z/VM 6.3 HiperDispatch

Bill Bitner
z/VM Development Client Focus and Care
bitnerb @ us.ibm.com

SHARE August 2014
Session 15744

#SHAREorg

SHARE in Pittsburgh 2014

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

IBM*
IBM Logo*
DB2*
Dynamic Infrastructure*
GDPS*
HiperSockets
Parallel Sysplex*
RACF*
System z*
System z10*
Tivoli*
z10 BC
z9*
z/OS*
z/VM*
z/VSE
zEnterprise*
System z196
System z114
System zEC12
System zBC12

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

OpenSolaris, Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
INFINIBAND, InfiniBand Trade Association and the INFINIBAND design marks are trademarks and/or service marks of the INFINIBAND Trade Association.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

All other products may be trademarks or registered trademarks of their respective companies.

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.

Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs and IFLs):

Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”).

No other workload processing is authorized for execution on an SE.

IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Agenda

- Objectives of the z/VM HiperDispatch enhancement

- A little about System z hardware and the PR/SM hypervisor
  – Machine structure
  – Behavior and features available in the hypervisor

- Key features of z/VM HiperDispatch
  – Use of vertical mode partitions
  – Running as widely as available power suggests
  – Reducing MP level when it appears z/VM overhead is a problem
  – Dispatching guests in a manner aware of physical and virtual topologies
  – Knobs you can twist or set

- Planning for z/VM HiperDispatch

- Workloads
  – Those that will benefit
  – Those that won’t

- Summary

Objectives and Strategies

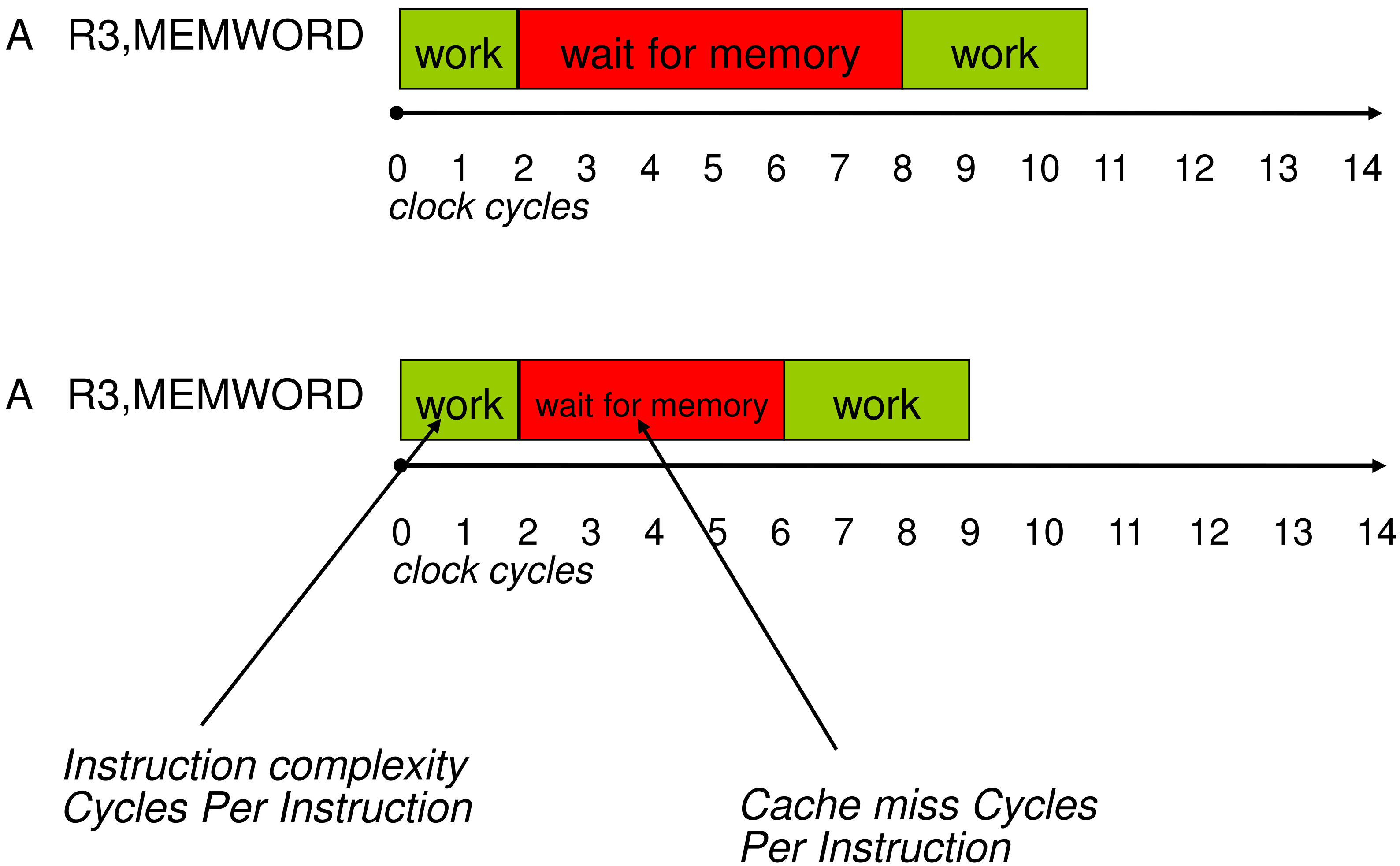
Objectives and Strategies of z/VM HiperDispatch

- Improve performance of your workloads, by …
  - Reducing CPU time needed per unit of work done, by …
  - Reducing the time needed for each instruction to run, by …
  - Reducing the time the CPU waits for memory contents to be brought to it.

- Improve performance of your workloads, by…
  - Sensing situations where z/VM Control Program overhead is a problem, and...
  - Changing the LPU configuration to try to reduce the overhead.

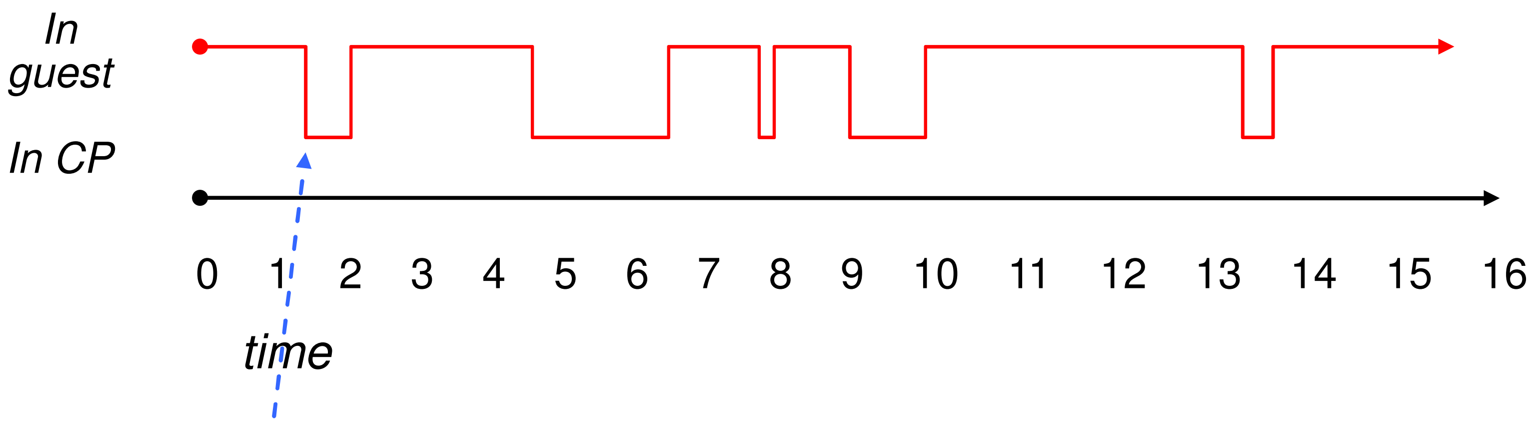
- Strategies:
  - Exploit PR/SM hypervisor features meant to help instruction speed
  - Be smarter about what the right MP-level is for the partition at the moment
  - Be smarter about the dispatching of guest virtual CPUs

What It Means to Reduce CPU Wait Time



What It Means to Reduce z/VM Overhead

CPU Consumption Timeline of a Virtual CPU



Some reasons guests go into CP:
- Issue a Diagnose
- Perform I/O
- Issue some other priv op
- Take a page fault

$$\text{T/V ratio} = \frac{\text{(CP time + guest time)}}{\text{guest time}}$$

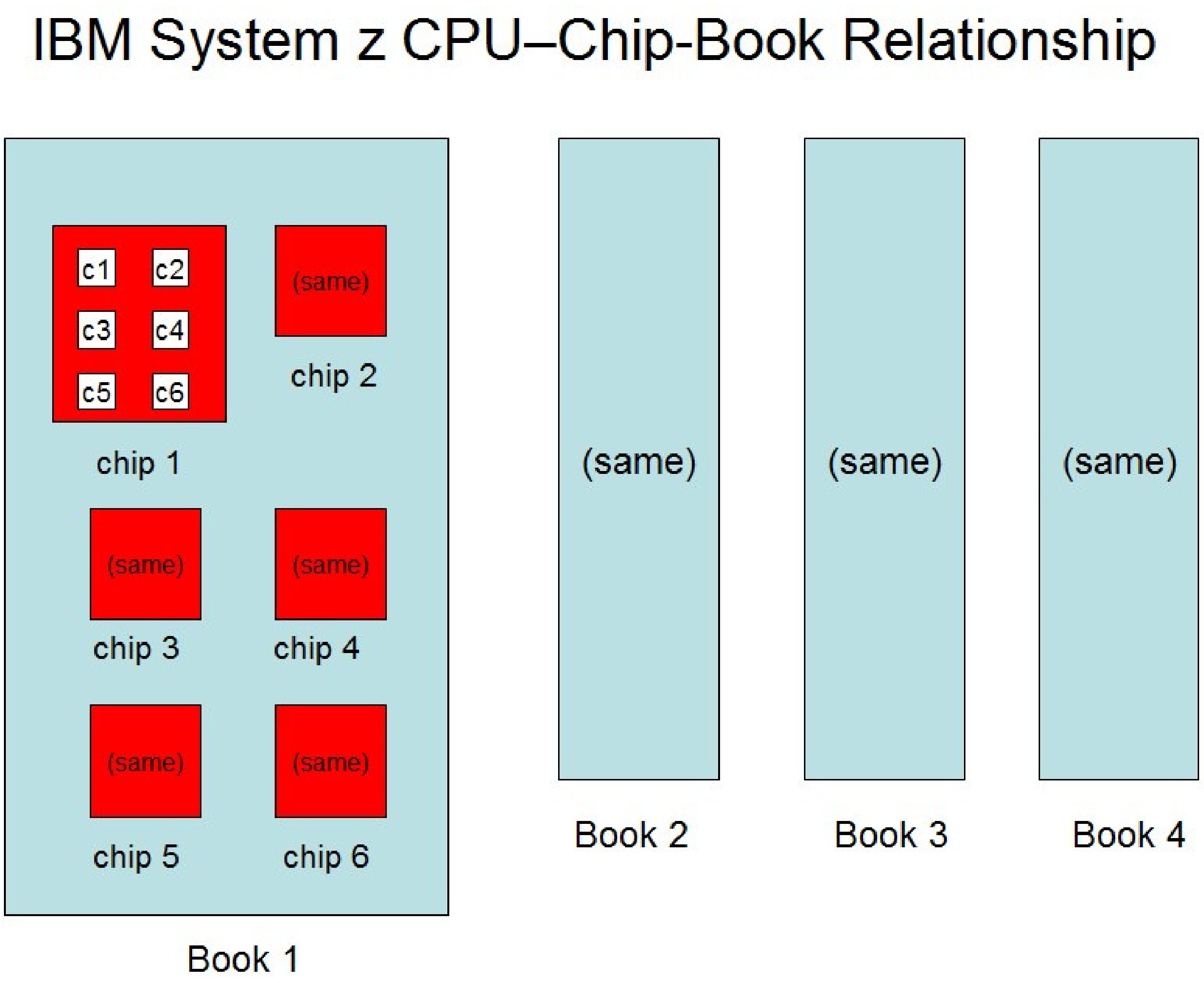
Things CP often does “down there”:
- Acquire a lock, for serialization
- Do some processing
- Release the lock
- Eventually, run the guest again

- Time spent spinning on locks is wasted time. We can reduce it by reducing the partition’s MP level.

