Cloud Performance Root Cause Analysis at Netflix

Brendan Gregg
Senior Performance Architect
Cloud and Platform Engineering
Experience: CPU Dips
# perf record -F99 -a
# perf script

[...]
java 14327 [022] 252764.179741: cycles: 7f36570a4932 SpinPause (/usr/lib/jvm/java-8
java 14315 [014] 252764.183517: cycles: 7f36570a4932 SpinPause (/usr/lib/jvm/java-8
java 14310 [012] 252764.185317: cycles: 7f36570a4932 SpinPause (/usr/lib/jvm/java-8
java 14332 [015] 252764.188720: cycles: 7f3658078350 pthread_cond_wait@@GLIBC_2.3.2
java 14341 [019] 252764.191307: cycles: 7f3656d150c8 ClassLoaderDataGraph::do_unloa
java 14341 [019] 252764.198825: cycles: 7f3656d140b8 ClassLoaderData::free_dealloca
java 14341 [019] 252764.207057: cycles: 7f3657192400 nmethod::do_unloading(BoolObje
java 14341 [019] 252764.215962: cycles: 7f3656ba807e Assembler::locate_operand(uns
java 14341 [019] 252764.225141: cycles: 7f36571922e8 nmethod::do_unloading(BoolObje
java 14341 [019] 252764.234578: cycles: 7f3656ec4960 CodeHeap::block_start(void*) c
[...]
1. Click start
2. Click end
3. See flame graph for that range
Observability
Methodology
Velocity
Root Cause Analysis at Netflix

Devices

Zuul 2 → Load

ELB → ASG Cluster → Netflix

Asg 1 → ASG 2 → SG

Roots

Atlas
Chronos
Zipkin
Vector

Eureka

Ribbon
Hystrix

gRPC

Service

Tomcat

JVM

Instances (Linux)

AZ 1
AZ 2
AZ 3

sar, *stat
ftrace
bcc/eBPF
bpftrace
PMCs, MSRs
Agenda

1. The Netflix Cloud

2. Methodology

3. Cloud Analysis

4. Instance Analysis
Since 2014

Asgard → Spinnaker
Salp → Zipkin
gRPC adoption
New Atlas UI & Lumen
Java frame pointer
eBPF bcc & bpftrace
PMCs in EC2

From Clouds to Roots (2014 presentation): Old Atlas UI
>150k AWS EC2 server instances

~34% US Internet traffic at night

>130M members

Performance is customer satisfaction & Netflix cost
Acronyms

AWS: Amazon Web Services
EC2: AWS Elastic Compute 2 (cloud instances)
S3: AWS Simple Storage Service (object store)
ELB: AWS Elastic Load Balancers
SQS: AWS Simple Queue Service
SES: AWS Simple Email Service
CDN: Content Delivery Network
OCA: Netflix Open Connect Appliance (streaming CDN)
QoS: Quality of Service
AMI: Amazon Machine Image (instance image)
ASG: Auto Scaling Group
AZ: Availability Zone
NIWS: Netflix Internal Web Service framework (Ribbon)
gRPC: gRPC Remote Procedure Calls
MSR: Model Specific Register (CPU info register)
P PMC: Performance Monitoring Counter (CPU perf counter)
eBPF: extended Berkeley Packet Filter (kernel VM)
1. The Netflix Cloud
The Netflix Cloud

- ELB
- EC2
- Applications (Services)
- Cassandra
- Elasticsearch
- EVCache
- S3
- SES
- SQS
Freedom and Responsibility

- Culture deck memo is true
  - https://jobs.netflix.com/culture
- Deployment freedom
  - Purchase and use cloud instances without approvals
  - Netflix environment changes fast!
Cloud Technologies

- Usually open source
- Linux, Java, Cassandra, Node.js, ...
- [http://netflix.github.io/](http://netflix.github.io/)
Cloud Instances

