CPU MF for Efficiency

Session 26998
February 26, 2020

John Burg
IBM Washington Systems Center
jpburg@us.ibm.com

Agenda – CPU MF Efficiency

- Value of CPU MF Counters
  What and Why
  Best Practice and TDA requirement
- z15 Update
- RNI and L1MP Based Decision LSPR Match
  – RNI is **not** a Performance metric
  – z14 Migration Studies and RNI update
- Efficiency
  – LPAR Controls – Example of 99-14s and VL CPI (Finite CPI)
  – Store Into Instruction Stream (SIIS)
  – COBOL - z14 Vector Packed Decimal Facility and CPU MF Usage
- Crypto, Encryption and zEDC Measurement Example
- z15 New zEDC Extended Counters
- Summary
- Backup
  – Fundamental Components of Workload Capacity Performance
    - Micro processor and Nest

CPU Measurement Facility

- Introduced in z10 and later processors
- Facility that provides hardware instrumentation data for production systems
- Two Major components
  - Counters
    - **Cache and memory hierarchy information**
    - SCPs supported include z/OS and z/VM
  - Sampling
- New z/OS HIS started task
  - Gathered on an LPAR basis
  - Writes SMF 113 records
  - z/OS implementation instructions http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TC000066
- New z/VM Monitor Records
  - Gathered on an LPAR basis – all guests are aggregated
  - Writes new Domain 5 (Processor) Record 13 (CPU MF Counters) records
  - z/VM implementation instructions http://www.vm.ibm.com/perf/tips/cpumfhow.html
- Minimal overhead

Value of CPU Measurement Facility (CPU MF)

- Recommended Methodology for successful z Systems Processor Capacity Planning
  - Need on “Before” processor to determine LSPR workload
  - **TDA process has been updated to require CPU MF Counters enabled**
- Validate achieved IBM Z processor performance
  - Needed on “Before” and “After” processors
- Provide insights for new features and functions
  - Continuously running on all LPARs
  - **Efficiency is even more important today, with new pricing models like Tailored Fit**

Capturing CPU MF data is an Industry “Best Practice”

z15 Update and Metrics

z15 Update

New

- Updated CPU MF Formulas
  - SMF 113 Updates
  - LSPR Workload Match Table
  - Formulas: Sourcing formulas similar to z14
  - RNI and TLB updates

- New z15 Crypto Counters
  - Elliptic-curve Cryptography (ECC) new with z15

- New z15 Extended Counters
  - z15 Integrated Adapter for zEDC (Synchronous)

- z14 Updates

z/OS SMF 113 Record

- SMF113_2_CTRVN2
  - “1” = z10
  - “2” = z196 / z114
  - “3” = zEC12 / zBC12
  - “4” = z13 / z13s
  - “5” = z14
  - “6” = z15

New

RNI-based LSPR Workload Decision Table

L1MP	RNI	LSPR Workload Match
< 3%	>= 0.75 < 0.75	AVERAGE LOW
3% to 6%	>1.0 0.6 to 1.0 < 0.6	HIGH AVERAGE LOW
> 6%	>= 0.75 < 0.75	HIGH AVERAGE

Current table applies to z10 EC, z10 BC, z196, z114, zEC12, zBC12, z13, z13s, z14/ZR1, and z15 CPU MF data

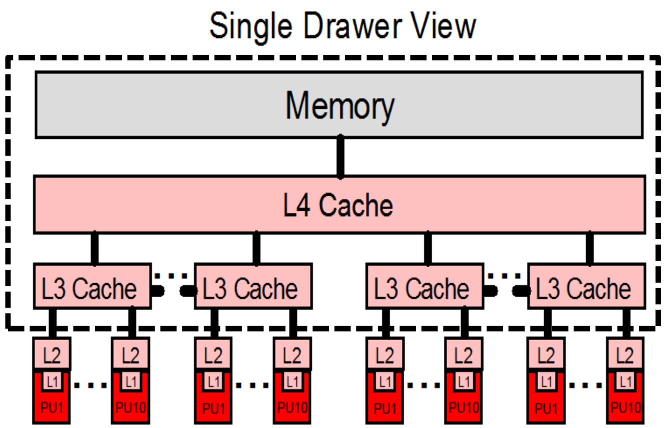
New

- RNI is **not** a Performance metric
  - RNI and L1MP allows one to match their workload to an LSPR workload
    - Any other use of RNI is not valid

z14 vs z15 Hardware Comparison

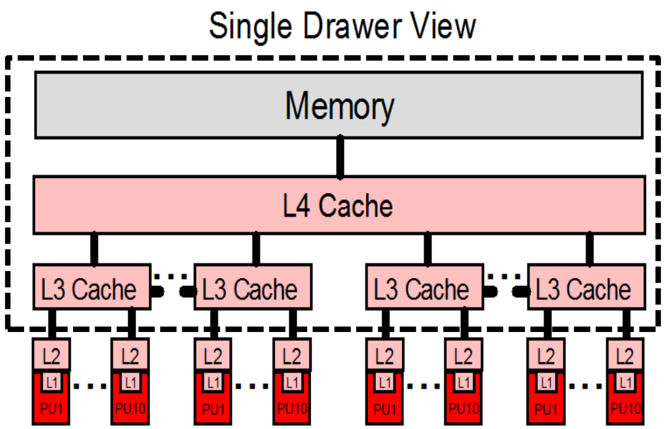
- **z14 (3906)**
  - CPU (14nm SOI)
    - 5.2 GHz
  - Caches
    - L1 private 128k i, 128k d
    - L2 private 2 MB i, 4 MB d
    - L3 shared 128 MB per chip
    - L4 shared 672 MB per drawer

- **Topology**
  - 10 cores + 1 L3 per CP chip
  - 2-or-3 CP chips per cluster
  - 2 clusters + 1 L4 per drawer
  - 4 drawers max per CPC
  - Book interconnect: NUMA star



- **z15 (8561)**
  - CPU (14 nm SOI)
    - 5.2 GHz
  - Caches
    - L1 private 128k i, 128k d
    - L2 private **4 MB i**, 4 MB d
    - L3 shared **256 MB** per chip
    - L4 shared **960 MB** per drawer

- **Topology**
  - **12** cores + 1 L3 per CP chip
  - **2** CP chips per cluster
  - 2 clusters + 1 L4 per drawer
  - **5** drawers max per CPC
  - Book interconnect: NUMA star



Formulas – z15

Workload Characterization
L1 Sourcing from cache/memory hierarchy

Metric	**Calculation** – *note all fields are deltas. SMF113-1s are deltas. SMF 113-2s are cumulative.*
CPI	B0 / B1
PRBSTATE	(P33 / B1) * 100
L1MP	((B2+B4) / B1) * 100
L2P	((E133+E136) / (B2+B4)) * 100
L3P	((E144+E146+E162+E164) / (B2+B4)) * 100
L4LP	((E147+E149+E156+E165+E167+E174+E150+E152+E158+E168+ E170) / (B2+B4)) * 100
L4RP	((E153+E155+E157+E171+E173+E175) / (B2+B4)) * 100
MEMP	((E145 + E148 + E151 + E154 + E163 + E166 + E169 + E172) / (B2+B4)) * 100
LPARCPU	(((1/CPSP/1,000,000) * B0) / Interval *in Seconds*) * 100

CPI – Cycles per Instruction
Prb State - % Problem State
L1MP – Level 1 Miss Per 100 instructions
L2P – % sourced from Level 2 cache
L3P – % sourced from Level 3 on same Chip cache
L4LP – % sourced from Level 4 Local cache (on same book)
L4RP – % sourced from Level 4 Remote cache (on different book)
MEMP - % sourced from Memory
LPARCPU - APPL% (GCPs, zAAPs, zIIPs) captured and uncaptured

B* - Basic Counter Set - Counter Number
P* - Problem-State Counter Set - Counter Number
See “The Load-Program-Parameter and CPU-Measurement Facilities” SA23-2260 for full description
E* - Extended Counters - Counter Number
See “IBM The CPU-Measurement Facility Extended Counters Definition for z10, z196/ z114, zEC12 /zBC12, z13/z13s, z14 and z15 SA23-2261-05 for full description
CPSP - SMF113_2_CPSP “CPU Speed”

Updated September 23, 2019

Note these Formulas may change in the future

Formulas – z15 Additional

Metric	**Calculation**– *note all fields are deltas. SMF113-1s are deltas. SMF 113-2s are cumulative.*
Est Instr Cmplx CPI	CPI – Estimated Finite CPI
Est Finite CPI	E143 / B1
Est SCPL1M	E143 / (B2+B4)
Rel Nest Intensity	2.9*(0.45*L3P + 1.5*L4LP + 3.2*L4RP + 6.5*MEMP) / 100
Eff GHz	CPSP / 1000

Updated September 23, 2019

Note these Formulas may change in the future

Est Instr Cmplx CPI – Estimated Instruction Complexity CPI (infinite L1)

Est Finite CPI – Estimated CPI from Finite cache/memory

Est SCPL1M – Estimated Sourcing Cycles per Level 1 Miss

Rel Nest Intensity –Reflects distribution and latency of sourcing from shared caches and memory

Eff GHz – Effective gigahertz for GCPs, cycles per nanosecond

Workload Characterization
L1 Sourcing from cache/memory hierarchy

B* - Basic Counter Set - Counter Number

P* - Problem-State Counter Set - Counter Number

See "The Load-Program-Parameter and CPU-Measurement Facilities" SA23-2260 for full description

E* - Extended Counters - Counter Number

See "IBM The CPU-Measurement Facility Extended Counters Definition for z10, z196/ z114, zEC12 /zBC12, z13/z13s, z14 and z15 SA23-2261-05 for full description

CPSP - SMF113_2_CPSP "CPU Speed"

Formulas – z15 Additional TLB

Metric	**Calculation** – *note all fields are <u>deltas</u>. SMF113-1s are deltas. SMF 113-2s are cumulative.*
Est. TLB1 CPU Miss % of Total CPU	((E130+E135) / B0) * (E143 / (B3+B5)) *100
Estimated TLB1 Cycles per TLB Miss	(E130+E135) / (E129+E134) * (E143 / (B3+B5))
PTE % of all TLB1 Misses	N/A with processor design change
TLB Miss Rate	(E129 + E134) / interval

Est. TLB1 CPU Miss % of Total CPU - Estimated TLB CPU % of Total CPU

Estimated TLB1 Cycles per TLB Miss – Estimated Cycles per TLB Miss

PTE % of all TLB1 Misses – Page Table Entry % misses

TLB Miss Rate – TLB Misses per interval (interval is defined by user for length of measurement and units)

B* - Basic Counter Set - Counter Number

P* - Problem-State Counter Set - Counter Number

See "The Load-Program-Parameter and CPU-Measurement Facilities" SA23-2260 for full description

E* - Extended Counters - Counter Number

See "IBM The CPU-Measurement Facility Extended Counters Definition for z10, z196/ z114, zEC12 /zBC12, z13/z13s, z14 and z15 SA23-2261-05 for full description

CPSP - SMF113_2_CPSP "CPU Speed"

Updated September 23, 2019

Note these Formulas may change in the future

Looking for z15 Migration “Volunteers” SMF data

- Want to validate / refine Workload selection metrics

Looking for “Volunteers”

(3 days, 24 hours/day, SMF 70s, 71s, 72s, 99 subtype 14s,113s per LPAR)

“Before z13 / z14” and “After z15”

Production partitions preferred

If interested send note to jpburg@us.ibm.com,

No deliverable will be returned

Benefit: Opportunity to ensure your data is used to influence analysis

Formulas – z14 Additional TLB

Metric	**Calculation** – *note all fields are deltas. SMF113-1s are deltas. SMF 113-2s are cumulative.*
Est. TLB1 CPU Miss % of Total CPU	((E130+E135) / B0) * (E143 / (B3+B5)) *100
Estimated TLB1 Cycles per TLB Miss	(E130+E135) / (E129+E134) * (E143 / (B3+B5))
PTE % of all TLB1 Misses	N/A with processor design change
TLB Miss Rate	(E129 + E134) / interval

Est. TLB1 CPU Miss % of Total CPU - Estimated TLB CPU % of Total CPU

Estimated TLB1 Cycles per TLB Miss – Estimated Cycles per TLB Miss

PTE % of all TLB1 Misses – Page Table Entry % misses

TLB Miss Rate – TLB Misses per interval (interval is defined by user for length of measurement and units)

B* - Basic Counter Set - Counter Number

P* - Problem-State Counter Set - Counter Number

See "The Load-Program-Parameter and CPU-Measurement Facilities" SA23-2260 for full description

E* - Extended Counters - Counter Number

See "IBM The CPU-Measurement Facility Extended Counters Definition for z10, z196/ z114, zEC12 /zBC12, z13/z13s and z14" SA23-2261-04 for full description

CPSP - SMF113_2_CPSP "CPU Speed"

Updated September 23, 2019

Note these Formulas may change in the future

End

z15 Update and Metrics

z14 / ZR1 RNI Update

New

- z14 / ZR1
  - Received numerous before/after z14 migration “volunteers” SMF data
    - Thank you!
  - SMF data is used to tune metrics and help design future processors
  - Effective Aug 2019, **there is No change to the z14 RNI formula**
  - The RNI-based (and L1MP) LSPR workload match table is unchanged
  - **Important for z15 Capacity Sizing**

SMF 30 Instruction Counts

- SMFPRMxx   SMF30COUNT|NOSMF30COUNT
  - –To receive non-zero data, the HIS component must be active and collecting the Basic Counter Sets
  - –These instruction counts may include instructions from z/OS events not attributable to the job
    - Degree of error to the counts can be significant and random impacting the validity of the data
  - –If Analysts understand and accept the variability the data may be used for tuning and problem isolation
    - Use the instruction counts with the CPU charged to create an individual Job CPI

My experience is Instruction Counts are valid / reasonable. You can correlate with SMF 30 EXCPs and SSCHs.(e.g. CPU / EXCP vs CPI). Typically when it has "not been reasonable" Instruction Counts are way too high, e.g. E18, and thus the CPI is very very low. Some valid examples will follow.

