35 YEARS OF PERFORMANCE MANAGEMENT

CPU Analysis and Tuning for Linux on System Z

- Barton@VelocitySoftware.com
- HTTP://VelocitySoftware.com

"If you can't Measure it,
I am Just Not Interested ™"

VELOCITY SOFTWARE

PROVEN PERFORMANCE

Processor Topics

Overview

CPU Management Hierarchy

- CEC/LPAR weights
- LPAR/Virtual Machine Share

Identifying CPU problem

CEC / LPAR Weights and analysis

Virtual Machine Shares

Affinity / DSPSLICE

SMT

CPU Overview

Importance of CPU Utilization

Most software licensing is based on number of CPUs

CPUs are purchased and licensed individually

CPUs operating at higher utilization cost less

Our objectives:

- Operate at the highest utilization
- Meet performance requirements

CPU Hierarchy

Processor Performance Concepts - Utilization

Utilization/Capacity is important:

- CEC Utilization (Capacity Planning)
- LPAR Weights (Entitlement)
- LPAR Utilization vs Entitlement
- Virtual Machine Setting (Share)
- Virtual CPU share
- Linux Process “niced”
- Workload requirements

CPU Utilization is used for:

- Performance Analysis
- Capacity Planning
- Accounting/Chargeback
- Operational Alerts

Higher utilization requires less hardware and software:

- Higher utilization requires correct configuration
- Higher utilization requires knowledge and management

Processor Measurement Concepts

Measured Utilization vs Reported Utilization

- Virtual Linux measures what?
  - Percent of wall clock time originally, now is “steal timer”
- z/VM measures what? CPU seconds (**hardware timer**)
- SMT: Core vs thread utilization is confusing
- Hardware measurement is the only valid method of measuring CPU

Percent of Percent is misleading

- Can not be used directly for capacity planning
- Can not be used directly for accounting/chargeback
- Is often misleading for performance analysis

Processor Performance Reporting - Utilization

All zVPS numbers are measured in CPU seconds

- Percent is always based on CPU seconds divided by wall clock time
- 200% means using 2 engines worth of CPU seconds
- Measured by the hardware in microseconds

This impacts the measurements of:

- Total IFLs/GPs
- LPARs
- z/VM Virtual Machines
- Linux processes
- VSE Jobs/Partitions
- z/OS Jobs/Partitions

Processor Utilization Components

LPAR Level:

- LPAR Physical Overhead
- LPAR Assigned time – Overhead
- LPAR Assigned time – Virtual

z/VM Level: (LPAR Assigned time – Virtual)

- System Time (z/VM Control Program)
- User Overhead (allocated system time)
- Emulation (z/VM guest time)

Linux Level: (Emulation – z/VM guest time)

- System time (kernel time)
- IRQ time
- User time (“real application work”)

IDLE

The Processor Distribution Hierarchy

At the CEC Level (Total Capacity):

- IFLs shared or dedicated at LPAR level (ESALPARS)

Shared Processor distribution "managed":

1. The LPAR is assigned a "weight"
   - An "entitlement" of the IFLs (ESALPAR)
2. z/VM LPAR Perspective (ESACPUU/ESACPUA) and "Parking"
3. The Virtual Machine is assigned a "share"
   - A "share" of the LPAR's available threads (ESAUSRC / ESAUSP2)
   - Thread time vs core time (ESAUSP5)
4. The Linux Processes have a "priority"
   - Processes are "prioritized" by "nice" settings (ESALNXC/P)

Detecting CPU Issues

Linux Perspective CPU Problem

Problems from a "Linux perspective" (bottom-up analysis):

- Workload is timing out
- Applications are running slowly
- "top" gives a limited view
- Linux admins complain about "steal time"

Steal time virtualization concept high level

- Virtual machines measured in CPU Wait
- Linux steal: Linux is ready to run, but not being dispatched
- z/VM steal: LPAR Level - vCPU is used by another LPAR

Other causes of CPU delays:

- Operation on GP, not on IFL engines (it happens)
- Cron jobs synchronized (100 processes across 100 servers)
- Spin locks – DIAG 44/9C (too many virtual machine vCPUs)

Identifying CPU Problems

Report: **ESAXACT** Transaction Delay Analysis

UserID /Class	<-Samples-> Total	In Q	<-----Percent non-dormant (Wait sta Run	Sim	**CPU**	SIO	Pag	D- SVM	T- SVM	CF	Tst Idl
15:45:00	52	34	50	2.9	29	0	0	0	0	0	18
Hi-Freq:	4740	2077	49	0.5	**19**	0.0	0	7.9	0.1	0	32
User Class Analysis											
Servers	840	47	0	0	0	2.1	0	8.8	4.3	0	94
ZVPS	660	12	17	0	8.3	0	0	0	0	0	75
TheUsers	2760	2018	50	0.5	19	0	0	11	0.0	0	30
Top User Analysis											
LXHDBLQ1	360	360	76	0.3	22	0	0	0	0	0	1.9
LXDDBLQ1	480	480	53	0.8	24	0	0	0	0	0	22
LXDDBLQ3	480	480	55	0.2	20	0	0	0	0	0	25
LXDDBLQ5	360	360	59	1.4	27	0	0	0	0	0	12
LX11PRX1	120	120	1.7	0	0.8	0	0	0	0	0	98
LX11PRX2	120	120	1.7	0	0.8	0	0	0	0	0	98
LX11RHCL	120	97	1.0	0	1.0	0	0	0	0	0	98

Running: Using CPU
CPU Wait: Waiting for CPU
Why is CPU Wait high?

- High CEC Utilization?
- High LPAR Utilization?
- Low LPAR Weight?
- Low Share?
- Affinity?

If CPU Problem:

Are there LPAR Configuration issues (weights, entitlements)?

Are there Capacity Issues (at very high utilization)?

Linux Configuration - Number of vCPUs?

- Spin locks (one vCPU waits for lock held on another vCPU)?
- Linux overhead of managing vCPUs?
- z/VM overhead of managing vCPUs?
- Hardware cache pollution?
- Valid shares per vCPU?

Settings correct (not default!)?

LPAR Weights and HiperDispatch

LPAR Analysis - ESALPARS

LPAR Analysis:

- Total IFL utilization (of the CEC - are there cycles to spare?)
- LPAR weight (entitlement)
- LPAR utilization

Virtual CPU Entitlement:

- LPAR entitlement divided by number of “cores” in the LPAR
- More vCPUs results in lower weighting per vCPU
- Hiperdispatch corrects this by “parking” vCPUs
- SMT threads share “core” entitlement

CEC Level Analysis

Evaluate CEC perspective first:

- There are 10 IFLs that are "shared"
- IFLS are total utilization 92%, assigned (903.1/1000)
- Note the system overhead – "Ovhd" and "Mgmt" (8.6 and 12.5)
- CPU delays can be expected! Shares/Weights need to be adjusted

Report: **ESALPARS** **Logical Partition Summary**

Totals by Processor type:

	<----------CPU-------->			<-Shared Processor busy->			
Type	Count	Ded	shared	Total	Logical	Ovhd	Mgmt
CP	7	0	7	511.9	501.5	4.5	5.9
IFL	**10**	**0**	**10**	**915.6**	**894.5**	**8.6**	**12.5**
ZIIP	3	0	3	23.9	22.3	0.4	1.2

LPAR Configuration (Entitlement) Summary

z/VM LPAR share of IFLs (always start here):

```
Report: ESALPARS      Logical Partition Summary
------------------------------------------------------------------------
          <--Complex--> <--------Logical Partition-------->  <-Assigned
          Phys Dispatch                Virt CPU  <%Assigned>  <---LPAR-->
Time      CPUs    Slice Name      Nbr CPUs Type Total  Ovhd  Weight  Pct
--------  ---- -------- -------- --- ---- ---- -----  ----  ------ ----
00:15:00    23  Dynamic Totals:    0   22  CP  506.0   4.5     999  100
                        Totals:    0   23 IFL  903.1   8.6    1000  100
                        ZVMQA     11    6 IFL  374.8   0.9     150 15.0
                        ZVMDEQ     9    4 IFL  131.6   2.0     100 10.0
                        ZVMPRD     8   10 IFL  333.7   4.9     650 65.0
                        ZVMSHR    12    3 IFL   63.0   0.8      80  8.0
Totals by Processor type:
<---------CPU-------> <-Shared Processor busy->
Type Count Ded shared  Total  Logical Ovhd Mgmt
---- ----- --- ------ ------ -------- ---- ----
IFL     10   0     10  915.6    894.5  8.6 12.5
```

- ZVMQA entitlement: 150/1000 (15%) of 10 shared IFLs
- So ZVMQA entitlement is 1.5 IFLs
- Virtual CPU entitlement:
  1.5 IFLs / 6 vCPUs (.25 IFL core per vCPU)
- ZVMQA is using 375% shared IFLs (more than 1.5 entitlement)
- IFLs running 915/1000% (91.6%) busy
- ZVMPRD entitlement: 6.5 IFLs but using 3.3 (Production has priority)

LPAR Options

Weights: Sets entitlement between Logical Partitions

- Set weights based on business requirements
- Creates the “guaranteed entitlement”

Virtual Processors

- If too many, HiperDispatch will “park”

Capping

- Limits Assigned Time to LPAR – use it carefully
- Useful for outsourcing or fixed contracts