Linux (Ubuntu)
- Optional Apache, memcached, non-Java apps (incl. Node.js, golang)
- Atlas monitoring, S3 log rotation, ftrace, perf, bcc/eBPF

Java (JDK 8)
- GC and thread dump logging
- Tomcat
  - Application war files, base servlet, platform, hystrix, health check, metrics (Servo)

Typical BaseAMI
5 Key Issues

And How the Netflix Cloud is Architected to Solve Them
1. Load Increases → Auto Scaling Groups

- Instances automatically added or removed by a custom scaling policy
- Alerts & monitoring used to check scaling is sane
- Good for customers: Fast workaround
- Good for engineers: Fix later, 9-5
2. Bad Push → ASG Cluster Rollback

- ASG red black clusters: how code versions are deployed
- Fast rollback for issues
- Traffic managed by Elastic Load Balancers (ELBs)
- Automated Canary Analysis (ACA) for testing
3. Instance Failure → **Hystrix Timeouts**

- Hystrix: latency and fault tolerance for dependency services
  - Fallbacks, degradation, fast fail and rapid recovery, timeouts, load shedding, circuit breaker, realtime monitoring
- Plus Ribbon or gRPC for more fault tolerance
4. Region failure → **Zuul 2 Reroute Traffic**

- All device traffic goes through the Zuul 2 proxy: dynamic routing, monitoring, resiliency, security
- Region or AZ failure: reroute traffic to another region
5. Overlooked Issue → **Chaos Engineering** (Resilience)

- **Instances**: termination
- **Availability Zones**: artificial failures
- **Latency**: artificial delays by ChAP
- **Conformity**: kills non-best-practices instances
- **Doctor**: health checks
- **Janitor**: kills unused instances
- **Security**: checks violations
- **10-18**: geographic issues
A Resilient Architecture

Load

ELB

ASG Cluster

SG

Application

Netflix

Devices

Zuul 2

Service

Tomcat

JVM

Instances (Linux)

AZ 1

AZ 2

AZ 3

A SG 1

A SG 2

Chaos Engineering

Some services vary:
- Apache Web Server
- Node.js & Prana
- golang

- Apache Web Server
- Node.js & Prana
- golang
2. Methodology
Why Do Root Cause Perf Analysis?

Often for:

- High latency
- Growth
- Upgrades
Cloud Methodologies

- Resource Analysis
- Metric and event correlations
- Latency Drilldowns
- RED Method

For each microservice, check:
- Rate
- Errors
- Duration
Instance Methodologies

- Log Analysis
- Micro-benchmarking
- Drill-down analysis
- USE Method

For each resource, check:
- Utilization
- Saturation
- Errors
Bad Instance Anti-Method

1. Plot request latency per-instance
2. Find the bad instance
3. Terminate it
4. Someone else’s problem now!

Bad instance latency
Terminate!

Could be an early warning of a bigger issue
3. Cloud Analysis

Atlas, Lumen, Chronos, ...
Netflix Cloud Analysis Process

Example path enumerated

1. Check Issue
   - Atlas Alerts
   - Atlas/Lumen Dashboards

2. Check Events
   - Atlas/Lumen Dashboards
   - Chronos
   - Atlas Metrics

Create New Alert

3. Drill Down
   - Atlas Metrics
   - Slalom
   - Zipkin

4. Check Dependencies
   - Redirected to a new Target

5. Root Cause
   - Atlas Alerts

Plus some other tools not pictured

Cost

Chat
Atlas: Alerts

Custom alerts on streams per second (SPS) changes, CPU usage, latency, ASG growth, client errors, …
Winston Diagnostics and Remediation

Did this correlate with a new or dying ASG?

Note: ASGs are listed in order of number of containers (decreasing with the highest) in the first ASG in the cluster.

Related to a Fast Property Change?

The highlighted portion of the graph shows the change in the Fast Property Change. The query performed filters things which we believe are NOT related to typical streaming issues. See the list of Fast Properties in future KB.
Winston: Automated Diagnostics & Remediation

Did this correlate with a new or dying ASG?