Sample WSC z15 CPU MF Metrics

CPI | Work Load Mix | CPI Micro Proc | CPI L2 & 'Nest' | How Often? | 100% of L1 Misses | How Far? | How Much? | LSPR Workload Match

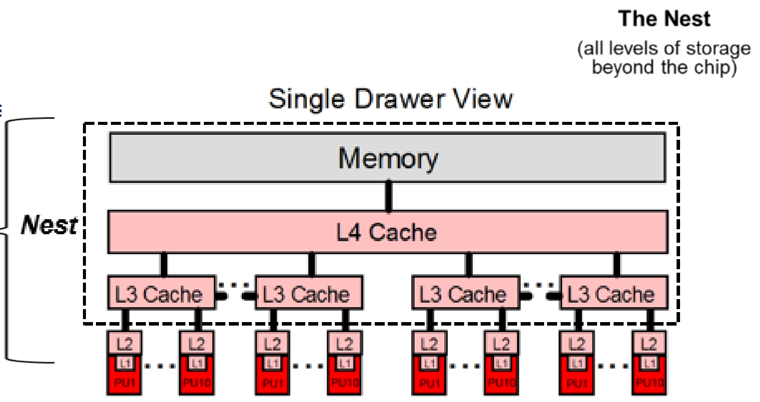
Hour	CPI	Prb State	Instr Cmplx CPI	Finite CPI	SCPL1M	L1MP	L2P	L3P	L4LP	L4RP	MEMP	Rel Nest Intensity	LPARCPU	Eff GHz	Machine Type	LSPR Wkld
11.58	1.44	53.4	1.07	0.37	17	2.1	83.4	15.6	0.3	0.0	0.7	0.35	1.6	5.2	Z15	LOW

Finite CPI
Important Metric for LPAR Controls

CPI – Cycles per Instruction – A rate of delivery metric
- **EICPI – Estimated Instruction Complexity CPI – Indicates portion of CPI related to the microprocessor**
- **EFCPI - Estimated Finite CPI – Indicates portion of CPI related to the L2 private and shared caches (Nes**

L1MP Sourced from Cache Hierarchy

Cycles / Instructions



Workload Characterization
L1 Sourcing from cache/memory hierarchy

CPU MF Efficiency

LPAR Controls

z13 / z14 / z15 Performance Recommendations

- Set weights and logicals to meet needs (GCPs and zIIPs)
  1. Understand LPAR capacity requirements across time
  2. Manage weights to meet capacity requirements
  3. Assign logicals to meet weights (CPs by weight)
     - Only 1-2 more than needed to meet CPs by weight
     - Optimize for VHs
  4. Utilize zPCR to help assess 2 and 3

  See Best Practice for ***Number of Logical CPs Defined for an LPAR***
  www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD106388

- Utilize SMF 99-14s and SMF 113s to understand topology and impact by polarity / logical processor

- Topology Change can occur for any change in
  – LPAR (de)activation, Weight, Logical Processor
    - Weight change includes IRD, WLM Capping (Defined Capacity and Group Capacity)

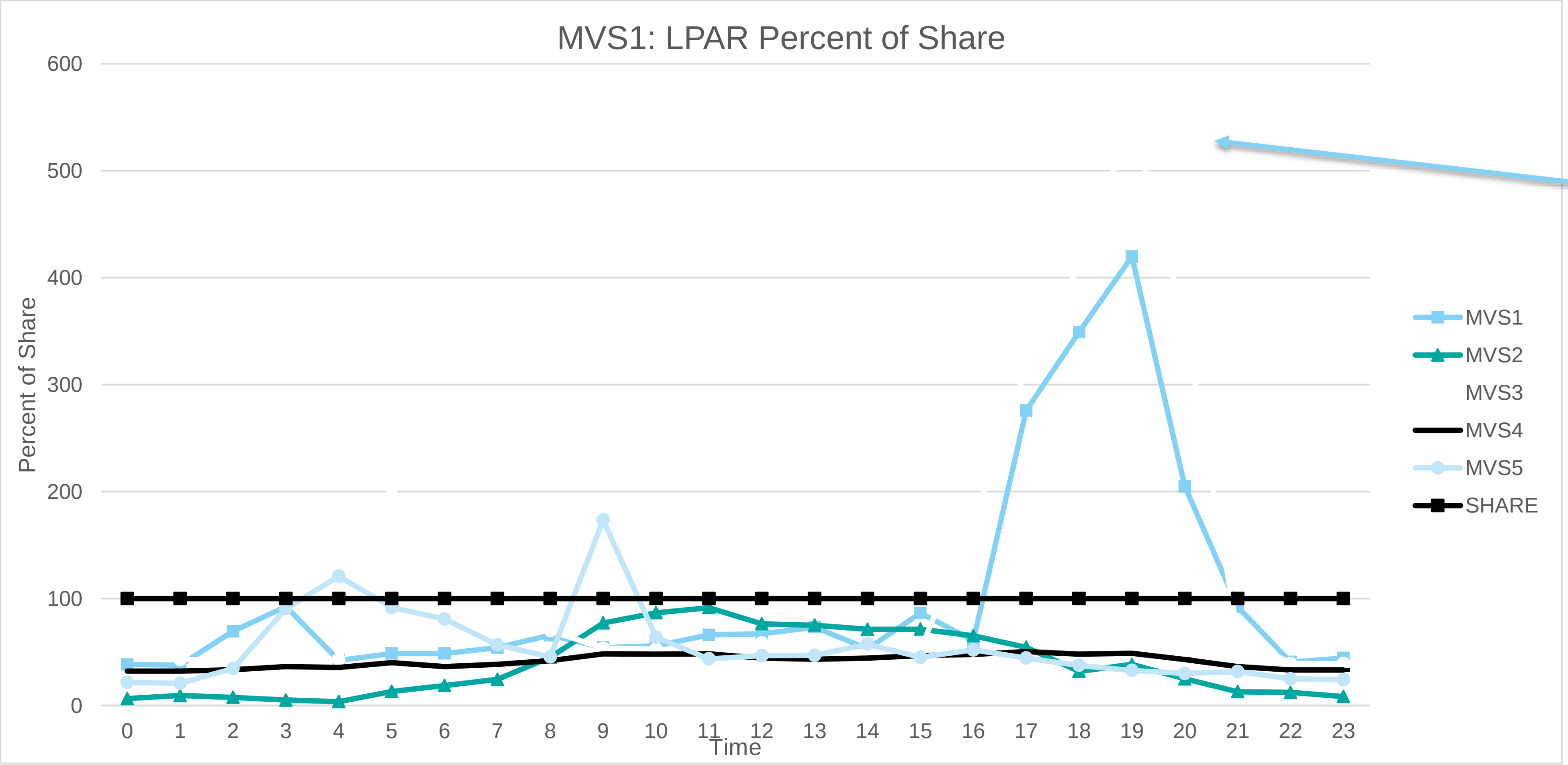
Recommendations for Defining Logical CPs

- New Best Practice document for defining logical CPs and zIIPs to an LPAR
  - www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD106388

- Recommendations
  - Define 1-2 more logicals than needed to meet CPs by weight
    - Don’t define all the logicals on a CEC to the LPAR
  - Reasons:
    - **Work runs most efficiently if you run with defined weight using VHs and VMs**
    - LPAR Busy value displayed on online monitors is relative to number of LCPs
    - LPAR time slice is sensitive to number of logicals, fewer logicals will lead to longer time slices
    - Reduce the impact of a CPU Loop, fewer logicals limits potential impact
    - z/OS operations like Quiesce need to be done even for parked logicals
    - Additional system resources utilized for each logical processor

Optimize LPAR Performance

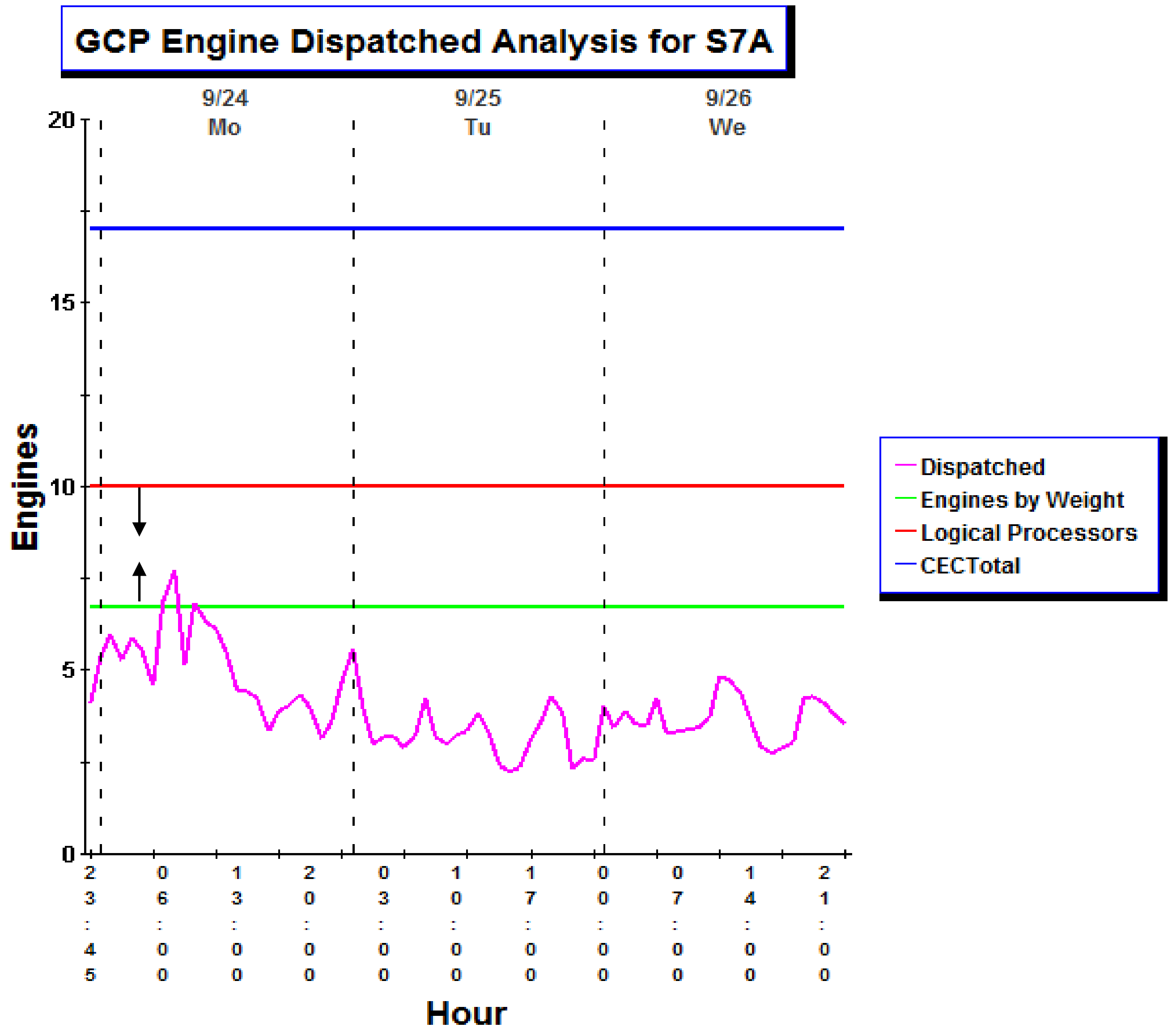
- Set weights to meet capacity needs
  - Optimize VHs to improve performance
- Need to manage weights across shift changes and business cycles



Increase weight to optimize VHs for MVS1, and MVS3

Evaluate Engine Performance

- More complete, overall view of capacity
  - Can see impacts of capping easier
- Dispatched (Busy) greater than Engines by Weight then using whitespace
  - Significant, continuous, use of VLs indicates need for weight change



