Managing Capacity and Performance in a Large Scale Production Environment

Presenter: Goranka Bjedov, Capacity Engineer, Facebook

Outline

1. Problem Description
2. Objectives/Goals
3. Start Simple/Small
4. Current System
5. What is Next?

Problem Description

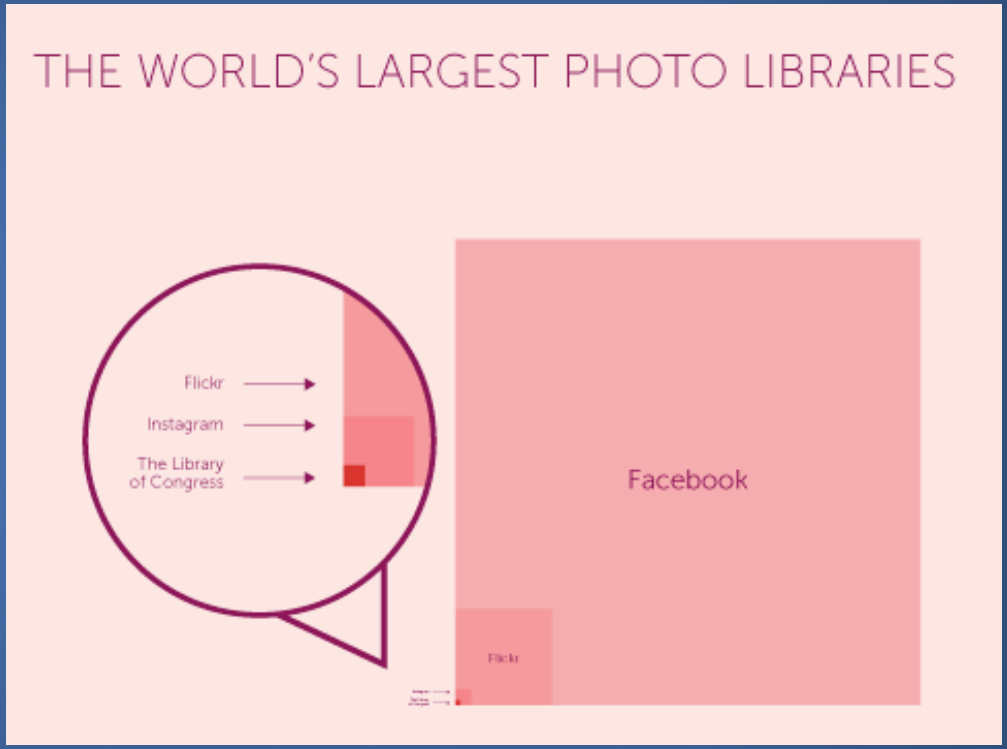
There is more to the story than meets the eye

- Facebook (> 1.7B MAU)
- Whatsapp (> 1B MAU)
- Messenger (> 1B MAU)
- Instagram (> 0.5B MAU)
- Oculus
- Internet.org

Photos (and Videos)

- User growth not reaching saturation
- Implicit contract to store content forever

How do you keep photos for 100 years?



What we do in 30 minutes...

- 3.8 trillion cache operations
- 108 billion queries run on MySQL
- 5 billion real-time messages sent
- 160 million newsfeed stories created
- 10 billion profile pictures served
- 50 million photos served

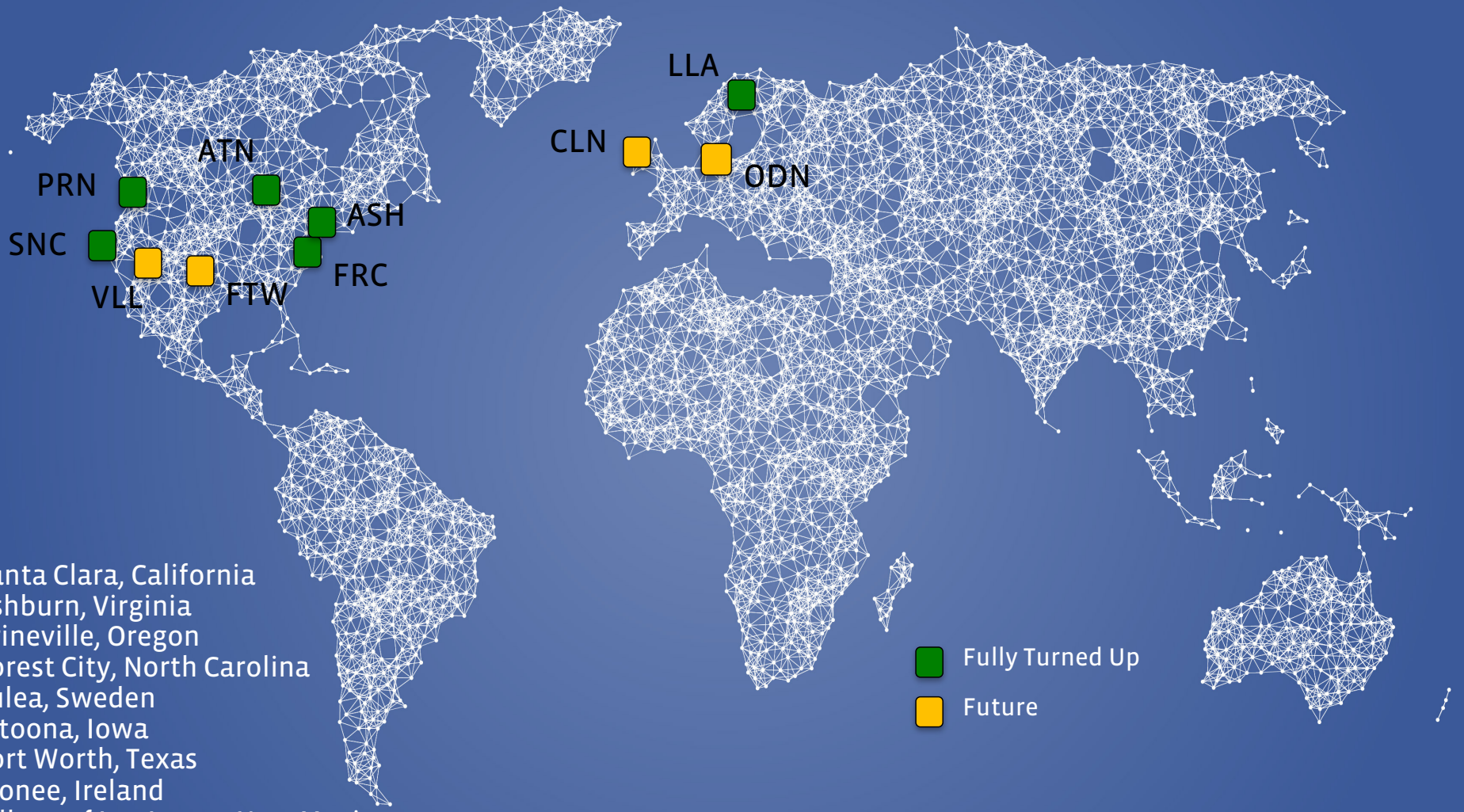
- 200 billion objects checked
- 300 million objects blocked
- 10 million photos uploaded

