

OpenShift Architecture and Performance Management

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

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





Performance Data Collection Technology

Container Architecture

- Docker
- OpenShift / Kubernetes
- ~~Rancher / Kubernetes~~
- zCX

OpenShift Performance Analysis

RHOS Overhead Analysis

Collecting Container Performance Data

z/OS zCX Data: Case Study

Performance Analysis

- Understanding system, application performance
- Resolving current performance issues (z/VM, Linux, network)

Operational Alerts

- Supporting 100's/1000's of servers/containers in many locations
- Defining and automating operational support

Capacity Planning

- Providing input to the financial acquisition process

Accounting / Charge back

- Building a financial model for resource billing
- Who is consuming the resource?

Performance management can NOT be the performance problem
Black boxes are not managed by definition

RHOS Servers are “closed”

- Not open to using SNMP directly or installing tools except for containerized
- No collectd
- Pushes data to Prometheus
- CPU data incorrect in SMT environment.

Prometheus agents are part of OpenShift (very heavy)

- HTTP interface
- “blob of data” (100k/pod, multiple megabytes per request) (yes, really “blob”)
 - <https://github.com/google/cadvisor/blob/master/docs/storage/prometheus.md#prometheus-container-metrics>
- Limited metrics
- Significant overhead in parsing every single metric of “blob”
- **Stil need “system / Linux” and “system / zVM” integration**

Generic implementation, parsing not too difficult, just expensive

Each metric tagged in blob:

- container=""
- id="/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-pod81ba15c5_32a2
- image=""
- name=""
- namespace="openshift-monitoring"
- pod="prometheus-k8s-1"} 65382.32 1677698845481 container_cpu_system_seconds_tota

And for comparison:

- container=""
 - id="/kubepods.slice/kubepods-besteffort.slice/kubepods-besteffort-pod953ec259_31
 - image=""
 - name=""
 - namespace="vsi-SNMPd-test"
 - pod="vsi-SNMPd-test-nzz4b"} 86.61 1677698849149
 - container_cpu_system_seconds_tot
- **Performance management can NOT be the performance problem**

Data sources are critical to performance management

(The performance management should not be the performance problem)

- Diagnostic tools are different, only run when there is a problem

SNMP installed in a container

- **SNMP is very efficient for collecting performance data!!!**
- Full access to systemwide metrics
- Additional metrics collected to align process data with containers
- SNMP is extremely efficient in every comparison
- **SNMP container installed on every node**

Now that we have visibility into it, understanding is easier

Containers are not magic, not an “extra layer of virtualization”

Each container managed by a “container manager”

- “Virtual” Processes inside container are mapped to “real”
- File systems are remapped
- Container “limited to what it can see”

Standard “relevant” Linux Process Tree for Docker

Report: **ESALNXC** LINUX Process Configuration

Node/ Name	<-----Process ID	Ident PPID	-----> GRP	-----> Appl	App1 Name
21:32:00					
DOCKER					
systemd	1	0	1	61176	systemd
dockerd	1218	1	1218	1218	dockerd
docker-c	1339	1218	1339	1218	dockerd
docker-c	32289	1339	32289	1218	dockerd
httpd	32312	32289	32312	32312	bouncer1
httpd	32370	32312	32312	32312	bouncer1
httpd	32371	32312	32312	32312	bouncer1
httpd	32372	32312	32312	32312	bouncer1
docker-c	32476	1339	32476	1218	dockerd
httpd	32498	32476	32498	32498	bouncer2 <---
httpd	32553	32498	32498	32498	bouncer2
httpd	32554	32498	32498	32498	bouncer2
httpd	32555	32498	32498	32498	bouncer2

Linux Process Tree

- Shows all processes
- “ppid” is parent process
- “1” is root
- Dockerd is manager
- Docker-c “container”

Result:

- CPU by process
- CPU by container

Docker Configuration APIs added to container SNMP MIB

- Exposed via SNMP
- Image, container names, status
- Internal index, **process ID, from "pid file"**
- Create / start / finish date/time

DOCKER Configuration Report

```

-----
Time / <-----Container Configuration-----> Status
Node   Index      ImageName  ContName      Procid
-----
11:25:00
sles12
      a2e334b2a401 stresscpu  stress4      36448  runn
      6dd81b413dfa stresscpu  stress3      36360  runn
      e96592194411 stresscpu  stress2      36219  runn
      ab0a179c44c9 stresscpu  stress1      36133  runn
      afc4f3819380 hello-worl hardcore_nash 0      exit
      f57e0f5fd61c hello-worl pedantic_lamarr 0      exit
  
```

If have container name, process ID, can capture CPU

Standard SNMP Linux data collection (VSI MIB):

- 45 metrics per process per minute

Container MIB provides container index and name

By container, data validated against process table data

- User CPU, System CPU

```

Report: ESADOCK2          DOCKER Transaction Report
-----
Node      <---Container-----> Interval <CPU Percent>
          Index          Name       Seconds  User   System
-----
11:25:00
sles12    a2e334b2a401 stress4   60.5    8.7    0
          6dd81b413dfa stress3   60.5    8.1    0.0
          e96592194411 stress2   60.5    5.2    0.0
          ab0a179c44c9 stress1   60.5    5.8    0.0
          afc4f3819380 hardcore_n 60.5    0      0
          f57e0f5fd61c pedantic_l 60.5    0      0
  
