

MFC for z/VM

Understanding and Using

- Barton@VelocitySoftware.com
- [HTTP://VelocitySoftware.com](http://VelocitySoftware.com)

“If you can’t Measure it,
I am Just Not Interested™”

- **Points of discussion:**
 - Is SMT adding capacity?
 - Is a large vCPU count a bad thing?
- **Hardware instrumentation: Mainframe Cache**
 - IBM Z Architecture
 - Understanding MFC (SMF113)
 - CPI
 - RNI
 - z/13 vs z/14, z/15 and z/16

Terms:

- L1/L2 Cache: Core level cache
- L3 Cache: Chip level cache
- L4 Cache: Node/Drawer level cache
- Local vs Remote (L4R)
- MFC: MainFrame Cache
- **CPI: Cycles per instruction**
- RNI: Relative nest intensity

Instruction components:

- L1 Load: Data, Instruction
- DAT: Direct Address Translation
- Execution

Cycles per instruction – CPI

- CPU utilization equals instruction rate time – CPI
- Lower CPI means more instructions are possible
- Higher CPI means higher CPU utilization/contention

For an instruction to execute, the L1 cache includes:

- Data (at possibly two locations)
- Instruction (and possibly a branch location)

If the data/instruction not in cache, it must load from:

- L2 cache (1 cycle)
- L3 cache
- L4 cache (local or remote)
- Memory

(The architecture changes on every CPU model)

Instruction use “n” cycles to execute

Most instructions execute one “cycle per instruction”

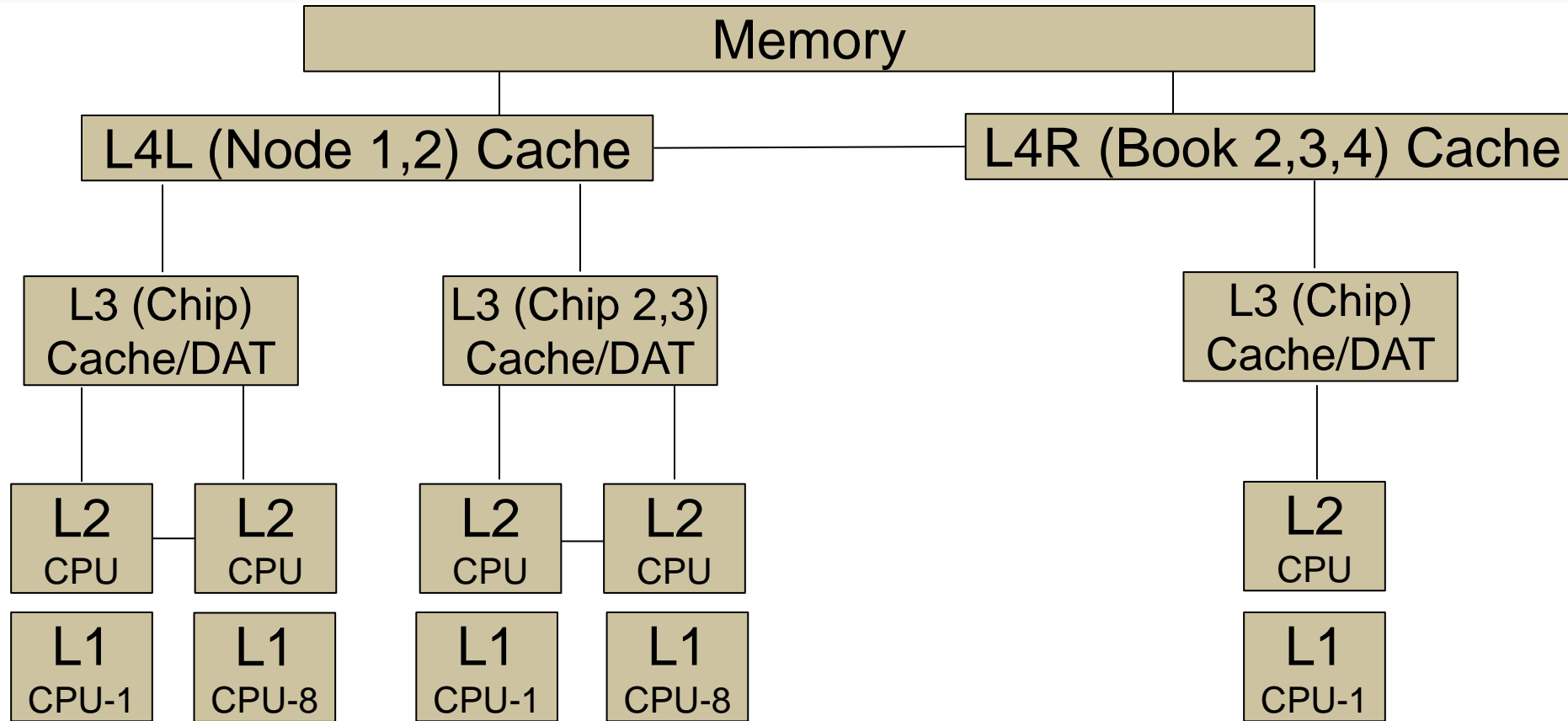
Pipeline reduces “cycles per instruction”