LPAR Entitlement

Entitlement field added! Validate assigned vs entitled

ESALPARS Logical Partition Summary

<-------Logical Partition------->				<%Assigned>		<-Assigned Shares---->		<VCPU Pct>		**Entitled**
Name	Nbr	Virt CPUs	CPU Type	Total	Ovhd	Weight	Pct	/SYS	/CPU	CPU Cnt
Totals:	00	387	IFL	4451	156	3860	100			
L1A1	21	4	IFL	2.6	0.4	50	1.3	0.32	44.3	1.77
L1D1	01	50	IFL	167.0	10.1	500	13.0	0.26	35.5	17.75
L1D2	02	40	IFL	490.8	38.7	900	23.3	0.58	79.9	31.94
L1D3	03	30	IFL	1.2	0.4	50	1.3	0.04	5.91	1.77
L1D4	04	14	IFL	1.3	0.5	10	0.3	0.02	2.52	0.35
L1E1	05	20	IFL	64.8	3.6	500	13.0	0.65	88.7	17.75
L1C1	11	**40**	IFL	**3228**	80.5	200	5.2	0.13	**17.7**	**7.10**
L1C2	12	31	IFL	11.1	0.7	10	0.3	0.01	1.14	0.35
L1C3	13	14	IFL	1.4	0.5	10	0.3	0.02	2.52	0.35
L1C4	14	14	IFL	1.0	0.4	10	0.3	0.02	2.52	0.35
L1A2	22	2	IFL	1.0	0.4	10	0.3	0.13	17.7	0.35
L1B1	25	20	IFL	310.7	7.1	300	7.8	0.39	53.2	10.65
L1B2	26	31	IFL	99.0	5.5	700	18.1	0.58	80.1	24.84
L1B3	27	30	IFL	1.2	0.4	50	1.3	0.04	5.91	1.77
L1B4	28	14	IFL	1.0	0.4	10	0.3	0.02	2.52	0.35
LN12	31	4	IFL	0	0	Ded	2.8	0	0	0
LOI3	32	14	IFL	64.0	4.7	500	13.0	0.93	127	17.75

Understanding z/VM Shares – Absolute and Relative

z/VM CPU Allocation Objective

SET SHARE Objective 1: Provide workload with "enough" CPU

Objective 2: Control looping users (manage performance)

- Manage "excess share"
- Manage share by vCPU

z/VM "Fair Share (Wheeler) Scheduler" controls looping users

- Guarantees "normalized share" to each vCPU
- Really misunderstood, works as designed…

Normalizing z/VM Shares

z/VM virtual machines have a SHARE of the LPAR

- Each Virtual Machine is assigned a **relative** or **absolute** share
- SRMRELDL is the value of "total relative inqueue" **(Very large is bad)**
- Share is then "normalized" to "normalized share"
  - Normalized = absolute
  - Normalized = (relative / (SRMRELDL)) * (100 – absolute)

"Normalized" share is a percent of the LPAR

- Normalized share is the "guarantee"
- Managing "normalized share" manages impact of loopers

Each vCPU has an equal part of the VM normalized share

- Linux process running on a virtual machine vCPU
  - Gets virtual machine vCPU share

Scheduling/Dispatching vCPUs is based on the normalized share

z/VM CPU Allocation

z/VM Shares – Absolute and Relative

- Absolute (ABS) Share is a percent of an LPAR
- Relative (REL) Share is comparable to LPAR “weight”

When to use Absolute vs Relative?

- If share should go up as workload increases (TCPIP, RACF), then use ABS
- If the resource requirement is to be guaranteed, also use ABS
- If share should go down as more users logon, then use REL

Calculation of Normalized Share

- **All ABSOLUTE and RELATIVE shares "normalized"**
  - Sum the Absolute shares of all VMDBKs in Dispatch list (SRMABSDL)
  - Sum the Relative shares of all VMDBKs in Dispatch List (SRMRELDL)

```
Report: ESASUM System Summary
Variable Average Minimum Maximum Description
-------- ------- ------- ------- ----------------------------------------

SRMTSLIC    5.00                 Minor time slice (ms) (SET SRM DSPSLICE)
SRMTSHOT    2.00                 Minor time slice (ms) for HOTSHOT users

SRMABSDL    52.0    48.0    55.0 Total absolute shares of VMDBKs in the dispat
SRMRELDL     818     550    1900 Total relative shares of VMDBKs in the dispat
```

Managing Virtual Processor Distribution

Objective: Provide sufficient CPU to meet workload/prod requirements

Managing Distribution of LPAR entitlement of IFLs

- Based on weight of the LPAR
- Weight is divided by current vCPU in the LPAR
- The more vCPUs, the less entitlement to each vCPU
- Horizontal vs Vertical using HiperDispatch

Managing Distribution – virtual machine share of LPAR

- Share is defined in relative or absolute
- Share is divided over the number of vCPUs
- The more vCPUs, the less entitlement to each vCPU

High Level Scheduler

Fair Share Scheduler (Wheeler scheduler):

- Allows prioritization of work
- Supports 1000's of concurrent virtual machines
- Maintains dispatch list to create fair share
- Allows wide range of workloads to effectively utilize resource

Also called DEADLINE SCHEDULING

- Every inqueue user assigned a deadline

Dispatch List Deadline Priority

Deadline priority is a “target” time of day

- Deadline = TOD + DelayFactor
- Based on ATOD (artificial time of day)

Dispatch list delay factor:

- Based on “Normalized” share
- Delay factor = DSPSLICE / (ncpus * normalized share)
  - 1% share will have 100 time slice delay (500ms)
- Deadline is calculated after every dispatch time slice is completed.

Excess Share Analysis (Oops?)

Starting with 3 looping users RELATIVE 100 share

- They all get equal share of the resources
- This is as we expected

```
Screen: ESAUSP2  Velocity Software-Test VSIVM4  ESAMON 3.778
 1 of 3  User Percent Utilization                CLASS * USER
                                <-------Main Storage-------->
         UserID   <Processor> <Resident->  Lock <-WSSize-->
Time     /Class   Total  Virt Total  Actv   -ed Total  Actv
-------- -------- ----- ----- ----- ----- ----- ----- -----
00:11:00 TSTLNX1  32.39 32.38 15862 15862    11 15536 15536
         TSTLX2   32.12 32.11 66136 66136   259 78478 78478
         TSTLX1   32.02 32.01 38219 38219   176 37790 37790
```

Excess Share Analysis (Broken by Default)

We now give TSTLX2 a RELATIVE 200 share

- Because that is a more important service
- (There is nothing with virtual 2-way)
- Not as expected, it gets the excess share

```
Screen: ESAUSP2  Velocity Software-Test VSIVM4  ESAMON 3.778
1 of 3  User Percent Utilization                CLASS * USER
                                <-------Main Storage-------->
         UserID   <Processor> <Resident->  Lock <-WSSize-->
Time     /Class   Total  Virt Total  Actv   -ed Total  Actv
-------- -------- ----- ----- ----- ----- ----- ----- -----
00:14:00 TSTLX2   68.71 68.68 66211 66211   258 78478 78478
         TSTLX1   14.00 14.00 38245 38245   256 37790 37790
         TSTLNX1  13.99 13.99 15879 15879    11 15536 15536
```

Excess Share Analysis

- **Now for the experiment – Set shares "correctly", perfect results**
  - Convert TCPIP, SSL, RACF from REL 3000 to ABS 2%
  - This ELIMINATES "EXCESS" bucket – looping users are now controlled

```
Screen: ESAUSP2  Velocity Software-Test VSIVM4  ESAMON 3.778
1 of 3  User Percent Utilization                CLASS * USER
                                  <-------Main Storage-------->
          UserID   <Processor> <Resident->  Lock <-WSSize-->
Time      /Class   Total  Virt Total  Actv   -ed Total  Actv
-------- -------- ----- ----- ----- ----- ----- ----- -----
00:20:00 TSTLX2   48.39 48.37 67141 67141   292 80047 80047
         TSTLNX1  24.19 24.19 16168 16168    11 15536 15536
         TSTLX1   24.19 24.18 39006 39006   241 37790 37790
```

Calculation of Normalized Share

- **If SRMABSDL is less than 100%**
  - Normalized share equals Absolute Share
  - Relative Share users get:

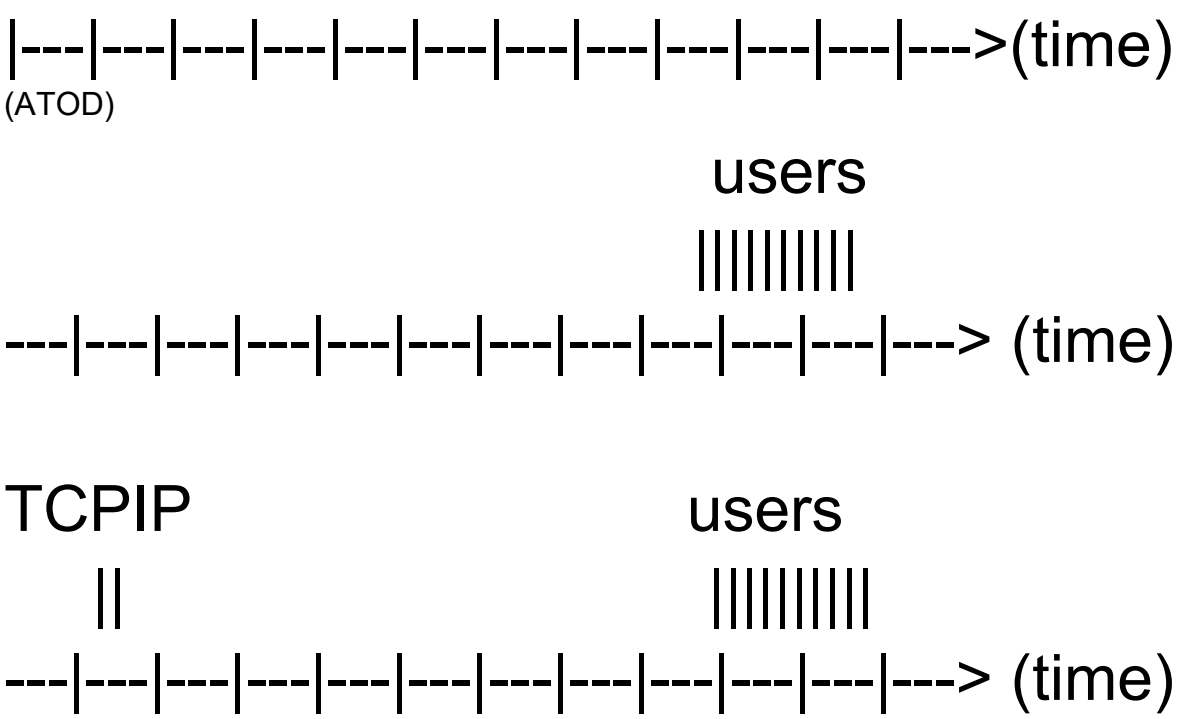
    (100 - SRMABSDL) x (relative share / SRMRELDL)
- **If SRMABSDL is greater than 99,**
  - Absolute shares “normalized” to 99
  - Relative users “share” 1 percent
  - Very dangerous situation

- **Normalized shares are percentages of the CPU resource**
- **Delay factor (OFFSET) is then DSPSLICE / “normalized” share**

Setting Deadline

Deadline time of day = current TOD + offset

Offset = (DSPSLICE / Normalized share) * bias



Dispatcher takes users in order by time from sorted deadline list

SHARE Impact on CPU Delivery Rate

- **CPU Delivery Rate for "one CPU system"**
- **If normal share is 10%, user will have:**
  - Delivery rate = 1 dispatch time slice out of 10
  - Offset = 10 dispatch time slices
- **If normal share is 50%, user will have:**
  - Delivery rate = 1 dispatch time slice out of 2
  - Offset = 2 dispatch time slices
- **If normal share is 1%, user will have:**
  - Delivery rate = 1 dispatch time slice out of 100
  - Offset = 100 dispatch time slices

Sample Deadlines - Comparison

Example 1:

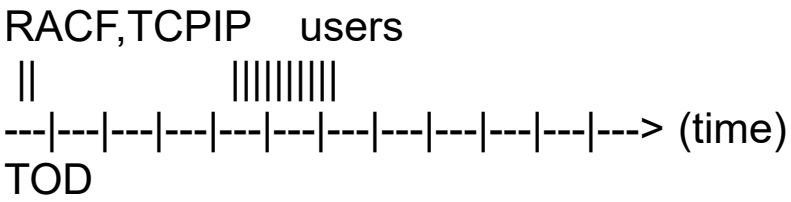
TCPIP offset 2.5 dspslice (Share 3000)

Users offset 250 dspslice (1.25 seconds)

Example 2: Change TCPIP/RACF share to ABS 10

TCPIP offset 5 dspslice

Users offset 84 dspslice (.42 seconds)



Sample Deadlines - Comparison

Did it make a difference to RACF/TCPIP to reduce share?

- NO. Still number one always on dispatch list

Did it make a difference to users?

- Yes. They are guaranteed 3 times the amount of CPU when looping users are on the system

Does setting shares too high for some users impact other users?

- Only when large CPU consumers (including loopers) exist
- IBM does not let looping users on their benchmark systems

Recommend low ABS shares when appropriate for servers

Measuring CPU Consumption

Processor Measurements – User View

```
Report: ESAUSP2       User Resource Rate Report
Monitor initialized: 05/06/08 at 12:00:00 on 2094 serial
-------------------------------------------------------------

         <---CPU time--> <----Main Storage (pages)----->
UserID   <(Percent)> T:V <Resident> Lock <-----WSS----->
/Class    Total  Virt Rat Totl Activ  -ed Totl Activ  Avg
-------- ----- ----- --- ---- ----- ---- ---- ----- ----
12:01:00 369.9 361.0 1.0  17M   17M  417  17M   17M 129K
 ***User Class Analysis***
*Servers  1.95  1.72 1.1 7566  7555   49 8674  7444  207
*Linux   184.0 180.6 1.0  15M   15M  305  15M   15M 185K
*Misc    183.7 178.5 1.0   2M 1642K   11   2M 1642K 328K
 ***Top User Analysis***
LXPWK001 183.5 178.4 1.0   2M 1641K    3   2M 1641K   2M
LXWKB215 37.63 37.01 1.0 782K  782K    1 782K  782K 782K
LXWKB211 33.97 33.88 1.0 514K  514K    0 514K  514K 514K
LXWKB210 17.64 17.55 1.0 298K  298K    2 298K  298K 298K
LXWKB214 16.86 16.68 1.0   1M 1188K    0   1M 1254K   1M
LXWKB228  6.01  5.98 1.0 731K  731K    3 731K  731K 731K
LXWKB222  5.06  4.94 1.0 621K  621K    5 621K  621K 621K
LXWKB183  4.70  4.57 1.0 231K  231K    0 230K  230K 230K
LXWKB220  3.69  3.66 1.0 125K  125K    8 124K  124K 124K
LXWKB225  3.65  3.52 1.0 780K  780K    0 780K  780K 780K
ESATCP    0.45  0.35 1.3 1038  1038    1 1037  1037 1037
TCPIP2    0.02  0.01 2.0 1142  1142   48  198   198  198
```

Measure CPU “thread” consumption:

- **ESAUSP2 (Traditional)**
- CPU Consumption (Percent)
  - Total of all users
  - By user
  - By class

Note:

- One server dominates the CPU

T:V Ratio is Total to Virtual

- Very traditional metric
- 1.0 is best / normal for Linux

Also Measure by vCPU View

Linux balances small tasks across vCPUs

Each vCPU waits ("wait time") and then does a small task

Having too many vCPUs just means the number of delays increases

Report: **ESAUSCP** Virtual Machine VCPU Analysis

UserID	<---CPU time-->						<---Percent		
CPUvadd	Cnt	<-Percent-> TOT	Virt	<-SHARE--> Type	Value	CPU TYPE	<-Samples-> Total	In Q	**Run**	**Sim**	**CPU**
17:01:00	0	430.6	425.0	.	.	.	3480	1682	18	0.2	21
LIMEDM1	4	**61.53**	59.64	REL	400	IFL	240	240	**15**	**1.3**	**36**
CPU-00		15.89	15.26	REL	100	IFL	60	60	17	1.7	35
CPU-01		13.77	13.41	REL	100	IFL	60	60	8.3	3.3	38
CPU-02		17.69	17.19	REL	100	IFL	60	60	15	0	32
CPU-03		14.18	13.78	REL	100	IFL	60	60	18	0	38
LIMEDI1	6	**49.66**	48.93	REL	600	IFL	360	360	5.3	0	8.9
CPU-00		8.85	8.67	REL	100	IFL	60	60	10	0	10
CPU-01		8.25	8.13	REL	100	IFL	60	60	6.7	0	6.7
CPU-02		7.24	7.11	REL	100	IFL	60	60	1.7	0	10
CPU-03		8.47	8.36	REL	100	IFL	60	60	1.7	0	8.3
CPU-04		10.13	10.04	REL	100	IFL	60	60	8.3	0	10
CPU-05		6.73	6.61	REL	100	IFL	60	60	3.3	0	8.3

Linux: How many Virtual Processors?

Linux is multiprocessor capable, thus requiring LOCKs

Global SPIN lock is large issue (changed to many locks)

- One virtual processor acquires lock
- Other virtual processors attempt to spin
- On 390 – spin converted to Diagnose 44 (now 9C), vCPU waits

Linux vCPU – Why Does it Hurt?

Too many vCPUs hurts for so many reasons:

- DIAG 9C overhead -> dispatch overhead
- Linux balances across all vCPUs
- Pollutes hardware cache – **cache impacts performance**
- vCPUs will wait for each other
- vCPUs will wait to be dispatched, but will do little work

Problem easily detected

- High Diagnose -> Instruction Simulation -> SIE
- High T:V ratio (Overhead to real work too high)
- Guideline: Minimize virtual processors

Affinity

Affinity Processing

What is Affinity?

- Virtual CPU will be dispatched on the same thread/CPU
- Theory: Reuses existing data in the hardware cache
- Theory: Reduces overhead and delays in re-loading cache
- Understanding this will pay your way to this free conference
- Reality: Probably good for z/OS
- Reality: Linux and TPF seriously break this model

How it works:

- Virtual CPUs are assigned to a PLDV - “Processor Local Dispatch Vector”
- Virtual CPUs stay assigned to the PLDV unless stolen
- Stealing is delayed 50 milliseconds to encourage affinity

Affinity Case Study Performance Perspective

ESALPARS

VML1 06 2 IFL 200.1 0.0 **Ded**

Report: **ESAXACT** Transaction Delay Analysis

UserID /Class	<-Samples-> Total	In Q	Run	Sim	**CPU**	SIO	Pag	<-----Percent non-dormant (Wait E- SVM	D- SVM	T- SVM
03/13/23										
Hi-Freq:	3180	1576	8.3	0.1	12	0.1	0	0	8.0	0.6
Hi-Freq:	3180	1580	6.2	0.1	5.1	0	0	0	10	0.8
Hi-Freq:	3180	1572	4.1	0.1	2.8	0	0	0	12	0.6
Hi-Freq:	3180	1575	4.3	0.2	3.0	0	0	0	14	0.6
Hi-Freq:	3180	1579	5.2	0.1	5.1	0	0	0	8.7	0.6
16:06										
Hi-Freq:	3180	1569	**7.3**	**0.1**	**10**	0	0	0	10	0.7
Hi-Freq:	3180	1568	6.8	0.1	8.4	0	0	0	10	0.4
Hi-Freq:	3180	1568	5.4	0.3	8.9	0	0	0	10	0.4

Why is CPU Wait high?