```

Docker “ram” consumption (from standard process data)

Storage Resource consumption by container

- Storage use, paging, file activity

```

Report: ESADOCK2          DOCKER Trans          Velocity Software Corpor
-----
Node      <---Container----->  <-----Storage in "MB"-----
Index      Name      Current  max  cache  rss  inact  activ
-----
11:25:00
sles12    a2e334b2a401 stress4   4.19  4.53  3.70  0.12  0.11  0.01
          6dd81b413dfa stress3   3.96  11.64  3.70  0.12  0.11  0.01
          e96592194411 stress2   4.23  5.32  3.70  0.13  0.11  0.01
          ab0a179c44c9 stress1   4.02  11.62  3.70  0.12  0.11  0.01
          afc4f3819380 hardcore_n 0      0      0      0      0      0
          f57e0f5fd61c pedantic_l 0      0      0      0      0      0
  
```

Filesystems have “link” to container, from standard HOST MIB

Report: **ESAHST2** LINUX HOST File Systems

NODE/ Time/ Date	Index	<-Utilization-> <MegaByte> Pct			-Storage-----> Description
-----	-----	Size	Used	Full	-----
sles12					
	76	389	0	0	/run/user/0
	25208	64.0	0	0	/var/lib/docker/containers/ab0a179c44c9
	25209	1946	0.0	0.0	/var/lib/docker/containers/ab0a179c44c9
	25211	64.0	0	0	/var/lib/docker/containers/e96592194411
	25212	1946	0.0	0.0	/var/lib/docker/containers/e96592194411
	25214	64.0	0	0	/var/lib/docker/containers/6dd81b413dfa
	25215	1946	0.0	0.0	/var/lib/docker/containers/6dd81b413dfa
	25217	64.0	0	0	/var/lib/docker/containers/ a2e334b2a401
	25218	1946	0.0	0.0	/var/lib/docker/containers/ a2e334b2a401

Linux File Systems matched to containers

File systems by container

- File size/utilization
- Description, R/W, boot flags

DOCKER File System Report

```

-----
NODE/Container      <-Utilization->
Time/      FileSys  <MegaByte>  Pct
Date       Index    Size   Used  Full   Errors  R/W  Boot  <----
-----  -----  -----  -----  -----  -----  ---  ----  -----
11:25:00
sles12
  stress4
    25217      64    0    0
    25218    1946    0  0.0
  stress3
    25214      64    0    0
    25215    1946    0  0.0
  stress2
    25211      64    0    0
    25212    1946    0  0.0
  
```

RHOS also introduces “pods”

Standard Linux data for RHOS

- “common” container manager
- “common” parent is root.

```

Report: ESALNXC          LINUX Process
Monitor initialized: 03/08/23 at 0
-----
Node/          <-----Process  Ident
Name          ID          PPID    GRP
-----
rhoscpl
systemd              1          0          1
auditd             1060        1        1060
dbus-dae           1097        1        1097
crio                1929        1        1929
kubelet            1970        1        1970
common             2387        1        2387
  kube-sch         2442        2387      2442
  common           2520        1        2520
  cluster-        2559        2520      2559
  common           2582        1        2582
  cluster-        2600        2582      2600
  common           2677        1        2677
  cluster-        2703        2677      2703
  common           8396        1        8396
  promethe         8455       8396      8455
  common           10850       1        10850
  grafana-        11073       10850    11073
  common           1509339     1        2011
  SNMPd           1509401   1509339  2073
  
```

Large Container RHOS Configuration (pid data, SNMP in container)...

Report: **ESAK8S1** Kubernetes Configuration Report Velocity Sof
 Monitor initialized: 06/22/23 at 00:00:00 on 8562 serial 040F78 First record

Linux Node/Time	<---OpenShift Pod Configuartion--> PodName	PodIndex	<-----Container Configuration Name	<--Process Ide ProcessID	Pro
00:15:00					
rhoscpl	insights-operator-7f	ba92ef4e1b29	insights-operator	13075	ins
	multus-admission-con	bbb779ebae39	kube-rbac-proxy	14520	kub
			multus-admission-con	12865	web
	etcd-rhoscpl.vsil.ve	c6088570034c	etcd	2276	etc
			etcd-metrics	2389	etc
			etcd-readyz	2941	clu
			etcd-health-monitor	3594	clu
	prometheus-operator-	c7875e16a183	prometheus-operator-	11955	pro
	kube-state-metrics-5	d5e9800a6a6c	kube-state-metrics	11658	kub
			kube-rbac-proxy-main	12519	kub
			kube-rbac-proxy-self	13456	kub
	prometheus-k8s-1	45f9f5becfa0	prometheus	14548	pro
			thanos-sidecar	15191	tha
			prometheus-proxy	15372	oau
			kube-rbac-proxy-than	15931	kub
			kube-rbac-proxy	15508	kub
			config-reloader	14820	pro
	packageserver-5f99c6	662518f1bc49	packageserver	13044	pac
	vsi-SNMPd-vk5vd	7e583397ff6e	vsi-SNMPd	19285	snm
	multus-9fj4w	8e5c35c1b612	kube-multus	4922	/en

