Robert Vaupel
STSM z/OS Workload Management
vaupel@de.ibm.com

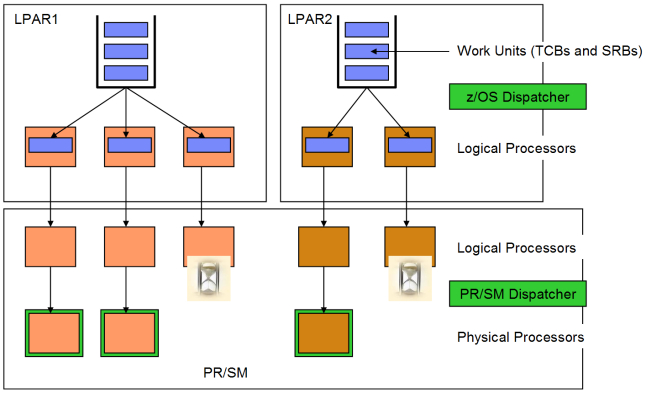
WLM and Hiperdispatch

- Dispatching in z/OS and LPAR
  - What is the problem and rationale for Hiperdispatch
- Hiperdispatch Function
  - New Terminology for Processors
  - What are Processor Shares?
  - How can this be observed thru RMF?
  - What does z/OS WLM do?
  - Special Processors
  - User interface
- Hiperdispatch Benefit on z196

Version 9a

This presentation gives and overview of Hiperdispatch how it is integrated in LPAR and z/OS. The presentation introduces the problem being addressed with Hiperdispatch and explains the new terminology with it.

Dispatching: z/OS and PR/SM



- 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

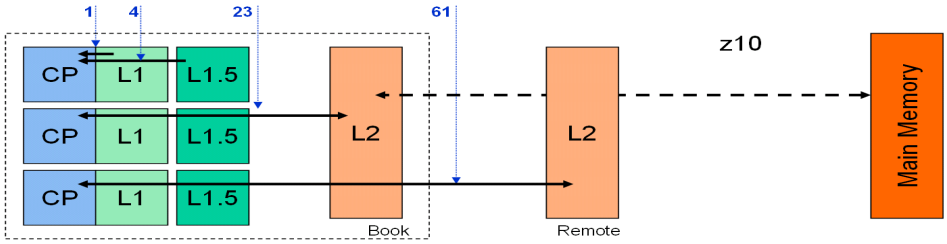
On today's System z environments there are always two dispatching (or work scheduling) processes:

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. PRE/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 re-dispatched 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 z10 system (1, 4, 23, 61 are relative access times)



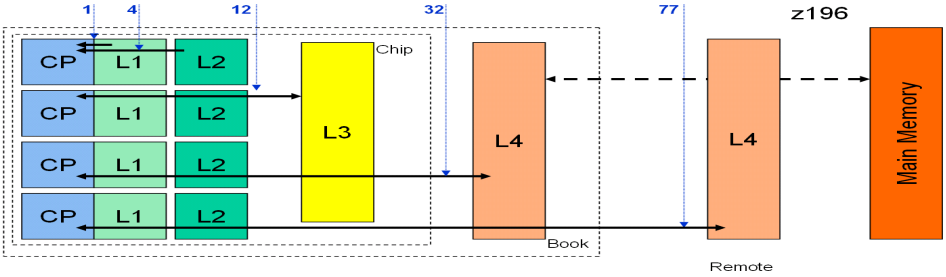
- Remarks: Dispatching w/o Hiperdispatch
  - PR/SM dispatching attempts to re-dispatch a logical processor on the same physical processor but can't guarantee that
  - In z/OS all logical processor select work from the same work unit queue
    - Therefore it is completely unpredictable where a UoW lands
- Result:
  - On a large scale computer this will result in un-wanted access to remote L2 caches
  - Thus the environment does not optimal scale

With System z10 the frequency of the processor became much faster then on any previous System z generations. As a result the access to memory or Level 2 cache gets longer in the sense that it requires more cycles where the processor must wait before the memory context has been retrieved. In addition the higher frequency also puts more limitations on Level 1 cache because the processor should be able to access information within 1 cycle of the Level 1 cache. An additional cache was introduced as a first improvement to this problem. The Level 1.5 cache is a processor only cache which allows much faster access than the Level 2 cache which is shared between the processors of the same book.

The graphic above shows the relative access times of a processor to the various cache structures. Access to the local L2 cache is 23 times more expensive than access to the Level 1 cache and retrieving information from the remote Level 2 cache is again 3 times more expensive than retrieving information for from the local L2 cache. As a result the desire comes up to improve dispatching in order to reduce the potentially required access to remote L2 cache structures and at the same time to increase the possibility that a unit of work can find cache context already within the L1 and L1.5 cache structures.

Hiperdispatch: Motivation

- Cache latency for a z196 system (1, 4, 12, 32, 77 are relative access times)



- On a z196 there is one more cache level
  - Access to L3 which is shared by 4 LCPs is faster than the access to “old” L2 on z10
  - But access to L4 (“old” L2 on z10) is worse
- Result:
  - Motivation increases to improve the performance of the system

For z196 the motivation becomes even higher than for z10. Now a another 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 z196. It can be observed that the access to the local L4 structure which was the local L2 cache structure on z10 is now a little more expensive than on z10. 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.

Hiperdispatch: Motivation …

- Design Objective
  - Keep work as much as possible local to a physical processor to optimize the usage of the processor caches
  - Expected Result
    - Cache reloads shall occur much less often
    - Cache misses and fetches from other books (and chips) should be avoided as much as possible
- Function: Hiperdispatch
  - Interaction between z/OS and PR/SM to optimize work unit and logical processor placement to physical processors
  - Consists of 2 parts
    - In z/OS (sometimes referred as Dispatcher Affinity)
      - → Because it attempts to create a temporary affinity between work and processors
    - In PR/SM (sometimes referred as Vertical CPU Management)
      - → Because it attempts to assign physical processors exclusively to logical processors (as much as possible)

Based on the previous discussion we can summarize the motivation for introducing Hiperdispatch as an optimized dispatching on large virtualized systems. Also based on the previous discussion it can be seen that Hiperdispatch is a function of the LPAR Hipervisor (PR/SM) and the z/OS operating system because both have to act together to ensure that dispatching can be optimized.