A Few Things About System z and PR/SM

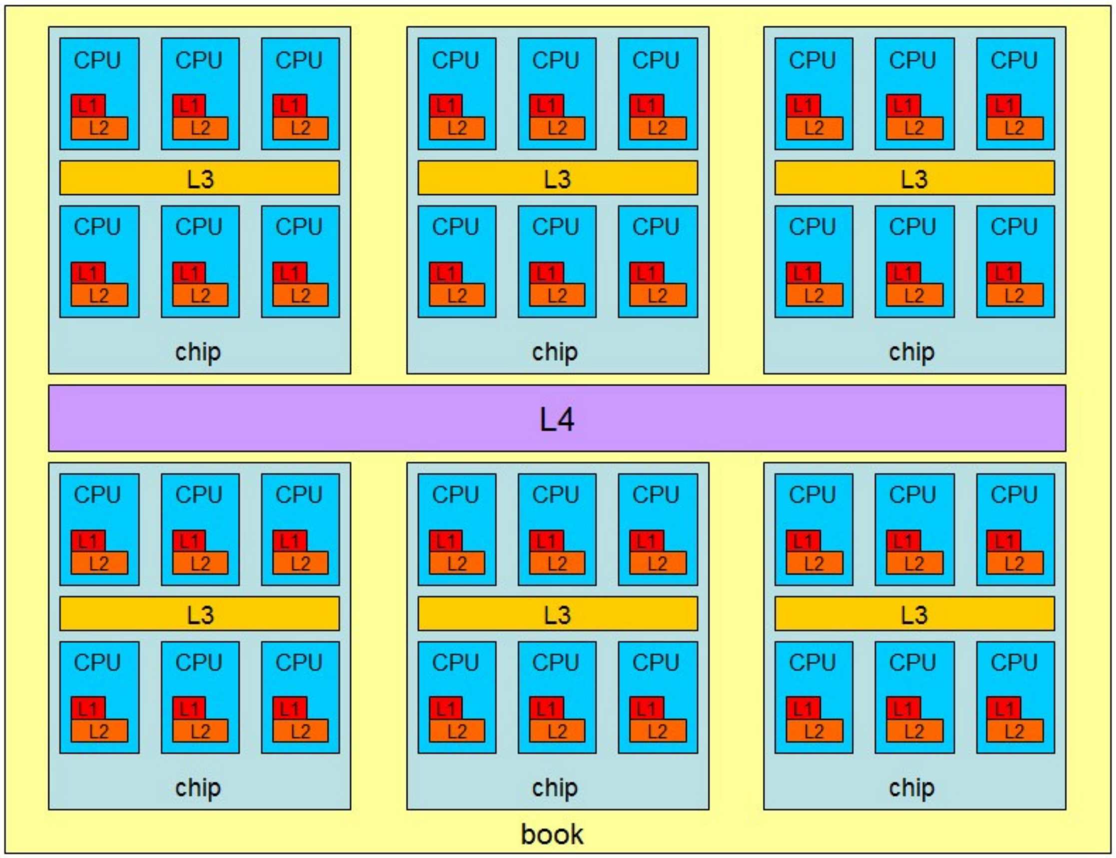
How is the hardware laid out?

IBM System z: Cores, Chips, and Books



IBM System z: Layered Cache Structure

IBM System z Cache Layering



Cache mantra:

- Closer, smaller, faster.

- Farther, larger, slower.

- Try to run a context in the same place over and over.

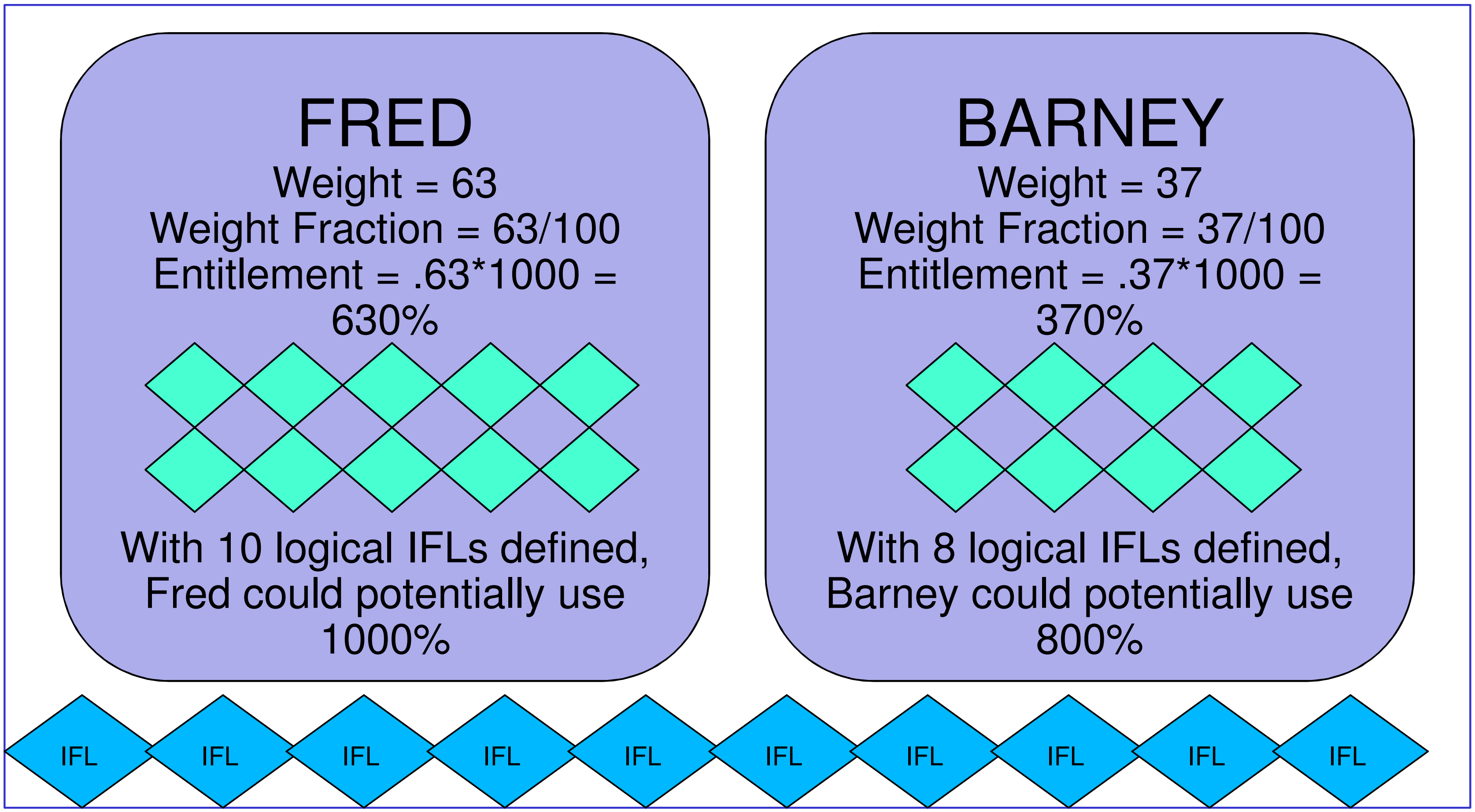
- Try to run related contexts near to one another.

- Try to run unrelated contexts apart from one another.

Okay, that's low-level, how does my LPAR get some CPU power?

HiperDispatch: Partition Entitlement vs. Logical CPU Count

Suppose we have 10 IFLs shared by partitions FRED and BARNEY
FRED has 10 logical IFLs defined and BARNEY has 8 logical IFLs defined.



IBM System z: Partition Entitlement vs. Logical CPU Count

Suppose we have 12 physical IFLs: 2 dedicated, 10 shared.

Partition	Weight	Weight Sum	Weight Fraction	Physical Capacity	Entitlement Calculation	Entitlement	Maximum Achievable Utilization
FRED, a logical 10-way	63	100	63/100	1000%	1000% x (63/100)	630%	1000%
BARNEY, a logical 8-way	37	100	37/100	1000%	1000% x (37/100)	370%	800%

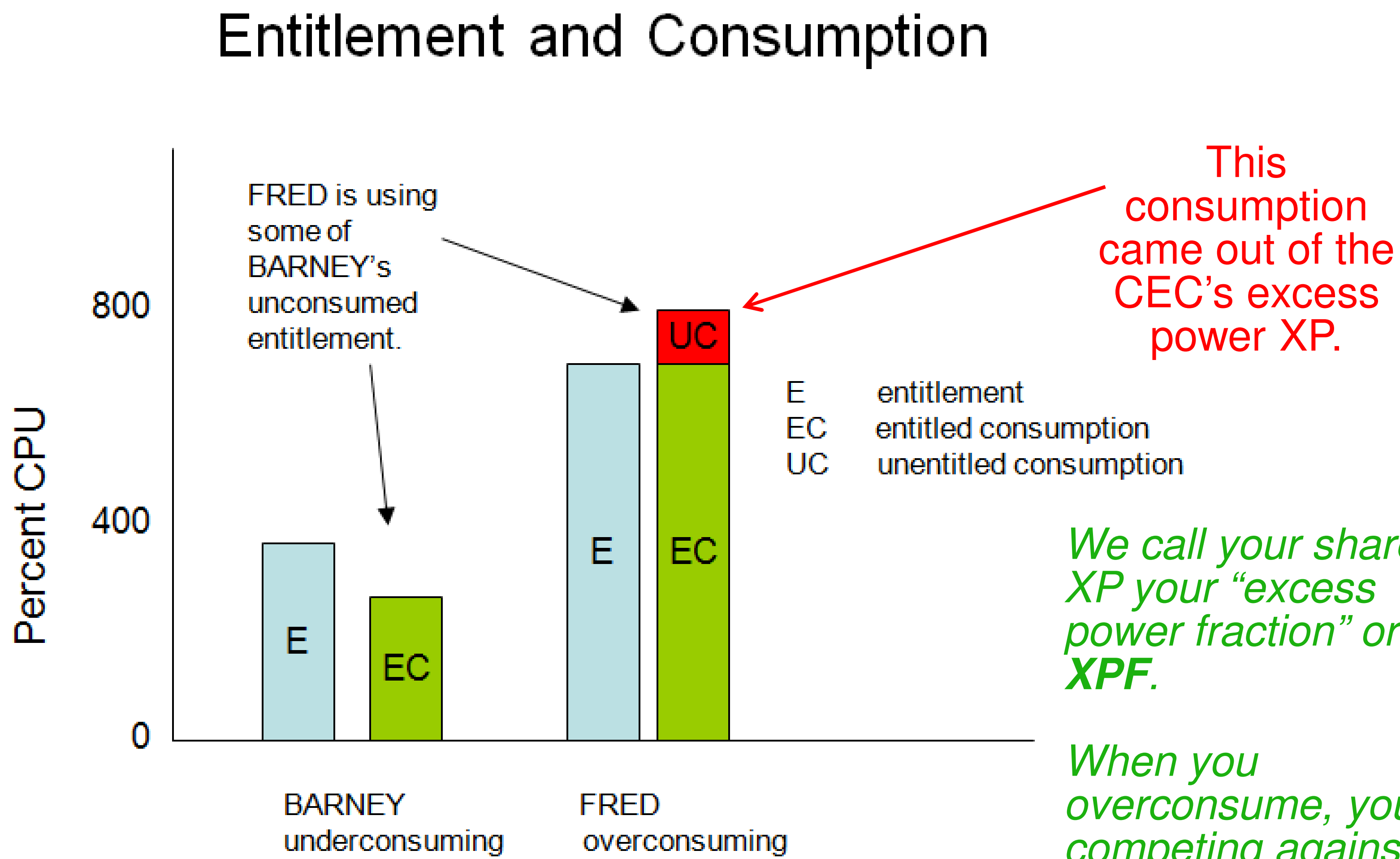
FRED can always run up to 630% busy. That's what *entitlement* means.