SMF 99 Subtype 14 – HiperDispatch Topology

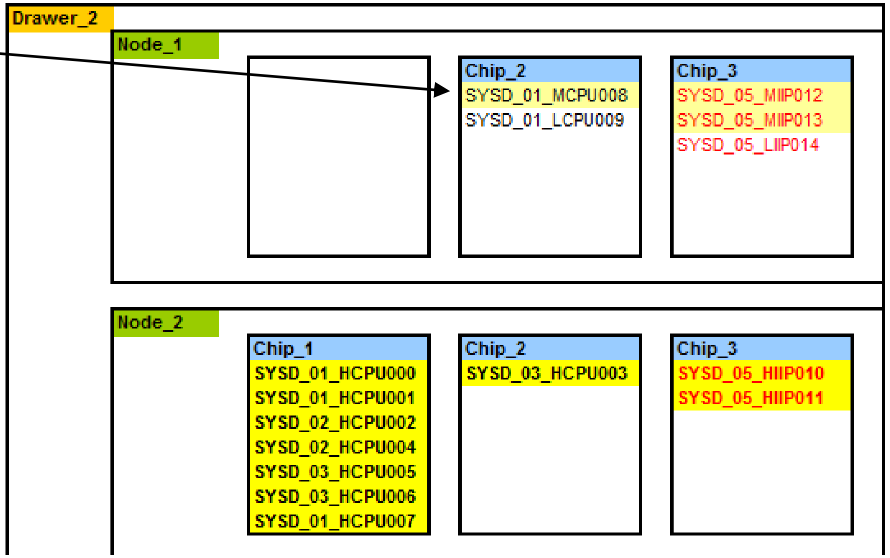
- SMF 99 Subtype 14 contains HiperDispatch Topology data including:
  - Logical Processor characteristics: Polarization (VH, VM, VL), Affinity Node, etc.
  - Physical topology information
    - zEC12 Book / Chip
    - z13 Drawer / Node / Chip
    - z14 Drawer / Node / Chip
    - z15 Drawer / Node / Chip
- Written every 5 minutes or when a Topology change occurs
    - e.g. Configuration change or weight change
- May be useful to help understand why performance changed
- Provides a “Topology Change” indicator
  - Can identify when the topology changed occurred
- Recommendation is to collect SMF 99 subtype 14s for each System / LPAR
- *WLM Topology Report* available to process SMF 99 subtype 14 records
  - http://www.ibm.com/systems/z/os/zos/features/wlm/WLM_Further_Info_Tools.html#Topology

z13 Topology Example

Description **“SYSD_01_MCPU008”**

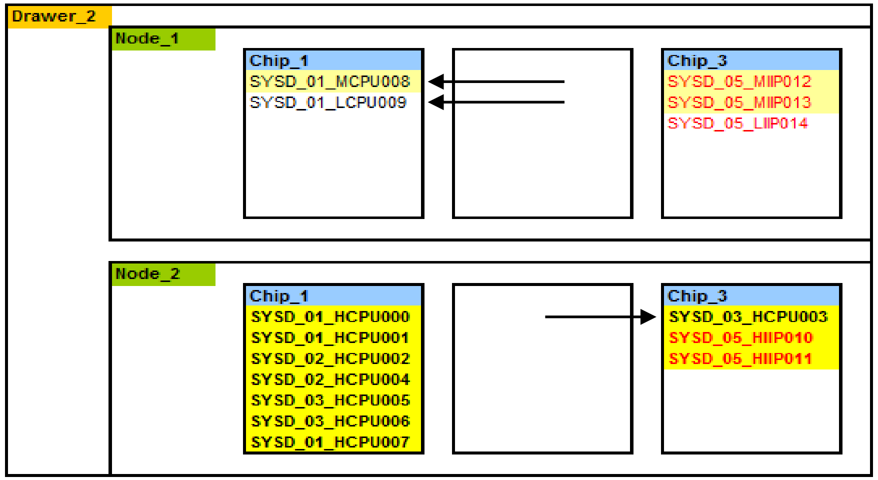
System	Affinity Node	Polarity H M L	CPU / zIIP	Logical Number
SYSD	01	M	CPU	008

Topology for 01/30/2015-14:08:32, System: SYSD



Topology before SYSD Tests

Topology for 01/30/2015-14:11:42, System: SYSD



Topology for 14:20 - 14:40 SYSD Tests

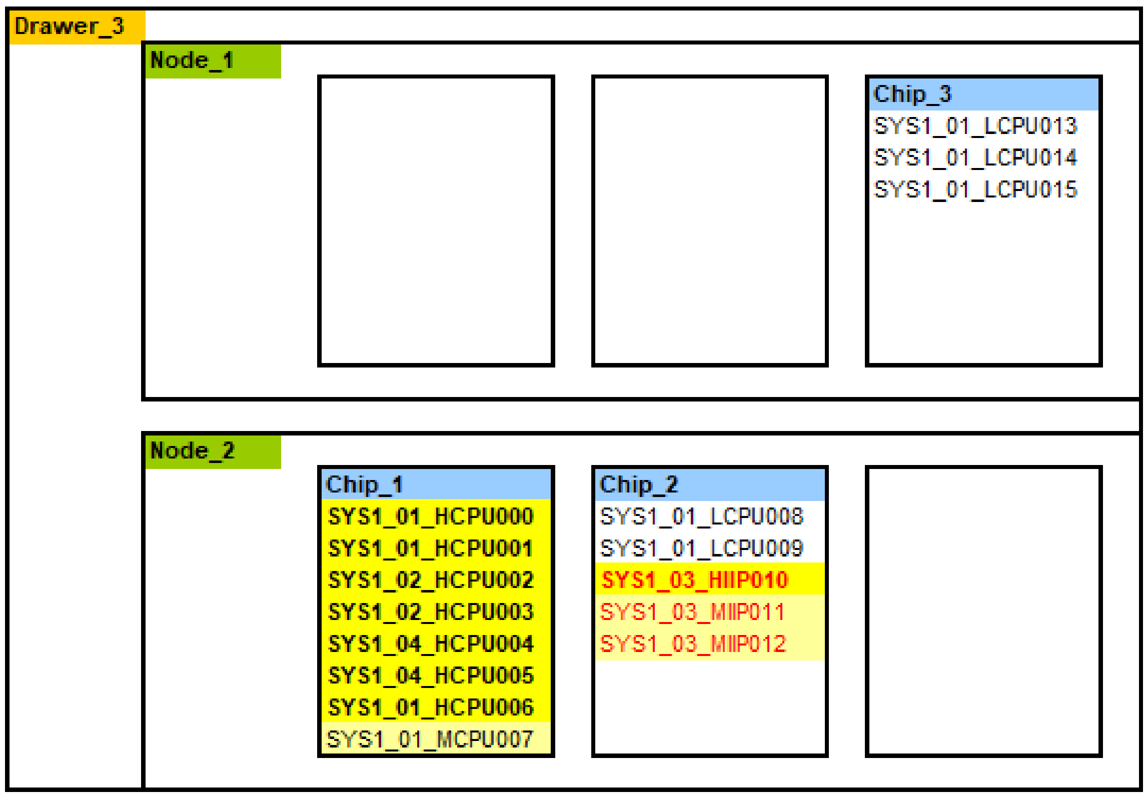
Changed at 14:11:42. Due to adding zIIPs on SYSB

z14 Simple Example – RMF and 99-14 Topology

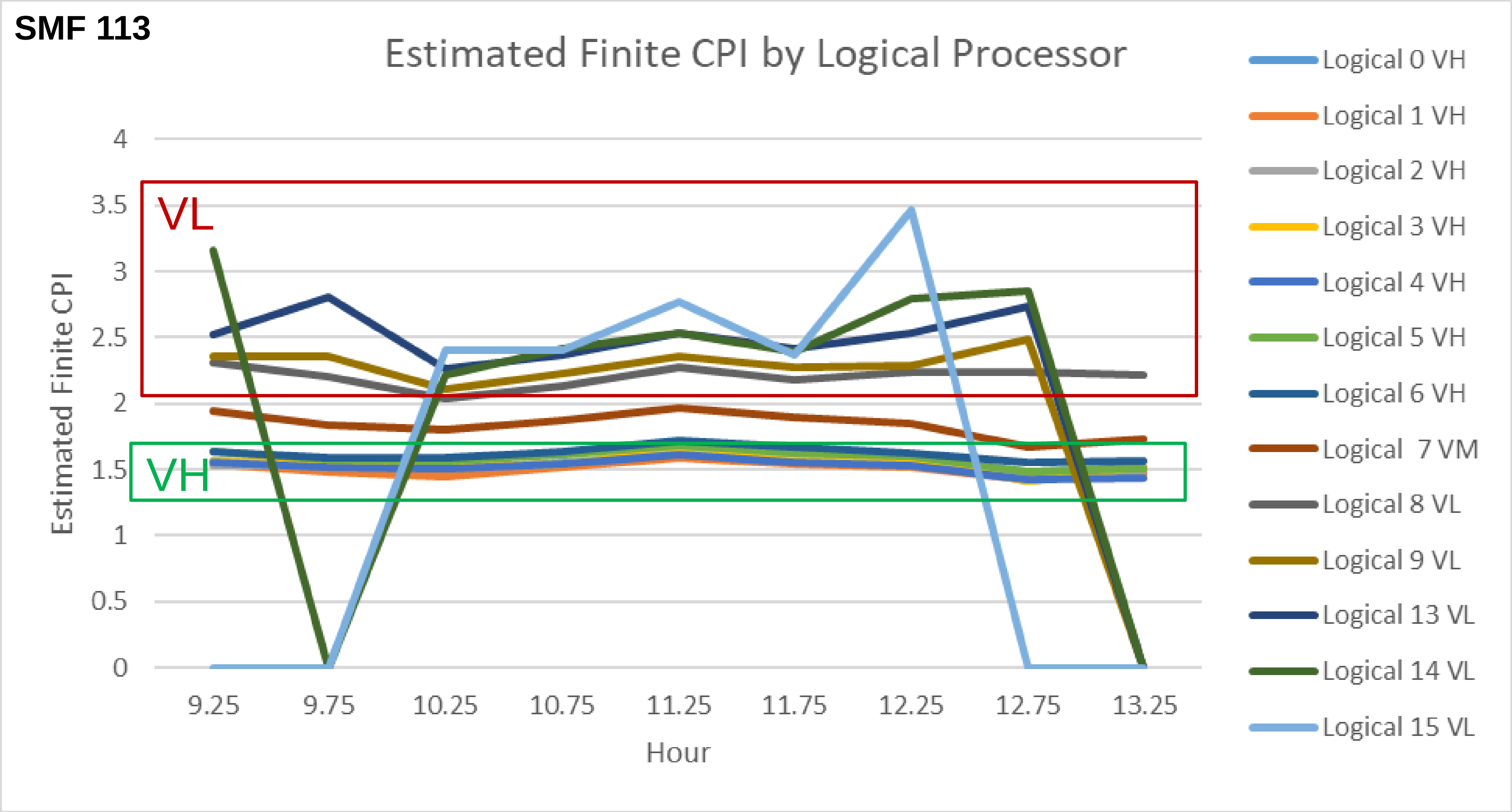
RMF CPU Activity View

---CPU---		---------------- TIME % ----------------				LOG PROC	
NUM	TYPE	ONLINE	LPAR BUSY	MVS BUSY	PARKED	SHARE %	
0	CP	100.00	80.63	80.51	0.00	100.0	HIGH
1	CP	100.00	65.50	65.44	0.00	100.0	HIGH
2	CP	100.00	85.57	85.48	0.00	100.0	HIGH
3	CP	100.00	73.23	73.20	0.00	100.0	HIGH
4	CP	100.00	82.00	81.94	0.00	100.0	HIGH
5	CP	100.00	77.40	77.37	0.00	100.0	HIGH
6	CP	100.00	81.48	81.37	0.00	100.0	HIGH
7	CP	100.00	50.29	51.85	0.00	71.2	MED
8	CP	100.00	28.42	32.53	8.00	0.0	LOW
9	CP	100.00	14.34	22.01	29.20	0.0	LOW
D	CP	100.00	3.24	14.19	72.88	0.0	LOW
E	CP	100.00	0.45	10.68	94.44	0.0	LOW
F	CP	100.00	0.00	-----	100.00	0.0	LOW
TOTAL/AVERAGE			49.43	65.00		771.2	
A	IIP	100.00	9.50	9.14	0.00	100.0	HIGH
B	IIP	100.00	1.01	1.01	0.00	59.2	MED
C	IIP	100.00	0.24	0.24	0.00	59.2	MED
TOTAL/AVERAGE			3.58	3.46		218.4	