In the following discussion we will understand why the PR/SM function is named “Vertical CPU Management”. The idea is to drastically improve the current attempts of PR/SM to re-dispatch logical processors on the same physical processor. We will also understand why the z/OS function is named dispatcher affinity. We already saw that it is possible and not advantageous if Z/OS potentially re-dispatches a units of work on any possible logical processor.

Hiperdispatch: PR/SM

- Optimize the number of logical processors to the minimum number needed of physical processors
- Based on the share of the logical partition

$$\text{Share(LPAR i)} = \frac{\text{Weight(LPA Ri)}}{\sum_{j=1}^{n} \text{Weight(LPA Rj)}}$$

$$\#\,\text{PP(LPARi)} = \text{Share(LPAR i)} \bullet \text{Total\_\# \_of\_PP}$$

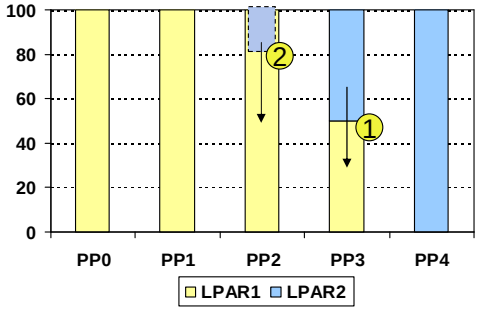
- Result
  - Form N.M with
    - N = number of physical processors which can be used completely by this partition
    - M = the fraction of a physical processor which must be used to satisfy the share of the partition

In order to optimize the access of logical processors to physical processors we have to determine the share of a partition and we are interested in the number of physical processors which can be fully used by a partition. The result of this calculation is a share in physical processors in the form N.M with N being the physical processors which could be solely used by this partition and M the fraction or remainder of the share.

Hiperdispatch: PR/SM …

- 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

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



In the next step we now try to 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.

The example now 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.

Hiperdispatch: RMF Example for Processor Types

C P U A C T I V I T Y

z/OS V1Rxx SYSTEM ID SMPX DATE 02/02/2009
CONVERTED TO z/OS V1Rxx RMF TIME 09.10.00

-CPU 2097 MODEL 729 H/W MODEL E64 SEQUENCE CODE 00000000000D56B2 HIPERDISPATCH=YES

0---CPU---			TIME %			LOG PROC		--I/O INTERRUPTS--	
NUM	TYPE	ONLINE	LPAR BUSY	MVS BUSY	PARKED	SHARE %		RATE	% VIA TPI
0	CP	100.00	93.57	95.03	0.00	100.0	HIGH	170.3	38.39
1	CP	100.00	96.07	97.05	0.00	100.0	HIGH	153.9	36.05
2	CP	100.00	94.52	95.65	0.00	100.0	HIGH	107.2	36.79
3	CP	100.00	94.26	95.34	0.00	100.0	HIGH	82.47	36.90
4	CP	100.00	92.45	94.11	0.00	100.0	HIGH	138.4	43.39
5	CP	100.00	95.39	96.63	0.00	100.0	HIGH	132.3	39.30
6	CP	100.00	93.47	94.66	0.00	100.0	HIGH	83.12	43.40
7	CP	100.00	93.55	94.82	0.00	100.0	HIGH	71.77	44.75
8	CP	100.00	89.38	91.71	0.00	100.0	HIGH	206.5	47.12
9	CP	100.00	94.33	96.00	0.00	100.0	HIGH	189.2	44.20
A	CP	100.00	90.93	92.47	0.00	100.0	HIGH	128.2	44.77
B	CP	100.00	90.57	92.32	0.00	100.0	HIGH	117.3	46.36
C	CP	100.00	91.09	92.70	0.00	100.0	HIGH	17204	15.92
D	CP	100.00	82.66	92.58	0.00	94.2	MED	104.1	48.21
E	CP	100.00	42.84	92.08	46.65	0.0	LOW	0.00	0.00
F	CP	100.00	39.49	92.52	51.33	0.0	LOW	0.00	0.00
10	CP	100.00	60.68	92.74	26.50	0.0	LOW	0.00	0.00
11	CP	100.00	56.99	94.10	31.33	0.0	LOW	0.00	0.00
12	CP	100.00	0.02	-----	100.00	0.0	LOW	0.00	0.00
13	CP	100.00	0.02	-----	100.00	0.0	LOW	0.00	0.00
TOTAL/AVERAGE			74.62	94.15		1394		18889	18.28

High Processors

Medium Processor

Low Processors
- 4 were partially un-parked
- 2 were always parked

Share of the Partition in Physical Processor (13.94)

The new state of processors can be easily observed in the RMF CPU Activity Report.

•For High Processors the “SHARE%” is always 100 and the state is also displayed right to the share number for z/OS V1.11 and above.

•For medium processors the remainder of the calculation is shown.

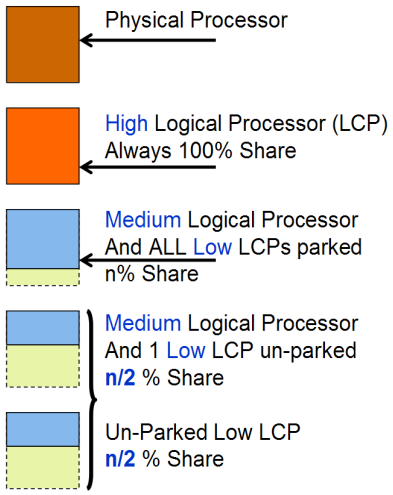
•For Low processors a “SHARE%” of 0 is depicted. We will see that this only reflects the share when the processor is parked.

A new column “PARKED” tells the you how long a low processor was parked meaning not used during the reporting interval.

Another important observation will be that the meaning of MVS BUSY changes with HIPERDISPATCH=YES. MVS BUSY only reflects how busy the z/OS system was for all un-parked processors. That means low processors which were in the park state are not used to determine this value. As a result MVS BUSY will be usually higher than on systems which ran with HIPERDISPATCH=NO before. It is also worth to mention that the meaning of LPAR BUSY hasn’t changed and still reflects the usage of all logical processors. It must be understood that this is meaningful because MVS BUSY reflects the state from the z/OS perspective and LPAR BUSY is PR/SM view.