- 105TB of data is scanned via Hive
- 10TB of logs loaded into Hadoop

- 225 TB network egress

* Numbers based on summer 2013

Facebook Private Cloud



SNC: Santa Clara, California
ASH: Ashburn, Virginia
PRN: Prineville, Oregon
FRC: Forest City, North Carolina
LLA: Lulea, Sweden
ATN: Altoona, Iowa
FTW: Fort Worth, Texas
CLN: Clonee, Ireland
VLL: Village of Los Lunas, New Mexico

2017 Servers (opencompute.org)

Standard Systems	I Web	III Database	IV Hadoop	V Haystack	VI Feed	VII Cold Storage
CPU	High Broadwell (16c) 1.8 GHz	High 2 x Broadwell (14c) 2.4 GHz	High 2 x Broadwell (14c) 2.4 GHz	Low 1 x Avoton (8c) 2.7 GHz	High 2 x Broadwell (14c) 2.4GHz	High 2 x Haswell (12c) 2.5 GHz
Memory	Low 32GB	High 256GB	Medium 128GB	Low 32GB	High 256GB	Medium 128GB
Disk	Low 256 GB Flash	High IOP 2 x 3.2TB Flash 128 GB mSata	High 15 x 8TB NL SAS	High 30 x 8TB NL SAS	Medium 2TB SAS (1.6 TB FLASH optional)	Very High 240 X 8TB 16 active disks
Number Servers/ Rack	120	30	18	9	30	6

2015 Servers (opencompute.org)

Standard Systems	I Web	III Database	IV Hadoop	V Haystack	VI Feed	VII Cold Storage
CPU	High 2 x Haswell (12c) 2.5 GHz	High 2 x Haswell (12c) 2.5 GHz	High 2 x Haswell (12c) 2.5 GHz	Low 1 x Avoton (8c) 2.7 GHz	High 2 x Haswell (12c) 2.5GHz	High 2 x Haswell (12c) 2.5 GHz
Memory	Low 32GB	High 256GB	Medium 128GB	Low 32GB	High 256GB	Medium 128GB
Disk	Low 500GB SATA	High IOP 2 x 3.2TB Flash 128 GB mSata	High 15 x 4TB (or 6TB) NL SAS	High 30 x 4TB (or 6 TB) NL SAS	Medium 2TB SAS (1.8 TB FLASH optional)	Very High 240 X 4TB 16 active disks
Number Servers/ Rack	30	30	18	9	30	6

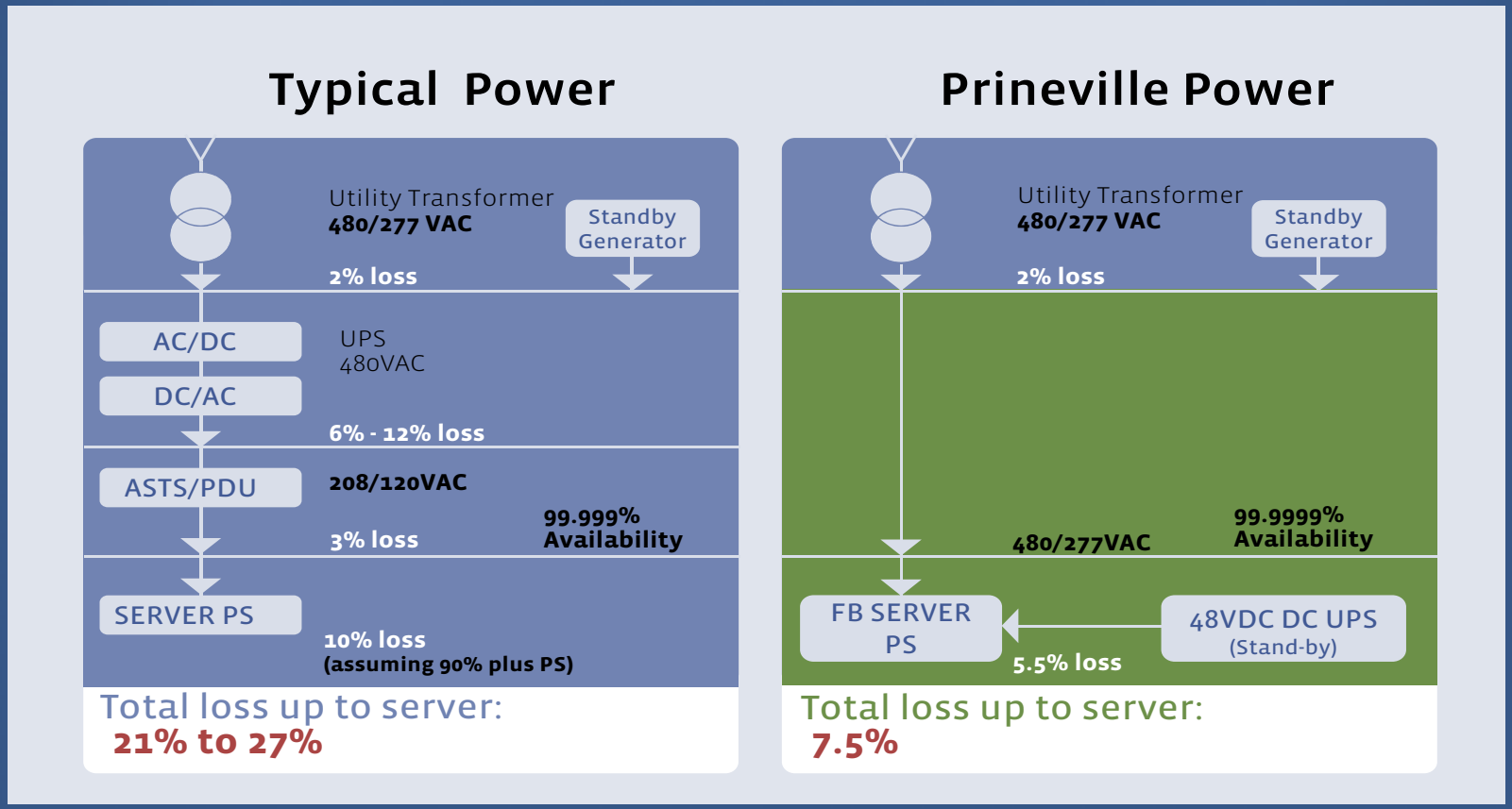
Server Generations

Type I Web Servers	2007	2008	2009	2010	2011	2012	2014	2015	2017
Rack Compo-sition	L5420 (VSS) Fully-buffered Dimms	L5420 (SC)	L5520 (NHM)	L5639 (WSM)	X5650 (XWSM)	E5-2670 (SNB)	E5-2680 (IVB)	Haswell	Broadwell
Cores / Speed	8 real cores 2.50 GHz	8 real cores 2.50 GHz	16 logical CPUs 2.27 GHz	24 logical CPUs 2.13 GHz	24 logical CPUs 2.67 GHz	32 logical CPUs 2.67 GHz	40 logical CPUs 2.8 GHz	48 logical CPUs 2.5 GHz	32 logical CPUs 1.8 GHz
RCUs	0.4	0.6	1	1.4	1.75	2.42	X	X	X