By CPU consumption by process

- Just the active processes

Report: **ESALNXP** LINUX HOST Process Statistics Report

node/ Name	<Process ID ID	<Ident PPID	Nice Valu	PRTY Valu	<-----CPU Tot	Percents sys	user	syst	usrt	
07:02:00										
rhoscp1	0	0	0	0	139	19.9	113	3.74	2.87	→ node totals
systemd	1	0	0	20	1.62	0.57	1.05	0	0	
crio	1929	1	0	20	6.48	0.15	0.58	3.24	2.51	
kubelet	1970	1	0	20	10.5	3.06	7.47	0	0	
etcd	3078	3061	0	20	15.4	5.91	9.5	0	0	
etcd	3131	3113	0	20	1.67	0.87	0.80	0	0	
cluster-	3959	3944	0	20	2.49	0.23	2.25	0	0	
cluster-	5072	4989	0	20	2.02	0.35	1.67	0	0	
promethe	8455	8396	0	20	64.1	3.47	60.6	0	0	
openshif	11221	11167	0	20	2.72	0.32	2.41	0	0	
kube-api	1245253	1245136	0	20	16.7	1.77	15.0	0	0	
oauth-ap	1830274	1830210	0	20	3.29	0.25	3.04	0	0	
rhoscp2	0	0	0	0	84.2	18.5	59.4	3.88	2.52	→ node totals
systemd	1	0	0	20	2.15	0.67	1.18	0.22	0.08	
crio	2056	1	0	20	5.85	0.18	0.58	3.10	1.98	
kubelet	2091	1	0	20	12.5	3.72	8.81	0	0	
etcd	3185	3169	0	20	14.3	5.33	9.00	0	0	
etcd	3230	3210	0	20	1.35	0.68	0.67	0	0	
cluster-	3277	3258	0	20	2.50	0.23	2.27	0	0	
cluster-	9096	9068	0	20	1.47	0.22	1.25	0	0	
cluster-	10370	10281	0	20	1.02	0.20	0.82	0	0	
openshif	10951	10905	0	20	2.22	0.22	2.00	0	0	
cluster-	11973	11886	0	20	1.50	0.23	1.27	0	0	

Report: **ESAK8S2** Kubernetes Resource Utilization Report
 Monitor initialized: 10/15/24 at 17:00:00 on 3932

```

-----
NODE/                               <---Container-->  <---Container CPU----->
Time/ PodName                       <---Process ID-->  <-----CPU Percents----->
Date  ContainerName                  ProcID ProcName          Tot   sys  user  syst  usrt
-----
18:00:00
*** Nodes *****
LPIOC330                               (Totals)           34.1  5.67  27.8  0.18  0.50
  apiserver-677df8ff8b                 759530 openshif  1.28  0.21  1.06   0     0
  openshift-apiserver
  kube-controller-manage                 755380 cluster-  2.71  0.62  2.09   0     0
  cluster-policy-controller
  kube-controller-manage                 3399299 kube-con  2.08  0.47  1.61   0     0
  etcd-vaausclupio330
  etcd                                   5084  etcd      4.65  1.35  3.30   0     0
  etcd-readyz                           2390617 cluster-  0.90  0.06  0.84   0     0
  apiserver-6475d96f67
  oauth-apiserver                       752737 oauth-ap  0.61  0.11  0.50   0     0
  packageserver-b67bbf
  packageserver                         700543 package-  1.33  0.23  1.10   0     0
  ovnkube-node-9xrrg
  ovnkube-controller                   5643  ovnkube   0.68  0.13  0.18  0.13  0.24
  machine-config-operator
  machine-config-operator               746794 machine-  0.58  0.11  0.13  0.12  0.23
  openshift-kube-scheduler
  kube-scheduler                       746438 kube-sch  0.52  0.07  0.46   0     0
  console-operator-ff4
  console-operator                     746794 machine-  0.51  0.06  0.45   0     0
  kube-scheduler
  console-operator-ff4
  console-operator                     746438 kube-sch  0.68  0.15  0.54   0     0
  kube-scheduler
  console-operator-ff4
  console-operator                     746438 kube-sch  0.61  0.12  0.49   0     0
  console-operator-ff4
  console-operator                     756863 console  0.68  0.12  0.57   0     0
  console-operator
  console-operator                     756863 console  0.55  0.11  0.44   0     0
  
```

Large OpenShift installation...

