



# Systems Behaving Badly

Darren Hoch – Director of Professional Services

# Performance Monitoring Introduction

## What This Tutorial Is:

- > Introduction to performance through systems architecture
- > Monitoring production boxes with standard tools
- > Quick methodologies to determine performance issues

Full Paper – <http://www.ufsdump.org>

# Performance Monitoring Introduction

Understand how Linux works

- > Seek to understand Linux before stats
- > System administrators not developers

Keep it simple

- > Use the tools that ship with the OS

Identify Linux Subsystems:

- > CPU, Memory, I/O, and Network

## Installing Monitoring Tools

Most systems ship with standard monitoring commands

- > vmstat
- > netstat
- > sysstat - sar, mpstat and iostat
- > ethtool
- > iptraf

Use system package managers

```
# yum install sysstat dstat iptraf
```

# Determine Baseline Statistics

## Baseline Statistics:

- > What should we expect?
- > Normal is relative to perception

```
# vmstat 1
r  b   swpd   free   buff  cache   si   so   bi   bo   in   cs us sy wa id

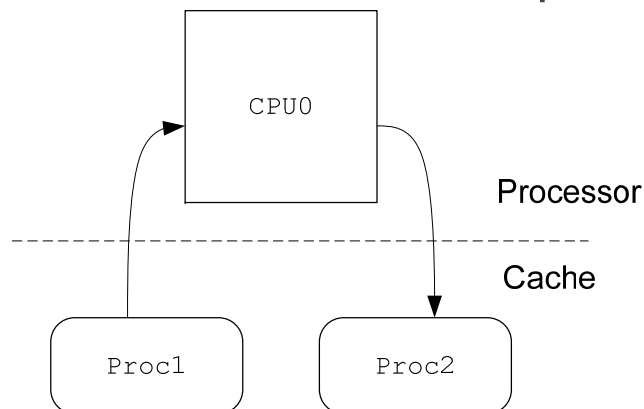
<Baseline>
0  0 138592 17932 126272 214244   0   0   0   0 117   49  0  0  0 100

<Utilized>
2  0 147092 13788 118600 212452   0 740   0 1324 620   61 92  8  0  0
```

# CPU Concepts

## Context Switches

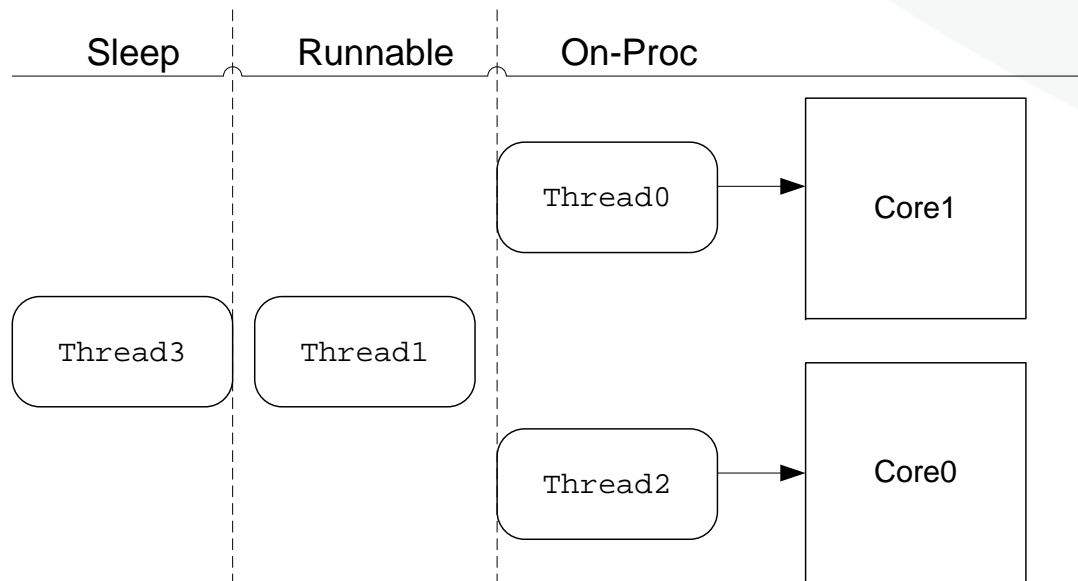
- > Everything process is a thread (single or multi)
- > The process of threads or interrupts (hardware actions) coming on and off the processor
- > High context switches = multiple busy threads



# CPU Concepts

## Run Queue

- > Runnable threads without a processor
- > Processor Load pulled from run queue



# CPU Concepts

## CPU Utilization

- > User Time - applications
- > System Time – kernel threads, interrupts
- > Wait on I/O – nothing runnable, all IO sleep
- > Idle – 0% - nothing going on