But for FRED to run *beyond* 630% busy, BARNEY has to leave some of its entitlement *unconsumed.*

Keep this in mind: (CEC's excess power XP) = (total power TP) - (consumed entitled power EP).

Excess power XP will become very important later.

IBM System z: Entitlement and Consumption



*We call your share of XP your "excess power fraction" or your **XPF**.*

When you overconsume, you are competing against other overconsuming partitions for XP.

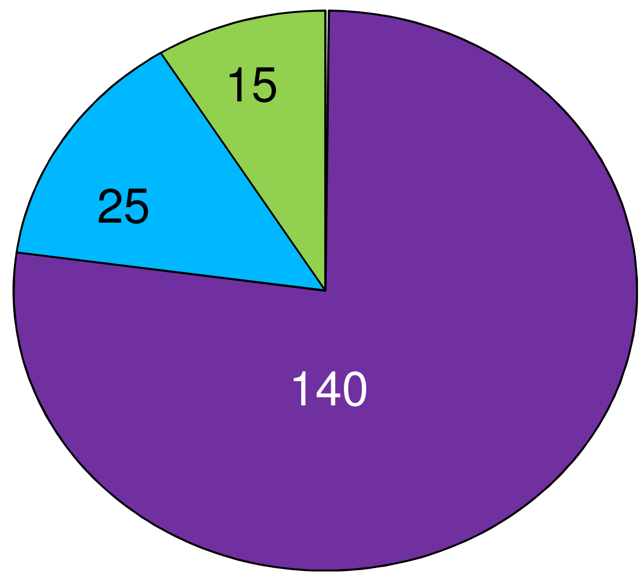
IBM System z: A Little More About XP and XPF

Suppose there is 180% left after all entitled consumptions are satisfied. XP=180%.

Suppose P1, P2, and P3 (me), all equal weights, are competing for it.
Their first-pass weight fractions of XP are therefore each 60%.

Case 1:
- P1 is overconsuming 15%
- P2 is overconsuming 25%

P3 can have (180-(15+25)) = 140%
if it wants it. XPF=140



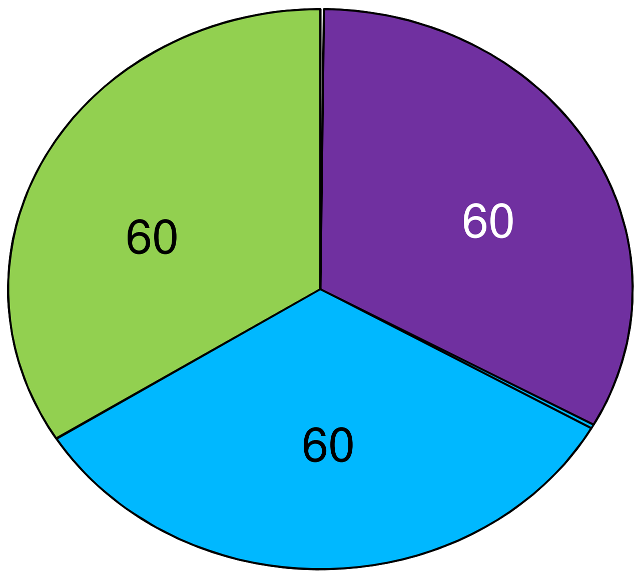
IBM System z: A Little More About XP and XPF

Suppose there is 180% left after all entitled consumptions are satisfied. XP=180%.

Suppose P1, P2, and P3 (me), all equal weights, are competing for it.
Their first-pass weight fractions of XP are therefore each 60%.

Case 2:
- P1 is overconsuming 90%
- P2 is overconsuming 90%

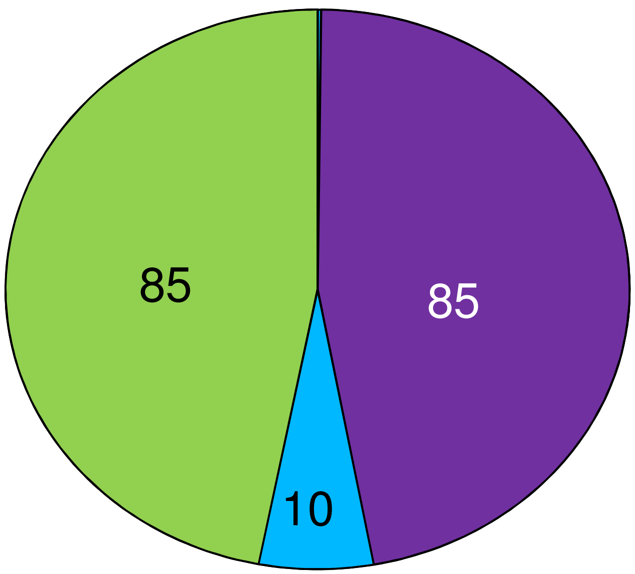
P3 can have 60% if it wants it.
XPF=60



IBM System z: A Little More About XP and XPF

Suppose there is 180% left after all entitled consumptions are satisfied. XP=180%.

Suppose P1, P2, and P3 (me), all equal weights, are competing for it.
Their first-pass weight fractions of XP are therefore each 60%.



Case 3:
- P1 is overconsuming 135%
- P2 is overconsuming 10%

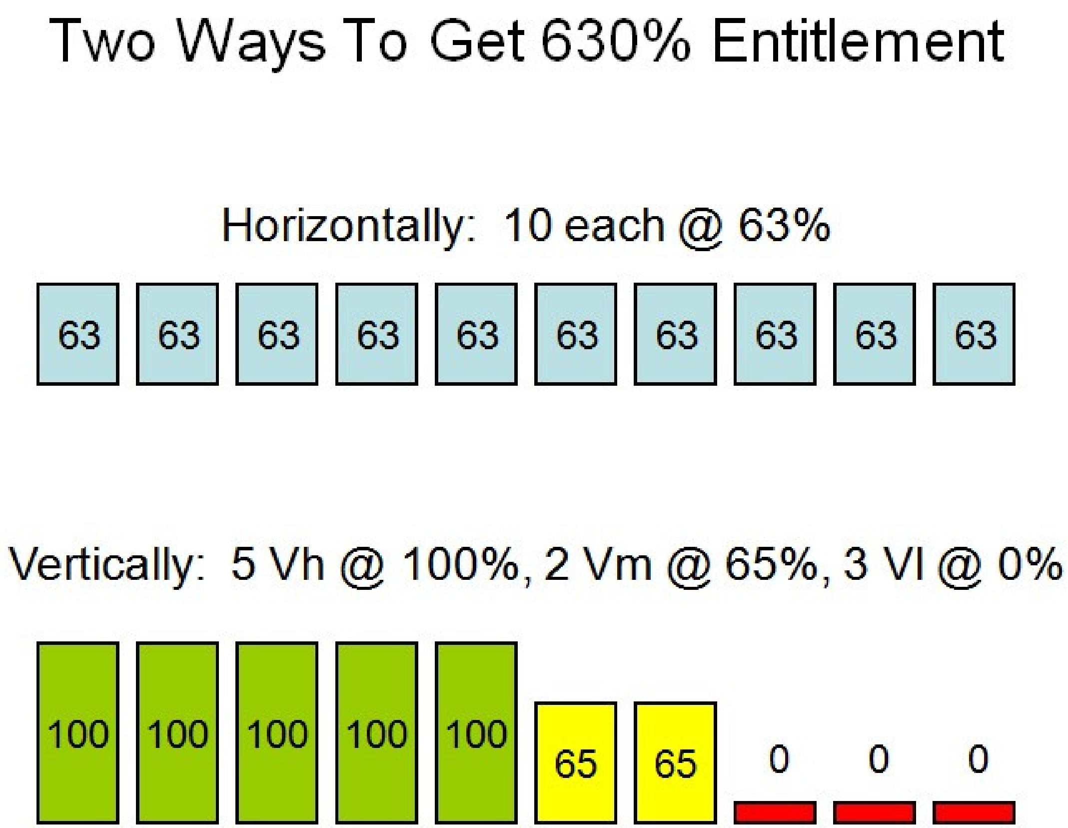
Round 1: P1+=60, P2+=10, P3+=60, s=130, r=50

Round 2: P1+=25, 3+=25, s=50, r=0

P3 can have 85% if it wants it. XPF=85

That's how PR/SM gives my LPAR CPU power, but how does that relate to cache usage?

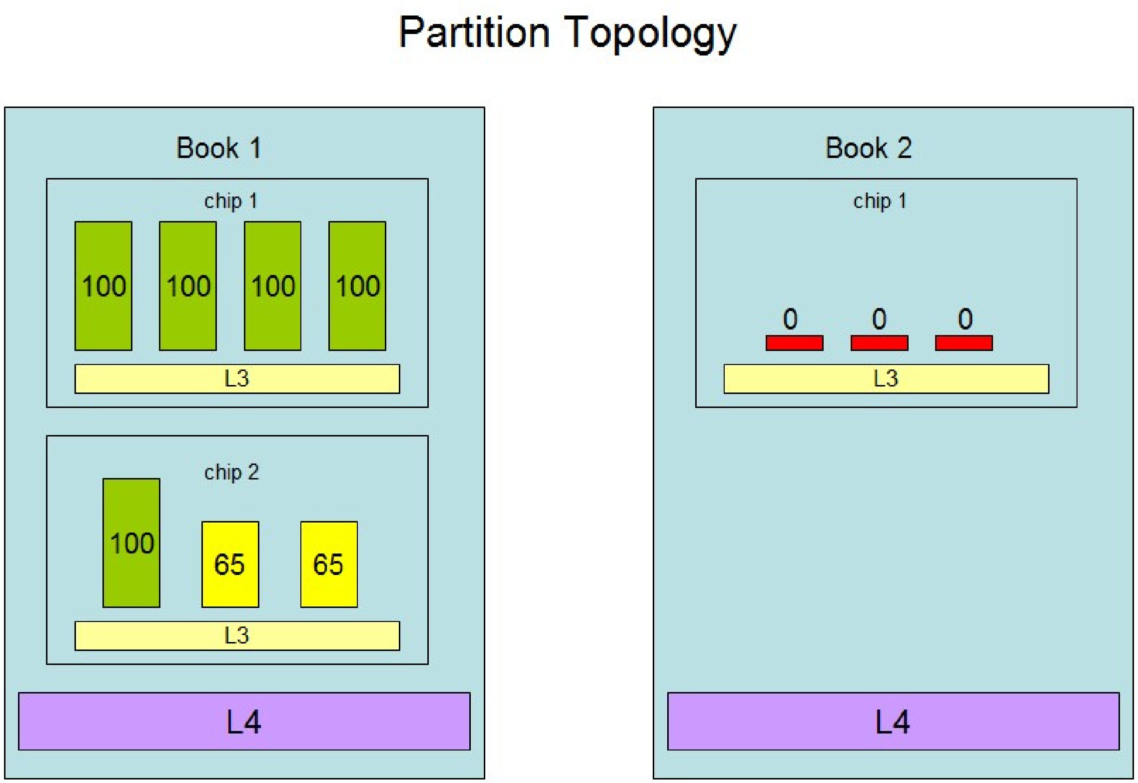
IBM System z: Horizontal and Vertical Partitions



In vertical partitions:

- Entitlement is distributed unequally among LPUs.
- The unentitled LPUs are useful only when other partitions are not using their entitlements.
- PR/SM tries very hard not to move Vh LPUs.
- PR/SM tries very hard to put the Vh LPUs close to one another.
- Partition consumes its XPF on its Vm and Vl LPUs.