Hiperdispatch: Processor Share

- Processor Share: M.N
  - Example: 13.94
    - 13 High Processors à 100% share
    - 1 Medium Processor with 94% share if all lows are parked
    - With 1 low unparked
      - → Medium: 47% share
      - → Unparked Low: 47% share
- High Logical Processor
  - Always 100% Share
  - That means
    - Always re-dispatched to its physical processor whenever it has demand
- Medium and Low Processors
  - Divide the share of the medium processors between them
  - That means
    - The share decreases per processor when more low processors become un-parked



➔ Share of Medium processor is used to “fuel” low processors

Finally we need to understand how PR/SM really manages the logical processors. PR/SM dispatches logical processors based on their share and how much of the processor share has been used for a specified time interval. The PR/SM Planning Guide describes processor shares in detail.

For High processors the share is always 100% and therefore they can always use their assigned physical processors when needed. The medium processor gets the remaining share assigned. So for the previous example the remainder is 0.94 which means that the processor gets a share of 94% assigned. The interesting part now starts when low processors are unparked. Because they also need a share the share of the medium processor is divided between the unparked low processors plus the medium processor. That nmeans if 1 low processor is unparked the low processor and the medium processor now have a share of 47% each and if another low processor is unparked the share is 31.3% for the medium and unparked low processors each.

As a result we can see that the share of the non-High processors gets reduced with each unparked low processor. So we still need to ensure that the unparked processors can really use the physical processor capacity and we can also see that it is not meaningful to keep low processors unparked but to optimize their usage.

Hiperdispatch: PR/SM Part …

- Optimization for medium share processors
  - If M is too small (M < 50%) the number of medium share processors for a partition is increased by 1 and the number of high share processors is reduced by 1
    - This is done to avoid that the logical processor receives a too small fraction of the physical processors
  - Calculation:

$$\text{Share(LPARi)} = \frac{\text{Weight(LPARi)}}{\sum_{j=1}^{n} \text{Weight(LPARj)}}$$

$$\#\text{PP(LPARi)} = \text{Share(LPARi)} \bullet \text{Total\_\#\_of\_PP} = \text{N.M}$$

$$\text{IF M} < 0.5 \text{ THEN } \{\text{N} = \text{N} - 1; \quad \text{M}_{\text{NEW}} = \frac{1 + \text{M}}{2}\}$$

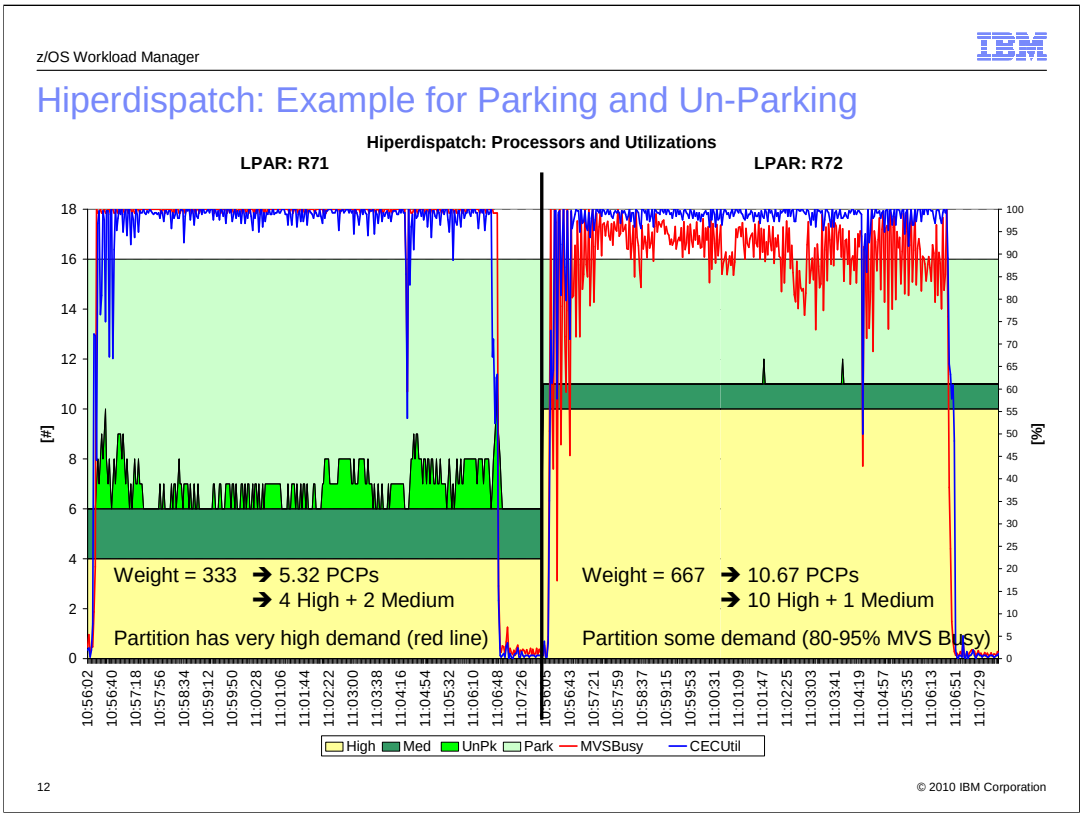
Another aspect is what should be done when the remainder of the calculation is too small? What if the share of the medium processor is already smaller than 50%? Unparking low processors will now very fast result in processors with too little share. So an optimization is included which avoids this situation. If the share for the medium processor is too smnall, smaller than 50%, a high processor is converted to a medium processor and the share of the former high processor plus the remainder of the original calculation is divided between the two medium processors. If now a low processor is unparked the share percentage must be divided by 3 processors and so forth.