# CPU Performance Monitoring

## Run Queues

- > 1-3 threads per processor

## Utilization

- > 60 – 65% User
- > 30 – 35% System
- > 0 – 5% Idle

## Context Switching

- > Acceptable if within the utilization stats

# Case Study

```
# vmstat 1
```

procs			memory			swap		io		system			cpu		
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	wa	id
2	1	207740	98476	81344	180972	0	0	2496	0	150	300	4	12	57	27
0	1	207740	96448	83304	180984	0	0	1968	328	810	2559	8	9	83	0
0	1	207740	94404	85348	180984	0	0	2044	0	829	2879	9	6	78	7
0	1	207740	92576	87176	180984	0	0	1828	0	689	2088	3	9	78	10
2	0	207740	91300	88452	180984	0	0	1276	0	565	2182	7	6	83	4
3	1	207740	90124	89628	180984	0	0	1176	0	551	2219	2	7	91	0
4	2	207740	89240	90512	180984	0	0	880	520	443	907	22	10	67	0
5	3	207740	88056	91680	180984	0	0	1168	0	628	1248	12	11	77	0
4	2	207740	86852	92880	180984	0	0	1200	0	654	1505	6	7	87	0
6	1	207740	85736	93996	180984	0	0	1116	0	526	1512	5	10	85	0

- High Run Queue (r)
- User time 4% system time 10% wait on IO 86%
- Sustained context switches and interrupts
- Conclusion – The CPU is stalled out and queuing up threads while a disk (in) is reading (bo) data into memory

## Introducing Virtual Memory

### Virtual Memory Pages

- > 4KB “Pages”
- > Read and writes from disk
- > Swap + RAM = virtual memory
- > Applications reserve virtual memory at start
- > Kernel maps VM to physical RAM

## Introducing Virtual Memory Continued

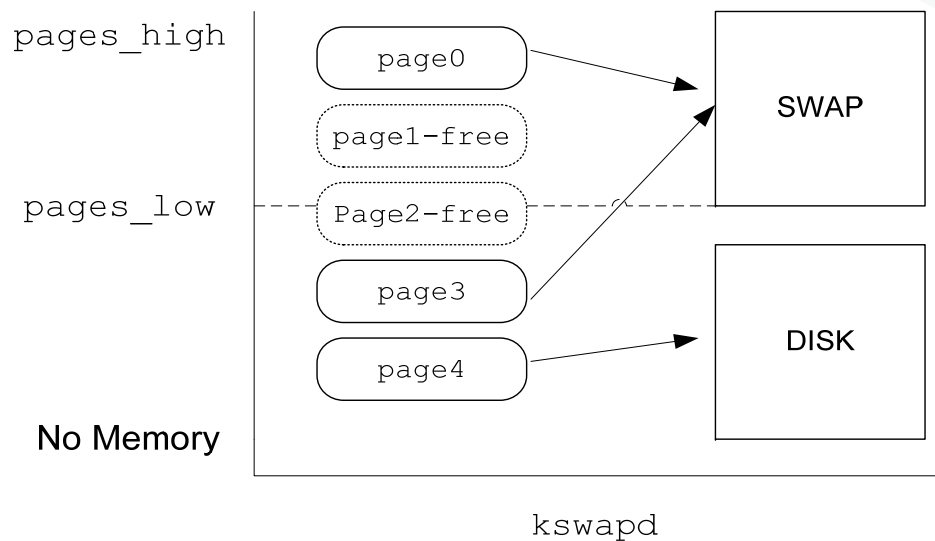
### Kernel Memory Paging

- > Kswapd checks low memory threshold
- > If low memory is below threshold, then scan for memory that can be “reclaimed” for other applications
- > Keep on scanning until memory is above high threshold
- > If kswapd can't keep up, it starts swapping to disk

# Introducing Virtual Memory Continued

## Kernel Memory Paging

- > Page Reclaiming
  - > Unreclaimable
  - > Swappable
  - > Syncable
  - > Discardable



# Case Study

```
# vmstat 3
```

procs			memory				swap		io		system			cpu		
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	
3	2	809192	261556	79760	886880	416	0	8244	751	426	863	17	3	6	75	
0	3	809188	194916	79820	952900	307	0	21745	1005	1189	2590	34	6	12	48	
0	3	809188	162212	79840	988920	95	0	12107	0	1801	2633	2	2	3	94	
2	3	809268	88756	79924	1061424	260	28	18377	113	1142	1694	3	5	3	88	
3	2	826284	17608	71240	1144180	100	6140	25839	16380	1528	1179	19	9	12	61	
6	1	854780	17688	34140	1208980	1	9535	25557	30967	1764	2238	43	13	16	28	
7	8	867528	17588	32332	1226392	31	4384	16524	27808	1490	1634	41	10	7	43	