Instructions wait for:

- Instruction in L1 cache
- Data in the L1 cache
- DAT time (address translation for the virtual address)
- Crypto chip
- Compression
- Other things?

If cycles per instruction increase, why?

Z (z13) Architecture



Cores, Chips (multiple Cores), Nodes/Books (multiple Chips)

What is the CPU Measurement Facility?

- Hardware instrumentation
- Statistics by virtual CPU/thread in z/VM LPAR

5.13 Monitor records – CPU domain (PRCMFC)

- Note: The z/OS SMF 113 record is equivalent

Basic Metrics:

- Cycles/Instructions
- Speed of the processor (5.2B cycles per second = 5.2GHz)
- Shows cycles used/instructions executed → CPI Ratio

Extended Metrics:

- Different for each – z10, 196, EC12, z/13, z/14, z/15, z/16

CPU Measurement Facility

What is the CPU Measurement Facility?

- Reported on ESAMFC, ESAMFCA and ESAMFCC
- CPU Percent Busy calculated on cycles used/rated speed
- Shows cycles consumed, instructions executed -> CPI

Report: **ESAMFC** MainFrame Cache Analysis Re
Monitor initialized: 02/27/15 at 20:00:00

```
-----  
                <CPU Busy> <-----Processor----->  
                <percent>  Speed/<-Rate/Sec->  
Time           CPU Totl User  Hertz Cycles Instr Ratio  
-----  
20:01:00      0  0.7  0.4  4196M  30.8M 8313K 3.709
```

↑
BC12...

CPU Measurement Facility for z/VM

Report: **ESAMFCA** MainFrame Cache Hit Analysis
Monitor initialized: 12/10/14 at 07:44:37

```
-----  
                <CPU Busy> <-----Processor----->  
                <percent>  Speed/<-Rate/Sec-> CPI  
Time           CPU Totl User  Hertz Cycles Instr Ratio  
-----  
07:48:35      0 20.8 18.4 5504M 1121M 193M 5.807  
              1 21.6 19.6 5504M 1161M 221M 5.264  
              2 24.4 22.5 5504M 1300M 319M 4.078  
              3 22.4 19.7 5504M 1248M 265M 4.711  
              4 19.6 17.6 5504M 1102M 194M 5.683  
              5 20.4 18.6 5504M 1144M 225M 5.087  
              6 23.9 22.0 5504M 1341M 341M 3.935  
              7 17.6 15.4 5504M  949M 160M 5.927  
              8 18.5 16.5 5504M 1005M 194M 5.195  
              9 22.5 20.6 5504M 1259M 347M 3.629  
-----  
System:           212 191 5504M 10.8G 2457M 4.733
```



EC12...