- CPU by pod
- Pod by container

Chargeback support – database extract:cpu by node by pod / space

Date	Time	NODE	VSIK8S.PODNAME	VSIK8S.PODSPACE	VSIK8S.TOTCPUPCT	VSIK8S.
20241015	173400	LPIOC330	*Totals	""	34.15	30731
20241015	173400	LPIOC330	apiserver-677df	openshift-apise	1.15	1038
20241015	173400	LPIOC330	openshift-kube-	openshift-kube-	0.25	228
20241015	173400	LPIOC330	kube-controller	openshift-kube-	0.39	350
20241015	173400	LPIOC330	kube-controller	openshift-kube-	0.53	480
20241015	173400	LPIOC330	kube-controller	openshift-kube-	2.08	1868
20241015	173400	LPIOC330	etcd-vaausclupi	openshift-etcd	4.65	4189
20241015	173400	LPIOC330	etcd-vaausclupi	openshift-etcd	0.41	373
20241015	173400	LPIOC330	etcd-vaausclupi	openshift-etcd	0.90	813
20241015	173400	LPIOC330	controller-mana	openshift-contr	0.33	300
20241015	173400	LPIOC330	route-controlle	openshift-route	0.21	191
20241015	173400	LPIOC330	apiserver-6475d	openshift-oauth	0.61	547
20241015	173400	LPIOC330	packageserver-b	openshift-opera	1.33	1199
20241015	173400	LPIOC330	ovnkube-node-9x	openshift-ovn-k	0.58	526
20241015	173400	LPIOC330	ingress-operato	openshift-ingre	0.15	135
20241015	173400	LPIOC330	node-resolver-k	openshift-dns	0.21	189
20241015	173400	LPIOC330	machine-config-	openshift-machi	0.51	458
20241015	173400	LPIOC330	openshift-kube-	openshift-kube-	0.61	546
20241015	173400	LPIOC330	nmstate-handler	openshift-nmsta	0.21	193

Other components?

- Node “groups” are defined groups of associated servers
- Totals for all RHOS “OpenShif” servers (**Linux perspective**)
- CPU time as measured by Linux – correct???

```

Report: ESALNXA          LINUX HOST Application Report
-----
Node/      Process/      ID      <---Processor Percent--->
Date       Application  <Process><Children>
Time       name        Total   sys   user  syst  usrt
-----
11:25:00
OpenShif *Totals*      0  351.0  53.4   278  11.8   7.9
            common      0  293.1   39.9   251   1.5   1.1
            crio         0  18.68    0.4    1.3  10.3   6.8
            kernel      0   1.37    1.4     0     0     0
            kubelet     0  31.91    9.5   22.4   0.0   0.0
            ovs-vswi     0   1.27    0.7    0.6     0     0
            ovssdb-se    0   0.15    0.0    0.1     0     0
            systemd    0   4.50    1.5    3.0     0     0
  
```

Some “better news” from z/VM based measurements

- CPU numbers are traditional, measured by Linux (Thread time)
- **Virtual Machine** with **SMT** “Prorate” are lower
- **IBM SMT numbers do not match reality....** (Same in zCX)
- <https://VelocitySoftware.com/tuneguide/smt.html> for a detailed explanation of SMT

```

Report: ESAUSP5           User SMT CPU Consumption Analys
-----
UserID      <-----CPU Percent Consumed (Total)---->
/Class      <Traditional> <MT-Equivalent> <IBM Prorate>
            Total   Virt   Total   Virtual   Total   Virtual
-----
07:02:00   414.9  408.0  322.7   317.3   239.7   235.8

***User Class Analysis***
OpenShif 355.0 350.3 276.0   272.3   204.9 202.2

***Top User Analysis***
RHOSCP1   142.4  140.8  110.1   108.9   82.93   82.01
RHOSCP3   125.2  123.8   97.38   96.34   72.35   71.60
RHOSCP2   86.79  85.04   68.00   66.64   49.31   48.30
    
```

Some even “better news”

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

Report: **ESAU5P5** User SMT CPU Consumption Analysis

UserID /Class	<-----CPU Percent Consumed (Total)---->				<-TOTAL CPU-->			
	< Traditional >		<MT-Equivalent>		<IBM Prorate>		< VSI Prorated >	
	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

Resource consumption model

- Correct the CPU first from SMT
- Aggregate containers into pods
- Aggregate pods by name
- Naming convention allows consumption model

RHOS Case study:

- Production environment
- 9 nodes, SNMP enabled

Linux Release: RHELCoreOS 412.86.202310210217-0

Linux s01vx9986726 4.18.0-372.76.1.el8_6.s390x

Case study: Production “Real” Storage over configured

- ESAUCD2 – 400GB defined, **300GB free**.
- CMM works. (CMM container of course)
- Can manage

Report: **ESAUCD2**

```

-----
Node/      <-----
Time/      <--Real Storage-->
Date       Total  Avail  Used
-----  -----  -----  -----
17:01:00
***Node Groups***
COREOS    410936  303K    98K
*** Nodes *****
RHOSDI1   24156  14201   9955
RHOSDI2   24156  13141  11015
RHOSDM1   24156  10483  13673
RHOSDM2   24156   8964  15192
RHOSDM3   24156  10811  13345
RHOSDWB   96731  90768   5964
RHOSDW1   64475  53776  10699
RHOSDW2   64475  52806  11670
RHOSDW3   64475  55161   9315
  
```

How much CPU (vCPUs) should be allocated to RHOS?