- High CPU Utilization?
- Only 2 IFLs dedicated to LPAR?

Affinity Case Study CPU Perspective

Why is CPU Wait high? (Apply a little statistical analysis)

- High CPU utilization? NO, 55% per thread
- Very basic Queuing theory MM1 – 1 server, 1 queue, 50% time in queue
- Queuing theory – 4 servers, 1 queue, (MM4) Should be 6% time in queue
- Affinity forces vCPU to stay on the PLDV (single queue / server model)

```
Report: ESACPUU       CPU Utilization Report
Monitor initialized: 03/13/23 at 16:00:00 on 3906 serial 06FCC8
-------------------------------------------------------------------
         <----Load---->            <--------CPU (percentages)----
         <-Users-> Tran     CPU    Total  Emul  User   Sys  Idle
Time     Actv In Q /sec CPU Type    util  time ovrhd ovrhd  time
-------- ---- ---- ---- -  ----   ----- ----- ----- ----- -----
03/13/23
-------------------------------------------------------------------
16:06:00   28 26.0  0.5  0  IFL     54.9  50.2   1.9   2.8  45.0
                         1  IFL     55.1  51.5   1.9   1.6  44.9
                         2  IFL     54.0  50.7   1.5   1.8  45.9
                         3  IFL     53.8  50.5   1.6   1.6  46.2
                                    ----- ----- ----- ----- -----
System:                             217.8 203.0   6.9   7.9 182.0
```

Linux vCPU – Why Does it Hurt?

Customer Performance Critsit:

- Customer has excess processor capacity
- Bad performance (“CPU WAIT”- ESAXACT)
- Totals by Processor type on box (25% utilization - ESALPARS)

```
Totals by Processor type:
<---------CPU-------> <-Shared Processor busy->
Type Count Ded shared  Total  Logical Ovhd Mgmt
---- ----- --- ------ ------ -------- ---- ----
IFL     70   0     70 1706.9   1628.8 30.1 48.0c
```

Linux vCPU – Why does it Hurt?

Linux balances small tasks across vCPUs

Each vCPU waits ("wait time") and then does a small task

Having more vCPUs just means the number of delays increases

Report: **ESAUSCP** Virtual Machine VCPU Analysis

UserID	<---CPU time-->						<---Percent		
CPUvadd	<-Percent->		<-SHARE-->		CPU	<-Samples->			
	Cnt	TOT	Virt	Type	Value	TYPE	Total	In Q	**Run**	**Sim**	**CPU**
17:01:00	0	430.6	425.0	.	.	.	3480	1682	18	0.2	21
LIMEDM1	4	**61.53**	59.64	REL	400	IFL	240	240	**15**	**1.3**	**36**
CPU-00		15.89	15.26	REL	100	IFL	60	60	17	1.7	35
CPU-01		13.77	13.41	REL	100	IFL	60	60	8.3	3.3	38
CPU-02		17.69	17.19	REL	100	IFL	60	60	15	0	32
CPU-03		14.18	13.78	REL	100	IFL	60	60	18	0	38
LIMEDI1	6	**49.66**	48.93	REL	600	IFL	360	360	5.3	0	8.9
CPU-00		8.85	8.67	REL	100	IFL	60	60	10	0	10
CPU-01		8.25	8.13	REL	100	IFL	60	60	6.7	0	6.7
CPU-02		7.24	7.11	REL	100	IFL	60	60	1.7	0	10
CPU-03		8.47	8.36	REL	100	IFL	60	60	1.7	0	8.3
CPU-04		10.13	10.04	REL	100	IFL	60	60	8.3	0	10
CPU-05		6.73	6.61	REL	100	IFL	60	60	3.3	0	8.3

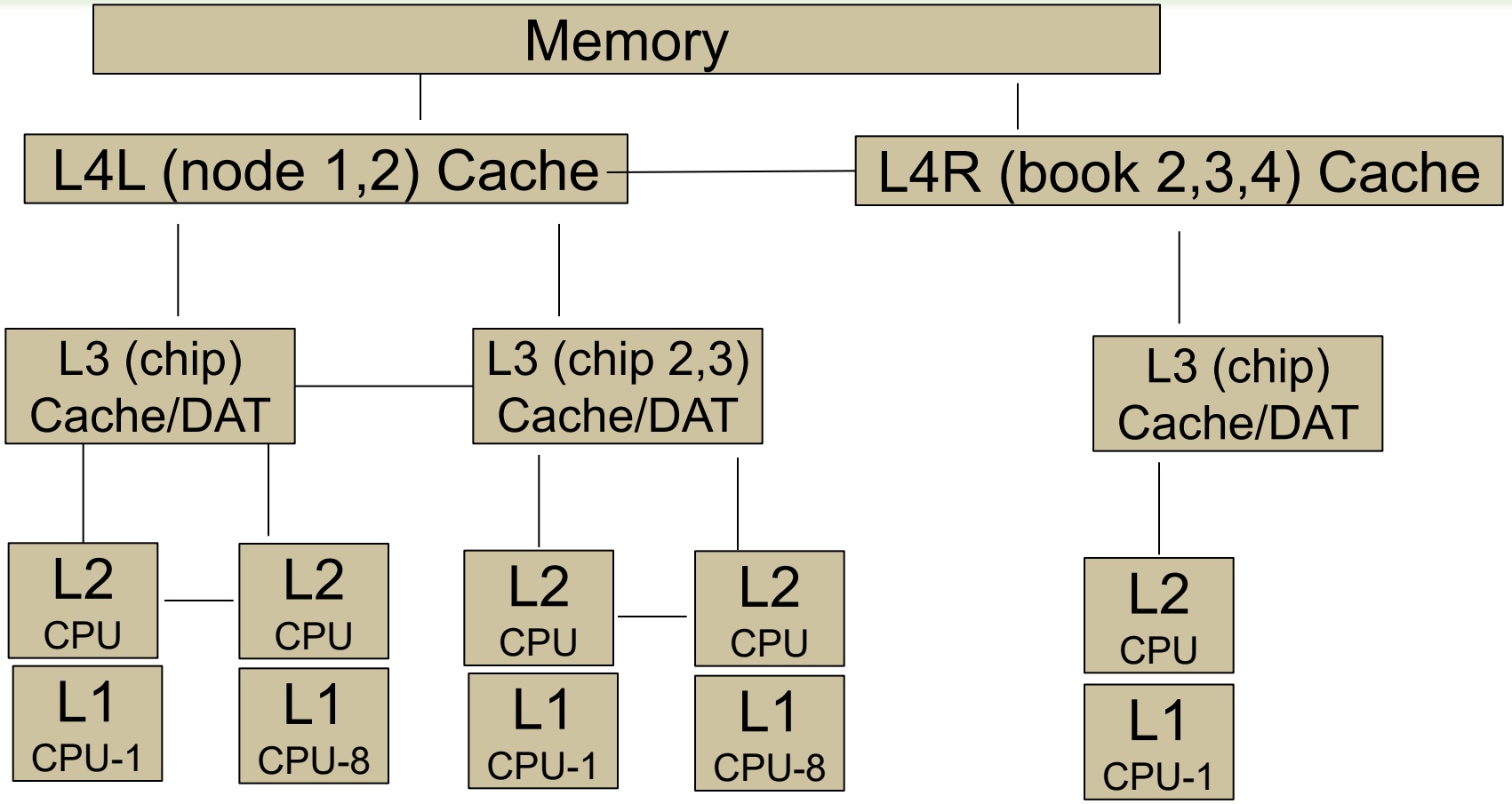
LINUXM1:

- Polling at 15,000 times / second (each “poll” waits for CPU – 15,000 / second)
- Dispatching every 70 microseconds
- Using 60% of one CPU
- 36 CPU seconds / 15,000 / 60 = 40 microseconds / dispatch

```
Report: ESAUSR3
--------------------
           <Dispatch>
UserID     <Rate/Sec>
/Class     Disp Waits
--------   ---- -----
17:01:00    45K 45271
LIMEDM1     15K 14578    (4 vCPU)
LIMEDI1    6261  6261    (6 vCPU)
```

Z Architecture – "Affinity"

Affinity keeps vCPU:CPU
Affinity keeps data intact?
Linux moves work to vCPU
Pollutes cache?
So why affinity?
Why many vCPU?



Affinity was meant to better utilize Z caching structure (Linux breaks that)

CPU Guidelines

But wait, there is more…

If 8 threads (4 cores) at 60% busy, queueing theory says:

- CPU Queue time should be: .017 time running
- Why is it 2 times running?