**What is the CPU
Measurement Facility?
(Basic)**

CPI: Cycles per Instruction

Why you should be interested in MFC

Report: **ESAMFC** MainFrame Cache Analysis Rep

| Time | CPU | <CPU Busy> <percent> | | <-----Processor-----> Speed/<-Rate/Sec-> | | | |
|----------|-----|-------------------------|------|---|--------|--------------|--------------|
| | | Totl | User | Hertz | Cycles | Instr | Ratio |
| 14:05:32 | 0 | 92.9 | 64.6 | 5000M | 4642M | 1818M | 2.554 |
| | 1 | 92.7 | 64.5 | 5000M | 4630M | 1817M | 2.548 |
| | 2 | 93.0 | 64.7 | 5000M | 4646M | 1827M | 2.544 |
| | 3 | 93.1 | 64.9 | 5000M | 4654M | 1831M | 2.541 |
| | 4 | 92.9 | 64.8 | 5000M | 4641M | 1836M | 2.528 |
| | 5 | 92.6 | 64.6 | 5000M | 4630M | 1826M | 2.536 |
| System: | | 557 | 388 | 5000M | 25.9G | 10.2G | 2.542 |
| 14:06:02 | 0 | 67.7 | 50.9 | 5000M | 3389M | 2052M | 1.652 |
| | 1 | 67.8 | 51.4 | 5000M | 3389M | 2111M | 1.605 |
| | 2 | 69.0 | 52.4 | 5000M | 3450M | 2150M | 1.605 |
| | 3 | 67.2 | 50.6 | 5000M | 3359M | 2018M | 1.664 |
| | 4 | 60.8 | 44.5 | 5000M | 3042M | 1625M | 1.872 |
| | 5 | 70.1 | 53.8 | 5000M | 3506M | 2325M | 1.508 |
| System: | | 403 | 304 | 5000M | 18.8G | 11.4G | 1.640 |

More work

Less CPU?



z13...

Why you should be interested – What is a MIP?

Report: **ESAMFC** MainFrame Cache Analysis Rep

| Time | CPU | <CPU Busy> <percent> Totl User | | <-----Processor-----> Speed/<-Rate/Sec-> Hertz Cycles Instr Ratio | | | |
|----------|-----|--------------------------------------|------|--|-------|--------------|--------------|
| 14:05:32 | 0 | 92.9 | 64.6 | 5000M | 4642M | 1818M | 2.554 |
| System: | | 557 | 388 | 5000M | 25.9G | 10.2G | 2.542 |
| 14:06:02 | 0 | 67.7 | 50.9 | 5000M | 3389M | 2052M | 1.652 |
| System: | | 403 | 304 | 5000M | 18.8G | 11.4G | 1.640 |

1830 MIPS (at 100%)

2828 MIPS (at 100%)

Workload changes:

- Cache benefit is better
- Cycles per instruction then improves
- Doing 10% more work at 20% lower utilization

Capacity benefits of SMT depend on workloads:

- Cache benefit drops if high dispatch rate
- Address translation time increases as cache benefit drops

Why you should be interested – What is a MIP?

Case Study (working backwards)

What happens when CPI doubles?

Report: **ESAMFC** MainFrame Cache Magnitudes

| Time | CPU | <CPU Busy> <percent> Totl | User | <-----Processor-----> Speed/<-Rate/Sec-> Hertz | Cycles | Instr | Ratio |
|---------|-----|------------------------------|------|--|--------|-------|--------------|
| System: | | 1741 | 1632 | 5200M | 90.7G | 50.2G | 1.808 |
| System: | | 1666 | 1559 | 5200M | 86.8G | 50.5G | 1.717 |
| System: | | 1727 | 1622 | 5200M | 89.9G | 52.2G | 1.724 |
| System: | | 1886 | 1787 | 5200M | 98.2G | 55.9G | 1.757 |
| System: | | 2414 | 2298 | 5200M | 126G | 50.2G | 2.504 |
| System: | | 3590 | 3491 | 5200M | 187G | 41.1G | 4.541 |
| System: | | 3591 | 3494 | 5200M | 187G | 42.9G | 4.349 |
| System: | | 3591 | 3487 | 5200M | 187G | 41.6G | 4.489 |
| System: | | 3590 | 3486 | 5200M | 187G | 45.6G | 4.091 |
| System: | | 3224 | 3091 | 5200M | 168G | 51.4G | 3.263 |
| System: | | 1948 | 1806 | 5200M | 101G | 49.7G | 2.041 |
| System: | | 2069 | 1937 | 5200M | 108G | 55.3G | 1.950 |

Why you should be interested in MFC

What happens when CPI doubles?