Objectives/Goals
Not getting any simpler

- Low latency for users
- Launch things quickly
  - Succeed or fail quickly
  - Don’t worry about efficiency
- Conserve Resources
  - Power
  - Money
  - Computers (RAM, CPU)
  - Network
  - Developer Time

Power Efficiency



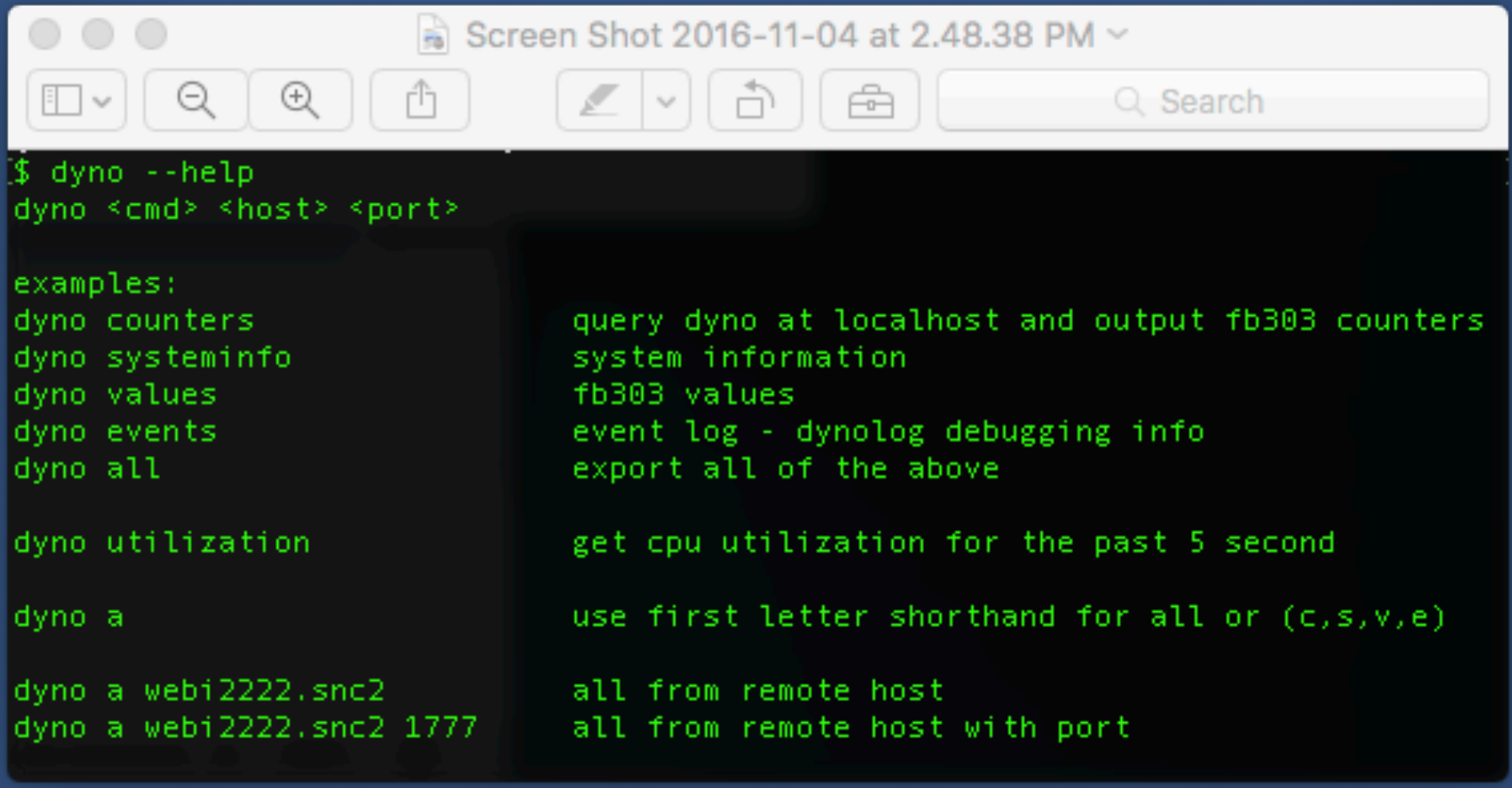
My Goals
Finally something simple

- Want (only) right things running on (only) right gear
- Want things to run efficiently
- Want to know if something is broken
- Want to know if something is about to break
- Want to know how (and why) things are growing (or not)
- Want to know who to blame… ☺

Approach
Start small

- Monitoring Daemon (dynolog)
  - Runs on all servers
  - Collects system/kernel data in one second granularity
  - Slim high-throughput aggregator and streaming Thrift service
  - Exports data to scribe, hadoop, ods, etc.
- What Data Does it Collect
  - Default: CPU, RAM, Network, IO, disk stats…
  - Customized: Requests + App performance