SMF 99-14 Topology View



z14 Simple Example – Finite CPI by Logical Processor

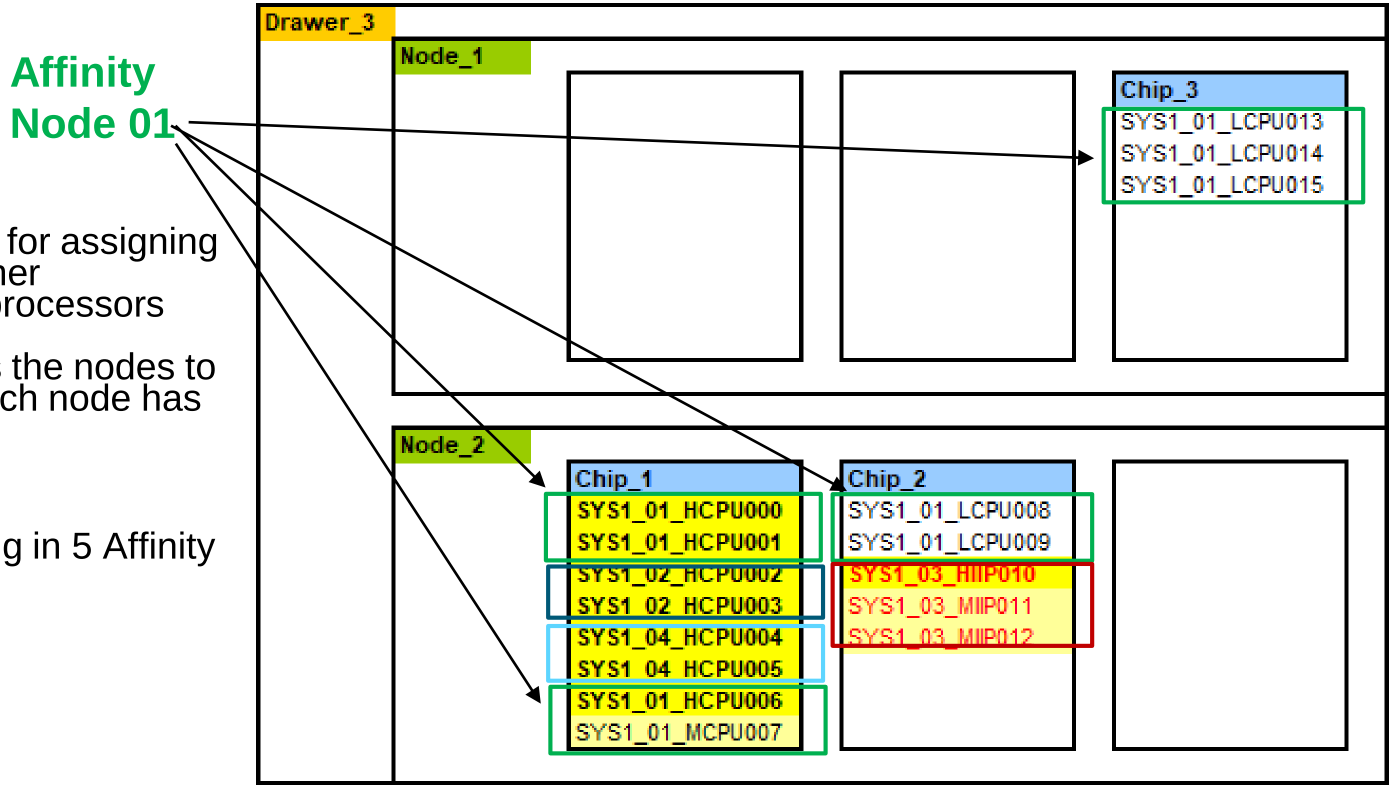


VHs have the lowest Estimated Finite CPI, followed by VM and then VLs

z14 Simple Example – Affinity Nodes

SMF 99-14 Topology View

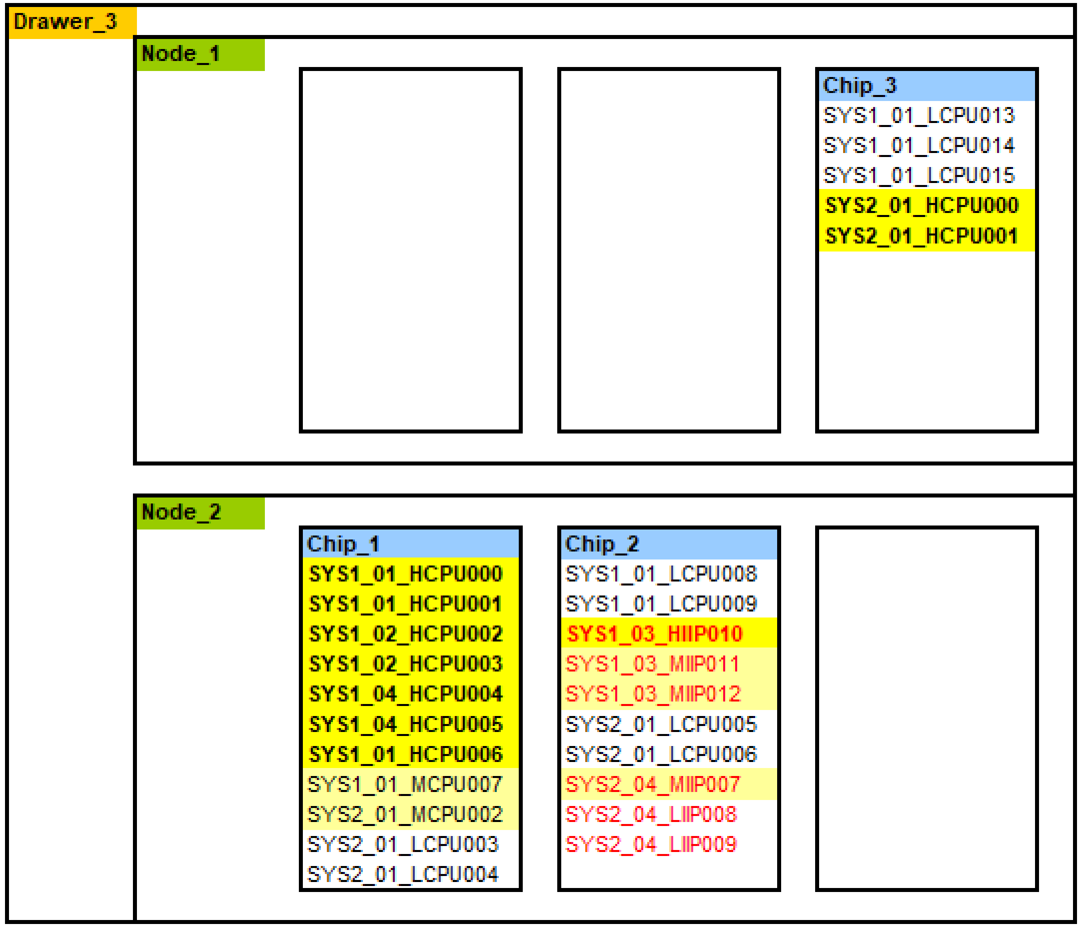
- Affinity Nodes impact where work is selected to run
- 4 Affinity Nodes
  - 01 9 Logical Proc (3 VH, 1 VM, 5 VL) GCPs
  - 02 2 Logical Proc (2 VH) GCPs
  - 04 2 Logical Proc (2 VH) GCPs
  - 03 3 Logical Proc (1 VH and 2 VM) zIIPs
- WLM HiperDispatch balancing algorithms are responsible for assigning work to affinity nodes every 2 seconds. The z/OS dispatcher responsibility is to schedule the work units on the logical processors that make up the affinity node.
  - WLM balances the units of work (TCBs/SRBs) across the nodes to equalize the utilization of nodes and to assure that each node has work of different priorities
- Optimize for VHs
  - Additional GCP Weight could add 1 more VH, resulting in 5 Affinity Nodes (4 GCPs, currently 3)



z14 Simple Example – Multiple Systems

- Multiple LPARs Share and Contend for Resources
- SYS1 and SYS2
  - Potential VH vs VL contention
- So match weight to requirement with minimal VLs
  - Adjust weights at time periods for business requirements

SMF 99-14 Topology View



CPU MF Efficiency

Store Into Instruction Stream

Efficiency – What is SIIS ?

- What is “Store Into Instruction Stream” (SIIS) ?
  - Modern Processors require codependence between their design and the code it expects to execute including the following characteristics:
    - Separating data and instructions, localizing storage references, no self modifying code
    - Cache line today is 256 bytes
  - Most modern compilers have been written with the microprocessor architecture in mind
  - “Old” (usually Assembler) programs with poor program practices continue to run
  - Updating these “SIIS” programs can result in significant CPU reductions

- DFSort APAR PI58848 corrects a SIIS programming error

Efficiency – CPU MF SIIS Indicator can help Identify potential SIIS

- CPU MF can be used to help identify potential SIIS timeframes
  - Based on % of certain I Writes / D Writes sourced
  - LPAR view, identifies when it happens, not who is causing it
    - Identify the program(s) running in the time period, e.g. via zBNA Top Programs
    - Use a hot spot analyzer to find the issue
    - Remediate the source code to correct the issue

Processor	SIIS Indicator %	Description
zEC12 / zBC12	E130 / B4 * 100%	D Writes sourced with L2 intervention / D Writes
z13 / z13s	E163 / B2 * 100%	I Writes sourced with L3 intervention / I Writes
z14 / ZR1	E164 / B2 * 100%	I Writes sourced with L3 intervention / I Writes
z15	E164 / B2 * 100%	I Writes sourced with L3 intervention / I Writes

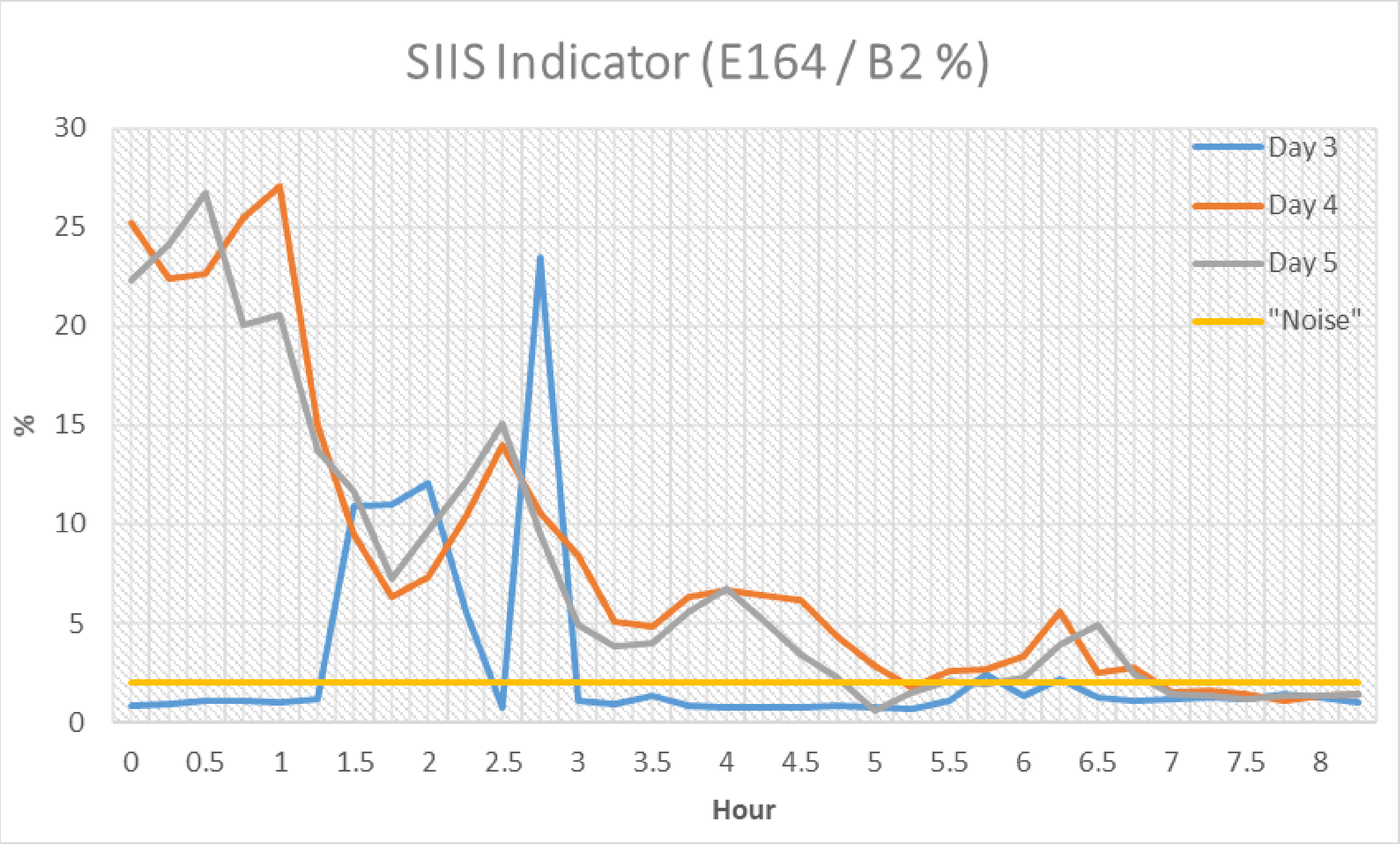
Efficiency – SIIS Indicator and Actions

- Based on the SIIS Indicator %, the following actions are recommended

SIIS Description	SIIS Indicator %	Action
Noise – it will never be 0%	< 2%	None
Minimal SIIS impact	2% < 5%	Low Priority but potential MSU savings
Noteworthy SIIS impact	5% < 10%	Medium Priority – Investigate and Remediate
Considerable SIIS impact	>= 10%	Top Priority – Investigate and Remediate

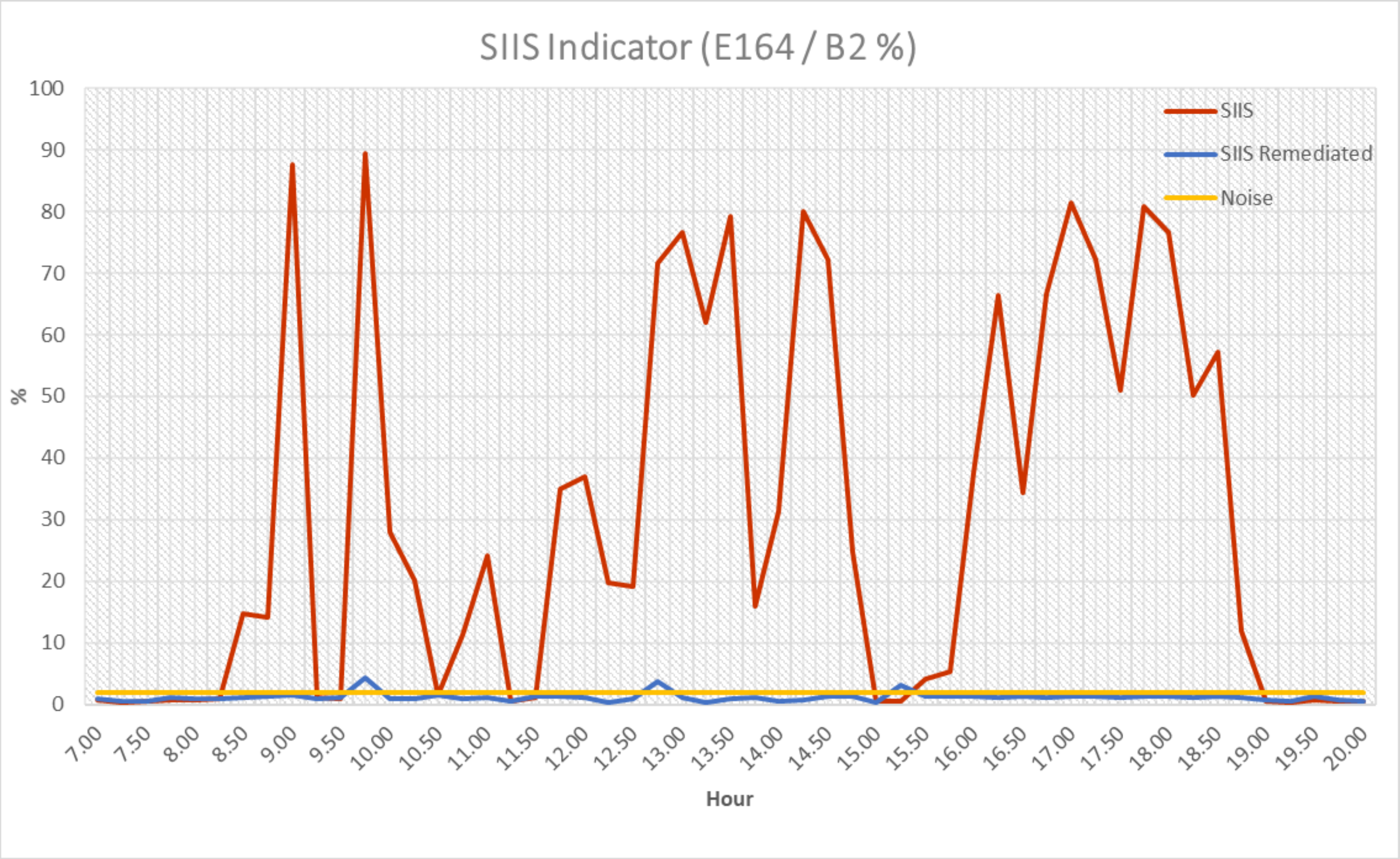
Efficiency – SIIS Customer Experience 1

- Customer z14 experience: SIIS activity detected consistently in Batch Window across 3 days