Problem: CPU Affinity processing delays 50ms before steal

- 85% of the time there is at least 1 available CPU – blocked by affinity
- The secret command undocumented (yet), used often (z/VM 7.5)

Decision time? (Based on response time needs vs capacity needs)

- Keep Affinity and increase CPU Wait
- No affinity, and decrease cache effectiveness
- Reduce nvcpu and no affinity is best option

Modlevels – "Secret" Command…

```
q syscontrol
DISPATCH THDAFFINITY ON
DISPATCH PREEMPTLOCAL OFF
DISPATCH TSEARLY  50
DISPATCH INCHIPBUSY     50000
DISPATCH INCHIPDELAY    50000
DISPATCH INNODEBUSY    100000
DISPATCH INNODEDELAY   100000
DISPATCH INSYSBUSY     200000
DISPATCH INSYSDELAY    200000
Ready; T=0.01/0.01 11:24:20

CP SET SYSCONTROL DISPATCH MODLEVEL 0
Ready; T=0.01/0.01 11:24:24
q syscontrol
DISPATCH THDAFFINITY OFF
DISPATCH PREEMPTLOCAL ON
DISPATCH TSEARLY   0
DISPATCH INCHIPBUSY         0
DISPATCH INCHIPDELAY        0
DISPATCH INNODEBUSY     50000
DISPATCH INNODEDELAY    50000
DISPATCH INSYSBUSY     200000
DISPATCH INSYSDELAY    200000
Ready; T=0.01/0.01 11:24:27

CP SET SYSCONTROL DISPATCH MODLEVEL 1…
```

If in CPU Wait, but have excess capacity, less "affinity" is enforced when set to MODLEVEL 0

Also, setting MODLEVEL 0 eliminates "steal time"

Default in z/VM 7.5

Modlevels Secret Command... – Turn Off Affinity

```
q syscontrol
DISPATCH THDAFFINITY ON
DISPATCH PREEMPTLOCAL OFF
DISPATCH TSEARLY  50
DISPATCH INCHIPBUSY     50000      ←Delay for steal on chip
DISPATCH INCHIPDELAY    50000
DISPATCH INNODEBUSY    100000      ←Delay for steal on node
DISPATCH INNODEDELAY   100000
DISPATCH INSYSBUSY     200000      ←Delay for steal on system
DISPATCH INSYSDELAY    200000
Ready; T=0.01/0.01 11:24:20
CP SET SYSCONTROL DISPATCH MODLEVEL 0
Ready; T=0.01/0.01 11:24:24
q syscontrol
DISPATCH THDAFFINITY OFF
DISPATCH PREEMPTLOCAL ON
DISPATCH TSEARLY   0
DISPATCH INCHIPBUSY         0
DISPATCH INCHIPDELAY        0
DISPATCH INNODEBUSY     50000
DISPATCH INNODEDELAY    50000
DISPATCH INSYSBUSY     200000
DISPATCH INSYSDELAY    200000
Ready; T=0.01/0.01 11:24:27
CP SET SYSCONTROL DISPATCH MODLEVEL 1...
```

Setting SYSCONTROL to 0 also alleviates "steal time"

Modlevels Secret Command... 2 – Turn Off Affinity

```
SYSCONTROL set to MODLEVEL 0
Ready; T=0.01/0.01 11:24:24
q syscontrol
DISPATCH THDAFFINITY OFF
DISPATCH PREEMPTLOCAL ON
DISPATCH TSEARLY   0
DISPATCH INCHIPBUSY         0
DISPATCH INCHIPDELAY        0
DISPATCH INNODEBUSY     50000
DISPATCH INNODEDELAY    50000
DISPATCH INSYSBUSY     200000
DISPATCH INSYSDELAY    200000
Ready; T=0.01/0.01 11:24:27
CP SET SYSCONTROL DISPATCH STEALBARRIER INCHIPBUSY 0 INCHIPDELAY 0
INNODEBUSY 0 INNODEDELAY 0 INSYSBUSY 0 INSYSDELAY 0
q syscontrol
DISPATCH THDAFFINITY OFF
DISPATCH PREEMPTLOCAL ON
DISPATCH TSEARLY   0
DISPATCH INCHIPBUSY         0
DISPATCH INCHIPDELAY        0
DISPATCH INNODEBUSY         0
DISPATCH INNODEDELAY        0
DISPATCH INSYSBUSY          0
DISPATCH INSYSDELAY         0
```

SMT

SMT Theory

SMT is about using unused cycles

If one thread

- Cycles wasted waiting for L1/L2 cache update
- Cycles wasted waiting for DAT (Dynamic Address Translation)

If two threads

- Wasted cycles are used by an alternate thread
- If there is contention for cache or DAT, work takes longer!
- Is there an increase in capacity?
- What is the performance impact?

SMT Theory

SMT Objective:

- Increases capacity (maybe at the cost of performance / response time)
- Better core utilization (more cycles for real work)

In theory: Processor cycles are sitting idle

- To execute an instruction, L1 cache is populated (data/instruction)
- Cycles wasted while L1 cache is loaded from L2/L3/L4/Memory
- SMT uses “wasted” cycles for another “thread”

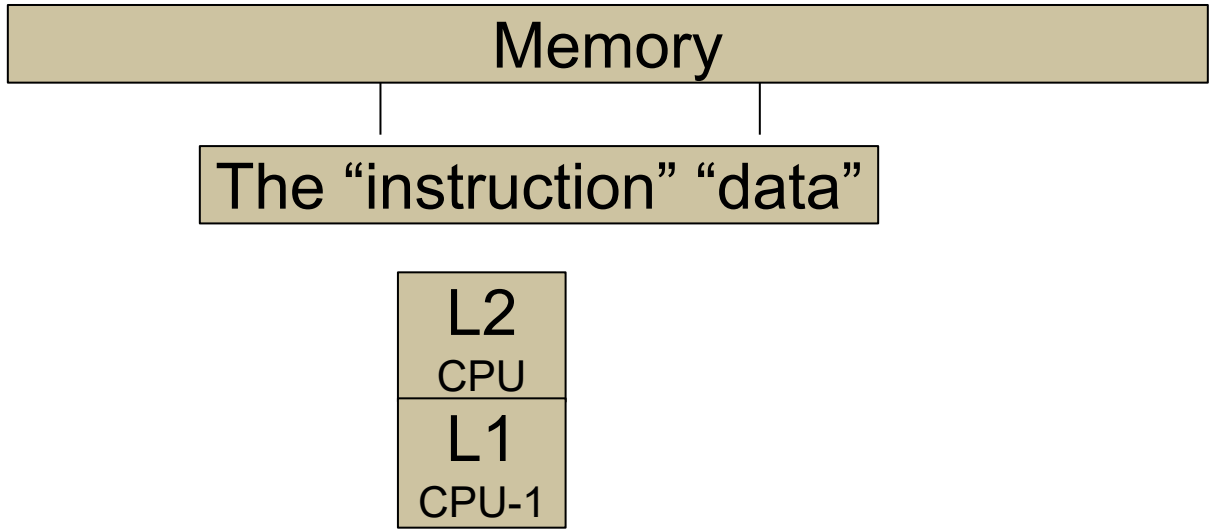
In practice: (Linux broke everything)

- Two threads share one core – and cache
- More processes share core – and cache
- Cache has more contention (always reloaded)
- Core has contention
- Wasted cycles are now being used

SMT Theory – Why Cache is important

Hardware – the way it works

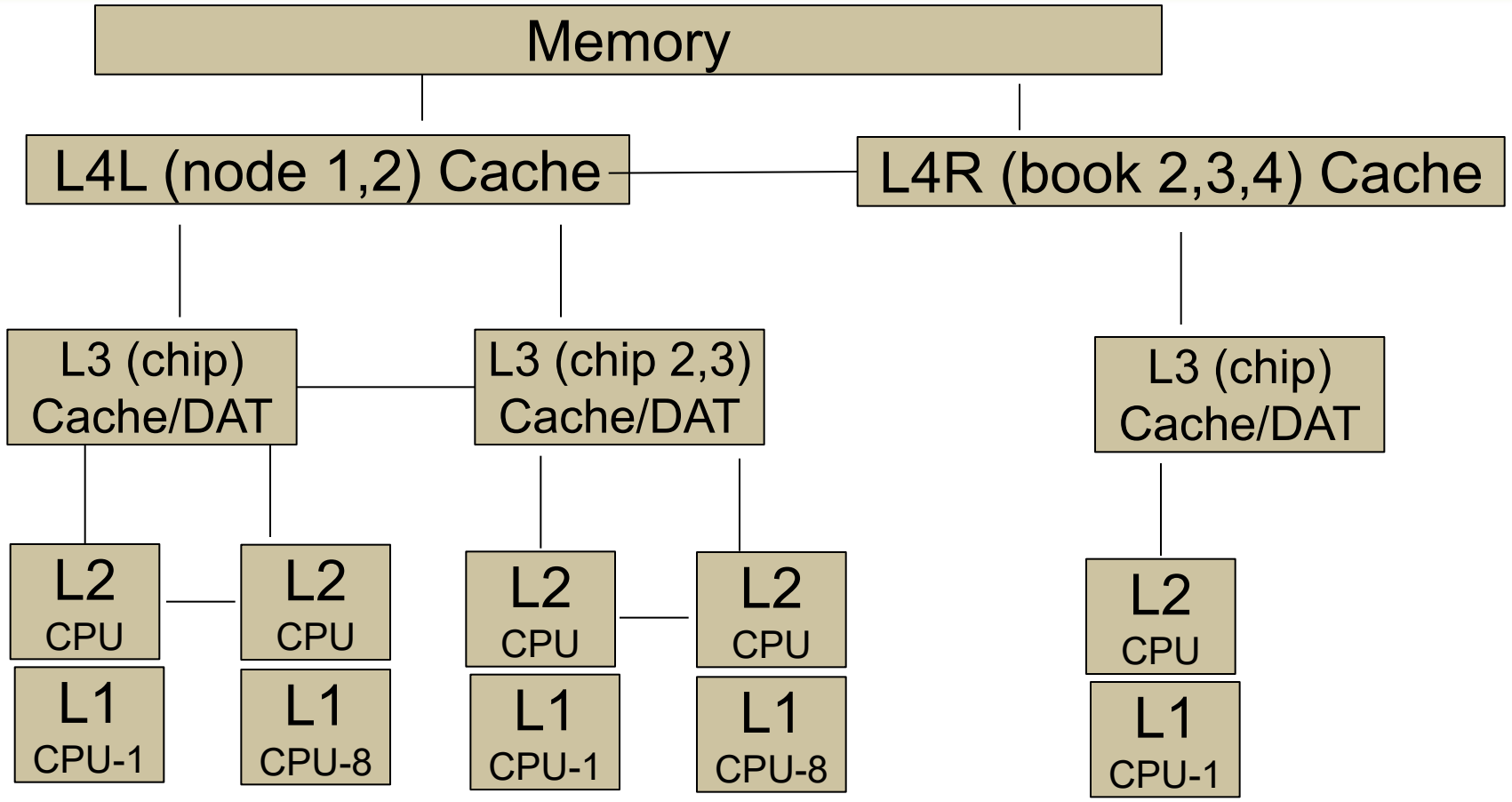
- Instructions executed on the CPU, from the L1 Cache
- Includes data (to/from locations)
- Same on INTEL
- IBM z Implements levels of cache