Hiperdispatch: RMF Report Example

```
 z/OS V1R9                SYSTEM ID R71            DATE 01/28/2009          INTERVAL 00.59.753
                               CONVERTED TO z/OS V1R10 RMF       TIME 11.02.00
-CPU  2097   MODEL  716   H/W MODEL  E26  SEQUENCE CODE 00000000000A73A2    HIPERDISPATCH=YES
0---CPU---    ---------------- TIME % ----------------     LOG PROC      --I/O INTERRUPTS--
 NUM  TYPE    ONLINE    LPAR BUSY    MVS BUSY   PARKED     SHARE %      RATE     % VIA TPI
  0    CP     100.00    99.50        100.0       0.00       100.0       29.40      0.00
  1    CP     100.00    99.88        100.0       0.00       100.0       18.14      0.00
  2    CP     100.00    99.83        100.0       0.00       100.0       31.71      0.00
  3    CP     100.00    99.78        100.0       0.00       100.0       16.82      0.00
  4    CP     100.00    72.24        100.0       0.00        66.4        0.00      0.00   Medium LCPs
  5    CP     100.00    72.30        100.0       0.00        66.4        0.00      0.00
  6    CP     100.00    35.16        100.0      46.14         0.0        0.00      0.00   Low Un-parked LCPs
  7    CP     100.00    52.22        100.0      24.06         0.0        0.00      0.00
  8    CP     100.00     0.00        -----     100.00         0.0        0.00      0.00
  9    CP     100.00     0.00        -----     100.00         0.0        0.00      0.00
  A    CP     100.00     0.00        -----     100.00         0.0        0.00      0.00
  B    CP     100.00     0.00        -----     100.00         0.0        0.00      0.00
  C    CP     100.00     0.00        -----     100.00         0.0        0.00      0.00
  D    CP     100.00     0.00        -----     100.00         0.0        0.00      0.00
  E    CP     100.00     0.00        -----     100.00         0.0        0.00      0.00
  F    CP     100.00     0.00        -----     100.00         0.0        0.00      0.00
 TOTAL/AVERAGE           39.43        100.0                  532.8       96.08      0.00
```


The following example shows how unparking of processors works. For this reason we look at a test scenario for a CEC with two partitions. The RMF CPU Activity Report for the smaller of the two partitions is shown above. The CEC has 16 physical processors and for both partitions 16 logical processors have been defined. The smaller partition has 33% of the share which results in 5 High and 1 Medium processor with 32% share. Based on the optimization which we just discussed 1 High processor is converted to a medium processor and therefore the smaller partition now has 4 High processors and 2 medium processors with 66% share each.

We can also observe from the RMF report that only 2 low processors were unparked during the reporting interval of 1 minute. And finally that the processors state: HIGH, MED, and LOW is not printed on this report because it was generated for a z/OS 1.9 system on a reporting system of z/OS 1.10 and the state was added with z/OS 1.11.



The graphic above now shows the SMF 99 information for the test run. The graphic is shows both partitions R71, the small partition which we just discussed on the left hand side and R72 the big partition on the right hand side. The test run started at 10:56 until 11:08. For R72 we can observe that based on the partition share 10 logical processors are treated as High processors and 1 as a medium processor because the share of 67% is high enough.

The work which runs on the partitions is very different. R71 shows a very high demand. The MVS BUSY value for R71 is always at 100%. For R72 the demand is not that high. The red line which depicts the MVS BUSY fluctuates between 75 and 95%. For R71 we can now also observe that low processors are being unparked. Based on the the demand of the partition which is always at 100% and the amount of cxapacity which is not used by R72 R71 unparks between 1 and 3 processors. But we can also observe that no low processor for R71 is unparked. This is the case if WLM determines that it is not efficient to use a low processor because there isn't enough capacity which can be used by the low processor.

As a result of this test scenario we learned:

•WLM parks and unparks the low processors

•WLM evaluates the park and unpark conditions every 2 seconds

•High demand is required to unpark a low processor. In fact WLM unparks a low processor when the MVS BUSY is above 95% and sufficient capacity on the system exists. The parking conditions will be discussed later. But this is also shows why the MVS BUSY has changed and only considers unparked processors

Hiperdispatch: PR/SM – Annotations

- What if there is only 1 High or 1 Medium share processor?
  - The high share processor will be converted to a medium share processor
    - So there isn't really just 1 High Share processor
  - A low share processor is always un-parked
  - The share of the medium processor is now equally divided between the two processors
    - So it is ensured that the system does not starve because there is just one processor online
- If "low share" processors exist there is also ALWAYS at least one medium processor
  - For example if the previous calculation would end with 2.0 meaning that 2 high processors exist and no medium AND in addition there is at least one low processor
    - One high processor is converted to a medium processor
      - → This is necessary to ensure that the low processors get some share of the shared processor pool when they need to be un-parked

At this point we are completed with the PR/SM part of Hiperdispatch. PR/SM determines the state of the processors whether they are treated as High, Medium or Low processors and PR/SM ensures that a High processors is always re-dispatched on the same physical processor. In fact PR/SM determines the full logical processor topology which is presented to z/OS. z/OS and within z/OS WLM then controls whether a low processor is in the park or unpark state.

Some final annotations are required for cases where only very few processors exist are listed above.

Hiperdispatch: PR/SM – Annotations

- PR/SM dispatches logical processors to physical processors in the same way whether it is HD=YES or HD=NO
  - Logical Processors have a “Share per PCP”
    - For HD=NO: the share is equal for all LCPs of the same partition
    - For HD=YES: High processors always have 100% share and medium and un-parked low processors use the remaining share together
  - For HD=NO
    - PR/SM attempts to re-dispatch LCPs on the same PCP or book but can’t guarantee this
  - For HD=YES
    - PR/SM always guarantees that vertical high processors are re-dispatched on the same physical processor
- Mix of HD=YES and HD=NO partitions
  - No special treatment
    - High processors of HD=YES partitions will be re-dispatched on the same physical processor
    - All other LCPs compete for the shared logical processors
      - →Shared logical processors are those processors which have no logical high processor assigned to them or which are not used by their logical high processor

A final remark is necessary about the way PR/SM dispatches processors. As earlier mentioned PR/SM uses the processor share to deterime the runtime of the logical processor on a physical processor. This hasn’t changed between HIPERDISPATCH=NO and HIPERDISPATCH=YES. The only difference is that the share is not equally distributed between the logical processors for the HIPERDISPATCH=YES case. This also shows that it is possible at any time to run partitions with HIPERDISPATCH=YES on the same CEC with partitions with HIPERDISPATCH=NO.