Related to a Fast Property Change?

Chronos: Possible Related Events

Links to Atlas Dashboards & Metrics
Atlas: Dashboards
Atlas: Dashboards

Netflix perf vitals dashboard

1. RPS, CPU
2. Volume
3. Instances
4. Scaling
5. CPU/RPS
6. Load avg
7. Java heap
8. ParNew
9. Latency
10. 99th tile
Atlas & Lumen: Custom Dashboards

- Dashboards are a checklist methodology: what to show first, second, third...

- Starting point for issues
  1. Confirm and quantify issue
  2. Check historic trend
  3. Atlas metrics to drill down

Lumen: more flexible dashboards eg, go/burger
Atlas: Metrics

The image shows a screenshot of the Atlas UI stable build with a focus on metrics. The dashboard is set to a specific Env: prod, test, Region: us-east-1, and the metric being displayed is CPU Utilization for node CPU. The chart illustrates the CPU usage over time, with data points showing a trend that reflects the CPU load on the system.
Atlas: Metrics

- **Region**: prod.us-east-1
- **Interactive graph**
- **Summary statistics**
- **Time range**: Prior day
- **Start**: Prior day
- **End**: 5 minutes ago
- **Application**
- **Metrics**
- **Presentation**
- **Env**: prod, test
- **Dashboard target**: prod.us-east-1

You can create or edit an Atlas query interactively here. Hover over any button or label for more information. Switch to Manual mode if you need additional features or to copy/paste a query.
Atlas: Metrics

- All metrics in one system
  - System metrics: CPU usage, disk I/O, memory, ...
  - Application metrics: latency percentiles, errors, ...
- Filters or breakdowns by region, application, ASG, metric, instance
- URL has session state: shareable
## Chronos: Change Tracking

### Predefined Queries
- All events except low critical

### Search
- Search chronos events

### Region
- Undefined
- us-event-1
- us-event-2

### Applied Filters
- Criticality
- exclude: low

### Add Filter
- Include
- Exclude

### Source App
- Include
- Exclude

### Select Value
- More

### Type to Search
- Undefined

### Event Log

<table>
<thead>
<tr>
<th>Start Time</th>
<th>Region</th>
<th>Application</th>
<th>Cluster</th>
<th>Stack</th>
<th>Source App</th>
<th>Action</th>
<th>Event Type</th>
<th>Name</th>
<th>Description</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-11-27 09:54</td>
<td>us-event-1</td>
<td>hadoop</td>
<td>map</td>
<td></td>
<td></td>
<td></td>
<td>delete</td>
<td>object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:54</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>create</td>
<td>scheduled Action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:54</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>create</td>
<td>scheduled Action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:53</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>publish</td>
<td>gutenberg</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:53</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>create</td>
<td>scheduled Action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:53</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>create</td>
<td>scheduled Action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:53</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>update</td>
<td>cluster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:53</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>create</td>
<td>scheduled Action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:53</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>create</td>
<td>scheduled Action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-11-27 09:53</td>
<td>us-event-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>end</td>
<td>workflow</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total Events Selected: 10 out of 42
Slalom: Dependency Graphing
Slalom: Dependency Graphing
Zipkin UI: Dependency Tracing
PICSOU: AWS Usage