z/VM Challenges, Linux Breaks the Theory

- **SMT on z/VM had challenges**
  - Why is SAP/Oracle better for SMT? (z13)
  - A 30% ITR (Internal Throughput Rate) improvement with SMT in SAP production LPAR
- **Dispatching 30,000 times per second on one thread**
  - How long is the task on CPU? (<30 microseconds)
  - 30 microseconds -> 15,000 cycles – 5k instructions?
  - How long does data remain in L1/L2 cache?
  - The more references further out, the worse things get
- **Relative Nest Intensity – RNI (John Burg, WSC)**
  - Provides relative wait times
  - Smaller means less time waiting for cache to be loaded

Why SMT?



For instructions to execute in CPU, must be in L1 cache
It takes time to load cache, no instructions execute during load

Cycle Requirements Per Source

SMT Analysis starts with Cycles Per Instruction

- CPU clocked at 5.2GHz, so 5.2 Billion CYCLES per second
- Instructions take 1-n cycles (Less when pipelined)
- If data/instructions not in L1 cache, they get loaded (1-30 cycles wasted)
- Capacity is based on instructions executed
- Using “wasted” cycles for real instructions increases capacity

- z13 was limited by address translation taking 30-50% of the cycles
- z14+ has 4 DAT units per core

- Objective is more instructions per given cycle

Cycle Requirements Per Source

What happens to RNI when a 2nd thread is added? (z13)

- Relative Nest Intensity derived by John Burg, WSC for z/OS as capacity measure
- Average CPI (Cycles Per Instruction) went from 1.25 to 1.40 (so some degradation)
- Average RNI (Relative Nest Intensity) went from .55 to .66 (cache contention)

Report: **ESAMFCA** MainFrame Cache Magnitudes R

		<CPU Busy>		<-------Processor------>				**RNI**	
		<percent>		Speed/<-Rate/Sec->			**CPI**	From	
Time	CPU	Totl	User	Hertz	Cycles	Instr	**Ratio**	**Burg**	
09:47:00	0	10.9	10.6	5208M	569M	454M	1.254	0.53	
09:48:00	0	11.9	11.6	5208M	621M	523M	1.187	0.42	
09:49:00	0	9.3	9.0	5208M	487M	385M	1.265	0.56	
09:50:00	0	9.5	9.2	5208M	497M	391M	1.270	0.54	
09:51:00	0	9.5	9.1	5208M	497M	380M	1.309	0.65	
09:52:00	0	10.0	9.5	5208M	520M	373M	1.394	0.62	←SMT Enabled
09:53:00	0	11.2	10.8	5208M	587M	448M	1.312	0.48	
09:54:00	0	9.8	9.3	5208M	512M	365M	1.403	0.68	
09:55:00	0	10.5	10.0	5208M	550M	390M	1.411	0.66	

CPU Measurement Facility with SMT

- Cycles by thread (total cycles used for both work and wait)
- Shows cycles used and instructions executed (thread CPI)
- Core CPI went down (instructions/capacity went up)
- Meaningful instructions per second – total (3.33G)
- Real cycles per instruction: 88% of 5208M / 3330M (1.37)

```
Report: ESAMFC          MainFrame Cache Magnitudes
Monitor initialized: 06/17/20 at 21:23:09 on 390
-----------------------------------------------
                <CPU Busy><-------Processor------>
                <percent> Speed/<-Rate/Sec->
Time      CPU Totl User Hertz Cycles Instr Ratio
-------- --- ---- ---- ----- ------ ----- -----
21:25:02   0 88.4 74.5 5208M  4607M 1652M 2.789 -> 1.37
           1 88.6 76.9 5208M  4617M 1678M 2.752
```

Back to – What is a CPU second?

- We charge for CPU seconds?
- Is it consistent? No!
- How much does it vary? (in instructions per second)
- Dependent on workload (cache residency)
- If more contention for cache, more time is spent waiting

SMT Data Points

System data points – hardware perspective

- Core time allocated to LPAR (both threads at a time)
- Thread busy vs thread idle (potential capacity)
- Instructions per second per core
- Cycles per instruction (low is good)

User data points

- Recognize core time vs thread time
- Change in thread time (response time)
- Change in cycles consumed (capacity)
- Does the data agree?

SMT – When to use it?

SMT was announced on z13 without much guidance

Some installations said "good stuff"

- Oracle, SAP workloads

Others said "not so good…"

- Java, Websphere workloads

The question is why?

And why is z14 (and up) so much better?

Does SMT provide more capacity?

Which approach is designed for the higher volume of traffic? Which road is faster?

**Illustrative numbers only*

- **Measurement:**
  - ”Person miles”?
  - Per Minute?
- **Add lanes and…?**

Does SMT Provide Contention?

Which approach is designed for the higher volume of traffic? Which road is faster?

**Illustrative numbers only*

- **(z/13) Not always faster…**

Capture Ratios – z/VM – NO SMT

Compare LPAR (SYTCUP) to z/VM (SYTPRP): Capture 99%

- CPU by CPU comparison accurate when SINGLE THREAD
- Some scheduling time likely lost

```
ESACAPT       Logical Partition Analysis
-------------------------------------------------------------------
 <----Logical Processor---> <---CPU (percentages----> Capture%
 VCPU  CPU <---%Assigned--> Total  Emul  User   Sys   LPAR
 Addr Type Total Ovhd  Emul  util  time ovrhd ovrhd
 ---- ---- ----- ---- ----- ----- ----- ----- ----- - -----

    0  IFL  15.7  0.5  15.2  14.9  12.0   1.3   1.6    0.98
    1  IFL  18.8  0.5  18.3  17.9  16.0   1.5   0.5    0.98
    2  IFL  20.7  0.4  20.3  20.0  18.1   1.4   0.5    0.98
    3  IFL  25.1  0.4  24.7  24.4  22.5   1.5   0.4    0.99
    4  IFL  27.2  0.4  26.8  26.5  24.6   1.4   0.5    0.99
    5  IFL  38.4  0.4  38.0  37.7  35.5   1.7   0.6    0.99
    6  IFL  64.8  0.6  64.3  64.0  60.4   2.8   0.8    1.00
    7  IFL   1.1  0.2   0.9   0.7   0.1   0.1   0.5    0.76
    8  IFL   0.8  0.0   0.7   0.7   0.6   0.0   0.1    0.95
 ---- ---- ----- ----- -----   ----- ----- ----- ----
Total  IFL 212.6  3.3 209.3 206.9 189.8  11.6   5.4 6  0.99
```

SMT Base Lines

Processor Utilization – No SMT

- Numbers agree and make sense
- Can capture virtual machine resources and believe it
- Have value for overheads

SMT Challenges

- Virtual machines share the CPU/core
- The more they share, the slower they go? (in theory, how slow?) BUT WAIT LESS
- Numbers are likely not repeatable based on workload
- How much added capacity with SMT for YOUR workload?
- How do you charge?
- (You must charge for “REAL” consumption)

Processor Capacity Planning Concepts

Processor utilization – What level is target?

- Performance – what level of performance is required?
- What level of performance management is required? Is available?
- Capacity Planning – what utilization level is needed financially?

Customer targets

- Target based on performance?
- 80% and higher hardware utilization requires management
- 50% CPU minimizes CPU queue – better performance – tradeoff
- Higher utilization is better financially

Capacity planning objective

- Provide resources to get work done in a timely fashion
- Meeting appropriate financial and performance objectives

Processor Measurement Concepts - Utilization

What is “CPU Utilization”? Need to agree on this first?

All zVPS numbers are measured in CPU seconds

- Percent is always based on CPU seconds divided by wall clock
- What is a CPU second if there are two threads with SMT?

SMT Impacts the measurements of:

- LPAR (percent of processor assigned to the partition)
- z/VM Virtual Machines (percent of “thread” assigned to a virtual machine)
- Linux processes (percent of a vCPU)

BUT DO WE AGREE ON WHAT IS IMPORTANT?