Hiperdispatch: PR/SM – Annotations

- LPAR with dedicated processors:
  - All processors are high share processors and nodes are created as for shared logical processors
  - Hiperdispatch is efficient too in this case:
    - z/OS part – re-dispatch work on a node of physically closely related processors
- Special Processors
  - Special processors (zAAPs and zIIPs) have their own processor pools
  - PR/SM divides the special processors into the same structure of high, medium and low share processors as it does with regular CPs
    - The mechanism is the same
  - PR/SM provides this information also to z/OS

Special processors are assigned separate pools with their own weight definitions. Other than that the topo0logy information, the state of processors and the way how processors are parked and unparked are the same for special processors than for regular processors.

Finally for dedicated processors HIPERDISPATCH=YES is meaningful from a z/OS perspective as we will see in the following charts

Hiperdispatch: Special Processors

C P U A C T I V I T Y

z/OS V1R9 SYSTEM ID R71 DATE 02/15/2009

CONVERTED TO z/OS V1R10 RMF TIME 20.31.00

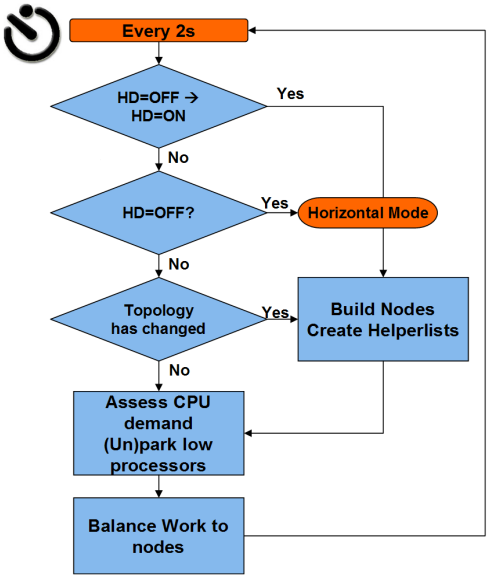
CPU 2097 MODEL 707 H/W MODEL E26 SEQUENCE CODE 0000000000019FC4 HIPERDISPATCH=YES

---CPU---		---------- TIME % ----------				LOG PROC	--I/O INTERRUPTS--		
NUM	TYPE	ONLINE	LPAR BUSY	MVS BUSY	PARKED	SHARE %	RATE	% VIA TPI	
0	CP	100.00	90.49	95.67	0.00	100.0	23.49	0.07	
1	CP	100.00	90.03	94.55	0.00	100.0	55.24	0.00	High
2	CP	100.00	91.94	95.99	0.00	100.0	21.97	0.08	
3	CP	100.00	69.13	96.10	0.00	50.0	0.07	0.00	Medium
4	CP	100.00	53.55	78.77	0.00	0.0	0.00	0.00	
5	CP	100.00	51.81	79.16	3.80	0.0	0.00	0.00	Un-parked
6	CP	100.00	0.00	-----	100.00	0.0	0.00	0.00	Parked
TOTAL/AVERAGE			63.85	90.11		350.0	100.8	0.03	
7	IIP	100.00	51.86	99.96	0.00	50.0			
8	IIP	100.00	51.52	99.95	0.00	0.0			Medium
9	IIP	100.00	51.40	99.94	0.00	0.0			
A	IIP	100.00	51.51	99.94	0.00	0.0			Un-parked
B	IIP	100.00	51.45	99.94	0.00	0.0			
TOTAL/AVERAGE			51.55	99.95		50.0			

The example on this foil shows an RMF CPU Activity report for a partition with regular and z/IIP processors. Based on the different weights and different number of processors two separate pictures of High, Med and Low processors can be observed.

Hiperdispatch: z/OS WLM

- z/OS WLM
  - Every 2s
  - Tests Hiperdispatch ON and OFF switch
  - Reads logical processor topology from PR/SM
  - Builds affinity nodes
  - Parks and un-parks low LCPs based on processor demand
  - Balances units of work to affinity nodes
- z/OS Dispatcher
  - Dispatches work on affinity nodes
  - Determines whether nodes need help



In the next step we want to take a look at z/OS and WLM. WLM evaluates every 2 second the state of HIPERDISPATCH, retrieves the topology if necessary, rebuilds its affinity nodes, parks and unparks low processors and balances the units of work across the affinity nodes. These steps will be discussed on the following pages.

Hiperdispatch: z/OS

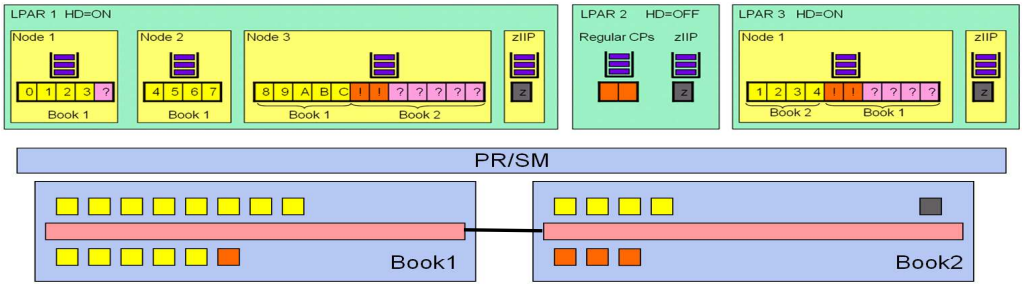
LPAR	Weight	LCPs	Share	PhysProc	High	Medium	Low
LPAR1	680	21	68%	14.28	13	2	6
LPAR2	80	2	8%	1.68	1	1	0
LPAR3	240	10	24%	5.04	4	2	4
	1000			21			

- Dispatcher Nodes
  - Nodes are created based on the high-share processors
    - Ideally a node has 4 high share processors (target for z10)
    - An additional node is created when at least 3 high share processors (or 2 high and 2 medium) can be placed in it
    - Ideally a node encompasses only high share processors of the same book
  - High shared processors are always re-dispatched on the same physical processor
  - Medium and low share processors are added to the created nodes based on their book placement to keep a node as much as possible on one book
    - Medium and Low share processors have no fixed physical processor placement
    - But book crossing nodes can't be avoided
    - Also even if a low processor is "assigned" to a book it may float to the other book if necessary

We will start with an example of a z10. The system contains three partitions with one big partition encompassing 13 high, 2 medium and 6 low processors. The system is a z10 with 21 physical processors. Even with assigning a polarization (High, Medium, Low) to the logical processors it is still necessary to limit the work to a subset of logical processors. This subset is named a node and ideally a node consists of 4 high processors. In fact if the partition is rather small than only 1 node is created and if the system has only 5 or 6 High processors also only 1 node is created because it is not possible to create a second node which would be too small.