- Breakdowns
- Cost per hour
- Details (redacted)
Latency is high in us-east-1

Sorry
We just did a bad push
Netflix Cloud Analysis Process

Example path enumerated

1. Check Issue
   - Atlas Alerts

2. Check Events
   - Atlas/Lumen Dashboards

3. Drill Down
   - Atlas Metrics

4. Check Dependencies
   - Chronos
   - Create New Alert
   - Redirected to a new Target

5. Root Cause
   - Slalom
   - Zipkin

Plus some other tools not pictured

PICSOU

Cost

Slack

Chat
Generic Cloud Analysis Process

1. Check Issue
2. Check Events
3. Drill Down
4. Check Dependencies
5. Root Cause

Example path enumerated

Create New Alert

Usage Reports

Cost

Messaging

Redirected to a new Target

Alerts

Custom Dashboards

Change Tracking

Metric Analysis

Dependency Analysis

Instance Analysis

Plus other tools as needed

Chat
4. Instance Analysis

1. Statistics
2. Profiling
3. Tracing
4. Processor Analysis
Linux Performance

Operating System

Applications

System Libraries

System Call Interface

VFS
File Systems
Volume Manager
Block Device Interface

Device Drivers

I/O Bus

Expander Interconnect

I/O Bridge

I/O Controller

Disk
Disk
Swap

Network Controller

Port
Port

Hardware

CPU Interconnect

CPU 1

Memory Bus

DRAM

http://www.brendangregg.com/linuxperf.html 2018
1. Statistics
## Linux Tools

- vmstat, pidstat, sar, etc, used mostly normally

```bash
$ sar -n TCP,ETCP,DEV 1
Linux 4.15.0-1027-aws (xxx) 12/03/2018 _x86_64_ (48 CPU)

<table>
<thead>
<tr>
<th>TIME</th>
<th>IFACE</th>
<th>rxpck/s</th>
<th>txpck/s</th>
<th>rxkB/s</th>
<th>txkB/s</th>
<th>rxcmp/s</th>
<th>txcmp/s</th>
<th>rxmcst/s</th>
<th>%ifutil</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:43:53 PM</td>
<td>lo</td>
<td>15.00</td>
<td>15.00</td>
<td>1.31</td>
<td>1.31</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>09:43:54 PM</td>
<td>eth0</td>
<td>26392.00</td>
<td>33744.00</td>
<td>19361.43</td>
<td>28065.36</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>09:43:53 PM</td>
<td>active/s</td>
<td>18.00</td>
<td>132.00</td>
<td>17512.00</td>
<td>33760.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>09:43:54 PM</td>
<td>atmptf/s</td>
<td>0.00</td>
<td>0.00</td>
<td>11.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
```

- Micro benchmarking can be used to investigate hypervisor behavior that can’t be observed directly
Exception: Containers

- Most Linux tools are still not container aware
  - From the container, will show the full host
- We expose cgroup metrics in our cloud GUIs: Vector
Vector: Instance/Container Analysis
2. Profiling
Experience:

“ZFS is eating my CPUs”
CPU Mixed-Mode Flame Graph

Application (truncated)

38% kernel time (why?)
2014: Java Profiling

Java Profilers

System Profilers
2018: Java Profiling
CPU Flame Chart (same data)
CPU Flame Graphs

e()

f()
g()
d()
c()
b()
a()

h()
i()
CPU Flame Graphs

- **Y-axis:** stack depth
  - 0 at bottom
  - 0 at top == icicle graph

- **X-axis:** alphabet
  - Time == flame chart

- **Color:** random
  - Hues often used for language types
  - Can be a dimension eg, CPI

Top edge:
Who is running on CPU
And how much (width)
Application Profiling

• Primary approach:
  - CPU mixed-mode flame graphs (eg, via Linux perf)
  - May need frame pointers (eg, Java -XX:+PreserveFramePointer)
  - May need a symbol file (eg, Java perf-map-agent, Node.js --perf-basic-prof)

• Secondary:
  - Application profiler (eg, via Lightweight Java Profiler)
  - Application logs
Vector: Push-button Flame Graphs

<table>
<thead>
<tr>
<th>Dashboard</th>
<th>Window</th>
<th>2 min</th>
<th>5 min</th>
<th>10 min</th>
<th>Interval</th>
<th>1</th>
</tr>
</thead>
</table>

Flamegraphs: CPU
100.65.20.44:7402

Flamegraph previous request status: IDLE
Profile duration:
20 sec

Start capture
Fetch url: cpufreqgraph/cpuflamegraph.1.png
View / download
Future: eBPF-based Profiling