Dyno

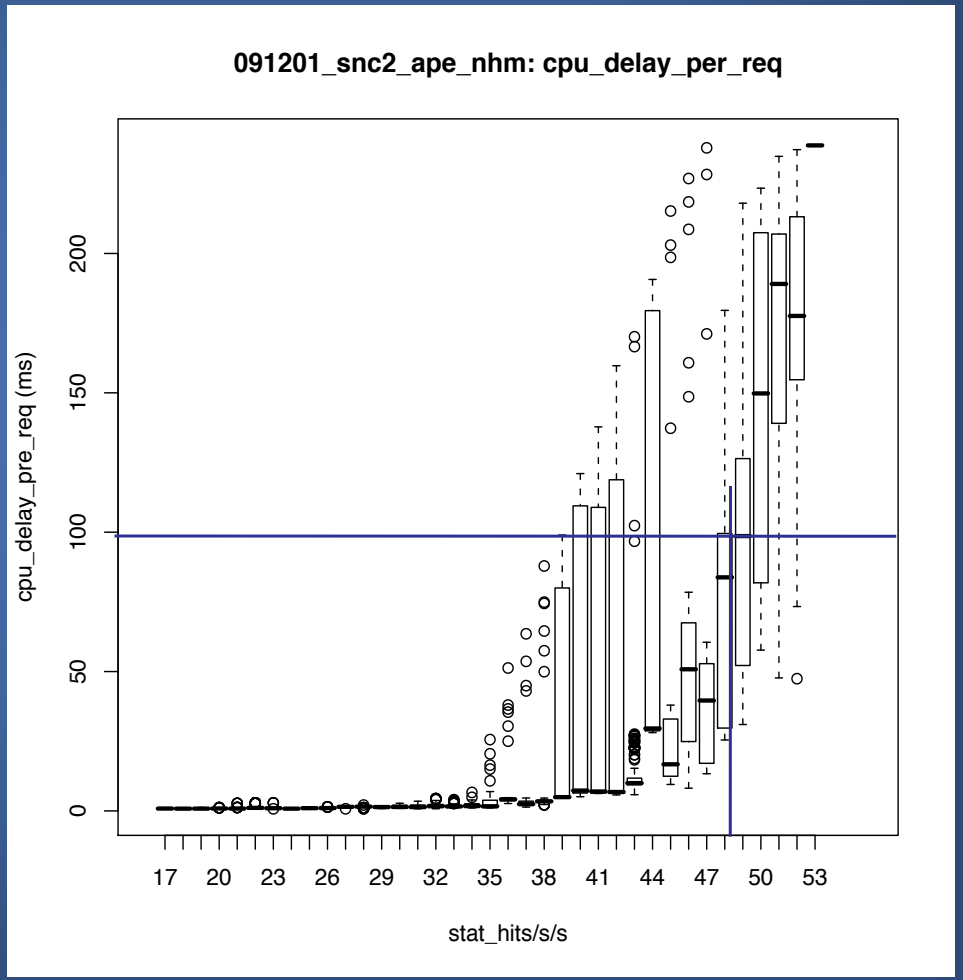

Screen Shot 2016-11-04 at 2.48.38 PM
Search
$ dyno --help
dyno <cmd> <host> <port>
examples:
dyno counters query dyno at localhost and output fb303 counters
dyno systeminfo system information
dyno values fb303 values
dyno events event log - dynolog debugging info
dyno all export all of the above
dyno utilization get cpu utilization for the past 5 second
dyno a use first letter shorthand for all or (c,s,v,e)
dyno a webi2222.snc2 all from remote host
dyno a webi2222.snc2 1777 all from remote host with port

Application level (Web)

```
duration_avg_ms: 132
duration_cpu_avg_ms: 8
duration_ms_p25: 4
duration_ms_p50: 8
duration_ms_p75: 13
duration_ms_p95: 18
duration_ms_p99: 18
dynolog_uptime_min: 2942
ego_renders_per_hit: 0
ego_renders_per_sec: 0
instructions_hit_percent_of_busy: 8
instructions_per_cpu_count: 252416
instructions_per_cpu_sec: 937187000
instructions_per_cpu_sec_p25: 263233135
instructions_per_cpu_sec_p50: 426773721
instructions_per_cpu_sec_p75: 639600511
instructions_per_cpu_sec_p95: 1056292963
instructions_per_kcycle: 621
```

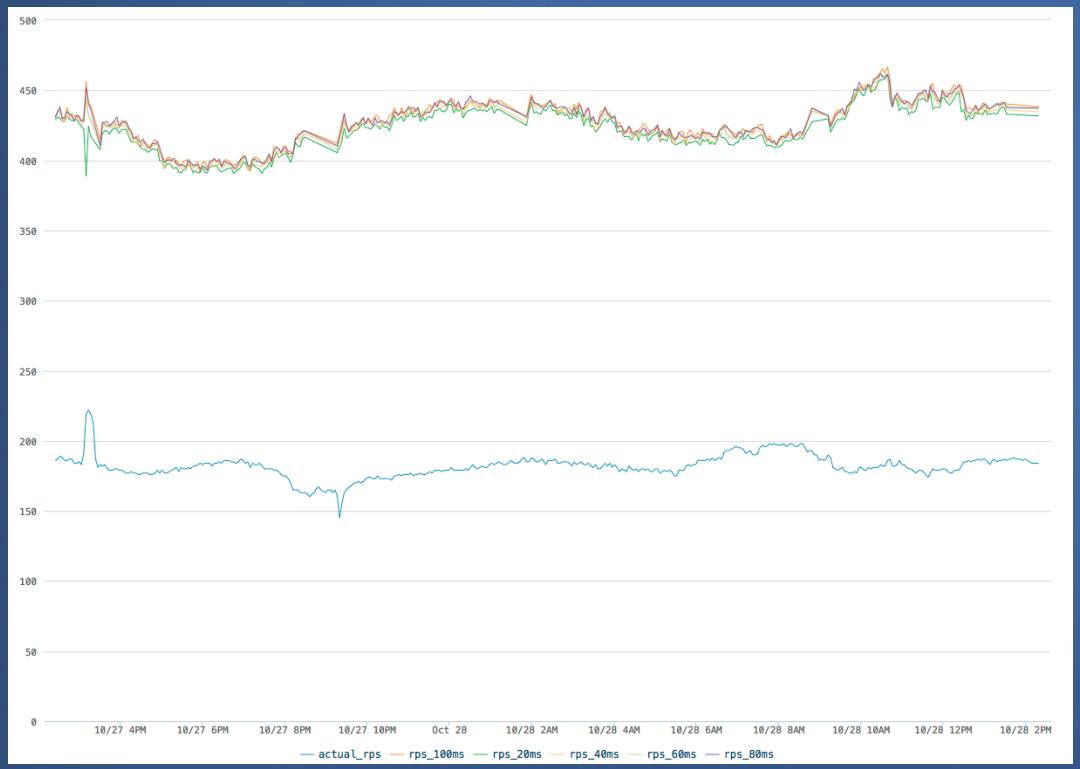
Measuring web capacity

- Hit duration increases with hit rate
- Defined 100 milliseconds of contention induced latency as 100% utilization—any more latency is easily perceived by the user
- The increase in duration comes from the process waiting in the kernel run queue.
- The NHM server to the right tops out at 48 hits/s