IBM System z: The Partition Knows Its Placement



In vertical partitions:

- Sense your placement
- Run work smartly in light of your placement
- Sense unentitled power
- Use LPUs smartly in light of unentitled power

Notice PR/SM has given this partition a "quiet place" to do its work, provided the partition runs its work on its Vh LPUs.

What HiperDispatch Does with All This

z/VM HiperDispatch: Use of Vertical Mode

```
indicate load
AVGPROC-000% 24
XSTORE-000000/SEC MIGRATE-0000/SEC
MDC READS-000000/SEC WRITES-000000/SEC HIT RATIO-000%
PAGING-0/SEC
Q0-00000(00000)                           DORMANT-00000
Q1-00000(00000)               E1-00000(00000)
Q2-00000(00000) EXPAN-000 E2-00000(00000)
Q3-00000(00000) EXPAN-000 E3-00000(00000)

PROC 0000-000% CP    VH      PROC 0001-000% CP    VH
PROC 0002-000% CP    VH      PROC 0003-000% CP    VH
PROC 0004-000% CP    VH      PROC 0005-000% CP    VH
PROC 0006-000% CP    VH      PROC 0007-000% CP    VH
PROC 0008-000% CP    VH      PROC 0009-000% CP    VH
PROC 000A-000% CP    VH      PROC 000B-000% CP    VH
PROC 000C-000% CP    VH      PROC 000D-000% CP    VH
PROC 000E-000% CP    VH      PROC 000F-000% CP    VH
PROC 0010-000% CP    VH      PROC 0011-000% CP    VH
PROC 0012-000% CP    VH      PROC 0013-000% CP    VH
PROC 0014-000% CP    VM      PROC 0015-000% CP    VM
PROC 0016-000% CP    VL      PROC 0017-000% CP    VL

LIMITED-00000
Ready; T=0.01/0.01 13:13:39
```

Here we see an assortment of LPUs:
- 20 Vh
- 2 Vm
- 2 Vl

(24-way with 2130% entitlement)

Note: these percent-busies are now *percent of a physical CPU*, not percent-not-deliberately-waiting as they used to be:
- Older releases: if the logical CPU never loaded a wait PSW, it showed 100% busy no matter what it was truly using.
- New release: these percentages are the *fraction of the capacity of a physical CPU* being used by the logical CPU.

z/VM HiperDispatch: Awareness of Topology

```
q proc topology
13:14:59 TOPOLOGY
13:14:59   NESTING LEVEL: 02  ID: 01
13:14:59     NESTING LEVEL: 01  ID: 01
13:14:59       PROCESSOR 00  PARKED     CP    VH  0000
13:14:59       PROCESSOR 01  PARKED     CP    VH  0001
13:14:59       PROCESSOR 12  PARKED     CP    VH  0018
13:14:59     NESTING LEVEL: 01  ID: 02
13:14:59       PROCESSOR 0E  MASTER     CP    VH  0014
13:14:59       PROCESSOR 0F  ALTERNATE  CP    VH  0015
13:14:59       PROCESSOR 10  PARKED     CP    VH  0016
13:14:59       PROCESSOR 11  PARKED     CP    VH  0017
13:14:59     NESTING LEVEL: 01  ID: 03
13:14:59       PROCESSOR 02  PARKED     CP    VH  0002
13:14:59       PROCESSOR 03  PARKED     CP    VH  0003
13:14:59       PROCESSOR 04  PARKED     CP    VH  0004
13:14:59     NESTING LEVEL: 01  ID: 04
13:14:59       PROCESSOR 05  PARKED     CP    VH  0005
13:14:59       PROCESSOR 06  PARKED     CP    VH  0006
13:14:59       PROCESSOR 07  PARKED     CP    VH  0007
13:14:59     NESTING LEVEL: 01  ID: 05
13:14:59       PROCESSOR 08  PARKED     CP    VH  0008
13:14:59       PROCESSOR 09  PARKED     CP    VH  0009
13:14:59       PROCESSOR 0A  PARKED     CP    VH  0010
13:14:59     NESTING LEVEL: 01  ID: 06
13:14:59       PROCESSOR 0D  PARKED     CP    VH  0013
13:14:59   NESTING LEVEL: 02  ID: 02
13:14:59     NESTING LEVEL: 01  ID: 02
13:14:59       PROCESSOR 14  PARKED     CP    VM  0020
13:14:59     NESTING LEVEL: 01  ID: 04
13:14:59       PROCESSOR 15  PARKED     CP    VM  0021
13:14:59       PROCESSOR 16  PARKED     CP    VL  0022
13:14:59       PROCESSOR 17  PARKED     CP    VL  0023
13:14:59     NESTING LEVEL: 01  ID: 05
13:14:59       PROCESSOR 0B  PARKED     CP    VH  0011
13:14:59       PROCESSOR 0C  PARKED     CP    VH  0012
13:14:59       PROCESSOR 13  PARKED     CP    VH  0019
Ready; T=0.01/0.01 13:14:59
```

Here we see the placements of our LPUs on the physical topology.

For example,
- LPU 00: Vh, book 1, chip 1
- LPU 15: Vm, book 2, chip 4

Nesting level just refers to book, chip, etc.   They are numbered from smallest to largest:

CP Monitor has been updated to log out logical CPU polarity.

Whoa, wait! What Does "Parked" Mean?

- A *parked* logical CPU is simply not participating in the running of the system's work.
- It is still varied-on
- It is still a configured logical CPU as far as PR/SM is concerned
- It still counts as far as software licensing is concerned
- It is sitting in a barely-enabled wait-state PSW waiting for somebody to wake it up
- It might sit there in a wait for a really long time
- When we need it, we will signal it aka *unpark* it.
- Unparking requires a SIGP and some wakeup processing. Much faster than VARY ON.

z/VM HiperDispatch: Running According to Available Power

Your available power A = your entitled power E + your excess power fraction XPF.

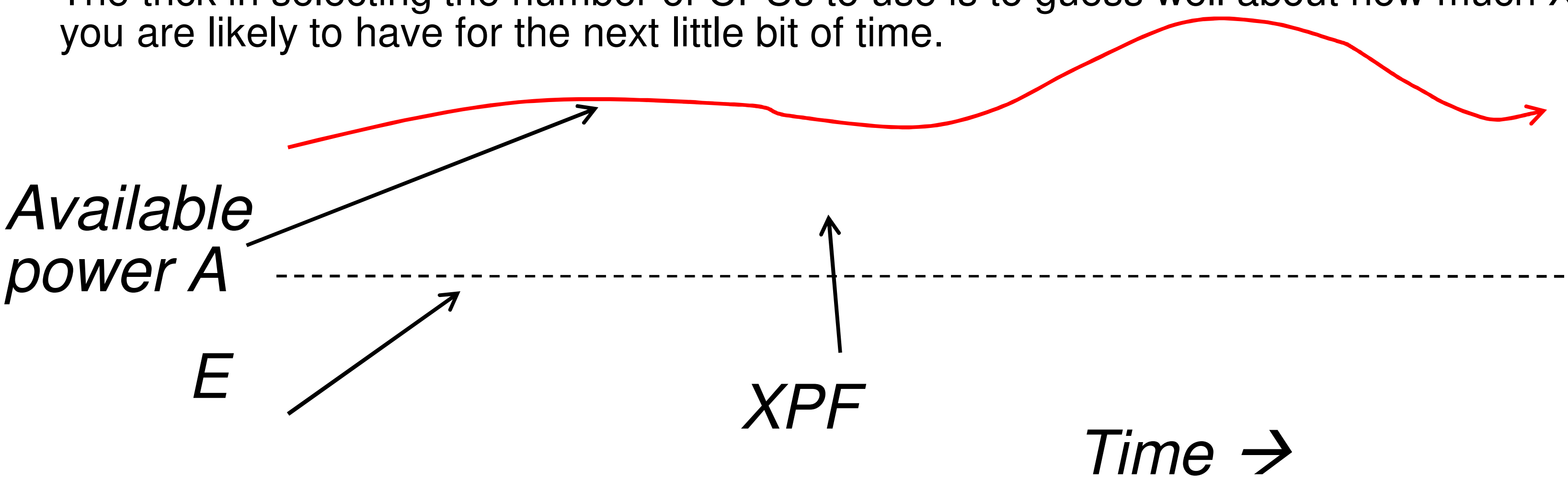
In other words, you can use your E plus what PR/SM will let you use from the excess power XP.

-- You can have all of the XP no one else wants, or your weight-fraction among your competitors.

You want to run with just the right number of CPUs to be able to consume E + XPF if you need it.

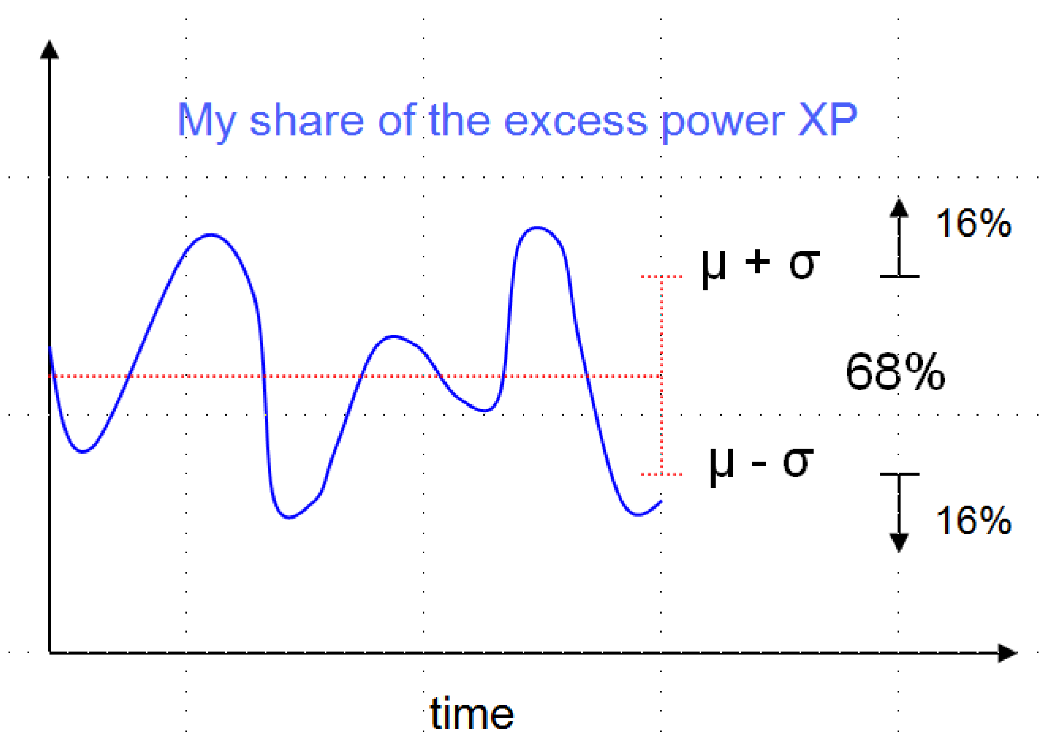
-- For example, if E+XPF = 1458%, you need 15 CPUs unparked to consume it.