- Is it processor utilization?
- Or work completed?

Capacity Benefits?

SMT adds how much capacity?

- How much more throughput?
- Workload dependencies
- How to predict

Z13/14/15/16/17 have larger cache sizes

- How long does cache last when 30,000 dispatches per second per processor?
- How much does enabling SMT impact cache?

Capacity Planning Thoughts

How much used capacity at the CEC level?

- Total IFL (Assigned) Utilization (ESALPARS/ESALPMGS)
- Totals by Processor type
- Shared processor total busy

<----------CPU-------->				<-Shared Processor busy->				
Type	Count	Ded	shared	Total	Logical	Ovhd	Mgmt	
CP	11	0	11	892.1	865.2	11.2	15.7	
IFL	**37**	6	**31**	**2466.7**	2412.0	30.9	23.8	**← 80% utilization**

z/VM: One core – Two threads

- "Assigned" – 933.7% or 4.1%
- Two threads are not always both active –> thread idle time
- Subtract 138% thread idle -> (933% - 4) * 2 – 138% = **1720**% thread time (z/VM time)
- (Thread idle time is likely available capacity)

			<--------Logical Partition-------->					
			Virt	CPU	<%Assigned>		<-Thread->	
Time	Name	Nbr	CPUs	Type	Total	Ovhd	Idle	cnt
21:25:00	Totals:	00	27	CP	876.3	11.2		
	Totals:	00	54	IFL	2443	30.9		
	ZVMQAXX	0B	**14**	IFL	**933.7**	4.1	**138.1**	2 ←

Capacity Planning Thoughts

Report: **ESACPUU** CPU Utilization Report

	<----Load---->					<--------CPU (percentages)-------->					
	<-Users->	Tran		CPU	Total	Emul	User	**Sys**	Idle	Steal
Time	Actv	In Q	/sec	CPU	Type	util	time	ovrhd	**ovrhd**	time	time
21:25:00	194	399	0.5	0	IFL	88.4	74.5	1.7	12.2	10.5	1.1
				1	IFL	88.6	76.9	1.9	9.8	10.3	1.1
				2	IFL	89.2	77.7	2.4	9.1	9.7	1.1
				3	IFL	89.2	77.7	1.5	10.1	9.6	1.2
				4	IFL	89.6	78.0	1.7	9.9	9.4	1.0
				5	IFL	89.1	77.7	2.3	9.1	9.9	1.1
				22	IFL	67.2	58.5	1.5	7.1	11.6	21.2
				23	IFL	66.8	58.4	1.4	6.9	12.0	21.3
				24	IFL	74.9	66.4	1.5	7.0	13.9	11.1
				25	IFL	74.4	66.3	1.6	6.5	14.4	11.2
				26	IFL	76.4	68.3	1.3	6.8	12.6	10.9
				27	IFL	75.6	68.2	1.6	5.8	13.4	11.0
System:						**1709**	**1499**	**36.2**	173.6	332.2	759.1

Thread time validates

How much used capacity in the z/VM LPAR?

- Total IFL Utilization (ESACPUU) 1,709% (capture 99%+)
- User billable – Traditional: (1499 + 36) – 1,535%?
- Steal time: Physical processor stolen

SMT – ~~Not~~ Always a Good Thing?

Workload helped by SMT? Is monitor user data valid? No….

- 1535% “thread time” (validated against CPU busy)
- 1192% core time
- 1051% “would be” time
- Used 1192% - could have been 1051
- Based on IBM monitor data, less capacity because of SMT? (13%)
- But the hardware said 933% assigned and that data is validated
- And still there is thread idle – how to account for that?

```
Report: ESAUSP5       User SMT CPU Consumption Analysi
Monitor initialized: 06/17/20 at 21:23:09 on 3906 ser
-------------------------------------------------------
          <------CPU Percent Consumed   (Total)---->
UserID    <Traditional> <MT-Equivalent> <MT Prorated>
/Class    Total  Virt   Total  Virtual  Total Virtual
--------  -----  -----  -----  -------  ----- -------
21:25:00   1535   1499   1051     1026   1192    1163
```

Processor Measurements for SMT Validity

ESAUSR5/ESAUSP5 show IBM's SMT user data

- Traditional: Thread time (response time)
- Equivalent: Time it would take if non-SMT
  - (**PERFORMANCE ratio 1051 / 1535**) – 50% slower
- Prorated: Cycles really used (approximate/prorated)
  - (**Capacity and Chargeback**)
  - Want to charge for 933% (physical assigned time to LPAR)
  - Prorated metrics are too high (1192 vs 933)

```
<------CPU Percent Consumed    (Total)---->
UserID   <Traditional> <MT-Equivalent> <MT Prorated>
/Class   Total  Virt   Total  Virtual  Total Virtual
-------- ----- -----   -----  -------  ----- -------
21:25:00  1535   1499   1051     1026   1192    1163
```

Processor Measurements – Valid Data?

ESAUSP5:

- CPU Percent Consumption:
  - Total for all users
  - By User
  - By Class

LPAR assigned time: 933.7%

z/VM thread assigned time: 1720%

User time: (1499 + 36 = 1535)

- Traditional (thread) measurements are valid
- 100% capture ratio
- IBM SMT prorated numbers 30% off?
- New “Velocity” Prorate

```
Report: ESAUSP5       User SMT CPU Consumption Analysis
Monitor initialized: 06/17/20 at 21:23:09 on 3906 seri
-------------------------------------------------------
          <------CPU Percent Consumed    (Total)---->
UserID    <Traditional> <MT-Equivalent> <MT Prorated>
/Class    Total  Virt    Total  Virtual  Total Virtual
-------- ----- -----    -----  -------  ----- -------
21:25:00  1535   1499    1051     1026   1192    1163
 ***User Class Analysis***
Servers    0.04   0.00    0.03     0.00   0.03    0.00
ZVPS       2.38   1.57    1.66     1.04   2.14    1.37
TheUsers   1532   1497    1049     1025   1189    1162
-------------------------------------------------------
```

```
                <Processor>  Captur
                Utilization   Ratio
Time            Total Virt.  (pct)
--------        ----- -----  ------
21:25:00         1709  1499  100.00
21:26:00         1642  1438  100.00
21:27:00         1641  1381  100.01
21:28:00         1639  1329   99.99
21:29:00         1561  1332  100.00
21:30:00         1528  1305   99.99
**********************
Average:         1629  1389  100.00
```

Processor Measurements – SMT Validity

ESALPARS	ASSIGNED	
ZVMQA00	**933.7**	4.1
ZVMQA00	897.6	4.2
ZVMQA00	908.8	5.6
ZVMQA00	905.1	5.9
ZVMQA00	883.2	7.4
ZVMQA00	873.5	8.2
ZVMQA00	894.9	7.0
ZVMQA00	915.2	4.8
ZVMQA00	901.2	5.3
ZVMQA00	917.3	6.2
ZVMQA00	906.4	6.2
ZVMQA00	923.1	6.5

ESAUSP5	THREAD		MT-PRORATED	
21:25:00	**1535**	1499	**1192**	1163
21:26:00	1477	1438	1146	1116
21:27:00	1431	1381	1115	1076
21:28:00	1382	1329	1077	1035
21:29:00	1379	1332	1081	1044
21:30:00	1350	1305	1061	1025
21:31:00	1445	1402	1129	1095
21:32:00	1469	1427	1141	1107
21:33:00	1413	1364	1097	1058
21:34:00	1452	1405	1134	1097
21:35:00	1430	1383	1117	1080
21:36:00	1454	1406	1137	1099

Compare assigned time to thread time to “prorated”

- Target is assigned time, maybe subtract thread idle

SMT Prorate Minute by Minute

Compute ESALPARS assigned and subtract thread idle

Prorate against ESAUSP5 total and get “new” prorate interval by interval (.56 - .59)

```
ratio:  0.563289902
ratio:  0.562660799
ratio:  0.583018868
ratio:  0.603183792
ratio:  0.577411168
ratio:  0.578814815
ratio:  0.560761246
ratio:  0.578012253
ratio:  0.591224345
ratio:  0.575172176
ratio:  0.576713287
ratio:  0.576925722
```

CPU by Component/Function

Some even "better news"

- CPU numbers are traditional, measured by Linux
- VSI Prorated is based on HMC data
  - Shows SMT is significantly better

Report: **ESAUSP5** User SMT CPU Consumption Analysis

	<------CPU Percent Consumed (Total)---->						<-TOTAL CPU-->	
UserID	<**Traditional**>		<MT-Equivalent>		<IBM Prorate>		**<VSI Prorated>**	
/Class	Total	Virt	Total	Virtual	Total	Virtual	Total	Virtual
07:02:00	414.9	408.0	322.7	317.3	239.7	235.8	208.2	204.7
User Class Analysis								
OpenShif	**355.0**	350.3	276.0	272.3	204.9	202.2	**178.1**	175.7
Top User Analysis								
RHOSCP1	142.4	140.8	110.1	108.9	82.93	82.01	71.43	70.65
RHOSCP3	125.2	123.8	97.38	96.34	72.35	71.60	62.80	62.14
RHOSCP2	86.79	85.04	68.00	66.64	49.31	48.30	43.55	42.67

Summary: Expectations of SMT?