Linux 2.6

```
perf record
```

```
perf.data
```

```
perf script
```

```
stackcollapse-perf.pl
```

```
flamegraph.pl
```

Linux 4.9

```
profile.py
```

```
flamegraph.pl
```
3. Tracing
Core Linux Tracers

- **Ftrace** 2.6.27+  Tracing views
- **perf** 2.6.31+  Official profiler & tracer
- **eBPF** 4.9+  Programmatic engine
  - **bcc**  -  Complex tools
  - **bpftrace**  -  Short scripts

Plus other kernel tech: kprobes, uprobes
Experience: Disk %Busy

Disk Percent Utilization

- Max: 0.000, Min: 0.000
- Avg: 0.000, Last: 0.000
- Tot: 0.000, Cnt: 240.000

Frame: 4h, End: 2016-08-18T13:01-07:00[US/Pacific], Step: 1m
Fetch: 144ms (L: 54.0, 12.0, 3.0; D: 3.2k, 2.9k, 720.0k)
```
# iostat -x 1

avg-cpu: %user %nice %system %iowait %steal %idle
        5.37  0.00  0.77  0.00  0.00  93.86

Device: rrqm/s  wrqm/s  r/s  w/s  rkB/s  wkB/s  avgrq-sz avgqu-sz  await  r_await w_await  svctm %util
  xvda      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
  xvdb      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
  xvdj      0.00    0.00  139.00 0.00  1056.00   0.00  15.19   0.88   6.19  6.19  0.00   6.30  87.60
```
# /apps/perf-tools/bin/iolatency 10
Tracing block I/O. Output every 10 seconds. Ctrl-C to end.

<table>
<thead>
<tr>
<th>&gt;=(ms) .. &lt;(ms)</th>
<th>I/O</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 -&gt; 1</td>
<td>421</td>
<td>###################################</td>
</tr>
<tr>
<td>1 -&gt; 2</td>
<td>95</td>
<td>#######</td>
</tr>
<tr>
<td>2 -&gt; 4</td>
<td>48</td>
<td>######</td>
</tr>
<tr>
<td>4 -&gt; 8</td>
<td>108</td>
<td>#######</td>
</tr>
<tr>
<td>8 -&gt; 16</td>
<td>363</td>
<td>###################################</td>
</tr>
<tr>
<td>16 -&gt; 32</td>
<td>66</td>
<td>######</td>
</tr>
<tr>
<td>32 -&gt; 64</td>
<td>3</td>
<td>#</td>
</tr>
<tr>
<td>64 -&gt; 128</td>
<td>7</td>
<td>#</td>
</tr>
</tbody>
</table>