The trick in selecting the number of CPUs to use is to guess well about how much XPF you are likely to have for the next little bit of time.



Mixed-engine environments: all of this is done by CPU-type-pool.

z/VM HiperDispatch: How We Calculate XPF'



CP Monitor has been updated to log out all of the observations and all of the predictions.

Every two seconds, we:

- Query all partitions' weights and consumptions, so we can…
- Figure out how much excess power is available to compete for, and…
- Who our competitors for it are…
- And this tells us what our XPF is.

We keep a history of our last 10 observations of XPF.

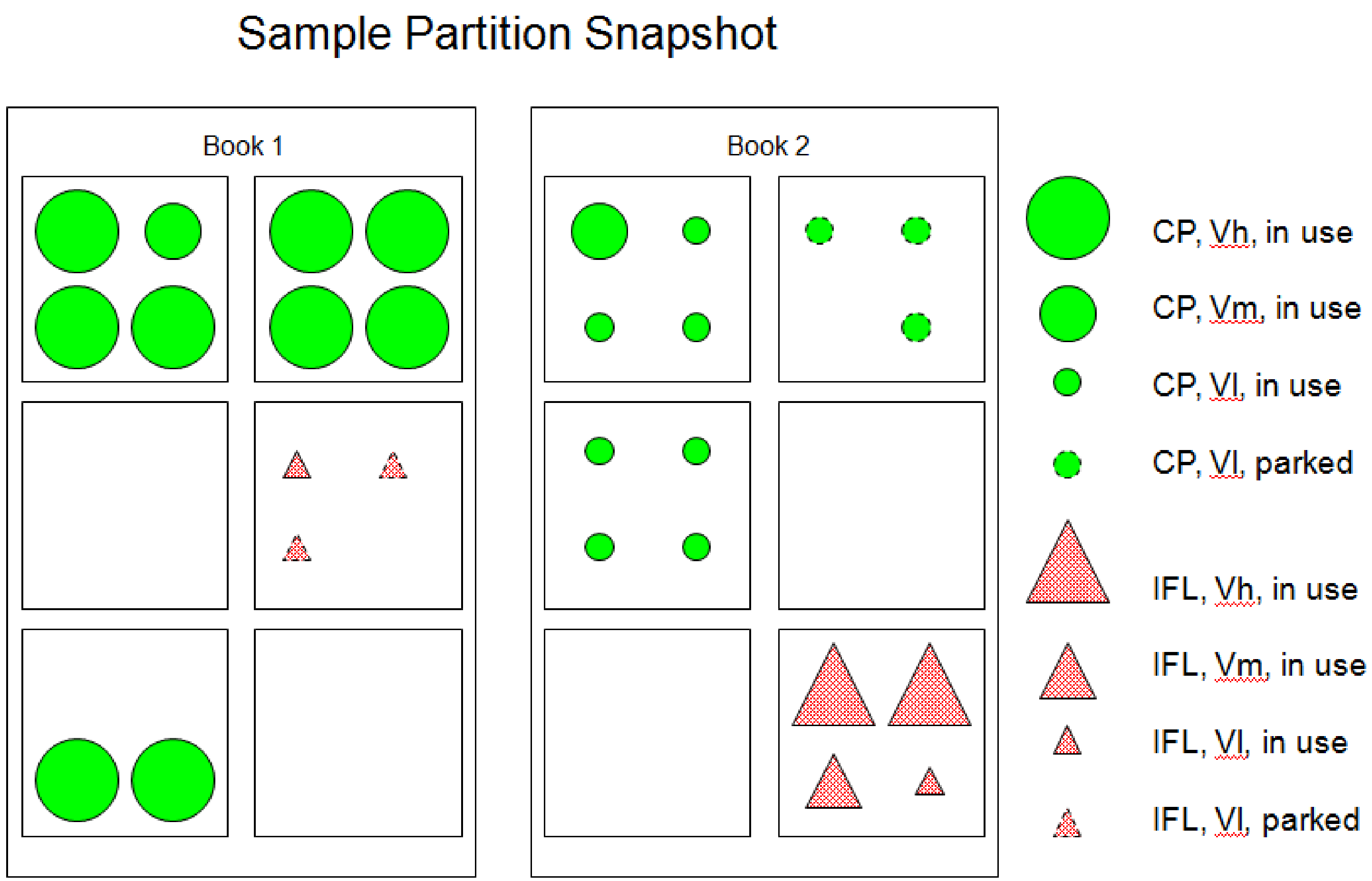
Using the observation history we statistically project a *floor* for XPF, called XPF', for the next two seconds.

And we then *park* or *unpark* according to the engines needed to consume predicted A' = E + XPF'.

z/VM HiperDispatch: Importance of Global Performance Data

- “Global Performance Data” is a setting in the partition’s activation profile, “Security” category
  - Also you can use the SE’s “Change LPAR Security” function to change it while the partition is up
  - z/VM can handle changes in GPD without a re-IPL
- GPD is on by default (in DR scenario, ask your partition provider about it)
- When it is on, the partition can see performance data about all partitions
  - Their weights
  - How much CPU they are consuming
- That performance data lets the z/VM system do all of these things:
  - Determine every partition’s entitlement
  - Determine how much entitled power is being consumed
  - Determine how much excess power is available (XP = TP – EP)
  - Determine which partitions are overconsuming
  - Calculate the z/VM system’s XPF
- z/VM HiperDispatch is substantially crippled if you fail to enable GPD for the partition
  - You might see HCP1052I, “Global performance data is disabled. This may degrade system performance.”
  - You can always use CP QUERY SRM to find out whether GPD is on for your partition

z/VM HiperDispatch: Which LPUs Do We Park?

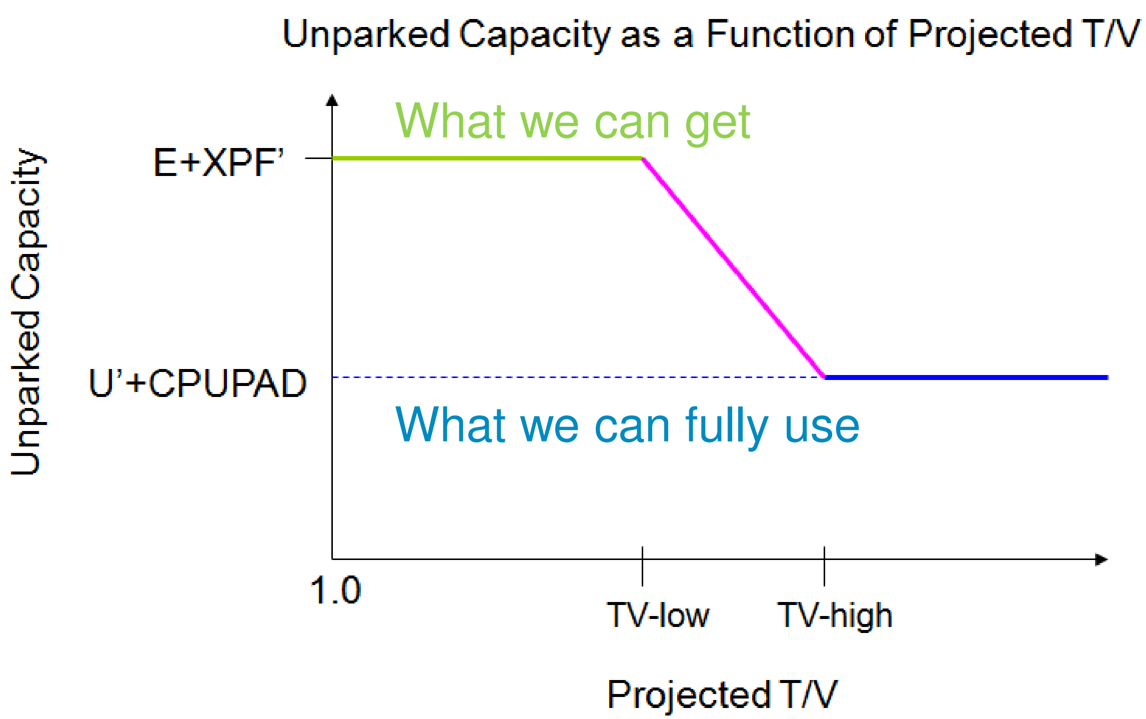


We park topological outliers.

CP Monitor has been updated to log out the park/unpark state every two seconds.

z/VM HiperDispatch: Reducing MP Level to Avoid Overhead

Sometimes, less is more.



Just as we project a floor on XPF, we also project:

- A *ceiling* U' on partition's CPU utilization.

- A *ceiling* T' on partition's T/V ratio.

Then, if U' is small enough and T' is large enough, we *park* LPUs to try to get rid of overhead.

Severity of parking below E+XPF' can be controlled by setting a safety margin or CPUPAD value that we add to U'.

CP Monitor has been updated to log out all of the observations and all of the predictions.

We do not park below E+XPF' on low T' because being wide is not hurting us and the parallelism is apparently there for us to use.

Bottom Line:
Depending on what we can consume, what we can get and the most efficient way to use our resources, we will change how many processors are currently in-use.
It’s normal to see processors parked, even if they are vertical highs.