Custom Monitoring

More info: https://www.facebook.com/note.php?note_id=203367363919

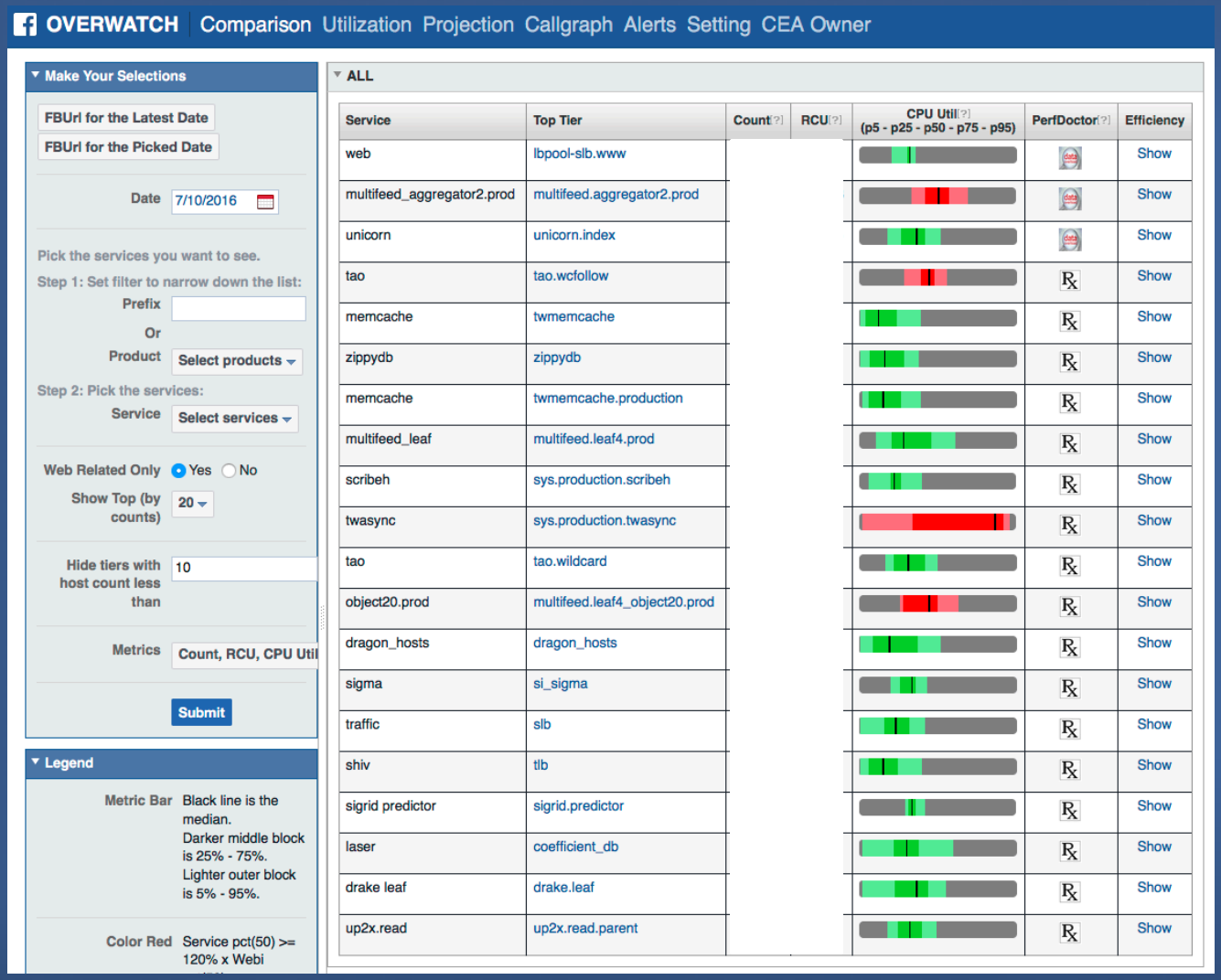


- InstantDyno:
  - Listen to dyno thrift interface
  - Make correlations between pairs of metrics
  - Adjust load on loadbalancers
- Live loadtests
- 24/7
- Every day, every second

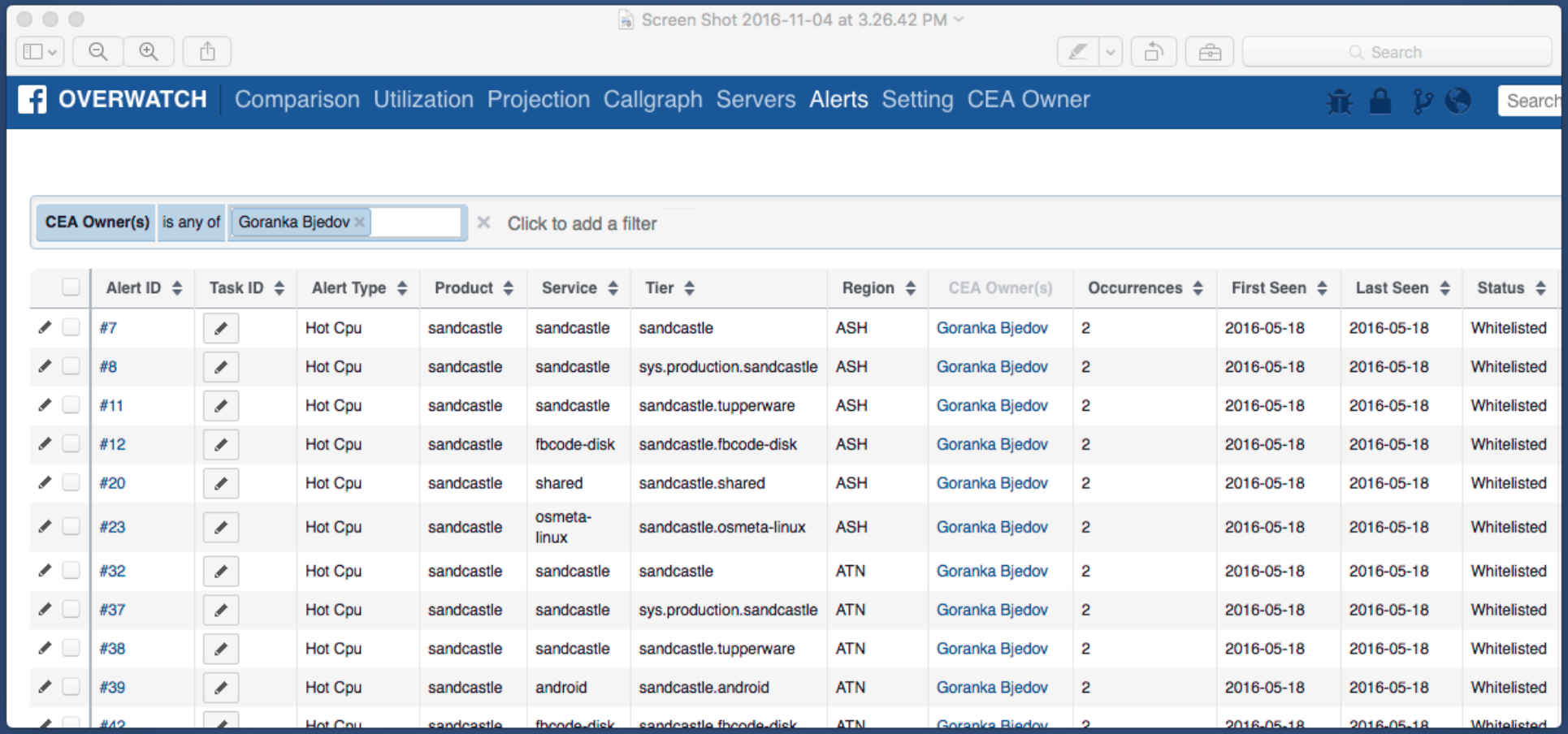
What Questions Can I Answer

- How is a group of servers performing?
  - Need to know which servers form a “group”
  - Need to process/organize/present that data
  - Our answer: Overwatch
- How is the code on those servers performing?
  - Is the code uniform?
  - CPU (instructions, cycles, etc.)
  - RAM
  - Disk, network, TOR, etc.
  - Our answer: Perf Doctor