Hiperdispatch: z/OS

LPAR	Weight	LCPs	Share	PhysProc	High	Medium	Low
LPAR1	680	21	68%	14.28	13	2	6
LPAR2	80	2	8%	1.68	1	1	0
LPAR3	240	10	24%	5.04	4	2	4
	1000			21			

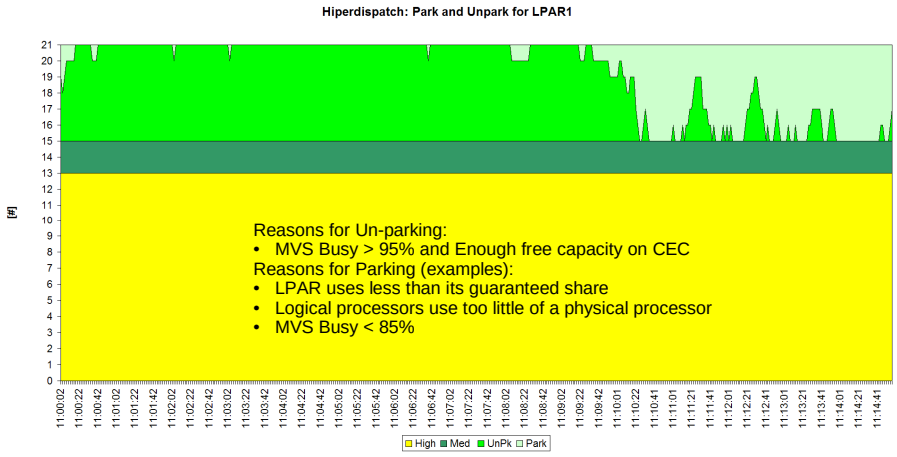


The chart above shows the topology and the nodes which have been created for the partitions. As mentioned before for LPAR3 only 1 has been created but for LPAR1 3 nodes have been built for the regular processors. The nodes are first created for the high processors and on which book the high processors are located. The third node has 5 high processors because it is not possible to form a node of one high processor.

After creating the node the medium and low processors are added to the nodes. This is again done by the location of these processors. As mentioned before it can't be guaranteed that a medium or low processor is always re-dispatched on the book it is assigned to but at least there is a high likelihood that this is possible. WLM now attempts to create the nodes in a way that book crossing activities are reduced as much as possible and if possible that these are restricted to as few nodes as possible. Therefore node 3 receives all medium and low processors which are located on the second book so that this node is mostly used for cross book activities. In this content it must be noted that node 1 receives a low processor which is assigned on book 1 the same book then the High processors.

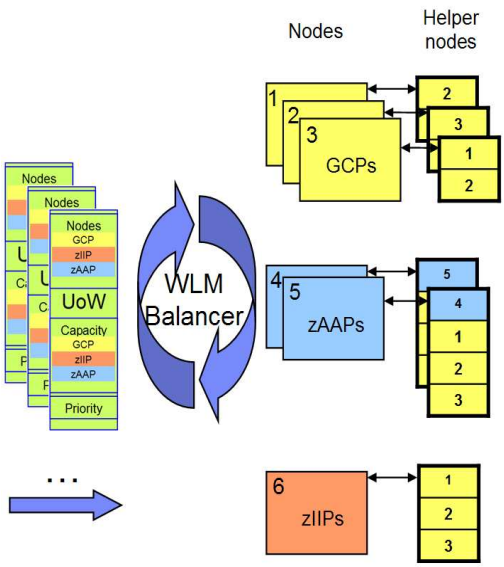
Hiperdispatch: Parking and Un-parking of Low Processors

- WLM tracks PR/SM white space attributes to dynamically address longer term workload requirements
  - Parks and un-parks low polarity LCPs based on available excess capacity



The next important part is to park and unpark the low processors. As already mentioned a low processor can be unparked if the MVS BUSY of all unparked processors exceeds 95% and if sufficient capacity exists on the CEC. Parking a low processor is triggered by more events of which some are listed above. The graphic in the background shows how the low processors were unparked for the big partition LPAR1 for a 15 minute time period.

Hiperdispatch: Work Balancing



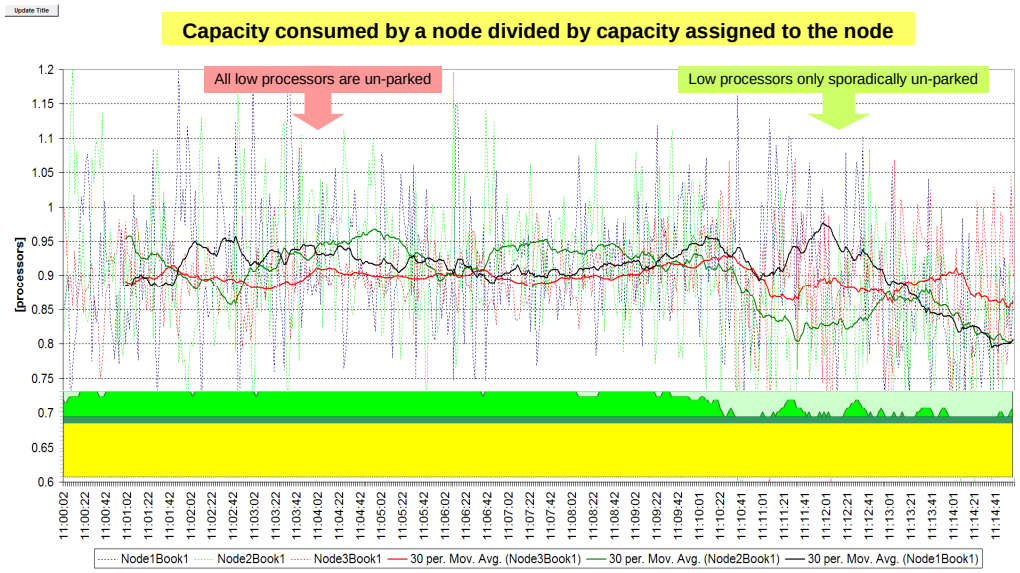
- WLM balances work across nodes
  - Each unit of work gets a home node assigned for CPs, zAAPs and zIIPs
    - Based on dispatch priority and consumed capacity
- Each node gets a helper list assigned
  - Required if a node has too much work
    - Work unit queue is too long
    - CPs, zAAPs and zIIPs do not enter wait within a specified time interval
  - Regular CPs can help any node