- RHOS manages CPU via “millicpu” increments
- PODS / containers do not start if not sufficient “millicpu”
- One vCPU = 1000 millicpu
- 500 reserved for system

- Work needed before “trimming vCPU counts”

Single node, (4 vCPU/threads) CPU balanced

Report: **ESALNXS** LINUX VSI System Analysis Report
 Monitor initialized: 06/12/25 at 09:59:46 on 8562 ser

```

-----
Node/      <---Load Numbers--> CPU <Processor Pct Util>
Time      Users Procs MaxProc NBR Total  Syst  User  Idle
-----
10:01:00
rhosgl1    0   488 4194304 Tot 130.5 24.3  100  255
           1   31.4  5.7 24.2 65.1
           2   31.3  5.5 24.1 65.3
           3   34.1  6.3 26.3 62.3
           4   33.5  6.7 25.2 62.6
  
```

Single node, “common” (RHOS) using 94% - Application analysis

Report: **ESALNXA** LINUX HOST Application Report

Node/ Date Time	Process/ Application name	ID	<---Processor Percent---> <Process><Children>				
			Total	sys	user	syst	usr
10:01:00	*Totals*	0	128.3	21.1	94.5	4.9	7.8
rhossgl1	bash	2539	2.15	0.0	0	0.3	1.8
	common	3044	94.15	14.3	77.9	0.6	1.4
	crio	2500	8.68	0.3	1.3	3.6	3.5
	kernel	1	0.62	0.6	0	0	0
	kubelet	4071	13.12	3.5	9.5	0.0	0.0
	ovs-vswi	1272	4.10	1.2	2.9	0	0
	ovsdb-se	1140	0.20	0.0	0.2	0	0
	systemd	0	4.88	0.9	2.6	0.4	1.0
	systemd-	879	0.22	0.1	0.1	0.0	0.0

Single node, common (RHOS) using 94%

Report: **ESAK8S2** Kubernetes Resource Utilization Report

```

-----
NODE/          <---Container--> Micro  <--Container CPU-----
Time/ PodName  <---Process ID--> <CPU>  <-----CPU Percents----
Date  ContainerName  ProcID ProcName      Tot  sys user syst usr
-----
rhossgl1      (Totals)          2546  93.9 14.2 77.7 0.58 1.3
  ovnkube-node-zfnxx          81  1.62 0.27 0.50 0.28 0.5
  ovn-controller             6320 ovn-cont  81  0.15 0.02 0.13  0
  ovnkube-controller        7571 ovnkube  81  1.43 0.25 0.37 0.27 0.5
  kube-controller-mana          81  1.85 0.48 1.37  0
  cluster-samples-oper         20   9.6 1.03 8.55  0
  multus-rqtdx                 10  0.48 0.03 0.05 0.10 0.3
  console-54c85c9767-q         10  0.23 0.02 0.22  0
  dns-default-6x68j            61  0.20 0.07 0.13  0
  prometheus-operator          6  0.15 0.05 0.10  0
  prometheus-k8s-0             76  7.13 0.50 6.63  0
  vsi-snmpd-kdwx6              2  2.07 0.43 1.63  0
  kube-apiserver-api.s        296  36.1 4.30 31.8  0
  kube-apiserver             669439 watch-te 296  35.7 4.20 31.5  0
  kube-apiserver-cert-       669569 cluster- 296  0.12 0.03 0.08  0
  kube-apiserver-check       669692 cluster- 296  0.23 0.05 0.18  0
  
```

SNMP works with containerized SNMP

- Standard network data

CPU analysis

- Linux process data
- Linux system data
- Docker / RHOS container / pod
- SMF records System (70), Job (30)

Storage / ram / Swap

- System
- Process
- Container / pod

Install zCX with Docker

- [“https://velocitysoftware.com/zcximpl.html”](https://velocitysoftware.com/zcximpl.html)

Install SNMP container

- And yes, it works, and what is it?
- (Linux 5.4.0, Ubuntu Distribution for 390)

SNMP server configuration

Description:

```
Linux 7f1752ffc4e3 5.4.0-146-generic
```

```
#163-Ubuntu SMP Fri Mar 17 18:32:31 UTC 2023 s390x
```

```
ObjectId: 01.03.06.01.04.01.BF.08.03.02.0A
```

zCX really is a standard Linux: Docker, not RHOS

Two network interfaces

- Ethernet, loopback
- Mac address even

Report: **ESATCP4**

Monitor initialized: 10/27/2

```

-----
Date/          <Total Octets>
Time          <-Per second->
Node          IFT Input  Output
-----
zcxinst1 - 1 555.0  554.96
           - 10 675.2  4715.0
  
```

NODE	Idx	Speed	<-Status->		Up	<-----Interface----->			
	Nbr	MTU	(Est)	Oper	Admin	Time	MACAddress	Description	Type
zcxinst1	1	65536	10M	UP	UP	0	00:20:20:20:20:20	lo	Software LoopBack
	10	1492	10G	UP	UP	0	02:42:AC:11:00:05	eth0	ETHERNET-CSMACD