^C
```plaintext
# /apps/perf-tools/bin/iosnoop
Tracing block I/O. Ctrl-C to end.

<table>
<thead>
<tr>
<th>COMM</th>
<th>PID</th>
<th>TYPE</th>
<th>DEV</th>
<th>BLOCK</th>
<th>BYTES</th>
<th>LATms</th>
</tr>
</thead>
<tbody>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>1670768496</td>
<td>8192</td>
<td>0.28</td>
</tr>
<tr>
<td>cat</td>
<td>6587</td>
<td>R</td>
<td>202,0</td>
<td>1727096</td>
<td>4096</td>
<td>10.07</td>
</tr>
<tr>
<td>cat</td>
<td>6587</td>
<td>R</td>
<td>202,0</td>
<td>1727120</td>
<td>8192</td>
<td>10.21</td>
</tr>
<tr>
<td>cat</td>
<td>6587</td>
<td>R</td>
<td>202,0</td>
<td>1727152</td>
<td>8192</td>
<td>10.43</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>620864512</td>
<td>4096</td>
<td>7.69</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>584767616</td>
<td>8192</td>
<td>16.12</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>601721568</td>
<td>8192</td>
<td>9.06</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>61067936</td>
<td>8192</td>
<td>0.97</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>1678557024</td>
<td>8192</td>
<td>0.34</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>55299456</td>
<td>8192</td>
<td>22.17</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>1625084928</td>
<td>4096</td>
<td>12.00</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>618895408</td>
<td>8192</td>
<td>16.99</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>581318480</td>
<td>8192</td>
<td>13.39</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>1167348016</td>
<td>8192</td>
<td>9.92</td>
</tr>
<tr>
<td>java</td>
<td>30603</td>
<td>RM</td>
<td>202,144</td>
<td>51561280</td>
<td>8192</td>
<td>22.17</td>
</tr>
</tbody>
</table>
[...]```
70.70%  251  java  [kernel.kallsyms]  [k]  blk_peek_request

---  blk_peek_request
     do_blkif_request
     _Blk_run_queue
     queue_unplugged
     blk_flush_plug_list
     blk_finish_plug
     _xfs_buf_ioapply
     xfs_buf_irequest

--88.84%--  _xfs_buf_read
     xfs_buf_read_map

--87.89%--  xfs_trans_read_buf_map

--97.96%--  xfs_imap_to_bp
     xfs_lread
     xfs_lget
     xfs_lookup
     xfs_vn_lookup
     lookup_real
     lookup_hash
     lookup_slow
     path_lookupat
     filename_lookup
     user_path_at_empty
     user_path_at
     vfs_fstat

--99.48%--  SYSC_newlstat
     sys_newlstat
     system_call_fastpath
     __lxstat64
     Lsun/nio/fs/UnixNativeDispatcher::lstat0
     0x7f8f963c847c
Disk Percent Utilization

10:45 11:00 11:15 11:30 11:45 12:00 12:15 12:30 12:45 13:00 13:15 13:30 13:45 14:00 14:15 14:30 14:45

- ((name=diskio.5min.avgload.xvda) / (name=diskio.5min.avgload.xvda))
  Max : 15.000 Min : 0.000
  Avg : 529.167m Last : 0.000
  Tot : 127.000 Cnt : 240.000

- ((name=diskio.5min.avgload.xvdb) / (name=diskio.5min.avgload.xvdb))
  Max : 0.000 Min : 0.000
  Avg : 0.000 Last : 0.000
  Tot : 0.000 Cnt : 240.000

- ((name=diskio.5min.avgload.xvdj) / (name=diskio.5min.avgload.xvdj))
  Max : 96.000 Min : 0.000
  Avg : 52.638 Last : 6.000
  Tot : 12.633k Cnt : 240.000

Frame: 4h, End: 2016-08-18T14:46-07:00[US/Pacific], Step: 1m
Fetch: 96ms (L: 30.0, 6.0, 3.0; D: 1.8k, 1.4k, 720.0k)
<table>
<thead>
<tr>
<th>TIME(s)</th>
<th>COMM</th>
<th>PID</th>
<th>DISK</th>
<th>T</th>
<th>SECTOR</th>
<th>BYTES</th>
<th>LAT(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000000000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>110824</td>
<td>4096</td>
<td>6.50</td>
</tr>
<tr>
<td>0.004183000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>111672</td>
<td>4096</td>
<td>4.08</td>
</tr>
<tr>
<td>0.016195000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4198424</td>
<td>4096</td>
<td>11.88</td>
</tr>
<tr>
<td>0.018716000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4201152</td>
<td>4096</td>
<td>2.43</td>
</tr>
<tr>
<td>0.019416000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4201160</td>
<td>4096</td>
<td>0.61</td>
</tr>
<tr>
<td>0.032645000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4207968</td>
<td>4096</td>
<td>13.16</td>
</tr>
<tr>
<td>0.033181000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4207976</td>
<td>4096</td>
<td>0.47</td>
</tr>
<tr>
<td>0.033524000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4208000</td>
<td>4096</td>
<td>0.27</td>
</tr>
<tr>
<td>0.033876000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4207992</td>
<td>4096</td>
<td>0.28</td>
</tr>
<tr>
<td>0.034840000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4208008</td>
<td>4096</td>
<td>0.89</td>
</tr>
<tr>