Efficiency – SIIS Customer Experience 2

- Customer z14 experience: Each Line represents the SIIS indicator for a day with “SIIS” and a day after the code was remediated “SIIS Remediated”
- Overall the customer saved ~3000 CPU seconds (top 15 jobs)



Efficiency – SIIS Summary

- Use CPU Counters "SIIS Indicator" to identify potential timeframes when inefficient "SIIS" programs may be running
- Look for repeating and high impact timeframes
  - Drill down to identify potential Jobs / Programs
  - Use Hot Spot analyzer / Examine / Remediate Source Code
  - Reduce CPU time and elapsed time
- With Tailored Pricing, all MIPS count
- 2006 TechDoc: *IBM System z and eserver zSeries Processor Performance: Processor Design Considerations*
  - *http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10208*
- Dec 2019 TechDoc: *Identifying "Store Into Instruction Stream" (SIIS) Inefficiency by Using CPU MF Counters*
  - *Includes SIIS Assembler remediation examples*
  - http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102806

New

CPU MF Efficiency

z14 New CPU MF COBOL Usage

Vector Packed Decimal Facility of z14

- Enterprise COBOL V6.2 adds support for exploiting the new Vector Packed Decimal Facility in z14 through the ARCH(12) compiler option.
- The Vector Packed Decimal Facility allows the dominant COBOL data types, packed and zoned decimal, to be handled in wide 16-byte vector registers instead of in memory.
  - Decimal and floating-point computationally intensive COBOL programs, which are optimized with Enterprise COBOL V6.2 and that target z14 ARCH(12), can deliver CPU time reduction on the z14 server over the same applications built with COBOL V6.1.
  - **No source changes are required** to take advantage of this new facility; just recompile with ARCH(12) to target z14.

Changed ARCH compiler option

- ARCH(7)   (still the default in 6.2)
  - 2094-xxx models (IBM System z9 EC)   2096-xxx models (IBM System z9® BC)
- ARCH(8)   (the default in 6.3)
  - 2097-xxx models (IBM System z10 EC)   2098-xxx models (IBM System z10 BC)
- ARCH(9)
  - 2817-xxx models (IBM zEnterprise z196 EC)   2818-xxx models (IBM zEnterprise z114 BC)
- ARCH(10)
  - 2827-xxx models (IBM zEnterprise EC12)   2828-xxx models (IBM zEnterprise BC12)
- ARCH(11)
  - 2964-xxx models (IBM z13)   2965-xxx models (IBM z13s)
- **ARCH(12)**
  - **3906-xxx models (IBM z14)**
- **ARCH(13)**
  - **8561-xxx models (IBM z15)**

z14 / z15 New CPU MF Counters to indicate COBOL “Modernization”

- 3 New z14 Extended Counters – See SA23-2261-04
  - E224 - Count of floating point execution slots used for finished Binary Coded Decimal to Decimal Floating Point conversions
  - E225 – Count of floating point execution slots used for finished vector arithmetic Binary Coded Decimal instructions
  - E226 – Decimal instructions dispatched
- Above Counters are not directly comparable to B01 (Instructions) or among each other. They could be used as an indicator of COBOL compiler “modernization”
  - E226 – Decimal “instructions”
  - E224 – Decimal Floating Point Converted – COBOL ARCH(10 |11)
  - E225 - New z14 Vector Packed Decimal Facility and z/OS 2.3 – COBOL V6.2 ARCH(12)
- One could identify when most Counter activity is occurring, then identify Jobs / Programs (e.g. zBNA) to investigate / re-compile for most impact
- See Performance examples in Back Up

Techniques for Making COBOL Applications Efficient

- New Best Practice document for improving efficiency of COBOL apps
  - http://w3.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102731

- Recommendations
  - Important to stay current with the COBOL compiler versions and exploit the use of the highest arch level feasible
  - Target the most performance critical sections of your application code for migration first
  - Performance sensitive code should be compiled with the latest compiler technology and with aggressive optimization
    - New Enterprise COBOL v6.2 compiler fully exploits the Vector Packed Decimal Facility
    - Improves decimal and floating point intensive applications by up to 38% over those compiled with COBOL v6.1[1]
  - Evaluate use of Automatic Binary Optimizer

[1] **Disclaimer: all performance results reported in this article are based on internal IBM compute-intensive test suites. Performance results from other applications may vary.**

Customer Example: COBOL Compile Level and SMF 30 CPU / Instruction Counts and COBOL "Modernization" Indicators

- Example z14 Job / COBOL Program compiled at different levels

	SMF 30 Subtype 4 COBOL Job/Step Statistics							SMF 113 - Mid Interval for entire System while COBOL Job running		
COBOL Compile Level	Total CPU Seconds	% Decrease Total CPU Seconds	Total Cycles	Total Instructions	% Decrease Total Instructions	Cycles per Instruction (CPI)	% Decrease in CPI	E224 - Count of Floating Point Execution Slots for Finished Binary Coded Decimal to Decimal FP conversions	E225 -Count of Floating Point Execution Slots for Finished Vector arithmetic Binary Coded Decimal Instructions	E226 - Decimal Instructions Dispatched (Count)
z14 5.1 ARCH(8)	2,632		7,252,382,778,339	3,880,516,200,794		1.868		0	0	28,617,900,000
z14 6.2 ARCH(11)	2,222	15.6%	6,122,126,138,058	3,305,548,383,929	14.8%	1.852	0.9%	1,937,530,000	0	32,197,700,000
z14 6.2 ARCH(12)	2,116	19.6%	5,830,006,070,220	3,364,473,345,571	13.3%	1.732	7.3%	0	24,836,700,000	29,231,400,000

- Overall lower CPU on z14 with ARCH(12)
  - More efficient instructions, as indicated by lower CPI
  - E225 shows z14 Vector Pack Decimal Facility used without code change

Most Advanced and Fit-for-Purpose Compilers

Compilers enable modernization and increases performance of critical business applications	Java enables delivery of rich, scalable and robust applications with speed and agility

Using COBOL 6.3 on average ***58% reduction in CPU usage*** over applications compiled with COBOL v4.2 on z15

Automatic Binary Optimizer v3.2 ***reduces CPU usage by up to 57%*** for compute intense apps built originally on COBOL 4.2

Up to 22% reduction in CPU usage on z15 over the same set of key numerically intensive double-precision floating-point applications built with z/OS v2.3 XL C/C++ on z14

Up to ***20% throughput improvements*** in general Java workloads

Takes advantage of new Integrated Accelerator for zEDC for ***up to 15x*** improvement over software and ***up to 2x*** faster elapsed times over zEDC Express

Pause-less garbage collection***: reducing pause times by up to 3x*** better throughput for constrained Service Level Agreements

Crypto and Encryption Measurement

Crypto CPU MF Enablement and Measurement

- CPU MF Crypto Counters <u>should</u> be enabled via the following command
  This example collects **B**asic, **E**xtended and **C**rypto counters
  – Modify HIS: “F HIS,B,TT=‘Text',CTRONLY,CTR=(B,E,**C**),SI=SYNC,CNTFILE=NO”

- CPU MF metrics can be calculated for SHA (AES or ECC) specific activity
  – Elliptic-curve Cryptography (ECC) new with z15

Metric	**Calculation** – *note all fields are <u>deltas</u>. SMF113-1s are deltas. SMF 113-2s are cumulative.*
CPI	C69 / C68 or C77 / C76 or C81 /C80
LPARCPU	(((1/CPSP/1,000,000) * C69) / Interval *in Seconds*) * 100

SA23-2260-06

Counter number	Counter
64	PRNG function count
65	PRNG cycle count
66	PRNG blocked function count
67	PRNG blocked cycle count
68	SHA function count
69	SHA cycle count
70	SHA blocked function count
71	SHA blocked cycle count
72	DEA function count
73	DEA cycle count
74	DEA blocked function count
75	DEA blocked cycle count
76	AES function count
77	AES cycle count
78	AES blocked function count
79	AES blocked cycle count
80	ECC function count
81	ECC cycle count
82	ECC blocked function count
83	ECC blocked cycle count
84-127	Reserved

Figure 2-8. Counters in the crypto-activity counter set for CSVN = 6

SHA Activity

Data Set and CF Encryption activity

Elliptic-curve Cryptography (ECC)

New with z15

Analyzing Encryption Performance – WSC Encryption Test

- **Tying it All Together – Example 2: WSC Encryption Test Jobs (w/o and with)**
  - IFASMFDP Jobs/Steps write same ~200 MB file to different output files
    - 1) Non Extended Format, 2) Extended Format and 3) Extended Format and Encrypted
    - 4) Extended Format**, zEDC Compressed** and then Encrypted

  - Analyzed the following SMF records
    - SMF 30

      Job / Step CPU, Instruction Counts, CPI , zEDC
    - SMF 42-6

      Data Set Characteristics – Encrypted Bytes and Encryption Indicator
    - SMF 82 31

      CPU MF Calls by Job, DFSMS Data Set Encryption Function (CSFKRR2)
    - SMF 113

      CPU MF Crypto Counters – AES CPU

Analyzing Encryption Performance – WSC Encryption Test

- Then ran a 4th combination with Extended Format, **zEDC Compressed** and then Encrypted
  - Resulted in ~9x smaller file and less AES CPU for Encryption

	SMF 42-6							SMF 82 Subtype 31			SMF 113			
	Data Set Name Written To	DSTYPE	Extended Format?	Encrypted ?	Encryption Type	Seq Write Blks	Write Bytes Eligible for or Encrypted	ENG CPACF	CSFKRR2	CSFSTAT	C76 -AES Function Count "Instructions"	C77 - AES Cycles	AES CPI	AES LPARCPU
1	JPBURG.WSCSYSD.SMF.SYSD.T03.PHYSEQ	PHY_SEQ	N	N	0	8,819	207,034,844							
2	JPBURG.WSCSYSD.SMF.SYSD.T03.EF	SEQX	Y	N	0	8,820	207,024,902							
3	JPBURG.WSCSYSD.SMF.SYSD.T03.EFENCR	SEQX	Y	**Y**	**AES-256**	8,820	**207,024,902**	**35,270**	**1**	**1**				
									5 Minute Interval Total		100,031	204,005,232	2039.42	**0.013**

zEDC Compressed and then Encrypted

	SMF 30 Subtype 4		
	zEDC_Comp_Req	zEDC_UnComp_In_Bytes	zEDC_Comp_Out_Bytes
4	802	207,040,137	22,837,452

	SMF 42-6							SMF 82 Subtype 31			SMF 113			
	Data Set Name Written To	DSTYPE	Extended Format?	Encrypted ?	Encryption Type	Seq Write Blks	Write Bytes Eligible for or Encrypted	ENG CPACF	CSFKRR2	CSFSTAT	C76 -AES Function Count "Instructions"	C77 - AES Cycles	AES CPI	AES LPARCPU
4	JPBURG.WSCSYSD.SMF.SYSD.T03.ZEDCENCR	SEQX	Y	Y	**AES-256**	404	**22,879,328**	**1,618**	**1**	**1**				
									5 Minute Interval Total		27,873	93,571,202	3357.06	**0.006**

Analyzing Encryption Performance – WSC Encryption Test

- Then ran a 4th combination with Extended Format, **zEDC Compressed** and then Encrypted
  - **Resulted in ~9x smaller file and less CPU for Encryption and all runs!**

	SYS	Date	Time	Job	Step	Program	CPU_SU	Tot_Ins	CPU_uSec_from_SUs	Tot_Cycles	CPI
1	SYSD	20181003	15:16:31	JPBURE1L	PHYSEQ1	IFASMFDP	11,845	358,623,301	157,687	821,231,618	2.29
2	SYSD	20181003	15:16:35	JPBURE1L	EF2	IFASMFDP	12,950	403,044,535	172,397	897,842,925	2.23
3	SYSD	20181003	15:16:53	JPBURE1L	ENCRYP6	IFASMFDP	14,418	409,395,430	191,940	999,621,567	2.44

zEDC Compressed and then Encrypted

SMF 30 Subtype 4

	zEDC_Comp_Req	zEDC_UnComp_In_Bytes	zEDC_Comp_Out_Bytes
4	802	207,040,137	22,837,452

SMF 30 Subtype 4

	SYS	Date	Time	Job	Step	Program	CPU_SU	Tot_Ins	CPU_uSec_from_SUs	Tot_Cycles	CPI
4	SYSD	20181003	15:20:55	JPBURE2L	ZEDENCR7	IFASMFDP	11,703	333,812,287	155,796	811,386,545	2.43

z15 Integrated Adapter for zEDC

- New z15 Integrated Adapter for zEDC distinguishes between Asynchronous and Synchronous exploiters:
  - Asynchronus zEDC exploiters continue to utilize “traditional” and new measurements: SMF 30, 74(10) EADM, 78(3) IOQ shows SAP % CMPR. They include:
    - BSAM/QSAM and DFHSM
  - Synchronous exploiters now are measured in z15 CPU MF Extended Counters. They include:
    - Java, MQ, Connect:Direct, Content Manager OnDemand