zCX, OpenShift run on zLIP in z/OS address space

Linux dashboard: Looks like any other Linux server to us

ESALNXP - VSI Linux Percent Usage by Process - VM4

Time	Node	Name	Process ID	PPID	GRP	Tot	sys	user	syst	usr	valu	valu	Size	RSS	Peak	Swap	2834	Data	Stk	EXEC	Lib	Lck	PTbl	min	maj	Faults
13:08:00	zcxinst1	*Totals*	0	0	0	10.7	6.2	2.7	1.5	0.3	0	0	24K	714	25K	306	2834	12.7	169	1981	336	11.6	17	0	2	
13:08:00	zcxinst1	systemd	1	0	1	0.1	0.0	0.0	0	0	0	20	164	7.4	226	0.7	20.5	0.1	1.3	13.7	0	0.1	0	0	0	
13:08:00	zcxinst1	ksoftirqd/0	10	2	0	0.1	0.1	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	
13:08:00	zcxinst1	rcu_sched	11	2	0	0.1	0.1	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	
13:08:00	zcxinst1	ksoftirqd/1	16	2	0	0.1	0.1	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	
13:08:00	zcxinst1	dmccrypt_writ	603	2	0	0.0	0.0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	
13:08:00	zcxinst1	btrfs-transa	682	2	0	0.0	0.0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	
13:08:00	zcxinst1	systemd-jour	903	1	903	0.1	0.1	0.0	0	0	-1	19	89.1	29.7	111	0.1	18.7	0.1	0.1	13.6	0	0.2	2	0	0	
13:08:00	zcxinst1	multipathd	1029	1	1029	0.1	0.1	0.0	0	0	0	-100	336	16.1	400	0	10.0	0.1	0.1	6.4	336	0.1	0	0	0	
13:08:00	zcxinst1	zcxauthplugi	1083	1	1083	0.1	0.1	0.0	0	0	0	20	469	5.8	469	0.2	142	0.1	2.7	1.9	0	0.1	0	0	0	
13:08:00	zcxinst1	containerd	1085	1	1085	1.1	0.7	0.4	0	0	0	20	1453	19.4	1517	3.4	134	0.1	25.6	1.9	0	0.2	0	0	0	

Linux CPU Utilization by Node zcxinst1 - VM4

Time	Node CPU Percent Used
13:02	9.0
13:03	8.8
13:04	9.5
13:05	10.2
13:06	13.5
13:07	8.2

ESALNXR - Linux Memory Analysis - VM4

Time	Node	Total	Free	Cache	Swap	Total	Activ	Inact	Stack	Slab	Reclm	Total	Used	Chunk	table	Total	Free	Rsvd	Size						
13:08:00	zcxinst1	1407	139.7	459.3	237.6	21.7	271.0	128.1	163.2	14.6	257.9	99.0	7.7	0.1	0	126.0	0.1	0	16.9	0	0	0	0	0	
13:07:00	zcxinst1	1407	139.8	458.9	237.0	21.7	271.0	128.1	163.2	14.7	257.8	99.0	7.7	0.1	0	126.0	0.1	0	16.9	0	0	0	0	0	0
13:06:00	zcxinst1	1407	140.2	458.5	236.1	21.7	271.0	128.1	163.2	14.7	258.0	99.0	7.6	0.3	0	126.0	0.1	0	16.9	0	0	0	0	0	0
13:05:00	zcxinst1	1407	132.2	457.7	235.0	21.7	280.6	137.7	163.2	14.7	258.1	99.0	7.6	0.1	0	126.0	0.1	0	17.2	0	0	0	0	0	0
13:04:00	zcxinst1	1407	141.5	457.4	234.6	21.7	270.9	128.1	163.2	14.7	258.1	98.9	7.5	0.2	0	126.0	0.1	0	17.0	0	0	0	0	0	0
13:03:00	zcxinst1	1407	142.0	457.2	234.2	21.7	271.0	128.0	163.2	14.7	258.0	98.9	7.5	0.0	0	126.0	0.1	0	16.9	0	0	0	0	0	0
13:02:00	zcxinst1	1407	142.0	456.9	233.6	21.7	271.0	128.1	163.2	14.6	258.0	98.9	7.5	0.2	0	126.0	0.1	0	16.9	0	0	0	0	0	0

ESALNXV - LINUX Virtual Processor Analysis Repo - VM4

Time	Node	VM	Node	<Linux Pct CPU>	<Process Data>	Capture Ratio	Prorate Factor	LPAR Name	
13:08:00	zcxinst1	INST1	TheUsrs	10.7	7.7	3.0	1.000	1.000	2 ZOS25A
13:07:00	zcxinst1	INST1	TheUsrs	8.4	6.0	2.3	1.000	1.000	2 ZOS25A
13:06:00	zcxinst1	INST1	TheUsrs	13.6	9.3	4.4	1.000	1.000	2 ZOS25A
13:05:00	zcxinst1	INST1	TheUsrs	10.2	7.5	2.7	1.000	1.000	2 ZOS25A
13:04:00	zcxinst1	INST1	TheUsrs	9.6	6.8	2.8	1.000	1.000	2 ZOS25A
13:03:00	zcxinst1	INST1	TheUsrs	8.7	6.3	2.5	1.000	1.000	2 ZOS25A
13:02:00	zcxinst1	INST1	TheUsrs	9.1	6.5	2.5	0.991	1.000	2 ZOS25A

