

# Processor Configuration and Analysis - Intro

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

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

- **Identifying CPU problem**
- **Processor Utilization**
- **CPU Management Hierarchy**
  - CEC/LPAR weights
  - LPAR/Virtual Machine Share
- **LPAR,**
- **Processor measurements**

Report: **ESAXACT** Transaction Delay Analysis

```

-----
                                <-----Percent non-dormant (Wait sta
UserID    <-Samples->          D-   T-          Tst
/Class    Total   In Q  Run Sim  CPU SIO Pag  SVM SVM   CF  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?

## 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 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)

## 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

## 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

## 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

## 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)
3. The **Virtual Machine** is assigned a “share”
  - A “share” of the LPAR (ESAUSRC / ESAUSP2)
  - Thread time vs core time (ESAUSP5)
4. The **Linux Processes** have a “priority”
  - Processes are “prioritized” by “nice” settings (ESALNXC/P)

## 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

## LPAR Weights and HiperDispatch

## 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 entitlement/performance
- Hiperdispatch corrects this by “parking” vCPUs
- SMT threads share “core” entitlement

## Evaluate CEC perspective first:

- There are **10** IFLs that are “**shared**”
- IFLS are 92% assigned (**915.6/1000**)
- 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
<b>IFL</b>	<b>10</b>	<b>0</b>	<b>10</b>	<b>915.6</b>	<b>894.5</b>	<b>8.6</b>	<b>12.5</b>
ZIIP	3	0	3	23.9	22.3	0.4	1.2

## 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
-----
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)

## **Weights: Sets entitlement between Logical Partitions**

- Set weights based on business requirements

## **Virtual Processors**

- If too many, HiperDispatch will “park” – this adds overhead

## **Capping**

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

## Entitlement field added! Validate assigned vs entitled

### ESALPARS Logical Partition Summary

```

-----
<-----Logical Partition-----> <---Assigned Shares---> Entitled
      Virt CPU  <%Assigned> <---LPAR--> <VCPUs Pct> CPU Cnt
Name      Nbr CPUs Type Total  Ovhd Weight  Pct /SYS /CPU
-----
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
  
```

## z/VM Shares – Absolute and Relative

## **Objective 1: Provide workload with CPU**

## **Objective 2: Control looping users (manage performance)**

- Manage “excess share”
- Manage share by vCPU

## **z/VM “Fair Share Scheduler” controls looping users**

- Guarantees “normalized share” to each vCPU

## 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

## IBM Defaults are not optimum... (REL 3000)

- (Creates excess share for looping users)

## 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”
- 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

- **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
  
```

- **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
```

- **Now for the experiment – Set shares “correctly”**
  - Convert TCPIP,SSL,RACF from REL 3000 to ABS 2%
  - This ELIMINATES “EXCESS” bucket

```

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
  
```

## Measuring CPU Consumption

**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 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

- 1.0 is best

## Managing Distribution – LPAR share of IFLs

- Based on weight of the LPAR
- Weight is divided by 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

# Affinity

## What is affinity processing?

- 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

**ESALPARS**

VML1            06            2 IFL    200.1    0.0            **Ded**

Report: **ESAXACT**            Transaction Delay Analysis

```

-----Percent non-dormant (Wait
UserID  <-Samples->          E-  D-  T-
/Class  Total   In Q  Run Sim CPU SIO Pag SVM SVM 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?
- 2 IFLs dedicated to LPAR

## Why is CPU Wait high?

- High CPU utilization? NO, 55% per thread
- Queuing theory – 1 server, 1 queue, 50% time in queue
- Queuing theory – 4 servers, 1 queue, 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
  
```

```
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”**