The last step is to assign work to the nodes. This is done based on dispatch priority and CPU consumption. The idea is that all nodes get a similar mix of work so that it doesn't happen that on one node there are only high priority tasks and on another node only high CPU consuming work. The node assignment is also done for offload processor nodes so that each unit of work gets up to 3 nodes assigned, one for regular processors one for zIIPs and one for zAAPs.

Result:

•WLM distributes the work based on CPU consumption and dispatch priority across the nodes in order to achieve an equal node distribution.

Hiperdispatch: Balancing and Node Utilization for LPAR1



In the chart above we can observe how the 3 nodes of LPAR1 are being utilized. We can observe that on a 2 second time scale the utilization can vary from 75 to 120% and we can easily derive that it is not possible to run more than 100% on a node. So we need a method to ensure that a node which temporarily has more demand then it can execute is being helped.

The above picture also shows that the node utilization on a 1 minute time scale is pretty consistent between the three nodes especially during the time period of very high demand when all low processors are unparked.

Hiperdispatch: Helper Processing

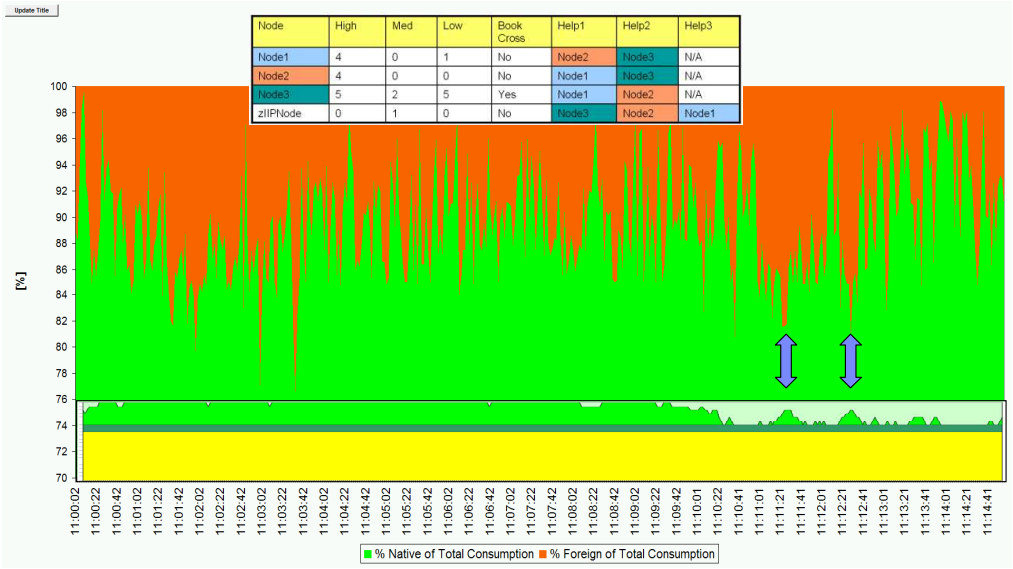
Node	High	Med	Low	Book Cross	Help1	Help2	Help3
Node1	4	0	1	No	Node2	Node3	N/A
Node2	4	0	0	No	Node1	Node3	N/A
Node3	5	2	5	Yes	Node1	Node2	N/A
zIIPNode	0	1	0	No	Node3	Node2	Node1

- Each node gets a helper list assigned
  - Required if a node has too much work
    - Work unit queue is too long
    - CPs, zAAPs and zIIPs do not enter wait within a specified time interval
  - Regular CPs can help any node
- Supervisor implements Needs-Help detection and action to address transient spikes in utilization
  - Maintains priority-based Affinity Node utilization statistics
  - Responsively acts on statistics by asking other LPs for “Help”
- Table: shows the internal structure of helper nodes per Node
  - Preferred are non book crossing nodes
  - All high processors can become helper processors

In order to ensure that no work starves on a node a list of helper nodes is assigned to each affinity node. If Supervisor now detects that the demand on a node is too high it enables processors of the helper nodes to also select work from the node which needs help. The helping processor helps the other node until it gets in a wait state. During this period the processor selects work from its node work queue and the node work queue being helped in priority order.

For helper nodes nodes which do not cross books are prefered. Therefore Node 3 is only listed as the second bets choice of being a helper in the example above.

Hiperdispatch: Helper Processing



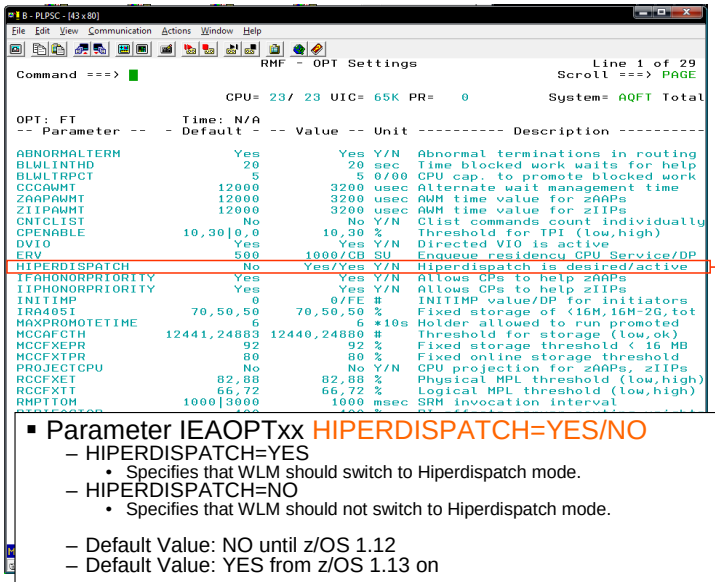
Native: executed on node 1, Foreign: executed on a helper node

The graphic above shows for Node1 of LPAR1 the percent of work which is executed locally by processors of the node (Native) or which is executed by processors of another node which helps Node 1 (Foreign). It can be observed that high help activity also results in unparking low processors. This can be especially seen on the right hand side when it is no longer required to have all low processors being unparked all the time.