ESALNXF - VSI Disk File System Performance - VM4

Time	Node	Name	Type	I/O Mrgd	/RdIO	/IO	I/O Mrgd	/WrIO	/IO	Prog	I/O	I/O	Device Path	
13:08:00	zcxinst1	dm-0	lvm	0	0	0	3.3	0	23.0	9.34	0	2.40	9.34 dm-0	
13:08:00	zcxinst1	dm-1	lvm	0	0	0	3.3	0	23.0	40.7	0	2.65	40.7 dm-1	
13:08:00	zcxinst1	dm-2	lvm	0	0	32.0	0	0.5	88.0	5.16	0	3.13	5.00 dm-2	
13:08:00	zcxinst1	vda	disk	0	0	0	3.4	0.7	23.9	8.40	0	2.98	3.90 ccw-0.0.0001	
13:08:00	zcxinst1	vda1	part	0	0	0	0.4	0.4	15.4	2.77	0	5.00	0.38 ccw-0.0.0001-part1	
13:08:00	zcxinst1	vda2	part	0	0	0	3.0	0.3	25.2	9.21	0	2.68	4.41 ccw-0.0.0001-part2	
13:08:00	zcxinst1	vdcc	disk	0	0	32.0	1.00	0.3	0.2	144	4.68	0	5.00	0.50 ccw-0.0.0004
13:08:00	zcxinst1	vdcc1	part	0	0	32.0	1.00	0.3	0.2	144	4.68	0	5.00	0.50 ccw-0.0.0004-part1
13:07:00	zcxinst1	dm-0	lvm	0	0	0	2.9	0	16.6	17.4	0	4.38	17.4 dm-0	

Linux process table – CPU (zIIP) by process captured
 Run some stressor processes (HTTP load)
 The Linux system number agrees with the process data
BUT SMT IS AN ISSUE FOR CAPACITY PLANNING/CHARGEBACK....

LINUX Virtual Processor Analysis Report

Node/ Name	VM ServerID	Node GroupID	<Linux Pct Total	<CPU> Syst	<Process User	<Data> Total	<Syst Syst	<User User	Capture Ratio
15:14:00 zcxinst1	INST1	TheUsers	88.4	5.0	83.4	88.4	5.0	83.4	1.000

Linux process table – CPU by process, standard Linux Run some stressor processes (HTTP load)

```

Report: ESALNXP          LINUX HOST Process Statistics Report
-----
node/      <Process Ident> Nice PRTY <-----CPU Percents----->
Name       ID      PPID  Valu Valu  Tot   sys user syst usrt
-----
15:14
zcxinst1   0        0     0    0   88.4 4.28 1.50 0.68 81.9
systemd-   889      1    -1   19  0.45 0.17 0.28  0    0
rsyslogd  1097     1     0   20  0.35 0.12 0.23  0    0
dockerd   2732     1     0   20  3.12 2.73 0.38  0    0
stresser  5518    5490  0    20  41.2  0    0  0.32 40.9
stresser  5655    5630  0    20  41.3  0    0  0.30 41.0
SNMPd    5778   5752  0   20 0.17 0.13 0.03  0    0
containe  5829     1     0   20  0.13 0.07 0.07  0    0
httpd     5883    5856  0    20  0.15 0.10 0.05  0    0
httpd     5884    5856  0    20  0.13 0.08 0.05  0    0
httpd     5885    5856  0    20  0.13 0.08 0.05  0    0
containe  6009     1     0   20  0.12 0.07 0.05  0    0
httpd     6065    6032  0    20  0.12 0.08 0.03  0    0
httpd     6066    6032  0    20  0.13 0.10 0.03  0    0
httpd     7079    5856  0    20  0.12 0.08 0.03  0    0
  
```

Docker containers (configuration, CPU)

- Note Docker assigns names if not provided

```

Report: ESADOCK1          DOCKER Configuration Report
-----
Time / <-----Container Configuration-----> Status
Node   ContainerName      ImageName      Index          Procid
-----
15:14:00
zcxinst1
clever_ramanujan localhost/ 7f1752ffc4e3    5752 runn
httpd1  httpd              2c9d7574ca87    6032 runn
httpd2  httpd              1fca2a98a85e    5856 runn
stress2      stresscpu        1c35d9534623    5518 runn
stress1      stresscpu        aa927ba72ee6    5655 runn
ibm_zcx_zos_ssh_  ibm_zcx_zo      a3cfd896b4d7    3436 runn
  
```

Docker containers (configuration, CPU)

- Note Docker assigns names if not provided – clever_ram is SNMP
- NOTE: DOCKER API GIVES INCORRECT CPU
- HTTPd containers getting hit many times a second
- Stress containers CPU intensive

Report: **ESADOCK2** DOCKER Transaction Report

```

-----
Node      <-----Container-----> <CPU Percent> <---Container Me
          Name             Index          ProcID      User  System  <---Storage met
          -----
15:14:00
zcxinst1  clever_ram  7f1752ffc4e3  5752        0.0   0.1    31.8  9.3  31.8
          httpd1    2c9d7574ca87  6032        0.1   0.3    3553  8.6  3617
          httpd2    1fca2a98a85e  5856        0.2   0.3    4736  7.7  4784
          stress2  1c35d9534623  5518        40.9   0.3    4.1   0.8  4.1
          stress1  aa927ba72ee6  5655        41.0   0.3    4.1   0.9  4.1
          ibm_zcx_zo a3cfd896b4d7  3436        0      0     0.9   0.0  0.9
  
```

From SMF 70 (we have one zIIP in SMT mode)

- V25A is our native z/OS LPAR
- Single zIIP is in **SMT**, z/OS sees two threads reported (92.8%)

```
Report: ZOSCPU          Z/OS CPU Report
-----
TIME/    <--CPU--> Sample <-CPU Utilization>

SYSID   ID   Type Count   Total   Wait   Parked
-----
15:12:00 - 15:13:00

V25A    0    GP      1      9.9    90.1    0
        2    GP      1      3.0    97.0    0
        4  zIIP    1     47.6    53.2    0
        5  zIIP    1     45.2    54.8    0
-----
Tot     GP      2     12.9   187.1    0
Tot    zIIP    2     92.8   108.1    0
```