New

- See SA23-2261-6

z15 Integrated Adapter for zEDC – WSC Test

- A simple Java (zEDC Synchronous) test driving z15 Encryption

CPU MF NXU Synchronous

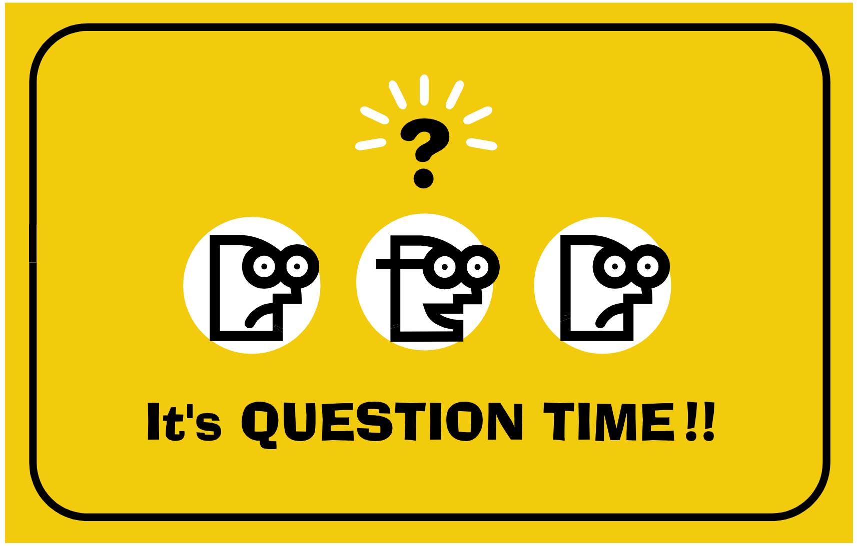
Hour	CPI	Prb State	Instr Cmplx CPI	Finite CPI	SCPL1M	L1MP	L2P	L3P	L4LP	L4RP	MEMP	Rel Nest Intensity	LPARCPU	Eff GHz	Machine Type	LSPR Wkld	NXU (zEDC) Synchronous Calls	NXU LPARCPU	NXU % LPARCPU	NXU "CPI"
11.58	1.44	53.4	1.07	0.37	17	2.1	83.4	15.6	0.3	0.0	0.7	0.35	1.6	5.2	Z15	LOW	61,055	0.06	3.92	16,275.7

SYSID	SH	DAY	HOUR	CPID	CPI RATIO	PRBSTATE %	RNI	LPARCPU %	ZE265	NXUCPU LPARCPU	NXUP %	NXUCPI CPI
SYSD	P	6	11.58	0	1.24	68.93	0.31	0.4	2415	0.00	0.60	16100.61
SYSD	P	6	11.58	2	1.82	7.84	0.68	0.0	6	0.00	0.04	17335.00
SYSD	P	6	11.58	4	1.77	0.51	0.30	0.1	0	0.00	0.00	---
SYSD	P	6	11.58	6	1.45	0.00	0.19	0.0	0	0.00	0.00	---
SYSD	P	6	11.58	8	1.76	8.19	0.48	0.1	0	0.00	0.00	---
SYSD	P	6	11.58	10	1.47	0.51	0.20	0.0	0	0.00	0.00	---
SYSD	P	6	11.58	12	1.95	6.60	0.64	0.1	0	0.00	0.00	---
SYSD	P	6	11.58	14	1.50	0.62	0.35	0.0	0	0.00	0.00	---
SYSD	P	6	11.58	16	1.42	68.70	0.36	0.4	31273	0.03	7.88	16277.24
SYSD	P	6	11.58	18	1.42	73.89	0.36	0.3	27361	0.03	9.20	16289.05

CPU MF Summary

- CPU MF Counters provide better information for more successful capacity planning and identifying efficiency opportunity
  - Match LPAR weights to capacity requirements with minimal VLs
  - Identify SIIS Opportunity
  - Identify COBOL Modernization exploitation
  - Identify z15 zEDC Synchronous exploitation
- Enable Crypto Counters to measure Pervasive Encryption
- Enable CPU MF Counters Today!
  - Continuously collect SMF 113s for all your systems

Capturing CPU MF data is an Industry "Best Practice"



Thank You for Attending!

Notices and disclaimers

-
- **U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**
- Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.
- IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."
- **Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**
- Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those
- customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.
- References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.
- Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.
- It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers continued

- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of 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. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**
- The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
- IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml

Complete your session evaluations for a chance at daily prizes!

To complete, visit www.share.org/evaluation and see your progress on the leaderboard!

PERSON

PROGRAM

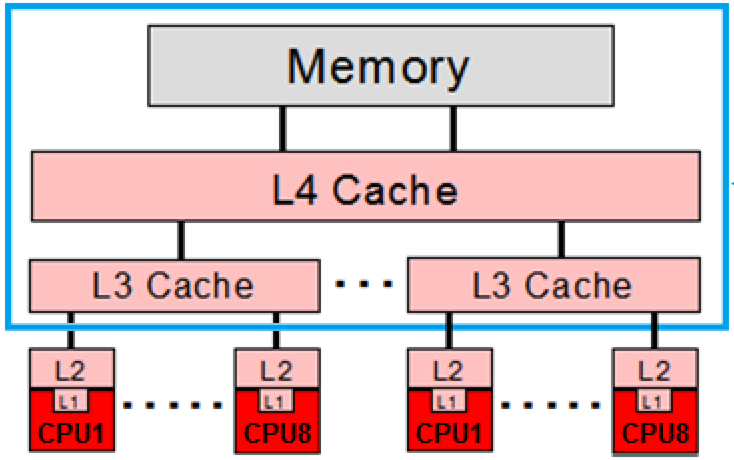
PROJECT

Back Up

Today’s z Systems Capacity Planning

Processor Design
- **CPU**
- **Memory Hierarchy (Nest)**

Hipervisor (PR/SM)
- **Amount of virtualization**



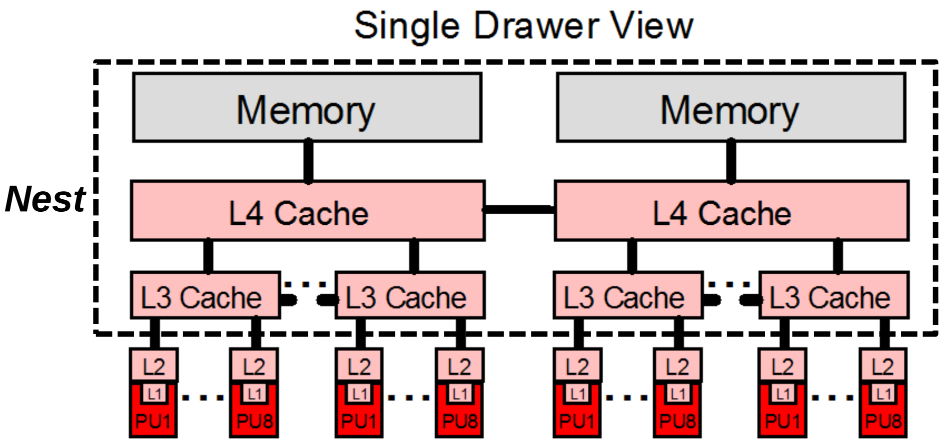
The Nest
(all levels of storage beyond the chip)

Operating System
- **Virtualization at address space level**

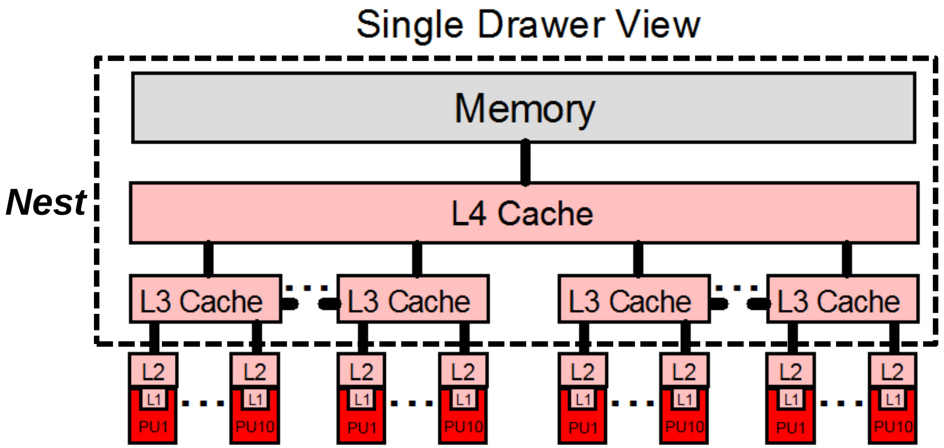
Workload Characteristics
- **Instructions**
- **Dispatch Profile**
- **I/O Rate**

z14 vs z13 Hardware and Topology Comparison

- z13
  - CPU
    - 5.0 GHz
    - Major pipeline enhancements
    - 1 picocoded translation engine
  - Caches
    - L1 private 96k i, 128k d
    - L2 private 2 MB i, 2 MB d
    - L3 shared 64 MB / chip
    - L4 shared 480 MB / node
      Plus 224 MB NIC



- z14 – *L3 clustering and cache sizes aside, topology strongly resembles zEC12*
  - CPU
    - 5.2 GHz
    - Logical directory w/ inclusive TLB
    - 4 HW-implemented translation engines
  - Caches
    - L1 private 128k i, 128k d
    - L2 private 2 MB i, 4 MB d
    - L3 shared 128 MB / chip
    - L4 shared 672 MB / ~~node~~ drawer



New

Fundamental Components of Workload Capacity Performance Part 1

- **Instruction Path Length for a transaction or job**
  - Application dependent, of course
  - Can also be sensitive to Nway (due to MP effects such as locking, work queue searches, etc)
  - But generally doesn't change much on moves between processors of similar capacity and/or Nway
- **Instruction Complexity (Micro processor design)**
  - Many design alternative
    - Cycle time (GHz), instruction architecture, pipeline, superscalar, Out-Of-Order, branch prediction, multi-threading and more
  - Workload effect
    - May be different with each processor design
    - But once established for a workload on a processor, does not change very much

Fundamental Components of Workload Capacity Performance Part 2

- **Memory Hierarchy or "nest"**

  – Many design alternatives
    - cache (levels, size, private, shared latency MESI protocol), controller, data buses

  – Workload effect
    - Quite variable
    - Sensitive to many factors: locality of reference, dispatch rate, IO rate, competition with other applications and/or LPARs, and more

  – Relative Nest Intensity
    - Activity beyond the private cache(s), is most sensitive area

      Due to larger latencies involved
    - Reflects activity distribution and latency to chip-level caches, book/node/drawer-level caches and memory
    - Level 1 cache miss percentage (L1MP) also important
    - Data for calculation available from CPU MF

Understanding CPU MF Metrics - 1

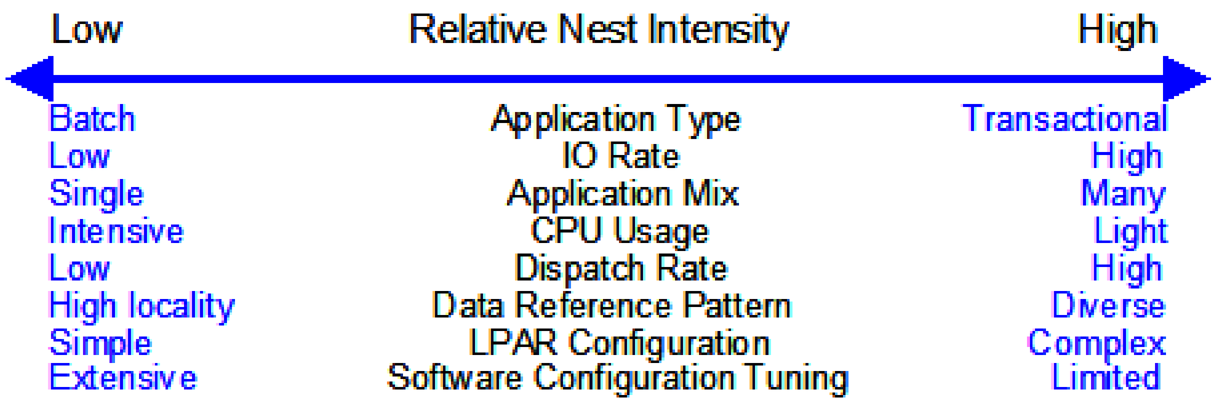
- CPI – Cycles per Instruction
  - EICPI – Estimated Instruction Complexity CPI – Indicates portion of CPI related to the microprocessor
  - EFCPI - Estimated Finite CPI – Indicates portion of CPI related to the L2 private and shared caches (Nest)
- PRB – The % of Problem State instructions. This is an indicator of the workload mix, so a changing of PRB%, may indicate different workload mixes running.
- ESCPL1M – Estimated sourcing cycles per L1 Miss
- L1MP – Level 1 Miss Percentage – The average Level 1 miss percentage per 100 instructions. It is an indicator of **"How Often"** the instructions and data are not found in the L1 cache, and must be sourced further out in the cache hierarchy. It is a component in matching to the LSPR workload. If L1MP is ~>6%, it may be an indicator of CICS Threadsafe opportunity.
- L2P - Level 2 Cache Miss Percentage – The percent of misses sourced from the private Level 2 cache
- L3P – Level 3 Cache Miss Percentage - The percent of misses sourced from the shared Level 3 cache
- L4LP – Level 4 Local Cache Miss Percentage - The percent of misses sourced from the shared Level 4 Local cache
- L4RP – Level 4 Remote Cache Miss Percentage - The percent of misses sourced from the shared Level 4 Remote cache

Understanding CPU MF Metrics - 2

- MEMP – The Memory Cache Miss Percentage - The percent of misses sourced from the shared memory
- RNI – The Relative Nest Intensity – **“How Far”** out in the Nest are Instructions and Data sourced. It is a component in matching to the LSPR workload.
  - z13 RNI: 2.3*(0.4*L3P + 1.6*L4LP + 3.5*L4RP + 7.5*MEMP) / 100
  - z14 RNI: 2.4*(0.4*L3P + 1.5*L4LP + 3.2*L4RP + 7.0*MEMP) / 100
  - z15 RNI: 2.9*(0.45*L3P + 1.5*L4LP + 3.2*L4RP + 6.5*MEMP) / 100
- LPARCPU – This is a measurement of **“How Much”** load is running. 100% equals 1 Engine
- LSPR WKLD – The LSPR Workload this system matches to based on its L1MP and Relative Nest Intensity (RNI).
- **TLB Metrics**
- ETLBCPUP –The estimated CPU % related to TLB misses. Some portion of this amount may be able to be reduced with Large Pages.
- PTEP - The Page Table Entry % of TLB misses. If PTEP is >40%, it may be an indicator of applicability of Large Pages to reduce CPU.
- ETLBCYPM – The estimated TLB sourcing cycles per TLB Miss

The Most Influential Factor Underlying Workload Capacity Curves is Relative Nest Intensity (RNI)

- Many factors influence a workloads capacity curve
- However, what they are actually affecting is the workload's RNI
- It is the net effect of the interaction of all these factors that determines the capacity curve
- The chart below indicates the trend of the effect of each factor but is not absolute. For example:
  - Some batch will have high RNI while some transactional workload will have low
  - Some low IO rate workloads will have high RNI, while some high I/O rates will have low



LSPR Workload Categories

LSPR workload categories are based on various combinations of measured workload primitives. Primitives include **CICS**, **DB2**, **IMS**, **OSAM**, **VSAM**, **WebSphere**, **COBOL**, **utilities**.