Report: **ESASSUM** Subsystem Activity

```
-----
      <---Users----> Transactions <Processor>
      <-avg number->   Per   Avg. Utilization
te
Time      On Actv In Q Minute   Resp Total Virt.
-----  -
14:01:00  95   70  191   88.0 0.030  1741  1632
14:02:00  95   69  191   89.0 0.028  1666  1559
14:03:00  95   71  193   89.0 0.030  1727  1622
14:04:00  95   69  194   86.0 0.035  1886  1787
14:05:00  95   73  189   81.0 0.029  2414  2298
14:06:00  95   70  189   83.0 0.192  3590  3491
14:07:00  95   70  188   83.0 0.083  3591  3494
14:08:00  95   71  198   86.0 0.070  3591  3487
14:09:00  95   70  196   81.0 0.097  3590  3486
14:10:00  95   75  189   88.0 0.088  3224  3091
14:11:00  95   70  191   86.0 0.067  1948  1806
```

Why you should be interested in MFC

```
Report: ESAXACT          Transaction Del
-----
<-----Percent n
UserID  <-Samples->
/Class  Total    In Q   Run   Sim   CPU   SIO
-----
Hi-Freq: 14220 11490  8.9  0.3  5.6   0
Hi-Freq: 14220 11487  8.3  0.2  4.4  0.0
Hi-Freq: 14220 11508  8.5  0.2  4.3  0.1
Hi-Freq: 14220 11541  9.3  0.2  6.1   0
Hi-Freq: 14220 11525  12  1.7  13  0.0
Hi-Freq: 14220 11367  19  3.6  30  0.0
Hi-Freq: 14220 11308  19  3.4  30  0.0
Hi-Freq: 14220 11281  19  3.0  31  0.2
Hi-Freq: 14220 11063  19  2.7  28  0.0
Hi-Freq: 14220 11064  17  0.9  22   0
Hi-Freq: 14220 11213  10  0.4  7.5  0.0
Hi-Freq: 14220 11268  11  0.6  7.8   0
```

CPU Utilization doubles:

- Transaction CPU (cycles/transaction) doubles
- CPU wait gets VERY large?

Why you should be interested in MFC

Case study summary – bad response time:

- Transaction delays CPU
- CPU was up 100%
- CPI was up
- Only way to define the problem is CPI

What makes up “Cycles per Instruction”?

Instruction time:

- Cache load time (instruction/data)
- Address translation (DAT)
- Wait for DAT
- The instruction (can take less than 1 cycle – pipeline)
- “Other delays” – Compression, Encryption, AI
- Wait time

Measurement metrics:

- RNI
- CPI

Instruction Load Time

Reported in “per 100 instructions”

- 0.7% miss is 7 misses per 1,000 instructions
- .02 MEM equates to one memory reference per 5,000 instructions
- Example (z/15) is low utilization

Report: **ESAMFCA** MainFrame Cache Hit Analysis

```

-----
--Processor-----> <-----Rate per 100 Instructions
<-Rate/Sec-> CPI   L1   <---Data source read from--->
Time    Cycles Instr Ratio MISS L2    L3    L4L  L4R  MEM
-----
09:28:00  488M  384M  1.272  0.701 0.590 0.074 0.008 0.000 0.029
09:29:00  485M  385M  1.260  0.701 0.594 0.074 0.007 0.000 0.027
09:30:00  537M  434M  1.238  0.686 0.584 0.070 0.007 0.000 0.024
09:31:00  524M  403M  1.301  0.726 0.608 0.076 0.010 0.000 0.032
09:32:00  535M  420M  1.273  0.720 0.570 0.114 0.009 0.000 0.027
09:33:00  618M  522M  1.184  0.671 0.584 0.061 0.006 0.000 0.020
  
```

Processor Cache Comparison

Clock speed 5,500 to 5,000 (10% slower?)