<td>0.035713000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4207984</td>
<td>4096</td>
<td>0.81</td>
</tr>
<tr>
<td>0.036165000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>111720</td>
<td>4096</td>
<td>0.37</td>
</tr>
<tr>
<td>0.039969000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>8427264</td>
<td>4096</td>
<td>3.69</td>
</tr>
<tr>
<td>0.051614000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>8405640</td>
<td>4096</td>
<td>11.44</td>
</tr>
<tr>
<td>0.052310000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>111696</td>
<td>4096</td>
<td>0.55</td>
</tr>
<tr>
<td>0.053044000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>111712</td>
<td>4096</td>
<td>0.56</td>
</tr>
<tr>
<td>0.059583000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>8411032</td>
<td>4096</td>
<td>6.40</td>
</tr>
<tr>
<td>0.068278000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4218672</td>
<td>4096</td>
<td>8.57</td>
</tr>
<tr>
<td>0.076717000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4218968</td>
<td>4096</td>
<td>8.33</td>
</tr>
<tr>
<td>0.077183000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>4218984</td>
<td>4096</td>
<td>0.40</td>
</tr>
<tr>
<td>0.082188000</td>
<td>tar</td>
<td>8519</td>
<td>xvda</td>
<td>R</td>
<td>8393552</td>
<td>4096</td>
<td>4.94</td>
</tr>
</tbody>
</table>

[...]
eBPF: extended Berkeley Packet Filter

User-Defined BPF Programs
- SDN Configuration
- DDoS Mitigation
- Intrusion Detection
- Container Security
- Observability
- Firewalls (bpfilter)
- Device Drivers

Kernel
- Runtime
  - verifier
  - BPF
    - BPF actions
- Event Targets
  - sockets
  - kprobes
  - uprobes
  - tracepoints
  - perf_events
Linux bcc/BPF Tracing Tools

- filetop
- filelifе
- fileslower
- vfscount
- vfsstat
- cachestat
- cachetop
- dcsstat
- dcsnoop
- mountsnoop
- trace
- argc
- dist
- funccount
- funcslower
- funclatency
- stackcount
- profile
- mdflush
- btrfsdist
- btrfslower
- ext4dist
- ext4slower
- xfsopt
- xfsllower
- zfsdist
- zfslower
- hardirqs
- ttysnoop
- biotop
- biosnoop
- biolatency
- bitsize
- c* java* node*
- php* python*
- ruby*
- ucalls
- uflow
- ugc
- uobj
- new
- ustat
- uthreads
- mysql
- qslower
- bashreadline
- gethostlatency
- memleak
- sslniff
- Other:
- capable
- syscount
- killsnoop
- execsnoop
- pidpersec
- cpu
-(pid)
- runqlat
- runglen
- deadlock_detector
- cpu
- unclaimed
- offcpu
- uptime
- wakeuptime
- offwakeuptime
- softirqs
- oomkill
- memleak
- slabratetop
- DRAM
- LLC
- CPU
- Profile

https://github.com/iovisor/bcc#tools 2018
```
<table>
<thead>
<tr>
<th>PID</th>
<th>COMM</th>
<th>LADDR</th>
<th>LPORT</th>
<th>RADDR</th>
<th>RPORT</th>
<th>TX_KB</th>
<th>RX_KB</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2509</td>
<td>java</td>
<td>100.82.34.63</td>
<td>8078</td>
<td>100.82.130.159</td>
<td>12410</td>
<td>0</td>
<td>0</td>
<td>5.44</td>
</tr>
<tr>
<td>2509</td>
<td>java</td>
<td>100.82.34.63</td>
<td>8078</td>
<td>100.82.78.215</td>
<td>55564</td>
<td>0</td>
<td>0</td>
<td>135.32</td>
</tr>
<tr>
<td>2509</td>
<td>java</td>
<td>100.82.34.63</td>
<td>60778</td>
<td>100.82.207.252</td>
<td>7001</td>
<td>0</td>
<td>0</td>
<td>15126.87</td>
</tr>
<tr>
<td>2509</td>
<td>java</td>
<td>100.82.34.63</td>
<td>38884</td>
<td>100.82.208.178</td>
<td>7001</td>
<td>0</td>
<td>0</td>
<td>15568.25</td>
</tr>
<tr>
<td>2509</td>
<td>java</td>
<td>127.0.0.1</td>
<td>4243</td>
<td>127.0.0.1</td>
<td>42166</td>
<td>0</td>
<td>0</td>
<td>0.61</td>
</tr>
<tr>
<td>12030</td>
<td>upload-mes</td>
<td>127.0.0.1</td>
<td>34020</td>
<td>127.0.0.1</td>
<td>8078</td>
<td>11</td>
<td>0</td>
<td>3.38</td>
</tr>
<tr>
<td>12030</td>
<td>upload-mes</td>
<td>127.0.0.1</td>
<td>21196</td>
<td>127.0.0.1</td>
<td>7101</td>
<td>0</td>
<td>0</td>
<td>12.61</td>
</tr>
<tr>
<td>3964</td>
<td>mesos-slav</td>
<td>127.0.0.1</td>
<td>7101</td>
<td>127.0.0.1</td>
<td>21196</td>
<td>0</td>
<td>0</td>
<td>12.64</td>
</tr>
<tr>
<td>12021</td>
<td>upload-sys</td>
<td>127.0.0.1</td>
<td>34022</td>
<td>127.0.0.1</td>
<td>8078</td>
<td>372</td>
<td>0</td>
<td>15.28</td>
</tr>
<tr>
<td>2509</td>
<td>java</td>
<td>127.0.0.1</td>
<td>8078</td>
<td>127.0.0.1</td>
<td>34022</td>
<td>0</td>
<td>372</td>
<td>15.31</td>
</tr>
<tr>
<td>2235</td>
<td>dockerd</td>
<td>100.82.34.63</td>
<td>13730</td>
<td>100.82.136.233</td>
<td>7002</td>
<td>0</td>
<td>0</td>
<td>4.18.50</td>
</tr>
<tr>
<td>2235</td>
<td>dockerd</td>
<td>100.82.34.63</td>
<td>34314</td>
<td>100.82.64.53</td>
<td>7002</td>
<td>0</td>
<td>0</td>
<td>8.56.73</td>
</tr>
</tbody>
</table>