Result:

•WLM and Supervisor have developed a mechanism which ensures that no work starves on a node while a node is temporarily overutilized.

Hiperdispatch: User Interface …



RMF Monitor 2

OPT Display
- Shows status of OPT parameters
- z/OS 1.11 and above
- There are two values shown on the panel. The first value reflects the OPT setting (inOPT) the second whether HIPERDISPATCH is really used in the system (Running)

- Parameter IEAOPTxx HIPERDISPATCH=YES/NO
  - HIPERDISPATCH=YES
    - Specifies that WLM should switch to Hiperdispatch mode.
  - HIPERDISPATCH=NO
    - Specifies that WLM should not switch to Hiperdispatch mode.
  - Default Value: NO until z/OS 1.12
  - Default Value: YES from z/OS 1.13 on

At the moment HIPERDISPATCH is still optional on System z and z/OS but with z/OS 1.13 it becomes the default for dispatching work. The state of HIPERDISPATCH can also be easily observed through the RMF Monitor II OPT display panel

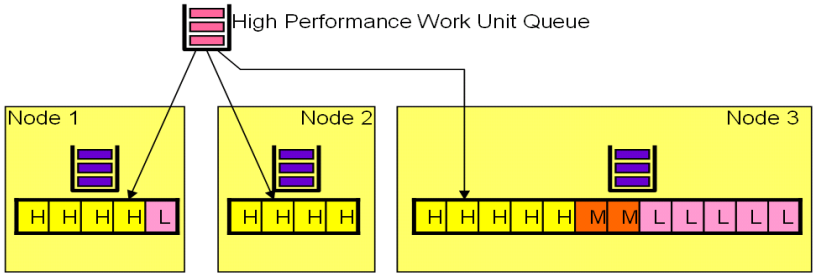
Hiperdispatch: Processor Latency Considerations

- SMPs are good at latency reduction
  - Higher probability of a processor to run important work
  - Running work anywhere reduces cache value
- Affinity nodes can limit access of work to processors
  - During periods of modest load (less than 95% busy), the CPU latency is very likely to increase if all else held constant
  - HiperDispatch is much more aggressive at having a processor running to service the work
  - The net can still be some increased CPU delay at modest utilization
  - HiperDispatch value is strongest at high utilization – which is where most clients expect z/OS to be tested
  - Clients analyze processor efficiency at all utilizations so HiperDispatch manages work to affinity nodes at all utilizations
- For WLM Service Definition
  - Hiperdispatch=Yes typically uses fewer logical processors than Hiperdispatch=No
    - Work may show higher CPU delays
    - This is most often the case for lower important work
    - This will effect the PI and achieved execution velocities
    - Consequence: Review your goal settings and potentially adjust execution velocity goals

This foil discusses the benefit of Hiperdispatch and what should not be expected from it.

Hiperdispatch: Special processing for SYSSTC

- Work classified to SYSSTC typically contains lots of short-running local SRBs required for transaction flow.
  - Examples of address spaces recommended to be classified into SYSSTC are VTAM, TCP/IP and IRLM.
- Local SRBs for address spaces classified into SYSTEM or SYSSTC can execute on any available logical processor in HiperDispatch mode.
- WLM service policies should be reviewed with this in mind.



27

Planning for HiperDispatch a good opportunity to review your WLM service definition to be sure that it reflects the needs of the business. The dispatching priorities produced by WLM have additional uses in HiperDispatch mode.

The SYSSTC is the service class that has the highest dispatching priority of those one can select. In an OLTP environment, many local SRBs are scheduled by address spaces like VTAM, TCP/IP etc. During the course of developing HiperDispatch, it became clear that there was little benefit to adhering to the rule that these SRBs must be hosted by the processors assigned to the affinity node of those address spaces. There is very little opportunity for cache reuse because the data touched is for many transactions. It was also learned that reducing the opportunity to service these short requests can elongate response time since the transaction is waiting to begin to deliver its output in may cases. Therefore, local SRBs in SYSTEM and SYSSTC service classes are executed on the first available unparked LP.

When you have the opportunity to convert to HiperDispatch mode, if you have some response time increase for a very high priority service class, which you feel is intolerable, and your application is one with many short requests for processor, you might try assigning it to SYSSTC. One should not do this if it compromises your other goals and importance settings.

Hiperdispatch: Summary

- Hiperdispatch is a combination of PR/SM and z/OS to provide more efficient dispatching on large scale processor environments
  - PR/SM provides a much better mapping of logical to physical processors
  - z/OS re-dispatches work on a subset of the logical processors
- Hiperdispatch is most efficient for systems with many logical processors
  - Provides the base to grow with many processors on System z
- White Paper published 2008 and republished 2009
- Hiperdispatch Benefit on z196:

Share of the partition - assumes 1.5 logical to physical ratio	Number of Physical CPs + zIIPs + zAAPs			
	<=16	17-32	33-64	65-80
0 <= share in processors < 1.5	0%	0%	0%	0%
1.5 <= share in processors < 3	2-5%	3-6%	3-6%	3-6%
3 <= share in processors < 6	4-8%	5-9%	6-10%	6-10%
6 <= share in processors < 12	5-11%	7-13%	8-14%	8-16%
12 <= share in processors < 24	-	8-16%	10-18%	11-21%
24 <= share in processors < 48	-	-	11-21%	12-24%
48 <= share in processors <= 80	-	-	-	14-26%

Hiperdispatch is most valuable for large partitions on large CECs. Nevertheless especially on z196 more benefit also exists for smaller partitions because of the cache structure which is shared by the processors of the same chip. The table above shows the expected benefit when Hiperdispatch=YES is used on z196 systems. In any case it should be noted that Hiperdispatch will not have a negative effect.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply. All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions. This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.