- How can a z/13 be faster than an EC12? Less CPI

Cache Sizes – EC12

- L1: 64K **Instruction**, 96K **Data**
- L2: 1MB Instruction, 1MB Data (Private CPU)
- L3: **48MB (Chip, Shared 6 CPUs)**
- L4: 384MB (Book, Shared over 20 CPUs)

Cache Sizes – z/13

- L1: 96K Instruction, 128K Data
- L2: 2MB Instruction, 2MB Data
- L3: **64MB (Chip, Shared over 8 CPUs)**
- L4: 480MB + 224M NIC (Per Node)

Processor Cache Comparison

Cache Sizes – z/15

- L1: 128K Instruction, 1286K Data
- L2: 4MB Instruction, 4MB Data (Private CPU)
- L3: 256MB (Chip, Shared 12 CPUs)
- L4: 960MB (Book, Shared over 20 CPUs)

Cache Sizes – z/16

- L1: 128K Instruction, 128K Data
- L2: 32MB Unified (Instruction/Data)
- L3: 224MB (Chip, Shared over 8 CPUs)
- L4: 1.75GB (Per drawer)

Measure of cache value:

- RNI – How far away is the data?
(Relative Nest Intensity is a z/OS metric defined by John Burg)

IBM RNI Calculations (per John Berg, WSC):

- z/16 RNI = **4.3** $(0.45 * L3P + 1.3 * L4LP + 5.0 * L4RP + 6.1 * MEMP) / 100$
- z/15/s RNI = **2.9** $(0.45 * L3P + 1.5 * L4LP + 3.2 * L4RP + 6.5 * MEMP) / 100$
- z/14/s RNI = 2.4 $(0.40 * L3P + 1.5 * L4LP + 3.2 * L4RP + 7.0 * MEMP) / 100$
- z/13 RNI = 2.6 $(0.40 * L3P + 1.6 * L4LP + 3.5 * L4RP + 7.5 * MEMP) / 100$
- EC12 RNI = 2.3 $(0.40 * L3P + 1.2 * L4LP + 2.7 * L4RP + 8.2 * MEMP) / 100$

Smaller is better – less time loading L1 cache

Higher means more opportunity for SMT?

Cycle Requirement per Source

z/15 – Based on RNI calculations (per John Burg)

- **Level 3 = 2.9 * .45 = 1.3 cycles**
- **Level 4L = 2.9 * 1.5 = 4.3 cycles**
- **Level 4R = 2.9 * 3.2 = 9 cycles**
- **Memory = 2.9 * 6.5 = 19 cycles**

A lot of cycles can be wasted:

- RNI is a measure
- Does not account for DAT

What happens to RNI when a 2nd thread is added?

- (Yes, it gets larger – per thread)

Dynamic Address Translation (DAT)

On a z/13, one DAT per core

- Address translation is performed on every address
- Translated addresses stored in the TLB (Translation Lookaside Buffer)

MFC data provides cycles waiting for DAT

During address translation, core cycles are wasted

The z/14 and beyond have 4 DATs per core

Hardware Metrics: TLB Analysis – z/13

DAT Translation:

- 30% of the cycles for **ONE** thread – below shows **one** thread at **30.69%**
- Two threads on one core leaves very little for real work

Report: **ESAMFC** MainFrame Cache Magnitudes Report ZMAP 4.2.4

```

-----
              <CPU Busy> <-----> <-Translation Lookaside buffer(TLB)-
              <percent>  Speed/      <cycles/Miss><Writs/Sec>  CPU Cycles
Time         CPU Totl User  Hertz  Ratio  Instr  Data Instr  Data  Cost  Lost
-----
07:45:01    0 25.9 24.4 5000M 1.704   159   742 473K 244K 19.77 257M
              1 35.9 34.7 5000M 1.491   138   731 530K 249K 14.17 255M
              2 15.8 13.9 5000M 2.868   206   826 419K 245K 36.30 289M
              3 16.6 15.4 5000M 2.508   212   825 411K 247K 34.90 291M
              23 18.1 17.0 5000M 2.144   197   815 412K 229K 29.44 268M
              24 21.4 19.9 5000M 1.865   114   533 598K 302K 21.35 229M
              25 26.2 24.9 5000M 1.742    98   503 736K 346K 18.71 246M
              26 12.9 11.6 5000M 2.050   154   631 378K 214K 29.92 194M
              27 13.1 11.9 5000M 1.987   156   630 378K 217K 29.64 195M
              ---
System:      514  476 5000M 2.257   176   724  14M 7641K 30.69 7917M
-----

```


TLB Analysis – Should SMT be Enabled?