- Applications request more virtual memory (increase in `swpd`, sustained blocks in or `bo`)
- The system is mapping more virtual memory into application space (`cache` increases)
- Free RAM levels out and does not increase (`free` stays at 17MB – low threshold)
- `kswapd` grabs pages from the buffer cache and tries to put them on the free list (`buff` decreases)
- `kswapd` can't keep find enough free memory, so it starts writing anonymous pages to swap (`so` increases)

## Introducing I/O Monitoring

### Reading and Writing Data

- > 4K Memory pages

```
# /usr/bin/time -v date
```

```
<snip>
```

```
Page size (bytes): 4096
```

```
<snip>
```

- > Minor fault – reclaim memory
- > Major fault – read/write to disk

# Introducing I/O Monitoring Continued

## Evolution – First Run

```
# /usr/bin/time -v evolution
<snip>
Major (requiring I/O) page faults: 163
Minor (reclaiming a frame) page faults: 5918
<snip>
```

## Evolution – Second Run

```
# /usr/bin/time -v evolution
<snip>
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 5581
<snip>
```

## IO Monitoring

### IO's Per Second (IOPS)

- > How many IO operation requests to the disk?
- > How big are those requests?
- > A 15K SAS drive should handle 250 – 300 IOPS

### Random Access Patterns

- > High IOPS, small requests
- > Web, mail, DNS servers

### Sequential Access Patterns

- > Low IOPS, large requests
- > RDBMS, imaging, rendering

# IO Monitoring

## Sequential Access Pattern - $53040/105 = 505\text{KB}$ per IO

```
# iostat -x 1
```

```
avg-cpu: %user   %nice   %sys    %idle
          0.00     0.00    0.00   57.14  42.86
```

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	svctm	%util
/dev/sda	0.00	12891.43	0.00	105.71	0.00	106080.00	0.00	53040.00	1003.46	1099.43	3442.43	26.49	280.00
/dev/sda1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
/dev/sda2	0.00	12857.14	0.00	5.71	0.00	105782.86	0.00	52891.43	18512.00	559.14	780.00	490.00	280.00
/dev/sda3	0.00	34.29	0.00	100.00	0.00	297.14	0.00	148.57	2.97	540.29	3594.57	24.00	240.00

## Random Access Pattern – $2640/102 = 23\text{KB}$ per IO

```
# iostat -x 1
```

```
avg-cpu: %user %nice %sys %idle
          2.04 0.00 97.96 0.00
```

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	svctm	%util
/dev/sda	0.00	633.67	3.06	102.31	24.49	5281.63	12.24	2640.82	288.89	73.67	113.89	27.22	50.00
/dev/sda1	0.00	5.10	0.00	2.04	0.00	57.14	0.00	28.57	28.00	1.12	55.00	55.00	11.22
/dev/sda2	0.00	628.57	3.06	100.27	24.49	5224.49	12.24	2612.24	321.50	72.55	121.25	30.63	50.00
/dev/sda3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

## IO Monitoring

### Good Performance

- > Random – IOPS equal to number of drives
- > Sequential – R/W size 500K+ for large files
- > A 15K SAS drive should handle 250 – 300 IOPS

### Bad Performance

- > Random – Too many IOPS for number of drives
- > Sequential – R/W size very small compared to drive and file sizes (8K IOP for 1GB file on 1TB SAN = bad)

# Example System Behaving Badly

```
# vmstat 1
procs -----memory----- ---swap-- -----io----- --system-- ----cpu----
 r  b      swpd   free  buff  cache   si   so    bi    bo   in cs   us sy id wa
17  0      1250   3248 45820 1488472    30 132   992    0 2437 7657 23 50  0 23
11  0      1376   3256 45820 1488888    57 245   416    0 2391 7173 10 90  0  0
12  0      1582   1688 45828 1490228    63 131  1348    76 2432 7315 10 90  0 10
12  2      3981   1848 45468 1489824   185  56  2300    68 2478 9149 15 12  0 73
14  2     10385   2400 44484 1489732     0  87  1112    20 2515 11620 0 12  0 88
14  2     12671   2280 43644 1488816    76  51  1812   204 2546 11407 20 45  0 35
```

1. There is a large amount of `bi`, signaling an intense read operation.
2. The read operation appears to be random due to the high `cs` and `int` values
3. The IO operations are depleting memory (`free` very low, `so/si` active but small)
4. The IO is enough that it is causing the CPU to stall (high `wa`)
5. Threads are backing up and not runnable due to the stall (high `run queue - r`)

## Possible Profiles:

- Web server dealing with too many inbound GET requests
- Web application making too many small read requests to an RDBMS
- Overloaded SMTP server