Workload Categories include:

- **Low** (rarely needs to rely on the nest for storage references)
  - Workload representing light use of the memory hierarchy
  - Similar to high-scaling CPU intensive workload primitives
- **Average** (average dependency on the nest for storage references)
  - Workload expected to represent the majority of customer workloads
  - Similar to the former LoIO-mix curve
- **High** (frequently needs to rely on the nest for storage references)
  - Workload representing heavy use of the memory hierarchy
  - Similar to the former DI-mix curve

zPCR extends published workload categories

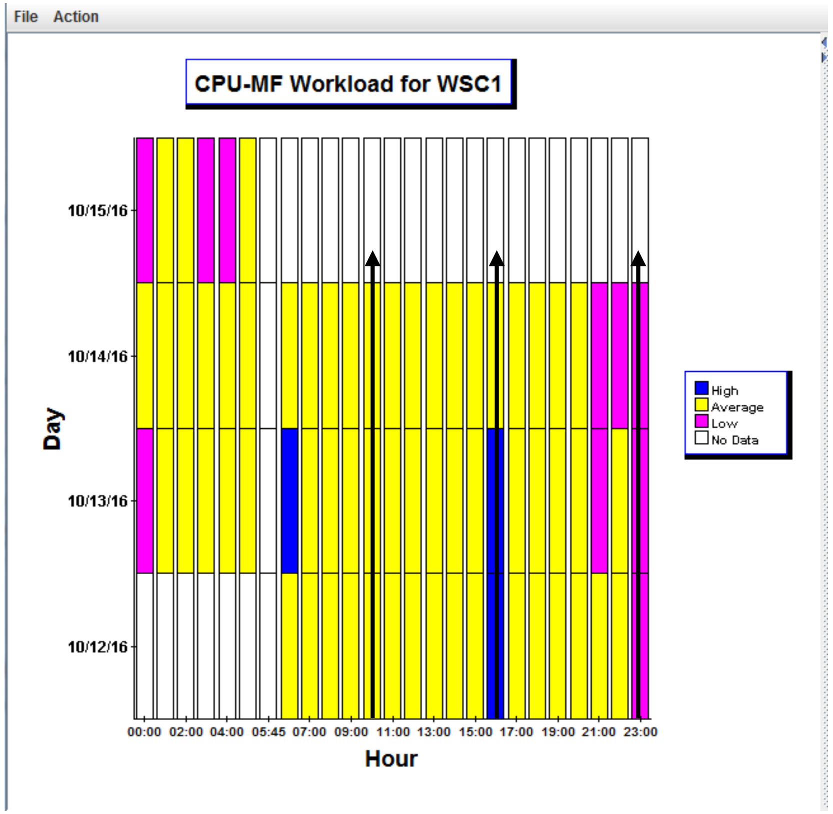
- **Low-Avg** (50% **Low** and 50% **Average**)
- **Avg-High** (50% **Average** and 50% **High**)

z Systems Capacity Planning

Processor	Features	Flag	MSU	LSPR Workload Category				
				Low	Low-Avg	Average	Avg-High	High
z13/700								
2964-701	1W	=	210	1,779	1,736	1,695	1,614	1,540
2964-702	2W	=	394	3,452	3,319	3,196	3,003	2,833
2964-703	3W	=	571	5,085	4,854	4,644	4,340	4,073
2964-704	4W	=	740	6,678	6,344	6,041	5,625	5,262
2964-705	5W	=	905	8,238	7,792	7,392	6,866	6,410
2964-706	6W	=	1,062	9,765	9,202	8,700	8,066	7,518
2964-707	7W	=	1,212	11,260	10,573	9,964	9,224	8,587
2964-708	8W	=	1,356	12,724	11,906	11,188	10,344	9,618
2964-709	9W	=	1,496	14,157	13,204	12,371	11,425	10,613
2964-710	10W	=	1,632	15,560	14,466	13,515	12,469	11,574

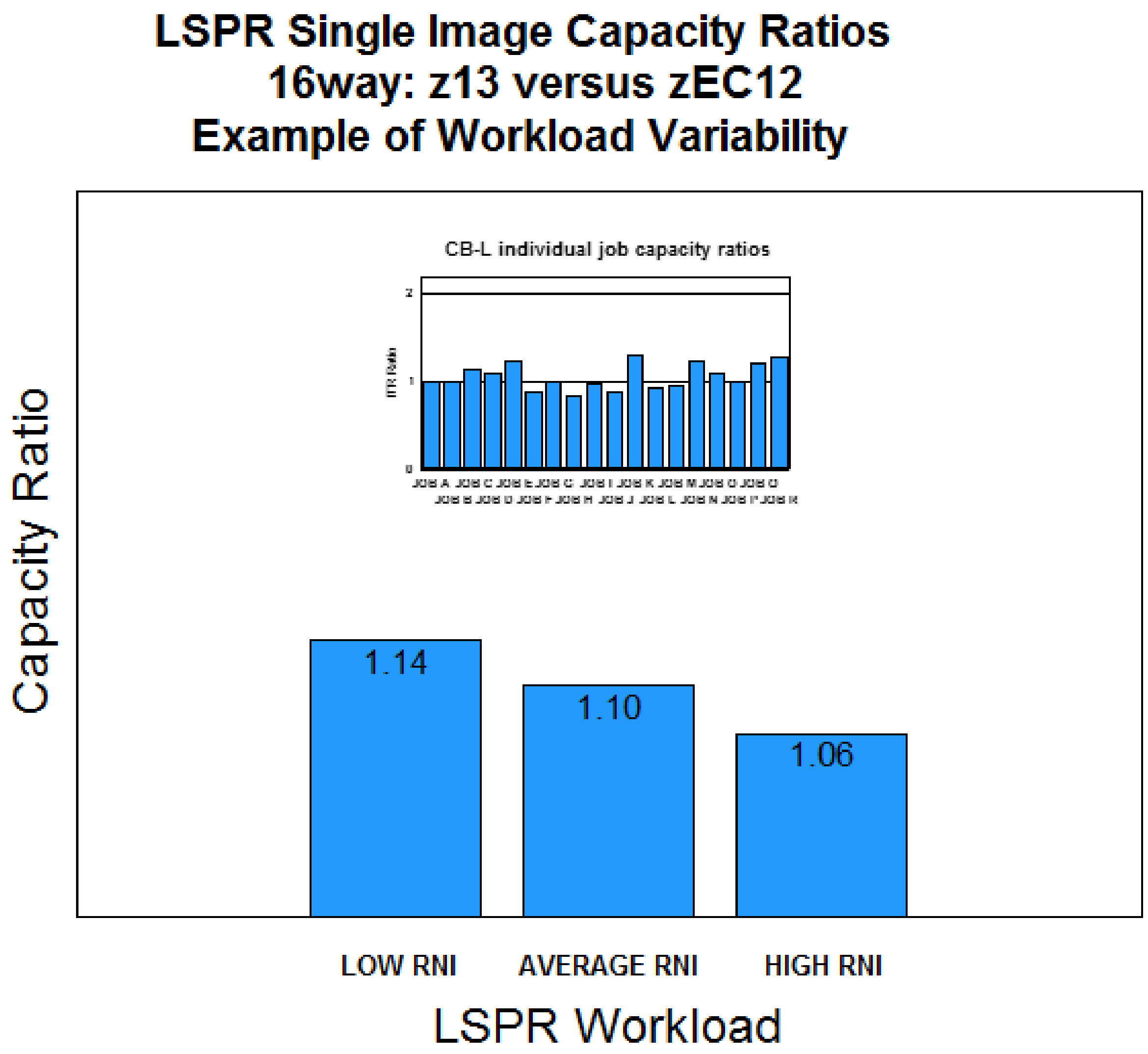
- **Relative Processor Capacity** varies by LPAR configuration and **LSPR Workload**
- **CPU MF data** used to select **LSPR Workload Match**
- **IBM Capacity Planning Tools** utilize **CPU MF** data to select a **workload**
  – zPCR, CP3000 and zBNA are all enabled for CPU MF

Customer LSPR Workload Match changes over time



- **zCP3000 graph (SYS41007) shows the LSPR Workload Match over time**
- **Use the Workload Match from the capacity planning interval that you are sizing**
  – If the LSPR workload is not consistent, then use different combinations in zPCR (e.g. AVG-HIGH at hour 16)

LSPR Single Image Capacity Ratios -
16 Way z13 versus zEC12



Additional Customer Value with CPU MF Counters data

- Counters can be used as a secondary source to:
  - Supplement current performance data from SMF, RMF, DB2, CICS, etc.
  - Help understand why performance may have changed
  - Supported by many software products including Tivoli TDSz
- Some examples of usage include:
  - Impact zEDC compression
  - HiperDispatch Impact
  - Configuration changes (Additional LPARs)
  - 1 MB / 2 GB Page implementation
  - Application Changes (e.g. CICS Threadsafe vs QR)
  - Estimating Utilization Effect for capacity planning
  - GHz change in Power Saving Mode
  - Crypto CPACF usage – Including RACF AESKDF and DFSMS Data Set Encryption
  - Identifying z14 Vector Packed Decimal opportunity

CPU MF Counters Enablement Resources

- CPU MF Webinar Replays and Presentations
  – http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS4922

- z/OS CPU MF - "Detailed Instructions" Step by Step Guide
  – http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TC000066

- z/VM Using CPU Measurement Facility Host Counters
  – http://www.vm.ibm.com/perf/tips/cpumf.html

z/OS Steps to Enable CPU MF Counters

- 1 - Configure the processor to collect CPU MF
  ___ Update the LPAR Security Tabs, can be done dynamically

- 2 - Set up HIS and z/OS to collect CPU MF
  ___ Set up HIS Proc
  ___ Set up OMVS Directory - required
  ___ Collect SMF 113s via SMFPRMxx

- 3 - Collect CPU MF COUNTERs
  ___ Start HIS
  ___ Modify HIS: "F HIS,B,TT='Text',PATH='/his/',CTRONLY,CTR=(B,E),SI=SYNC"

  – **Recommend to start HIS, Modify for Counters, and continuously run**

z/OS Steps to Enable CPU MF Counters with z/OS 2.2 (or z/OS 2.1 with APAR OA43366)

HIS Counters without USS File System

- 1 - Configure the processor to collect CPU MF
  ___ Update the LPAR Security Tabs, can be done dynamically

- 2 - Set up HIS and z/OS to collect CPU MF
  ___ Set up HIS Proc
  ___ ~~Set up OMVS Directory - required~~
  ___ Collect SMF 113s via SMFPRMxx

- 3 - Collect CPU MF COUNTERs
  ___ Start HIS
  ___ Modify HIS: “F HIS,B,TT=‘Text',CTRONLY,CTR=(B,E),SI=SYNC,**CNTFILE=NO**”

  – **Recommend to start HIS, Modify for Counters, and continuously run**

SMF 113s Space Requirements Are Minimal

- The SMF 113 record puts minimal pressure on SMF
  - 452 bytes for each logical processor per interval
- Example below is from 3 z196s processors
  - 713, 716 and 718
  - 10 Systems
  - 5 Days, 24 hours
- SMF 113s were 1.2% of the space compared to SMF 70s & 72s

RECORD TYPE	RECORDS READ	PERCENT OF TOTAL	**AVG. RECORD LENGTH**	MIN. RECORD LENGTH	MAX. RECORD LENGTH	RECORDS WRITTEN	Total Size (with AVG. Record Size)	% Total Size (with AVG. Record Size)
70	14,250	1.8%	14,236	640	32,736	14,250	202,865,850	15.1%
72	744,014	93.5%	1,516	1,104	20,316	744,014	1,128,252,590	83.7%
113	37,098	4.7%	**452**	**452**	**452**	37,098	16,768,296	**1.2%**
TOTAL	795,362	100.0%	1,695	18	32,736	795,362	1,347,886,736	100.0%

zBNA – Top Programs

- Guidance
  - The Top Programs should be used as part of any CPU (MIPS) reduction study as it represents the most CPU. So one can use it to drive the conversation as to what level/version of program, was it compiled with the latest compiled version. Also is it a candidate for ABO, or a “hot spot” analyzer to further improve efficiency

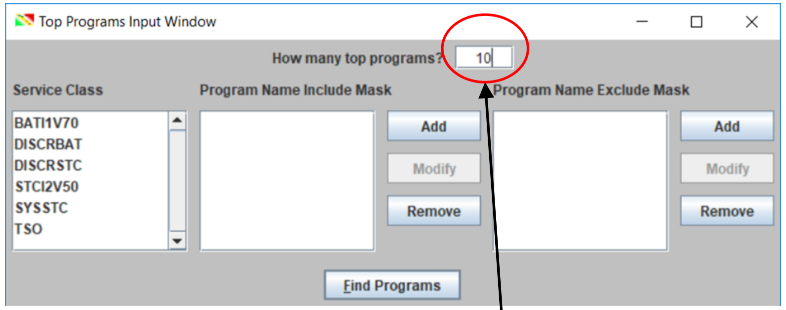
Top 15 Programs by Total CPU Time Report

The following table gives information on the top 10 programs sorted by total CPU time descending after filters are applied. This data spans from Sep 25, 2018 9:00:00 PM until Sep 26, 2018 4:00:00 AM.