Evaluate other data points:

- CPU Cost is “percent of cycles”
- z/VM Linux workloads issue: **VERY HIGH dispatches**
- Why z14 should be great...
- Don't enable SMT if one thread is consuming all of your DAT

Report: **ESAMFC**

MainFrame Cache Magnitudes Report

```
-----
      <CPU Busy>
      <percent>   Speed
  ##   Totl User   Hertz
  ----  - - - - -
Mem1   907  874  5504M
Mem2  1188 1140  5000M
VLB4  1703 1366  5000M
z13N   216  212  5000M
TCPN   892  757  5000M
MTRN   947  868  5000M

      <-Translation Lookaside buffer(TLB)->
      <cycles/Miss><Writs/Sec>  CPU Cycles
      Instr  Data Instr  Data  Cost  Lost
      ----  - - - - -
          54   232   117M   36M 29.55 14.8G
          147  364    30M   26M 23.62 14.0G
          185  567    66M   46M 44.59 38.2G
          192  598 3084K 1802K 15.94 1669M
          217  947    32M   17M 51.46 23.0G
          265 1283    33M   17M 65.25 30.8G
-----
```

TLB Problem, z/14-z/15 Advantages

z/13 (z/VM) Problem:

- z/VM does NOT support large pages, it needs 256 times as much TLB
- Linux with Java/WebSphere has VERY high dispatch rates (30K/sec?)
- Address translation (DAT) required for all parts of an instruction
- Sometimes no cycle are left after address translation...

z/14 – z/15:

- The fix – If one DAT per core is the bottleneck, put in 4 (“Quad TLLB”)
- Waiting for DAT still degrades performance
- z/14 – SMT is better
- z/15 – Just as good

TLB Analysis – Should SMT be Enabled?

Report: **ESAMFC** MainFrame Cache Magnitudes Rate ZMAP 5.1.0 initialized:

| <CPU Busy> | | <-----Processor-----> | | | | | <-Translation Lookaside buffer(TLB)-> | | | | | |
|------------|------|-----------------------|-------|--------|-------|-------|---------------------------------------|------|--------------|------|------------|------|
| <percent> | | Speed/<-Rate/Sec-> | | | | | <cycles/Miss> | | <Writes/Sec> | | CPU Cycles | |
| CPU | Totl | User | Hertz | Cycles | Instr | Ratio | Instr | Data | Instr | Data | Cost | Lost |
| 0 | 29.5 | 28.0 | 5208M | 1535M | 822M | 1.867 | 177 | 284 | 243K | 364K | 9.55 | 147M |
| 1 | 26.8 | 25.3 | 5208M | 1399M | 748M | 1.871 | 178 | 294 | 248K | 359K | 10.71 | 150M |
| 2 | 37.3 | 35.1 | 5208M | 1945M | 877M | 2.219 | 135 | 210 | 446K | 818K | 11.91 | 232M |
| 3 | 36.9 | 34.8 | 5208M | 1925M | 914M | 2.107 | 136 | 212 | 449K | 821K | 12.19 | 235M |
| 4 | 22.6 | 20.9 | 5208M | 1181M | 530M | 2.228 | 158 | 263 | 316K | 445K | 14.18 | 167M |
| 5 | 23.4 | 21.8 | 5208M | 1219M | 590M | 2.066 | 156 | 260 | 316K | 449K | 13.63 | 166M |
| 6 | 23.9 | 21.5 | 5208M | 1248M | 615M | 2.030 | 170 | 284 | 236K | 364K | 11.49 | 143M |
| 7 | 26.9 | 25.5 | 5208M | 1402M | 730M | 1.921 | 166 | 265 | 237K | 391K | 10.19 | 143M |
| 8 | 31.2 | 29.5 | 5208M | 1628M | 792M | 2.055 | 163 | 257 | 338K | 507K | 11.39 | 185M |
| 9 | 32.9 | 31.3 | 5208M | 1715M | 878M | 1.954 | 159 | 247 | 326K | 508K | 10.34 | 177M |
| 10 | 20.9 | 19.4 | 5208M | 1093M | 504M | 2.171 | 166 | 276 | 257K | 391K | 13.79 | 151M |
| 11 | 23.4 | 22.1 | 5208M | 1223M | 658M | 1.859 | 162 | 265 | 247K | 401K | 11.95 | 146M |
| 12 | 22.3 | 20.5 | 5208M | 1162M | 526M | 2.209 | 173 | 302 | 321K | 443K | 16.32 | 190M |

