discretionary” logical processors are not needed to allow the partition to consume the physical processor resource associated with its weight. These logical processors may be parked. In a parked state, discretionary processors do not dispatch work; they are in a long term wait state. These logical processors are parked when they are not needed to handle the partition’s workload (not enough load) or are not useful because physical capacity does not exist for PR/SM to dispatch (no time available from other logical partitions).

When a partition wants to consume more CPU than is guaranteed by its share and other partitions are not consuming their full guaranteed share, a parked processor can be unparked to start dispatching additional work into the available CPU cycles not being used by other partitions. An unparked discretionary processor may assist work running on the same processor type. When examining an RMF CPU activity report in HiperDispatch mode, one may now see very different processing utilizations across different logical processors of a logical partition.

Setting of the HiperDispatch Mode in SYS1.PARMLIB

The HiperDispatch state of the system is determined by the number of logical processors defined on an LPAR and the HIPERDISPATCH=YES|NO keyword IEAOPTxx of SYS1.PARMLIB.

All partitions with more than 64 logical processors defined at IPL are forced to run with HIPERDISPATCH=YES. LPARs with more than 64 logical processors defined are also unable to switch into HIPERDISPATCH=NO after IPL.

For all partitions with less than 64 logical processors HIPERDISPATCH is enabled or disabled by the HIPERDISPATCH=YES|NO keyword in parmlib member IEAOPTxx. This parameter can be changed dynamically with the use of the SETOPT command. This enables the operating system to choose the desired mode of operation.

When a new hardware generation is installed, for any z/OS image(s) that are running with HiperDispatch disabled, the system programmer should reevaluate whether those z/OS image(s) should be migrated to running with HiperDispatch enabled in the new environment. On earlier System z and z/OS releases, HiperDispatch disabled is the default. However, customers are encouraged to run with HiperDispatch enabled on z10 and later machines to take advantage of the processing benefits.

Beginning with z/OS V1R13 on IBM zEnterprise machines, HiperDispatch enabled is the default. With z/OS V1R13 running on a z196 or EC12, z/OS partitions with share greater than 1.5 physical processors will typically experience improved processor efficiency with HiperDispatch enabled. z/OS partitions with share less than 1.5 physical processors typically do not receive a detectable performance improvement with HiperDispatch enabled, but IBM recommends running those LPARs with HiperDispatch enabled when the performance improvement is greater than or equal to HiperDispatch disabled.

– – –