Overwatch - Comparison across services



Overwatch - Alerting



Alert ID	Task ID	Alert Type	Product	Service	Tier	Region	CEA Owner(s)	Occurrences	First Seen	Last Seen	Status
#7		Hot Cpu	sandcastle	sandcastle	sandcastle	ASH	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#8		Hot Cpu	sandcastle	sandcastle	sys.production.sandcastle	ASH	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#11		Hot Cpu	sandcastle	sandcastle	sandcastle.tupperware	ASH	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#12		Hot Cpu	sandcastle	fbcode-disk	sandcastle.fbcode-disk	ASH	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#20		Hot Cpu	sandcastle	shared	sandcastle.shared	ASH	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#23		Hot Cpu	sandcastle	osmeta-linux	sandcastle.osmeta-linux	ASH	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#32		Hot Cpu	sandcastle	sandcastle	sandcastle	ATN	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#37		Hot Cpu	sandcastle	sandcastle	sys.production.sandcastle	ATN	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#38		Hot Cpu	sandcastle	sandcastle	sandcastle.tupperware	ATN	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#39		Hot Cpu	sandcastle	android	sandcastle.android	ATN	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted
#42		Hot Cpu	sandcastle	fbcode-disk	sandcastle.fbcode-disk	ATN	Goranka Bjedov	2	2016-05-18	2016-05-18	Whitelisted

Dyno and Perf Doctor

- Perf Doctor consumes nearly all of Dyno's system counters
- Raw time-series values are used to determine:
  - Does a tier have performance issues?
  - What is possible remediation?
- Breakdown done by machine architecture
- Most counters are also plotted

Dyno system counters in ODS

CPU

- CPU instructions
- CPU cycles
- Instructions-per-cycle
- CPU frequency
- HHVM foreground inst
- CPU core count
- Turbo status

Memory

- Memory bandwidth
- Local memory bw
- Remote memory bw
- Memory bw utilization

Kernel

- Interrupts etho per sec
- Interrupts
- tlbshootdowns
- CPU busy
- Page faults
- Context switches

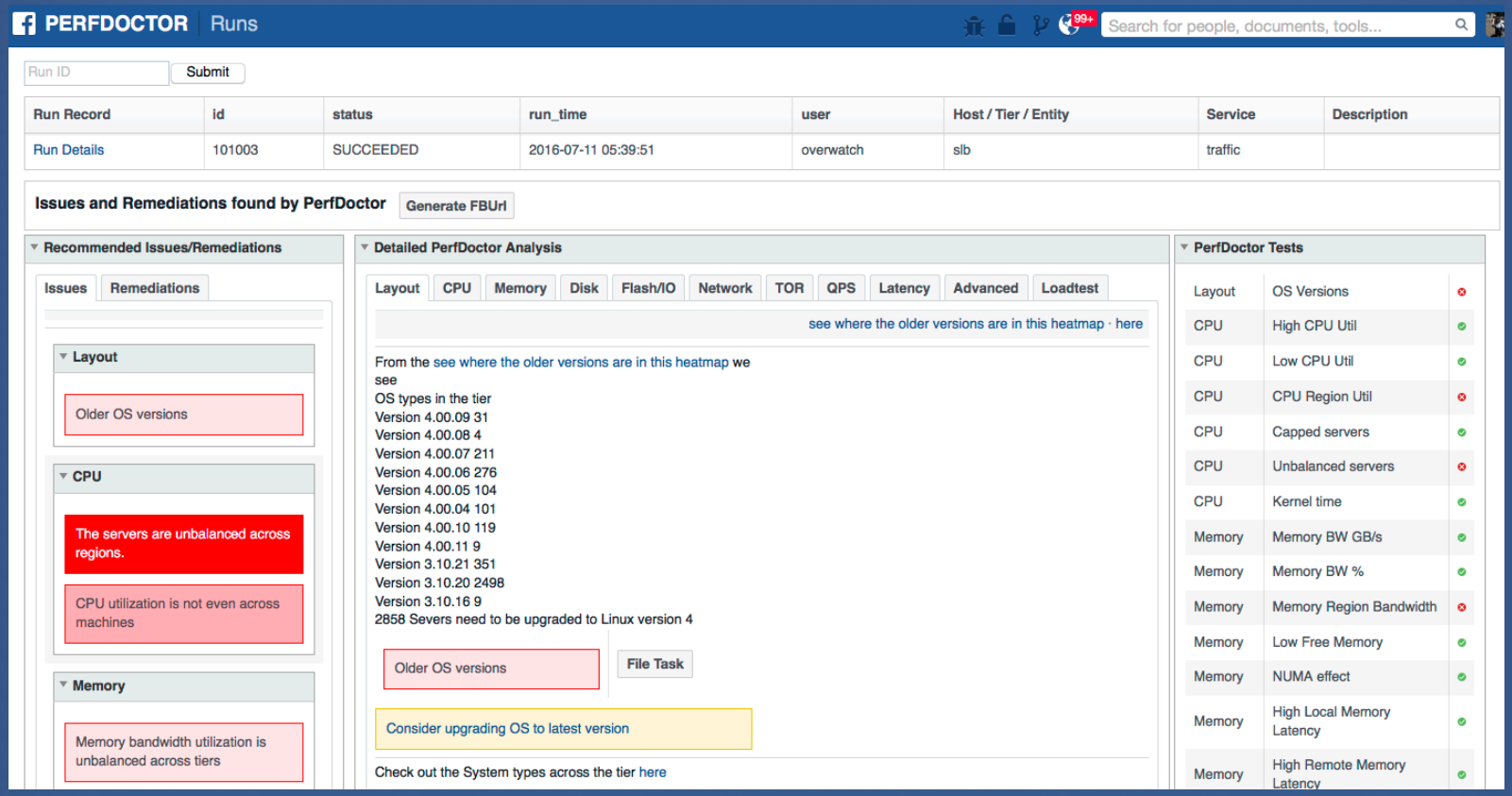
Storage

- Disk read/write iops
- Flash/disk rd/wr bytes
- Flash/disk rd/wr ms
- Flash/disk rd/wr ios

Network

- Tx/Rx packets
- Tx/Rx bytes
- Tx/Rx errors
- Tx/Rx drops

PerfDoctor - Is A Service Healthy?