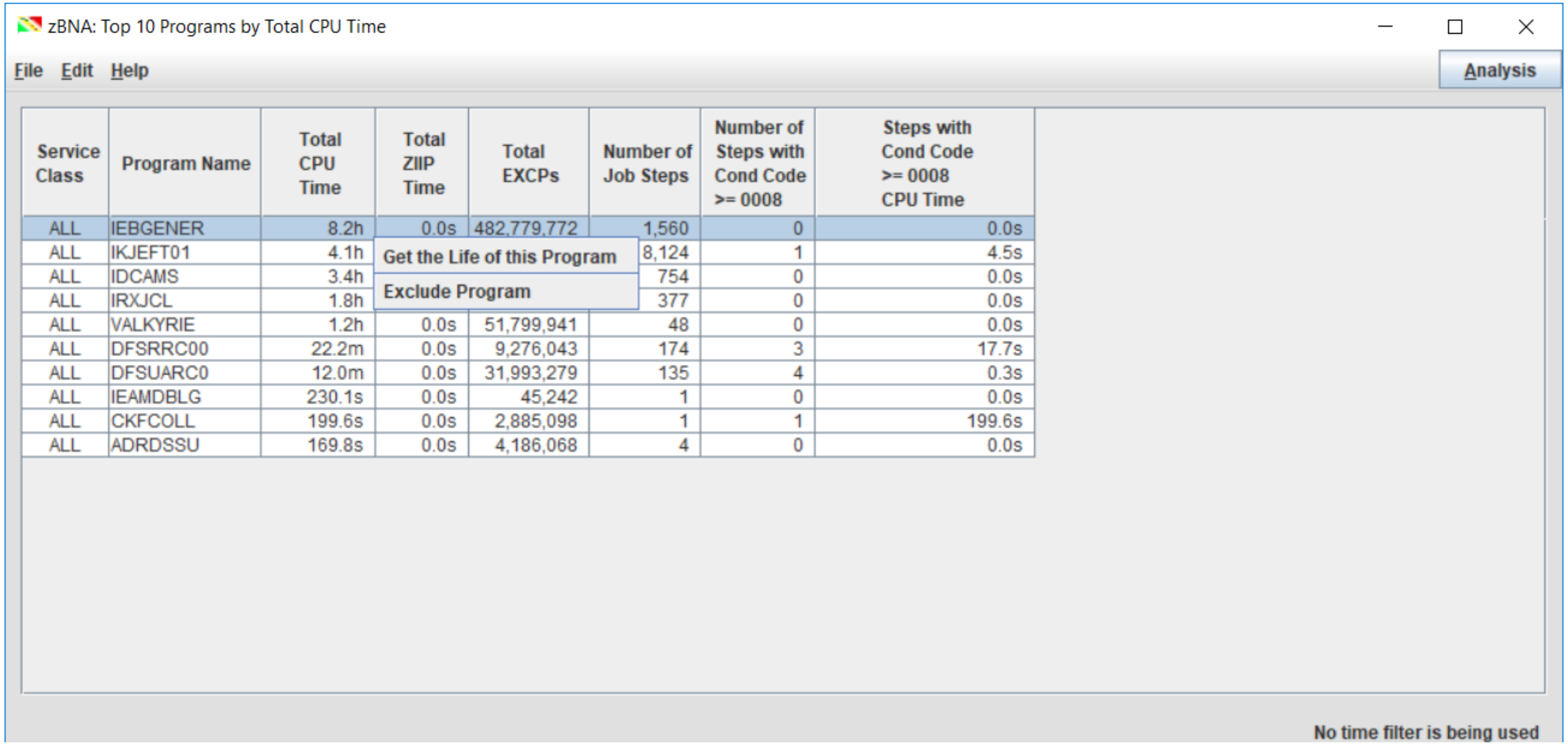
Service Class	Program Name	Total CPU Time	Total ZIIP Time	Total EXCPs	Number of Job Steps	Number of Steps with Cond Code >= 0008	Steps with Cond Code >= 0008 CPU Time
ALL	IKJEFT01	10,005.8	320.5	38,593,100	5,087	364	1,420.1
ALL	CDBUTIL	7,030.8	11,419.0	71,659,286	625	4	13.5
ALL	DRLPLC	4,798.6	0.0	32,569,678	17	2	12.6
ALL	CDBSERVR	3,962.2	1,802.7	80,362,879	335	0	0.0
ALL	DSNUTILB	2,088.5	5,805.0	4,892,858	147	2	0.5
ALL	ADUUMAIN	1,824.8	166.1	7,380,502	716	0	0.0
ALL	DFSRRC00	1,585.0	0.0	35,459,927	5,251	3	28.2
ALL	SASLPA	1,552.1	2.8	12,044,382	1,088	3	47.4
ALL	SORT	1,015.3	3,452.3	18,826,796	7,626	0	0.0
ALL	CTRCTR	644.8	0.0	1,935,591	3,794	0	0.0

zBNA Top Programs – New in V1.8.4

- **Top Programs** added to Application menu
  - **Summarizes which programs utilized most CPU**, and by Condition Code >= 8
  - Can drive "Life of this Program" to see jobs/steps that execute the programs
  - Include / Exclude Lists



Set how many Top Programs to display, e.g. 10



Service Class	Program Name	Total CPU Time	Total ZIIP Time	Total EXCPs	Number of Job Steps	Number of Steps with Cond Code >= 0008	Steps with Cond Code >= 0008 CPU Time
ALL	IEBGENER	8.2h	0.0s	482,779,772	1,560	0	0.0s
ALL	IKJEFT01	4.1h			8,124	1	4.5s
ALL	IDCAMS	3.4h			754	0	0.0s
ALL	IRXJCL	1.8h			377	0	0.0s
ALL	VALKYRIE	1.2h	0.0s	51,799,941	48	0	0.0s
ALL	DFSRRC00	22.2m	0.0s	9,276,043	174	3	17.7s
ALL	DFSUARC0	12.0m	0.0s	31,993,279	135	4	0.3s
ALL	IEAMDBLG	230.1s	0.0s	45,242	1	0	0.0s
ALL	CKFCOLL	199.6s	0.0s	2,885,098	1	1	199.6s
ALL	ADRDSSU	169.8s	0.0s	4,186,068	4	0	0.0s

Example 1 – Unsigned Packed Decimal Add – 4.85x Faster

```
01  WS-VAR-1   COMP-3   PIC s9(7)
01  WS-VAR-2   COMP-3   PIC s9(7).
01  WS-VAR-3   COMP-3   PIC s9(7).
. . .
ADD WS-VAR-1 TO WS-VAR-2
             GIVING WS-VAR-3.
```

Timing (100 million times in a loop)

```
COBOL V4:   3.648 cpu seconds
ARCH(11):   2.195 cpu seconds
ARCH(12):   0.752 cpu seconds
```

ARCH(12) is 4.85 times faster than COBOL V4
ARCH(12) is 2.91 times faster than ARCH(11)
80% less CPU compared to V4!!!!

V4
ARCH(6|7|8|9|10)
- Use in memory instructions
- Explicit sign setting

```
MVC     168(4,R9),160(R9)
OI      171(,R9),X'0F'
MVC     352(4,R13),152(R9)
OI      355(,R13),X'0F'
AP      168(4,R9),352(4,R13)
OI      171(,R9),X'0F'
```

ARCH(11)
- Convert to DFP
- Conversion overhead

```
CDPT    FP0,160(4,R9),0x9
CDPT    FP1,152(4,R9),0x9
ADTR    FP0,FP0,FP1
LPDFR   FP0,FP0
CPDT    FP0,168(4,R9),0xb
```

ARCH(12)
- Use new ARCH(12) facility
- No conversions, no explicit sign setting

```
VLRL    VRF16,160(,R9),0x3
VLRL    VRF17,152(,R9),0x3
VAP     VRF16,VRF16,VRF17,0x7,14
```


89

Example 2 – Large Decimal Divide – 135x Faster

```
01 WS-VAR-1   COMP-3   PIC s9(29)
01 WS-VAR-2   COMP-3   PIC s9(3).
01 WS-VAR-3   COMP-3   PIC s9(25)v9(6).
. . .
DIVIDE WS-VAR-1 BY WS-VAR-2
     GIVING WS-VAR-3.
```

Without ARCH(12)

- Call out to LE library routine
- Pre shifting operation
- Piecewise divide, call overhead

```
ZAP    336(16,13),16(2,2)
MVC    352(32,13),58(10)
MVC    366(15,13),0(2)
NI     380(13),X'F0'
MVN    383(1,13),14(2)
L      3,92(0,9
L      15,180(0,3)     V(IGZCXDI )
LA     1,180(0,10
BASR   14,15
```

Timing (100 million times in a loop)

COBOL V4 or
COBOL V5/V6 w/ARCH(11): 2.319 cpu seconds
ARCH(12): 0.027 cpu seconds

ARCH(12) is 135 times faster than COBOL V4 (or COBOL V5/V6 with ARCH(11) or less)!
99% less CPU compared to pre-ARCH(12)!!!

With ARCH(12)

- Use new ARCH(12) facility
- Inline hardware accelerated shift+divide

```
VLRL    VRF24,_WSA[0x12c] 0(,R3),0xe
VLRL    VRF25,_WSA[0x12c] 16(,R3),0x1
VSDP    VRF24,VRF24,VRF25,0x6,0
```


90

Example 3 – Large Decimal Multiply – 39x Faster

```
01 WS-VAR-1   COMP-3   PIC s9(14)v9(4).
01 WS-VAR-2   COMP-3   PIC s9(14)v9(4).
01 WS-VAR-3   COMP-3   PIC s9(14)v9(2).
MULTIPLY WS-VAR-1 BY WS-VAR-2
     GIVING WS-VAR-3.
```

Timing (100 million times in a loop)

```
COBOL V4 or
COBOL V5/V6 w/ARCH(11):   2.797
cpu seconds
ARCH(12):                 0.072
cpu seconds
```

ARCH(12) is 39 times faster than COBOL V4 (or COBOL V5/V6 with ARCH(11) or less)!
97.5% less CPU compared to pre-ARCH(12)!!!

Without ARCH(12)
- Call out to LE library routine
- Piecewise multiply, call overhead
- Post shifting operation

```
L     3,92(0,9)
L     15,188(0,3)   V(IGZCXMU )
LA    1,171(0,10)
BASR  14,15
NI    388(13),X'0F'
MVN   396(1,13),399(13)
ZAP   32(9,2),388(9,13)
```

With ARCH(12)
- Use new ARCH(12) facility
- Inline hardware accelerated multiply+shift

```
VLRL    VRF16,152(,R9),0x9
VLRL    VRF17,168(,R9),0x9
VMSP    VRF16,VRF16,VRF17,0x6,0
```

91

Example 4 – Zoned Decimal Computation – 3.05x Faster

```
01 WS-VAR-1 PIC 9(8) value 1352435.
01 WS-VAR-2 PIC s9(8)v9(2).
01 WS-VAR-3 PIC s9(10)v9(2).
01 WS-VAR-4 PIC s9(8)v9(2).
. . .
COMPUTE WS-VAR-4 = (WS-VAR-1 / 365) *
          (WS-VAR-2 + 1) - WS-VAR-3.
```

Timing (100 million times in a loop)

```
COBOL V4:   1.469 cpu seconds
ARCH(11):   0.837 cpu seconds
ARCH(12):   0.482 cpu seconds
```

ARCH(12) is 3.05 times faster than COBOL V4
ARCH(12) is 1.74 times faster than ARCH(11)
67% less CPU compared to V4!!!!

V4
ARCH(6|7|8|9)
- **Use in memory instructions**

```
PACK  296(8,13),0(8,2)
SRP   298(6,13),2(0),0
...
DP    296(8,13),40(2,10)
ZAP   264(16,13),296(6,13)
PACK  280(16,13),8(10,2)
...
PACK  296(8,13),24(12,2)
SRP   296(8,13),2(0),0
SP    268(12,13),296(8,13)
...
```

ARCH(10|11)
- **Convert to DFP**

```
CDZT    FP1,_WSA[0x12c] 0(8,R3),0x8
SLDT    FP0,FP1,2
...
DDTR    FP0,FP0,FP1
FIDTR   FP1,9,FP0
LXDTR   FP0:FP2,0,FP1
CDZT    FP1,_WSA[0x12c] 8(10,R3),0x8
...
MXTR    FP4:FP6,FP0:FP2,FP8:FP10
CXZT    FP0:FP2,_WSA[0x12c] 24(12,R3),0x8
SLXT    FP8:FP10,FP0:FP2,2
SXTR    FP0:FP2,FP4:FP6,FP8:FP10
...
```

ARCH(12)
- **Use new ARCH(12) facility**

```
VPKZ    VRF24,_WSA[0x12c] 0(,R3),0x7
VSRP    VRF24,VRF24,0xa,0x2,2
...
VLIP    VRF25,0x365,0
VDP     VRF24,VRF24,VRF25,0xa,0
...
VMP     VRF24,VRF24,VRF25,0x15,0
VPKZ    VRF25,_WSA[0x12c] 24(,R3),0xb
VSRP    VRF25,VRF25,0xe,0x2,0
VSP     VRF24,VRF24,VRF25,0x16,0
...
```


92