z/14 is better:

- DAT x (4) gives a lot of cycles back
- 12% DAT cycles (SMT) vs 30% on the z13 – NO SMT...

Objective – More work per cycle

- Less L1 cache misses
- Less address translation
- Lower RNI (the data is closer)
- Less cache pollution

Affinity

- Dispatching a virtual machine on the same core uses L1/L2 cache
- Keeping work on the same CHIP maintains L3 cache
- Keeping work on the same drawer maintains L4 cache
- “Steals” from one thread to another is delayed

What impacts (pollutes) cache:

- High dispatch rate with new process loaded
- High number of virtual CPUs accessing the same data

Nesting Steals – Is Affinity Working?

Report: **ESAPLDV** Processor Local Dispatch Vector Activity

| Time | CPU | <VMDBK Moves/sec> | | <-CPU Steals | | | |
|----------------|-----|-------------------|-------------|-----------------------|---------------------|------------|----------|
| | | Steals | To Master | Dispatcher Long Paths | <-From Nesting Same | NL1 | NL2 |
| 19:47:00 | 0 | 7442.9 | 16.5 | 34163.5 | 3034 | 4408 | 0 |
| | 1 | 5854.5 | 0 | 29842.1 | 2313 | 3542 | 0 |
| | 2 | 5363.9 | 0 | 23112.3 | 2466 | 2898 | 0 |
| | 25 | 5900.3 | 0 | 25649.6 | 847 | 5053 | 0 |
| | 26 | 7022.2 | 0 | 28863.4 | 1035 | 5987 | 0 |
| | 27 | 5907.7 | 0 | 25927.4 | 799 | 5109 | 0 |
| System: | | 161948 | 16.5 | 757754.4 | 67K | 95K | 0 |

A z/13 – 60 IFLs Total, 14 IFLs in LPAR (SMT enabled)

Steals: VMDBLKs moved to a different processor (20% of the time)

Dispatcher Long Paths:

- VMDBLKs dispatched (**30K/sec/CPU**)
- **NL1: Different Chip (L3) (check affinity)**
- **NL2: Different book (L4) – No NL2, are smaller LPARs better?**

Nesting Steals – Is Affinity Working?

```
Report: ESAPLDV      Processor Local Dispatch Vector Activity
-----
<-CPU Steals from Other CPUs->
      <VMDBK Dispatcher <-From Nesting Levels (/sec)->
Time    CPU Steals Long Paths Same  NL1  NL2  NL3  NL4  NL5
-----
21:25:00  0  924.1    8387.1  632  42.9  249    0    0    0
          1  855.0    7697.4  597  39.3  219    0    0    0
          2 1047.0    7640.9  781  45.6  220    0    0    0
          3  931.6    7081.6  691  41.2  199    0    0    0
          4  935.7    7552.7  682  40.0  213    0    0    0
          5  851.6    7651.8  619  37.8  195    0    0    0
-----
System:      21363  139331.4  9917  1292  10K    0    0    0
```

**A z/14 LPAR – 14 IFLs/28 threads
(SMT enabled)**

**NL2: Different Cluster/Book (L4) –
LPAR too big?**

LPAR defined across Books

Why NL2?