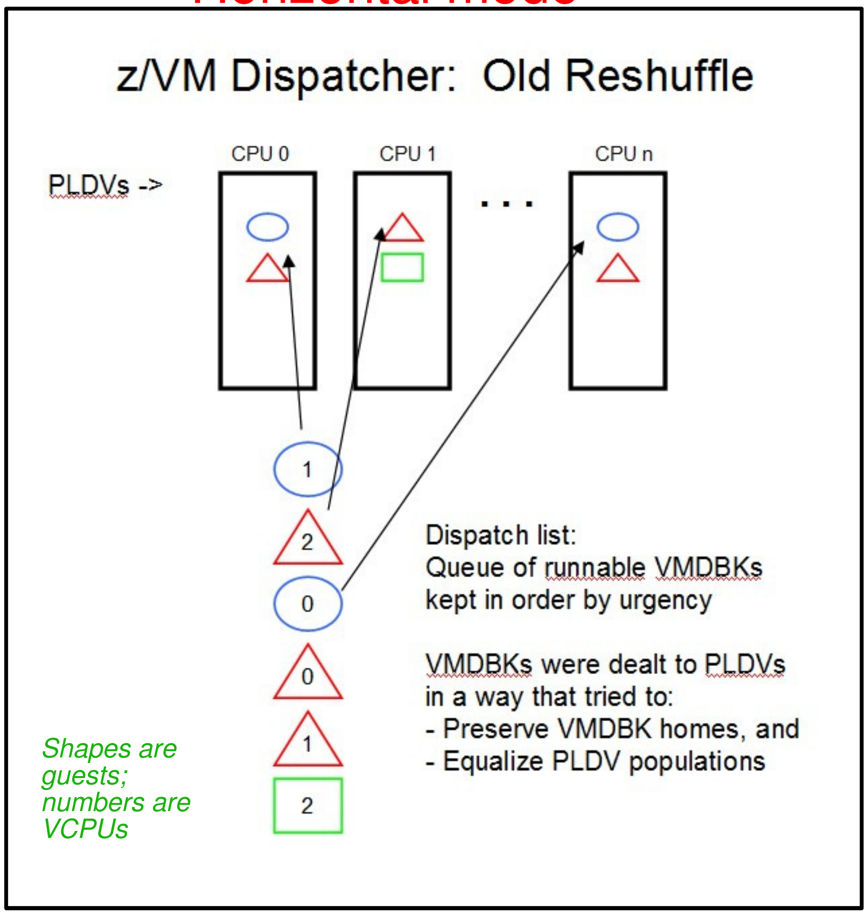
z/VM HiperDispatch: Guest Dispatch Objectives and Strategy

- Objectives: compared to earlier z/VM releases,
  - Reduce movement of virtual CPUs
  - Try to place the virtual CPUs of an N-way guest close to one another

- Strategies:
  - We made several small changes or additions:
    - Reshuffle
    - VMDBK steal
    - Work stacking wakeup
    - Needs help
  - We added a new work distribution algorithm:
    - Rebalance

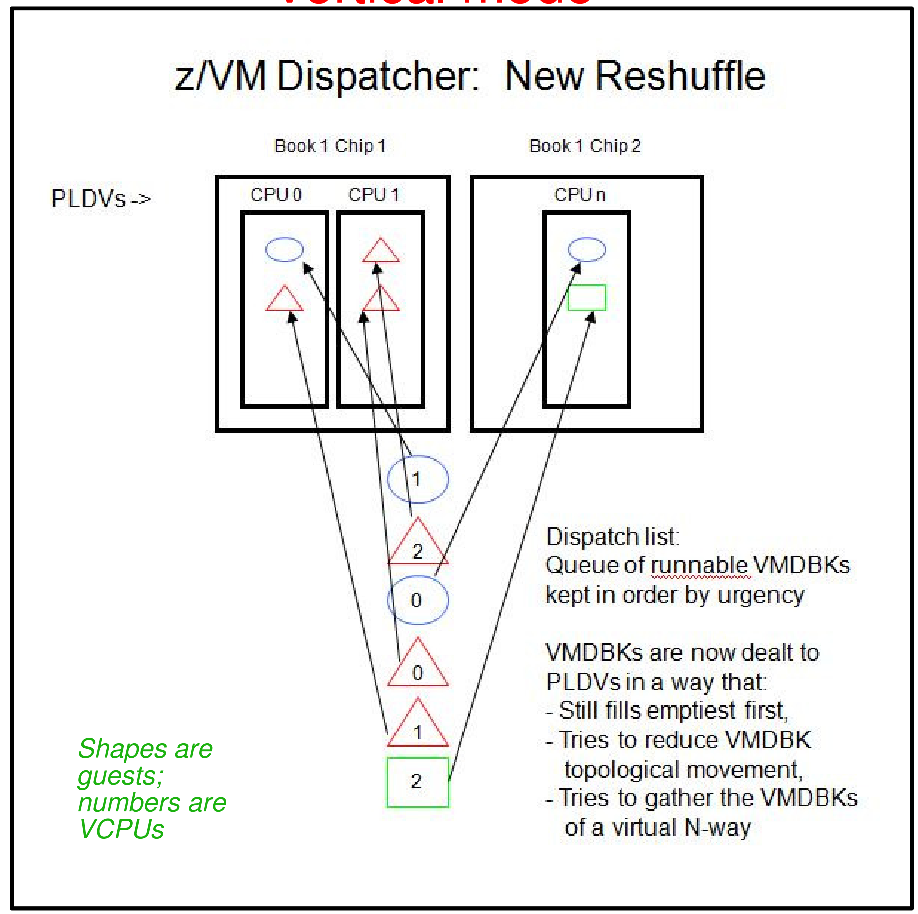
z/VM HiperDispatch: Reshuffle Changes

Horizontal mode



- Balances PLDV populations.
- If not home, then anywhere.
- No awareness of virtual N-ways.

Vertical mode



- Still balances PLDV populations.
- If not home, then hunt outward topologically.
- Collects virtual N-ways.

z/VM HiperDispatch: VMDBK Steal

OLD WAY

0 → 1 → 2 → 3 → 4 … → 19 → 0

Steal from neighbor by CPU number.

Work your way around the ring.

This is not topologically informed.

NEW WAY

(Easy) Steal within your chip.

(Harder) Steal within your book.

(Still harder) Steal across books.

This is topologically informed.

Barriers are for vertical mode only.

CP Monitor has been updated to log out steal behavior as a function of topology drag distance.

z/VM HiperDispatch: Work Stacking CPU Wakeup

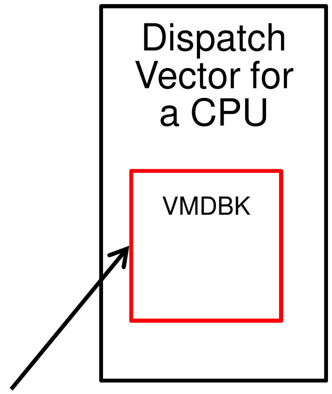
Horizontal mode

- Stack work

- If target CPU is busy,

-Find first wait-state
CPU right of stack target
(CPU 0, 1, 2, 3, …)

-Wake up the found
CPU to prowl for steal



Vertical mode

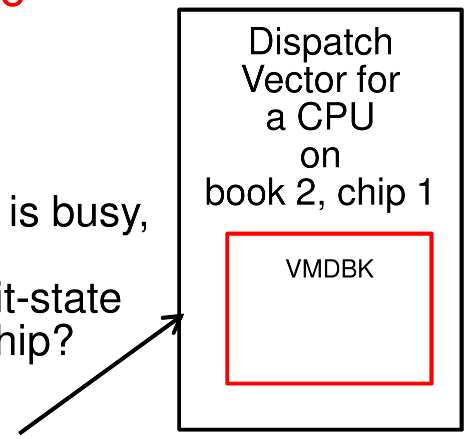
- Stack work

- If target CPU is busy,

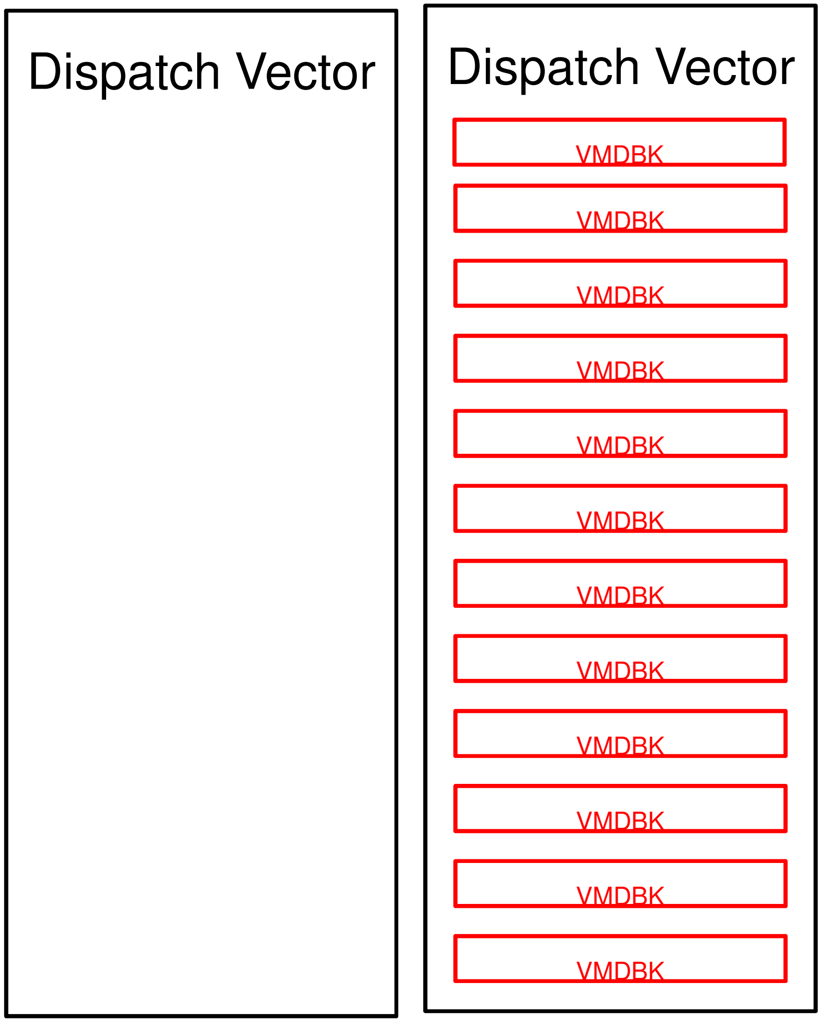
- Is there a wait-state
CPU in this chip?

-Is there a wait-state
CPU in this book?

- Is there a wait-state
CPU anywhere?

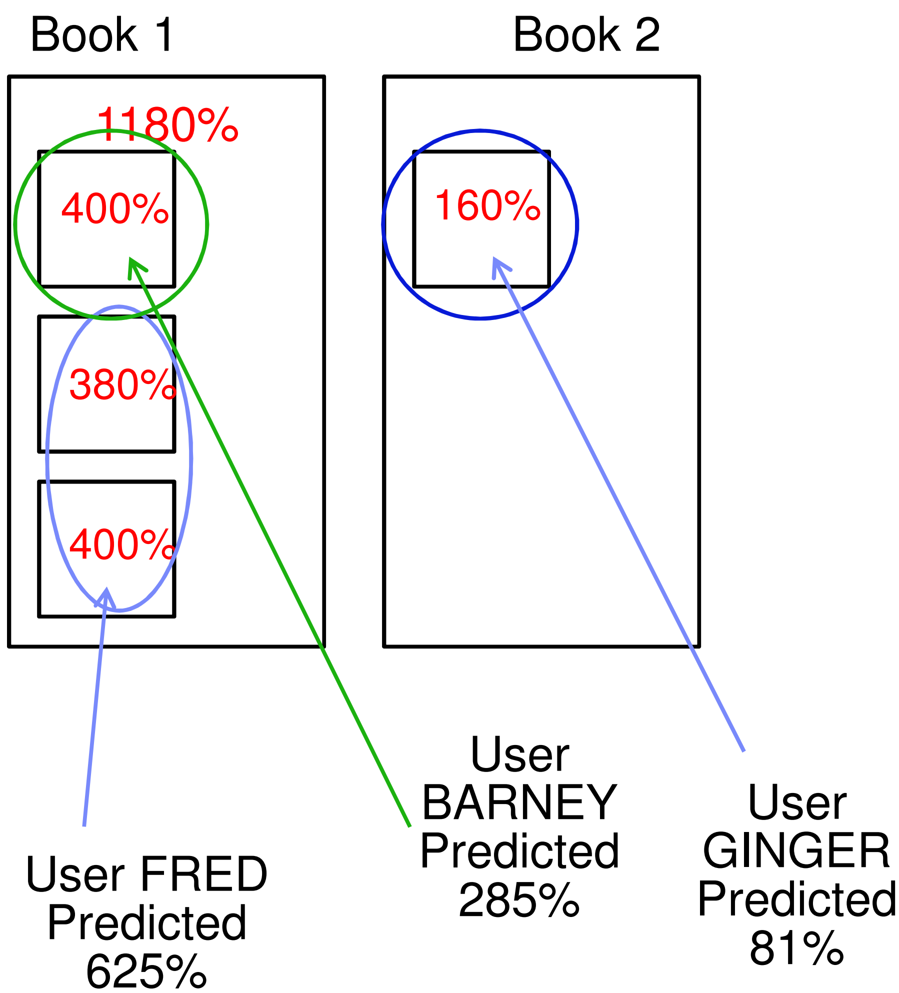


z/VM HiperDispatch: Needs Help



- Come out of wait
- Start working off my dispatch list’s VMDBKs
- About every minor time slice, calculate, “How long since I woke up?”
- If greater than a very long time, wake up the topologically closest waiter anywhere in the system so as to start him prowling to steal
- If greater than only a moderate time, wake up the topologically closest waiter in my book so as to start him prowling to steal

A new choice for scheduling: Rebalance



CP Monitor has been updated to log out the decisions of rebalance.