IBM Monitor data "MT Prorated" is incorrect

IBM Monitor data "MT-Equivalent" is not validated

Need a validated prorate factor for chargeback

Low utilization:

- Capacity is not really an issue
- Response time should not change

High utilization – Intense workloads (SAP, Oracle):

- Capacity WILL see improvements

High utilization – Polling workloads (WAS, DB2):

- Response times WILL get worse, TURN off AFFINITY
- Capacity increases - Validate with MFC…

Maximizing Utilization – Case Study

“Can I get more out of my box that is 97% busy?”

The Basic Challenge

The CEC is BUSY!

- There are 40 IFLs that are "shared"
- IFLS are 97% assigned (3867/4000)
- Note the system overhead – "Ovhd" and "Mgmt" (4.9 and 4.1)
- CPU delays can be expected! Shares/Weights need to be managed

Report: **ESALPARS** **Logical Partition Summary**

Totals by Processor type:

<----------CPU-------->				<-Shared Processor busy->			
Type	Count	Ded	shared	Total	Logical	Ovhd	Mgmt
CP	8	0	8	711.0	708.8	1.4	0.7
IFL	40	0	40	3867.5	3858.5	4.9	4.1
ICF	1	0	1	24.2	19.6	0.4	4.2
ZIIP	4	0	4	268.3	266.4	0.6	1.2

97% busy

Identifying CPU Problems (PROD2)

Report: **ESAXACT** Transaction Delay Analysis

UserID /Class	<-Samples-> Total	In Q	<-----Percent non-dormant (Wait sta Run	Sim	CPU	SIO	Pag	D- SVM	T- SVM	CF	Tst Idl
15:01:00	117	137	13	0	15	0	0	0	0	0	72
Hi-Freq:	20280	8108	10	0.1	**11**	0	0.0	0.5	0.0	0	78
User Class Analysis											
P_AP_JB	6300	2990	12	0.1	13	0	0	0	0	0	75
PROD_MQ	180	119	0.8	0	1.7	0	0	0	0	0	97
P_OTHER	1320	299	5.4	0	9.4	0	0	0	0	0	85
P_SUB_JB	3720	1519	6.5	0.2	8.4	0	0	0	0	0	85
P_UTIL	360	239	14	0	16	0	0	0	0	0	70
PROD_WAS	2640	1207	9.0	0.1	15	0	0.2	0	0	0	76
RT_AP_JB	1680	769	11	0	12	0	0	0	0	0	77
RT_OTHER	120	59	10	0	12	0	0	0	0	0	78
RT_SB_JB	240	59	19	0	8.5	0	0	0	0	0	73
RTST_WAS	1320	662	15	0	6.2	0	0	0	0	0	79
zVPS	720	55	15	0	0	0	0	0.1	0	0	85
Other	1680	131	6.1	0	3.8	0	0	6.5	0.8	0	89
CPU POOL User Analysis											
MQPOOL	180	119	0.8	0	1.7	0	0	0	0	0	97

Running: Using CPU
CPU Wait: Waiting for CPU
Why is CPU Wait high?

- High CEC Utilization?
- High LPAR Utilization?
- Low LPAR Weight?
- Low Share?
- Affinity?

DEV2: CPU Wait Analysis

Report: **ESAXACT** Transaction Delay Analysis

UserID /Class	<-Samples-> Total	In Q	Run	Sim	CPU	SIO	Pag	E- SVM	D- SVM
			<-----Percent non-dormant wait						
03/16/26 15:01:00	186	302	5.6	0	43	0	0	0	0
Hi-Freq:	25740	18004	6.3	0.1	**35**	0.0	0.1	1.5	0
User Class Analysis									
D_AP_JB	2040	1995	6.4	0.1	30	0	0.4	0	0
DEV_MQ	120	119	3.4	0	29	0	0	0	0
D_QM_OTH	1080	1058	4.4	0	28	0	0.1	0	0
D_SUB_JB	600	591	8.8	0.2	44	0	0.7	0	0
D_UTIL	600	369	4.1	0	20	0	0.5	0	0
D_WAS	720	715	8.4	0	53	0	0	0	0
GT_AP_JB	1800	1416	4.7	0.1	24	0	0.1	0	0
GT_OTHER	360	353	4.5	0	16	0	0	0	0
GT_SB_JB	1320	376	10	0	43	0	0.5	0	0
GTST_WAS	1320	1176	3.7	0	25	0	0	0	0
QM_AP_JB	2280	2245	8.9	0.1	39	0	0.2	0	0
QM_MQ	120	120	5.8	0	58	0	0	0	0
QM_SB_JB	960	837	10	0	60	0	0	0	0
QM_WAS	600	593	5.6	0	41	0	0	0	0
STAGE_MQ	360	357	11	0.3	34	0	0	0	0

Why is CPU Wait very high?

- **CPU limited by LPAR weight**
- High CPU Utilization!
- Affinity lasts for a time slice
- Time slice is 10ms default
- Minimum is 1ms
- (Modlevel already set)

LPAR Configuration Summary

z/VM share of IFLs (always start here):

```
Report: ESALPARS     Logical Partition Summary
------------------------------------------------------------------------
          <--Complex--> <--------Logical Partition-------->  <-Assigned
          Phys Dispatch              Virt CPU  <%Assigned>  <---LPAR-->
Time      CPUs    Slice Name     Nbr CPUs Type Total  Ovhd  Weight  Pct
--------  ---- -------- -------- --- ---- ---- -----  ----  ------ ----
15:01:00    53  Dynamic Totals:   00   14  CP  710.2   1.4     966  100
                        Totals:   00   46 IFL   3863   4.9     436  100
                        VMPROD2   06   13 IFL  945.3   2.8     128 29.4
                        VMDEV1    07   10 IFL  928.4   0.2      90 20.6
                        VMDEV2    08   10 IFL  929.7   0.2      90 20.6
                        VMPROD1   05   13 IFL   1060   1.7     128 29.4
Totals by Processor type:
<----------CPU-------> <-Shared Processor busy->
Type Count Ded shared  Total  Logical Ovhd Mgmt
---- ----- --- ------ ------ -------- ---- ----
IFL     40   0     40 3867.5   3858.5  4.9  4.1
```

- VMPROD2 entitlement: 128/436 (12%) of 40 shared IFLs
- So VMPROD2 entitlement is 11.7 IFLs
- VMPROD1 entitlement: 11.7 IFLs but using 10.6
- VMDEV1/2 are using 930% shared IFLs (more than 8.3 entitlement)
- IFLs running (97%) busy
- Not much left…

Running at 95% With "Available Capacity"…

<--------Logical Partition------->				<%Assigned>		<-Assigned				Entitled
		Virt	CPU			<---LPAR-->		<-Thread->		CPU
Name	Nbr	CPUs	Type	Total	Ovhd	Weight	Pct	Idle	cnt	Cnt
Totals:	00	14	CP	710.2	1.4	966	100			
Totals:	00	46	IFL	3863	4.9	436	100			
VMPROD1	05	13	IFL	**1060**	1.7	128	29.4	**356.0**	2	**11.74**
VMPROD2	06	13	IFL	945.7	2.8	128	29.4	**437.7**	2	11.74
VMDEV1	07	10	IFL	928.4	0.2	90	20.6	0.00	2	8.26
VMDEV2	08	10	IFL	929.3	0.2	90	20.6	0.03	2	8.26

PROD not using entitled but "idle thread time"
DEV using all of entitled, no "idle thread time"
PROD has EXTRA CAPACITY available
PROD has CPU wait time equivalent to run time

Modlevels Secret Command... 2 – Turn Off Affinity

```
SYSCONTROL set to MODLEVEL 0
Ready; T=0.01/0.01 11:24:24
q syscontrol
DISPATCH THDAFFINITY OFF
DISPATCH PREEMPTLOCAL ON
DISPATCH TSEARLY   0
DISPATCH INCHIPBUSY          0
DISPATCH INCHIPDELAY         0
DISPATCH INNODEBUSY      50000
DISPATCH INNODEDELAY     50000
DISPATCH INSYSBUSY      200000
DISPATCH INSYSDELAY     200000
Ready; T=0.01/0.01 11:24:27
CP SET SYSCONTROL DISPATCH STEALBARRIER INCHIPBUSY 0 INCHIPDELAY 0
INNODEBUSY 0 INNODEDELAY 0 INSYSBUSY 0 INSYSDELAY 0
q syscontrol
DISPATCH THDAFFINITY OFF
DISPATCH PREEMPTLOCAL ON
DISPATCH TSEARLY   0
DISPATCH INCHIPBUSY          0
DISPATCH INCHIPDELAY         0
DISPATCH INNODEBUSY          0
DISPATCH INNODEDELAY         0
DISPATCH INSYSBUSY           0
DISPATCH INSYSDELAY          0
```

Large LPARS cross chips (within nodes)
Turn off node delay ALSO?

Maximizing CPU Utilization

One more item

- Affinity likely for one time slice
- Default time slice 10 ms
- Average used time: less than 1 ms

SET SRM DSPSLICE 1 MS

- Also impacts fair share scheduler

CPU Summary

CPU Guidelines

Linux: Minimize vCPU to meet requirements (Use zVRM)

z/VM: Avoid CPU Wait (default dspslice is 5ms or 10ms with SMT)

- Reduce vCPU count and increase work per dispatch best
- Or, reduce wait time: SET SRM DSPSLICE 1
- Calculate CPU consumed / dispatch rate

Each vCPU waits, then does a small task

- 30 microseconds average CPU / dispatch
- (But waits 5ms – 10ms)

Summary – Maximize Your Investment!

Set LPAR weights correctly

Set Shares correctly -

Set virtual CPUs correctly (LPAR, Virtual Machines)

Remove affinity

Reduce time slice to 1ms….