From SMF 70 – LPAR “assigned time” from HMC

- One core assigned (47.6%), two concurrent threads

```

Report: ZOSLPARS      Z/OS LPAR Summary Report
Monitor initialized:      at 15:12:00 on Z15S serial 0E0F78
-----
                                <-----OnLine CPU----->
TIME/      <--Logical Partition--> <-CPU--> <---%Assigned----->
Date       Name      ID Serial SYSID Type Cnt Total Virtual Entitl
-----
15:12:00 - 15:13:00
           ZOSLP1    11 0E0F78 V25A GP      2  13.1    12.9    0.44
           ZOSLP1    11 0E0F78  V25A IIP      1  47.8    47.6    0.33
    
```

From SMF 30 (job records), only one user of zIIP

- Why is “zIIP on CP”, and why “GP CPU percent”?
- OpenShift is not “free” in terms of GP requirements

z/OS Job/Step CPU/Resources

SYSID	Job Name	JobID	Step Nbr	GCPU Percents			zIIP Pct				
				Total	STD	SRB	I T	Tot	Enc	Dep	on CP
15:13:00 - 15:14:00											
V25A	Totals		.	8.92	4.45	3.95		59	0	0.2	1.6
	INST1	STC05196	1	2.88	1.43	1.10		59	0	0.1	1.4
	ZOSMNV2	STC04764	1	0.02	0.02	0		0	0	0	0
	ZOSMNV4	STC04762	1	0.02	0.02	0		0	0	0	0

So, do we have accurate data?

From SMF 70 – LPAR “assigned time”

- One zIIP core assigned: 47.6% (smf70) (HMC, capacity planning/chargeback)
- Two concurrent threads: 92.8% (smf70, 46% average thread))
- Job data zIIP time 59% (smf30)
- Linux (total thread time): 88.4% (from Linux/SNMP)

Why is “zIIP on CP”, and why “GP CPU percent”?

- Thread time looks valid within 5%
- Prorate the SMF30 CPU time to “assigned” for SMT. ($59 / 47.6 = 1.24$)
- Solution to get accurate (or at least MUCH better) CPU data:
 - Prorate the SMF30 zIIP to / SMF70 HMC Assigned
 - Apply to Linux container CPU data from SNMP

zCX has swap defined

- Linux uses swap when short on storage
- When performance gets bad, is it swapping to disk?
- CMM is NOT an option for zCX
- IBM: Add storage to eliminate swap?
- **What is zCX plan for alerts for swapping?**
- **Note that we get swap space per process, so look at process table**

```
Report: ESAUCD2          LINUX UCD Memory Analysis Report
Monitor initialized: 10/27/23
-----
Node/      <-----Storage Sizes (MegaBytes)
Time/      <--Real Storage--> <-----SWAP Storage-----> Total
Date       Total  Avail  Used  Total  Avail  Used  MIN   Avail
-----
15:14:00
*** Nodes *****
zcxinst1 1407.0 312.5 1095 1980 1697 283.3 15.6 2009
```

If there is a “bug” or other problem?

- “reboot”???
- Add resources and reboot to hide the problem (for a while)?
- Understand and fix the problem? (30 “uwsgi-core” processes sleeping)
- **81 zombie processes....**

Report: **ESALNXP** LINUX Process Statistics Report Velocity Softw

node/ Name	<Process Ident>		<-----CPU Percents----->					<-----Stor			
	ID	PPID	Tot	sys	user	syst	usrt	Size	RSS	Peak	Swap
ZOSLP1	0	0	49.4	0.20	0.21	0.08	48.9	17K	332	17K	573
uwsgi-co	1177	1121	0.01	0	0.01	0	0	79	4	79.1	20.9
uwsgi-co	1178	1121	0	0	0	0	0	80	6	79.7	19.2
uwsgi-co	1179	1121	0	0	0	0	0	78	4	78.2	19.3
uwsgi-co	1180	1121	0	0	0	0	0	81	6	80.8	20.5
uwsgi-co	1181	1121	0.01	0	0.01	0	0	79	4	79.5	20.8
uwsgi-co	1182	1121	0	0	0	0	0	79	4	79.3	21.1
uwsgi-co	1183	1121	0	0	0	0	0	78	5	78.5	19.1
uwsgi-co	1184	1121	0	0	0	0	0	79	4	78.7	20.4
uwsgi-co	1185	1121	0	0	0	0	0	79	4	79.5	21.2
uwsgi-co	1186	1121	0	0	0	0	0	79	4	79.7	21.2
uwsgi-co	1187	1121	0	0	0	0	0	79	5	79.4	19.9
uwsgi-co	1188	1121	0.01	0	0.01	0	0	81	7	82.6	18.9

OpenShift / Kubernetes, Docker has the data for:

- Performance Analysis (zombies?)
- Operational Alerts (swapping)
- Capacity Planning
- Chargeback (accurate CPU?)

Data collection with SNMP

- Inexpensive
- Validated
- Measurable by container
- z/OS data needs effective prorate technology

Black boxes will be problematic