Rebalance highlights:

- Periodic rework of the assignments of all guests to the topological containers

- Reassigns every guest every pass, not just the VMDREADY, dispatch-list-resident VMDBKs as reshuffle does

- Predicts all guests' near-future utilizations

- Assigns guests to containers like this:
  - Predicted heaviest guests first
  - Spreads load over all containers
  - Tries not to split guests

- Good for situations where:
  - Guests' utilizations are easily distinguished from one another
  - A few heavy guests need not to move around
  - Movement of light users is OK
  - VCPU:LCPU ratio not too big

z/VM HiperDispatch: Knobs

Concept	Knob
Horizontal or vertical	SET SRM POLARIZATION { HORIZONTAL | VERTICAL }
How optimistically to predict XPF floors	SET SRM [TYPE cpu_type] EXCESSUSE { HIGH | MED | LOW }
How much CPUPAD safety margin to allow when we park below available power	SET SRM [TYPE cpu_type] CPUPAD nnnn%
Reshuffle or rebalance	SET SRM DSPWDMETHOD { RESHUFFLE | REBALANCE }

Defaults:
- Vertical mode
- EXCESSUSE MEDIUM (70%-confident floor)
- CPUPAD 100%
- Reshuffle

CP Monitor has been updated to log out the changes to these new SRM settings.

z/VM HiperDispatch: Horizontal Mode vs. Vertical Mode

- Horizontal mode
  - All unparked all the time
  - Reshuffle, but old-style
    - Not topologically aware
    - Does not gather virtual N-ways
  - Steal looks topologically outward
  - Barrier-free steal
  - Work-stack wakeup is not topologically aware
  - Needs-help is in effect
  - LPU dedicate to guest is OK

- It's very much like z/VM 6.2

- Vertical mode
  - Unparks according to A' = E+XPF'
  - Parks below A' if U' seems low and T/V' seems high
  - Reshuffle is new-style
    - Knows system topology
    - Knows about virtual N-ways
  - Steal looks topologically outward
  - Difficulty barriers in steal
  - Work-stack wakeup is topologically aware
  - Needs-help is in effect
  - *Cannot dedicate an LPU to a guest*

- More topological awareness

z/VM HiperDispatch: Aspects of Dedicated Partitions

- The physical PUs backing the partition are not part of the shared physical CPU pool
- If it is a mixed-engine partition, all CPU types are dedicated
- There's no such thing as "weight"
- Its entitlement E is Number of online CPUs * 100%
- A dedicated partition never consumes from XP. XPF=0 always.
- If you run a dedicated partition in vertical mode,
  - All of the logical PUs are vertical highs (Vh)
  - z/VM will park a logical PU only because of high T/V projections

Planning for z/VM HiperDispatch

z/VM HiperDispatch: Planning for It

- Normal best practices for migrating from an earlier release certainly apply

- While you are still on the earlier release, collect measurement data:
  - Know what your key success metrics are and what their success thresholds are
  - Transaction rates – *only you* know where these are on your workloads
  - MONWRITE files – some tips:
    - When: Daily peaks? Month-end processing? Quarter-end processing?
    - Collection tips: http://www.vm.ibm.com/devpages/bkw/monwrite.html
    - CPU MF tips: http://www.vm.ibm.com/perf/reports/zvm/html/620con.html
    - CPU MF reduction: http://www.vm.ibm.com/perf/tips/cpumf.html

- Remember to turn on Global Performance Data for your z/VM partition

- Then go ahead and try z/VM 6.3
  - Remember the default for z/VM 6.3 is vertical mode
  - Consider asking IBM whether your workload is amenable to using rebalance

- When you start running on z/VM 6.3, collect the very same measurement data

- Compare z/VM 6.3 back to z/VM 6.2 to see what the effect is on your workload

- If you like, you can revert to horizontal mode with these means:
  - CP SET SRM POLARIZATION HORIZONTAL
  - SRM statement in the system configuration file

Comments on Workloads

z/VM HiperDispatch: Traits of Workloads

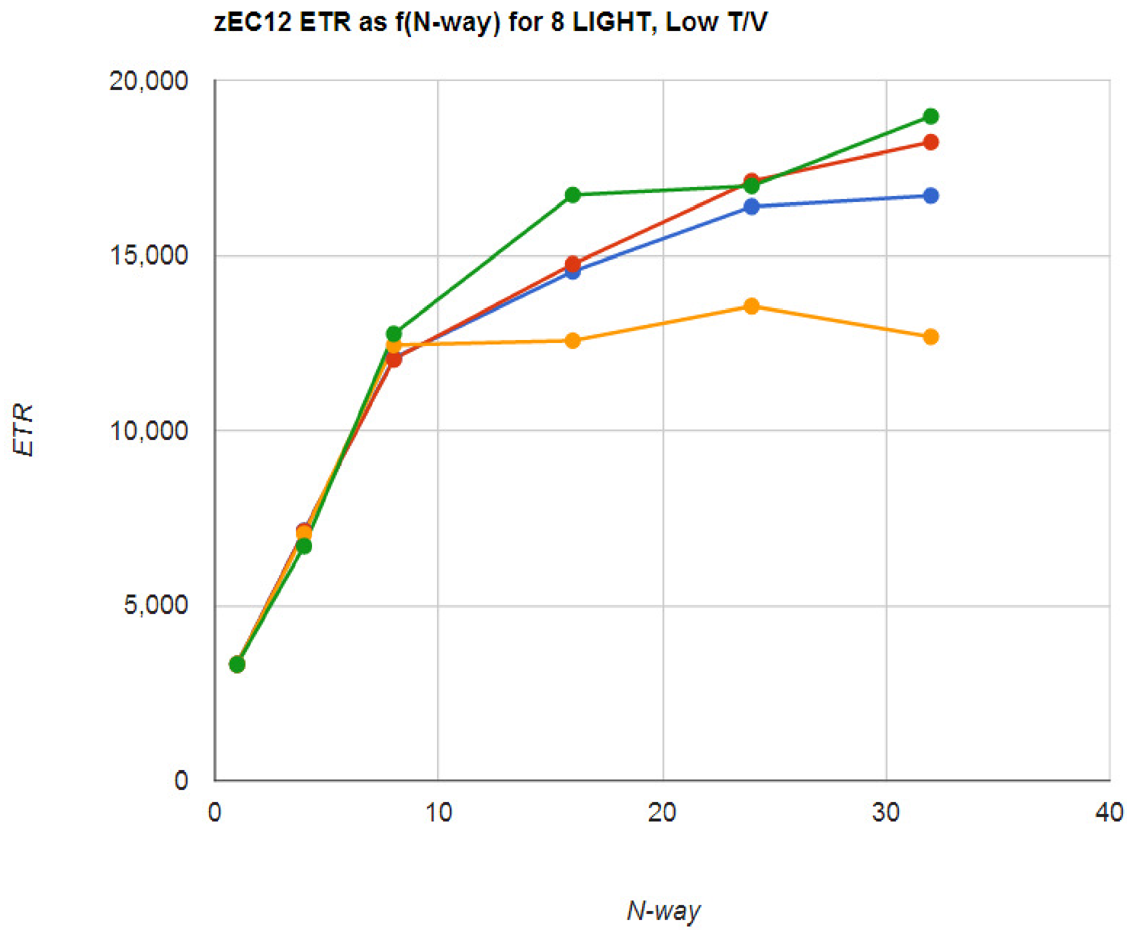
- Amenable workloads for z/VM HiperDispatch:
  - High-CPU, CPU-constrained workloads
    - Improving cache behavior stands to improve performance
  - Active Virtual CPU : Logical CPU ratio isn't too large
    - High ratio has too much context switching to feel much effect
  - Runs in a partition having multiple topology containers
    - Gives z/VM an opportunity to separate guests from one another

- Compare those statements to IBM's statements about PR/SM and partitions

- Indifferent workloads for z/VM HiperDispatch
  - Constrained by something else, such as I/O
  - Memory-overcommitted
  - High Virtual CPU:Logical CPU ratio with every virtual CPU active just a little bit
  - Workloads with bad memory access habits

- Remember that vertical mode also keeps your partition away from the other partitions

z/VM HiperDispatch: Various Numbers of LIGHT Tiles



Blue – 6.2.0
Red – 6.3.0 Horizontal with reshuffle
Orange – 6.3.0 Vertical with reshuffle
Green – 6.3.0 Vertical with Rebalance