PerfDoctor - Calculating CPI

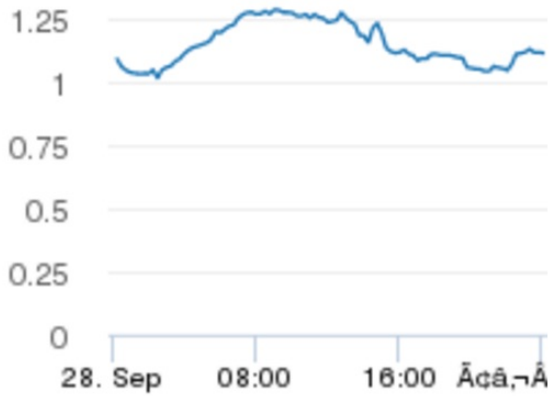
Layout | CPU | Memory | Disk | Flash/IO | Network | TOR | QPS | Latency | Advanced | Loadtest

Max CPI graph · Context switch graph · TLB Shootdowns graph · Lock count · System Calls · LLC Misses · LLC Misses (Hot Functions) · DTLB Misses · dTLB Misses (Hot Functions) · ITLB Misses · Branch Mispredictions · Branch Mispreds (Hot Functions) · Sharing · User cycles (Hot Functions) · Kernel cycles (Hot Functions) · Longma hot process experiment · RCU Distribution

From the Max CPI graph we see
Peak CPI is 1.29.
This is high

The CPI is high for the Application

File Task



Consider using text on huge pages

PerfDoctor - Calculating DTLB miss rate

From the dTLB Misses (Hot Functions) we see
The hottest functions causing dTLB misses are

free	15.71%
std::_MakeUniq<facebook::unicorn::internal_layeredindex::LayeredIndexIterator<facebook::unicorn::int	12.23%

dTLB miss rate is high on hot functions

File Task

LongMa Deep Dive: dTLB Misses (Hot Functions)

Check the hot functions for high dTLB misses and see if huge pages can be used

PerfDoctor - Calculating LLC miss rate

From the LLC Misses we see
The Last Level Cache miss rate is 0.65
This is acceptable

LongMa Deep Dive: LLC Misses

From the LLC Misses (Hot Functions) we see
The hottest functions causing last level cache misses are

copy_page_range	6.76%
page_remove_rmap	4.68%
copy_page_rep	4.33%

LongMa Deep Dive: LLC Misses (Hot Functions)

LLC miss rate is high on hot functions

File Task

Check the hot functions for high LLC misses and optimize memory behavior

Deep Dive - Longma

- A tool to simplify the collection of performance data, store it and present it in accessible way
- Can use INTEL's EMON to get low-level hardware counters
- Interprets hardware counters and presents the information

EMON Counters

4	(EDP 1.5a) name (sample #1 - #32382) all events/metrics are normalized	aggregated	95th percentil
5	metric_CPU operating frequency (in GHz)	2.1996	2.2000
6	metric_CPU utilization %	43.6180	76.0868
7	metric_CPU utilization% in kernel mode	4.1249	5.9215
8	metric_CPI	1.3206	1.5279
9	metric_kernel_CPI	3.0300	3.5594
10	metric_% cycles uops retired	28.9780	32.9625
11	metric_branch mispredict ratio	0.0324	0.0365
12	metric_loads per instr	0.2685	0.2758
13	metric_stores per instr	0.1519	0.1694
14	metric_locks retired per instr	0.0009	0.0013
15	metric_uncacheable reads per instr	0.0000	0.0000
16	metric_streaming stores (full line) per instr	0.0000	0.0000
17	metric_streaming stores (partial line) per instr	0.0000	0.0000
18	metric_L1D MPI (includes data+rfo w/ prefetches)	0.0199	0.0300
19	metric_L1D demand data read hits per instr	0.2546	0.2605
20	metric_L1-I code read misses (w/ prefetches) per instr	0.0086	0.0102
21	metric_L2 demand data read hits per instr	0.0043	0.0057
22	metric_L2 MPI (includes code+data+rfo w/ prefetches)	0.0210	0.0503
23	metric_L2 demand data read MPI	0.0045	0.0103
24	metric_L2 demand code MPI	0.0032	0.0041
25	metric_L2 Any local request that HITM in a sibling core (per instr)	0.0001	0.0001
26	metric_L2 read miss latency (in core clocks) - BROKEN	78.4953	110.9336
27	metric_LLC MPI (includes code+data+rfo w/ prefetches)	0.0029	0.0041
28	metric_LLC data read MPI (includes prefetches)	0.0020	0.0027
29	metric_LLC code read MPI (includes prefetches)	0.0002	0.0002
30	metric_LLC LLC prefetch data read MPI	0.0000	0.0000
31	metric_LLC LLC prefetch RFO read MPI	0.0002	0.0003
32	metric_LLC LLC prefetch code read MPI	0.0000	0.0000
33	metric_LLC total HITM (per instr)	0.0002	0.0002
34	metric_LLC total HIT clean line forwards (per instr)	0.0001	0.0002
35	metric_LLC writebacks per instr	0.0011	0.0015
36	metric_Average LLC data read (demand+prefetch) miss latency (in ns)	119.4158	122.3266
37	metric_Average LLC data read (demand+prefetch) miss latency (in core	262.6712	268.8380
38	metric_ITLB MPI	0.0003	0.0004
39	metric_ITLB large page MPI	0.0000	0.0000
40	metric_DTLB large page load MPI	0.0002	0.0003
41	metric_DTLB store MPI	0.0002	0.0003
42	metric_DTLB store miss latency (in core clks)	55.1831	60.5893