| CPU | Percent | Type | CPU | Percent | Type | CPU | Percent | Type |
|-----|---------|------------|-----|---------|------------|-----|---------|------|
| 1 | 1.037 | CP | 28 | 0.384 | IFL | 52 | 0.003 | IFL |
| 2 | 0.884 | CP | 29 | 1.102 | CP | 53 | 0.002 | IFL |
| 3 | 1.163 | CP | 32 | 0.416 | IFL | 54 | 0.675 | IFL |
| 4 | 0.838 | CP | 33 | 2.155 | CP | 55 | 0.605 | IFL |
| 5 | 1.870 | CP | 34 | 2.169 | CP | 57 | 0.855 | IFL |
| 6 | 0.463 | IFL | 38 | 0.234 | IFL | 58 | 0.683 | IFL |
| 8 | 0.459 | IFL | 40 | 1.337 | IFL | 59 | 0.702 | IFL |
| 9 | 1.678 | CP | 41 | 0.777 | IFL | 60 | 0.743 | IFL |
| 11 | 2.314 | ZII | 42 | 0.551 | IFL | 61 | 0.770 | IFL |
| 12 | 0.398 | ZII | 43 | 0.793 | IFL | 62 | 1.200 | IFL |
| 13 | 0.859 | ZII | 45 | 0.765 | IFL | 63 | 1.495 | IFL |
| 14 | 0.844 | ZII | 46 | 0.850 | IFL | 65 | 1.115 | CP |
| 18 | 0.983 | IFL | 48 | 0.002 | IFL | 66 | 1.644 | IFL |
| 24 | 0.795 | IFL | 49 | 0.002 | IFL | 67 | 1.572 | IFL |
| 26 | 0.441 | IFL | 50 | 0.002 | IFL | 68 | 1.586 | IFL |
| 27 | 0.722 | IFL | 51 | 0.002 | IFL | 74 | 0.235 | IFL |
| | | | | | | 75 | 1.727 | CP |

z/14 – Why does NL2 happen?

- 10 Cores/Chip
- 2-3 Chips/Cluster
- 2 Clusters/Drawer
- 4 Drawers maximum
- CPU order or just bad luck – (ESALPAR)?

<-----CPU----->

| Type | Count | Ded | shared |
|------|-------|-----|--------|
| CP | 11 | 0 | 11 |
| IFL | 37 | 6 | 31 |
| ICF | 4 | 3 | 1 |
| ZIIP | 4 | 0 | 4 |

Moving virtual machines to a different nesting level

LPAR IFLs are across two drawers

IFLs on the wrong Drawer – Off Drawer Memory reads

Report: **ESAMFCC** MainFrame L1 Cache Write Analysis

| Time | CPU | <CPU Busy>-----> | | <-L4 Cache-> | | < Sourced from Memory / Sec> | | | Off Drawer |
|----------|-----|------------------|------|--------------|--------------|-------------------------------------|---------|----------|-------------------|
| | | <percent> | | OnBook | OffBk | On Chip | On Book | Off Book | |
| 21:25:02 | 0 | 88.4 | 74.5 | 535093 | 72532 | 197648 | 398002 | 659588 | 0 |
| | 1 | 88.6 | 76.9 | 548073 | 71167 | 187255 | 386960 | 590812 | 0 |
| ... | | | | | | | | | |
| | 7 | 89.0 | 77.7 | 561408 | 70493 | 202484 | 417522 | 637009 | 0 |
| | 8 | 89.9 | 77.7 | 594979 | 76359 | 208859 | 427118 | 650971 | 0 |
| | 9 | 89.0 | 77.9 | 591668 | 74152 | 207873 | 423139 | 647480 | 0 |
| | 10 | 12.9 | 11.8 | 94593 | 39694 | 14984 | 15844 | 32611 | 137067 |
| | 11 | 12.7 | 11.8 | 96285 | 38869 | 16900 | 17382 | 36391 | 133331 |
| | 12 | 13.7 | 12.7 | 87874 | 58750 | 13900 | 14098 | 29191 | 135545 |