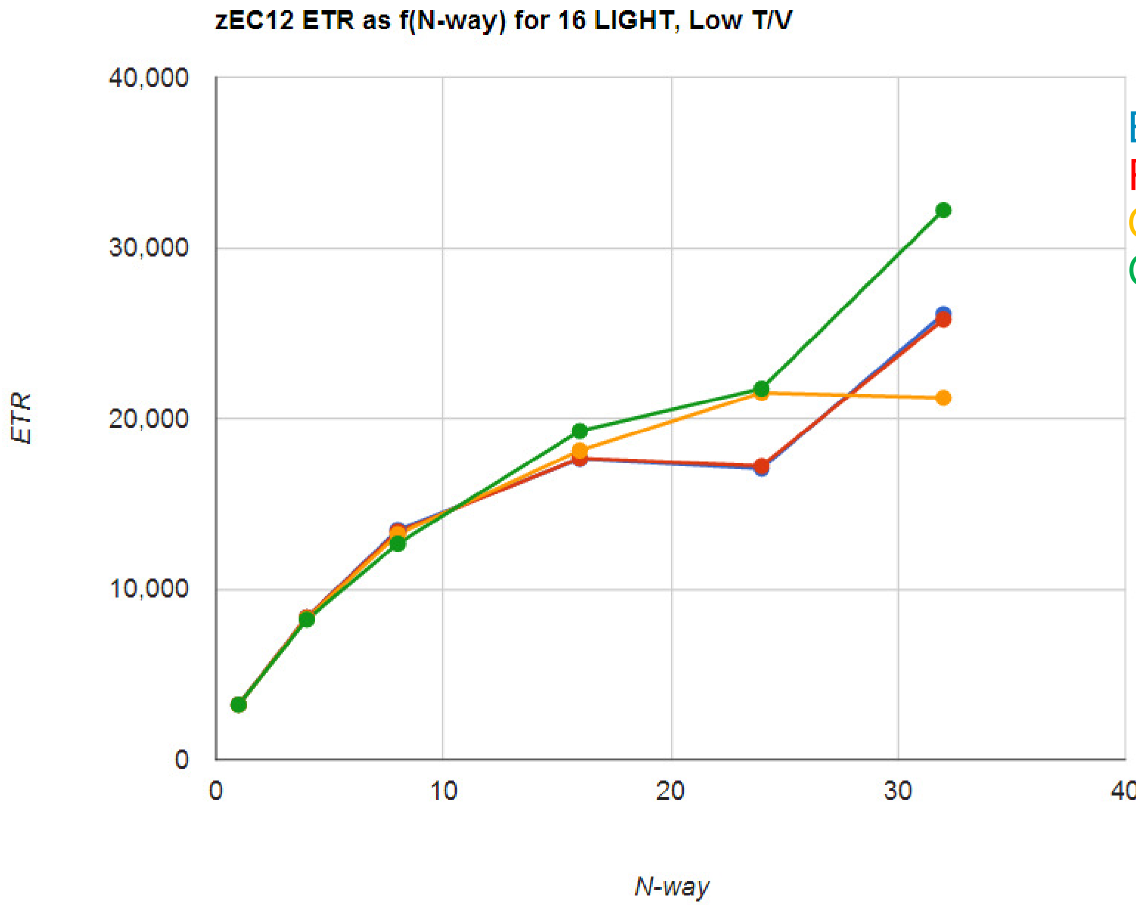
Synthetic, memory-touching workload

A LIGHT tile is 81% busy:
- 1 1-CPU guest - 15% busy
- 1 2-CPU guest with each CPU 33% busy

- No I/O, paging, etc.

- ETR = External Throughput Rate
  - a measure of wall clock time

z/VM HiperDispatch: Various Numbers of LIGHT Tiles



Blue – 6.2.0
Red – 6.3.0 Horizontal with reshuffle
Orange – 6.3.0 Vertical with reshuffle
Green – 6.3.0 Vertical with Rebalance

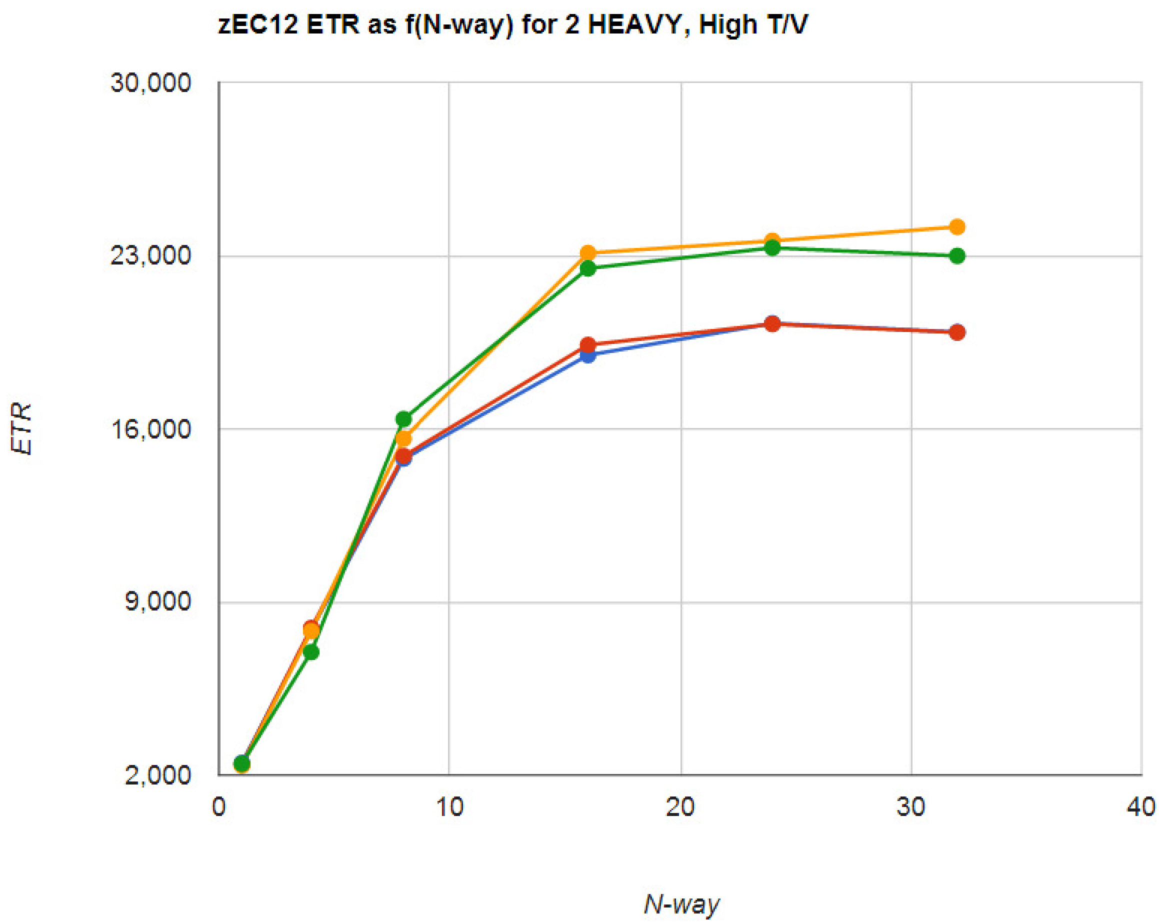
Synthetic, memory-touching workload

A LIGHT tile is 81% busy:
- 1 1-CPU guest - 15% busy
- 1 2-CPU guest with each CPU 33% busy

- No I/O, paging, etc.

- ETR = External Throughput Rate
  - a measure of wall clock time

z/VM HiperDispatch: Various Numbers of HEAVY Tiles



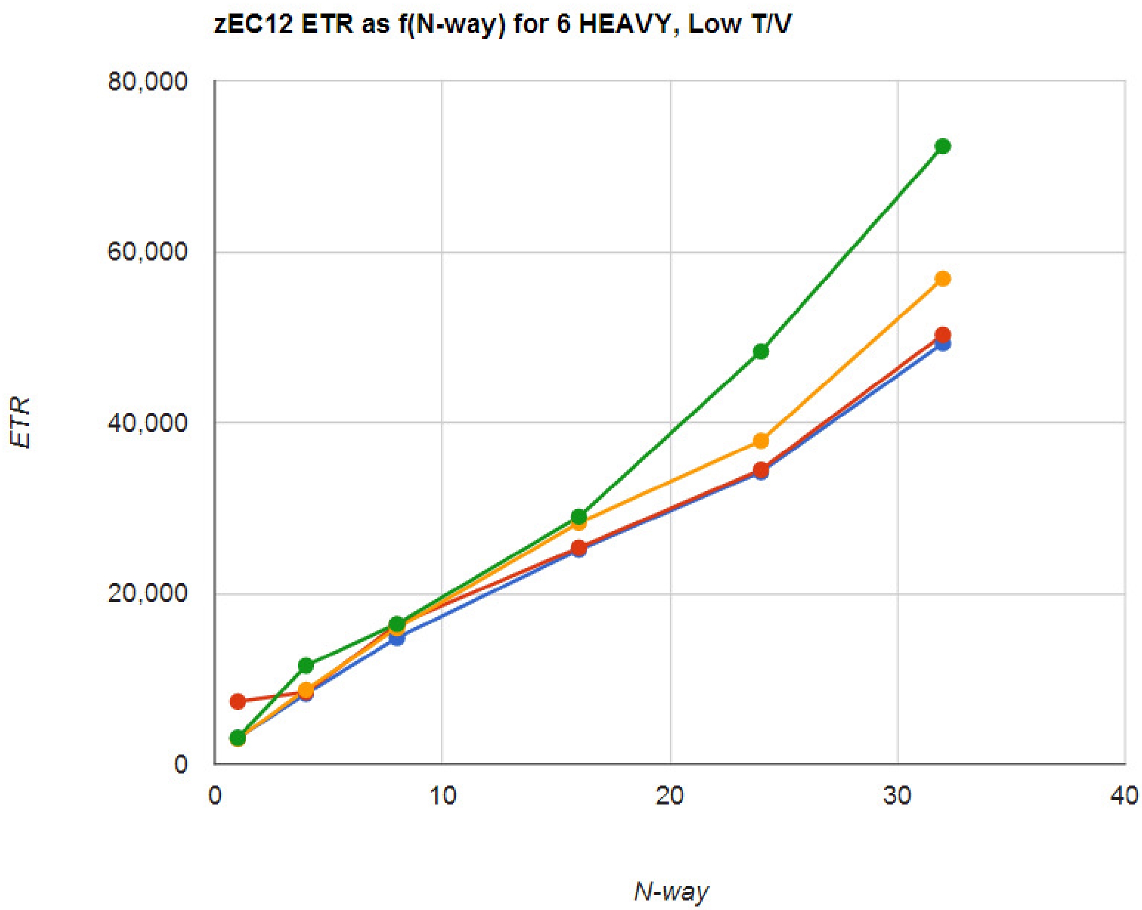
Blue – 6.2.0
Red – 6.3.0 Horizontal with reshuffle
Orange – 6.3.0 Vertical with reshuffle
Green – 6.3.0 Vertical with Rebalance

Synthetic, memory-touching workload

A HEAVY tile is 540% busy:
- 1-CPU guest - 15% busy
- 4-CPU guest with each CPU 31% busy
- 8-CPU guest with each CPU 50% busy

- No I/O, paging, etc.

z/VM HiperDispatch: Various Numbers of HEAVY Tiles



Blue – 6.2.0
Red – 6.3.0 Horizontal with reshuffle
Orange – 6.3.0 Vertical with reshuffle
Green – 6.3.0 Vertical with Rebalance

Synthetic, memory-touching workload

A HEAVY tile is 540% busy:
- 1-CPU guest - 15% busy
- 4-CPU guest with each CPU 31% busy
- 8-CPU guest with each CPU 50% busy

- No I/O, paging, etc.

Summary

z/VM HiperDispatch: Summary

- Objective: try to help CPU performance

- Strategies: pay attention to topology and to z/VM system overhead

- z/VM can now run in vertical mode
  - Runs just widely enough to be able to consume available power
  - Runs more narrowly when it looks like system overhead is a problem
  - Guest dispatch pays more attention to recent run location and to virtual N-way
  - CPU wakeup tries to be topologically friendly
  - VCPU steal tries to be topologically friendly

- Planning: not too difficult, just remember to measure before and after

- Amenable workloads should see improvements

- CP Monitor conveys the new information

- z/VM Performance Toolkit has been updated

z/VM HiperDispatch: References

- z/VM Planning and Administration – nice abstract writeup on HiperDispatch
- z/VM Performance – points to P&A
- z/VM CP Commands and Utilities – descriptions of the new commands
- z/VM Performance Report on www.vm.ibm.com/perf/.
- z/VM HiperDispatch article: http://www.vm.ibm.com/perf/reports/zvm/html/630hd.html
- This presentation cites two www.vm.ibm.com articles describing z/VM and the CPU Measurement Facility.

Please remember to do an evaluation.

Session 15744

Thanks!