43	metric_NUMA %_Reads satisfied by local DRAM (LLC prefetches exclude	43.7154	46.9211
44	metric_NUMA %_Reads satisfied by remote DRAM (LLC prefetches exclu	45.9279	49.7454
45	metric_NUMA %_Reads satisfied by remote caches (Hitm+HitF; LLC pref	10.3566	17.2653
46	metric_NUMA %_Reads that are code misses and satisfied by remote ca	1.0181	2.1003
47	metric_NUMA %_Reads that are code misses and satisfied by remote m	1.8602	3.8883
48	metric_QPI speed - GT/s	7.9949	7.9996
49	metric_QPI Data transmit BW (MB/sec) (only data)	2,973.5779	4,601.8066
50	metric_QPI total transmit BW (MB/sec) (includes control)	4,464.1978	6,935.6200
51	metric_QPI Transmit utilization_% (includes control)	6.9798	9.2192
52	metric_QPI % cycles transmit link is half-width (L0p)	0.0000	0.0000
53	metric_QPI % cycles receive link is half-width (L0p)	0.0000	0.0000
54	metric_QPI % cycles receive link is in L0s sleep state	0.0000	0.0000
55	metric_QPI % cycles transmit link is in L0s sleep state	0.0000	0.0000
56	metric_HA - Reads vs. all requests	0.7200	0.7497
57	metric_HA - Writes vs. all requests	0.2800	0.3083
58	metric_DDR data rate (MT/sec)	1,332.6932	1,333.5669
59	metric_memory bandwidth read (MB/sec)	4,334.6072	6,796.0721
60	metric_memory bandwidth write (MB/sec)	1,869.1081	2,910.5233
61	metric_memory bandwidth total (MB/sec)	6,203.7153	9,677.4048
62	metric_IO_bandwidth_disk_or_network_writes (MB/sec)	34.6664	57.8164
63	metric_IO_bandwidth_disk_or_network_reads (MB/sec)	34.6204	59.0679
64	metric_IO_number of partial PCI writes per sec	0.4792	0.0000
65	metric_IO_number of partial PCI reads per sec	0.1846	0.0000
66	metric_IO_write cache miss(disk/network reads) bandwidth (MB/sec)	32.3574	55.7779
67	metric_IO_read cache miss(disk/network writes) bandwidth (MB/sec)	33.6207	56.5899
68	metric_memory reads vs. all requests	0.6987	0.7266
69	metric_memory Page Misses vs. all requests	0.0837	0.1008
70	metric_memory % Cycles where all DRAM ranks are in PPD mode	39.1766	53.6138
71	metric_memory % Cycles all ranks in critical thermal throttle	0.0000	0.0000
72	metric_memory % Cycles Memory is in self refresh power mode	0.0007	0.0000
73	metric_power % cycles max frequency limited by thermal issues	0.0000	0.0000
74	metric_power % cycles max frequency limited by OS	100.0000	100.0000
75	metric_power % cycles max frequency limited by power	0.0000	0.0000
76	metric_ItoM operations (fast strings) that reference LLC per instr	0.0003	0.0004
77	metric_ItoM operations (fast strings) that miss LLC per instr	0.0002	0.0003
78	metric_Topdown Frontend bound (%)	19.2757	26.2533
79	metric_Topdown Retiring (%)	20.4631	23.9712
80	metric_Topdown Bad speculation (%)	6.1023	7.3639
81	metric_Topdown Backend bound (%)	54.1589	63.6967

Longma – Process Count

Find the hottest functions and get a breakdown of memory and CPU usage by process and thread.

Process Counters

Process	Thread Name	Thread ID	User Time %	User process %	User Start %	Kernel Time %	Kernel process %	Kernel Start %	Soft IRQ	Rss MB	Dirty MB	Size MB
Total	Total	0	27.69	0	0	2.45	0	0	0.43	0	0	0
leaf_main	leaf_main	3833196	14.18	100	100	0.44	100	100	0	130796	130715	156156
aggregator	aggregator	4059488	10.46	100	100	0.68	100	100	0	1227.66	1176.11	7682.04
aggregator	(PYMKAggregator-)	4061437	1.85	17.65	7.2	0.02	2.81	1.99	0	0	0	0
aggregator	(PYMKAggregator-)	4061426	1.77	16.89	3.96	0.04	5.62	1.13	0	0	0	0
aggregator	(PYMKAggregator-)	4061428	1.68	16.06	3.77	0.03	5.06	1.09	0	0	0	0
leaf_main	(PYMKLeaf-pri3-2)	3885496	1.65	11.64	3.64	0.01	2.22	0.95	0	0	0	0
leaf_main	(PYMKLeaf-pri3-1)	3885481	1.53	10.81	1.89	0.02	4.44	0.56	0	0	0	0
leaf_main	(PYMKLeaf-pri3-2)	3885470	1.42	10.02	4.01	0	1.11	1.01	0	0	0	0
leaf_main	(PYMKLeaf-pri3-1)	3885486	1.4	9.88	3.8	0.01	3.33	0.98	0	0	0	0
leaf_main	(PYMKLeaf-pri3-4)	3885472	1.4	9.85	3.43	0.01	2.22	0.91	0	0	0	0
leaf_main	(PYMKLeaf-pri3-1)	3885485	1.36	9.6	3.92	0	1.11	0.98	0	0	0	0
aggregator	(PYMKAggregator-)	4061432	1.36	13.01	4.43	0.02	2.81	1.29	0	0	0	0
leaf_main	(PYMKLeaf-pri3-1)	3885469	1.31	9.26	3.38	0.01	2.22	0.9	0	0	0	0
leaf_main	(PYMKLeaf-pri3-9)	3885477	1.29	9.12	2.85	0.01	2.22	0.78	0	0	0	0
leaf_main	(PYMKLeaf-pri3-8)	3885476	1.28	9.05	3.32	0.01	3.33	0.87	0	0	0	0
aggregator	(PYMKAggregator-)	4061427	1.15	11.02	4.21	0.02	2.81	1.22	0	0	0	0
leaf_main	(PYMKLeaf-pri3-2)	3885494	1.1	7.75	3.13	0	1.11	0.83	0	0	0	0
aggregator	(PYMKAggregator-)	4061430	1.09	10.4	5.06	0.02	2.81	1.47	0	0	0	0
leaf_main	(PYMKLeaf-pri3-2)	3885488	0.92	6.51	2.7	0	1.11	0.71	0	0	0	0
leaf_main	(PYMKLeaf-pri3-3)	3885499	0.85	6.02	3.88	0	1.11	0.99	0	0	0	0
leaf_main	(PYMKLeaf-pri3-2)	3885490	0.82	5.75	3.42	0	1.11	0.89	0	0	0	0
leaf_main	(PYMKLeaf-pri3-5)	3885473	0.59	4.13	3.11	0.01	2.22	0.81	0	0	0	0
aggregator	(PYMKAggregator-)	4061431	0.59	5.65	8.48	0.01	1.12	2.44	0	0	0	0
aggregator	(TierUpdNotify)	4059976	0.52	4.93	1.14	0.04	5.62	0.68	0	0	0	0

Other daemons

- I can still not tell if (only) the right things are running on the right hardware
- Want to be able to tell automatically why a particular (set of) machine(s) behaves differently at a certain time
- Our solution: two more daemons
  - atop daemon: take atop every X seconds, store output on a local drive
  - Automatically analyze and correlate output
  - Strobelight: get all stacktraces every Y minutes and analyze those

Questions?