[...]
# biolatency.bt
Attaching 3 probes...
Tracing block device I/O... Hit Ctrl-C to end.
^C

@usecs:
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[256, 512)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>[512, 1K)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>[1K, 2K)</td>
<td>426</td>
<td></td>
</tr>
<tr>
<td>[2K, 4K)</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>[4K, 8K)</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>[8K, 16K)</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>[16K, 32K)</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>[32K, 64K)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>[64K, 128K)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>[128K, 256K)</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
bpftrace: biolatency.bt

#!/usr/local/bin/bpftrace

BEGIN
{
    printf("Tracing block device I/O... Hit Ctrl-C to end.\n");
}

kprobe:blk_account_io_start
{
    @start[arg0] = nsecs;
}

kprobe:blk_account_io_completion
/@start[arg0]/
{
    @usecs = hist((nsecs - @start[arg0]) / 1000);
    delete(@start[arg0]);
}
Future: eBPF GUIs
4. Processor Analysis
What “90% CPU Utilization” might suggest:

What it typically means on the Netflix cloud:

Busy | Waiting ("idle")

Busy | Waiting ("stalled") | Waiting ("idle")
PMCs

- Performance Monitoring Counters help you analyze stalls

Some instances (e.g., Xen-based m4.16xl) have the architectural set:

<table>
<thead>
<tr>
<th>Event Name</th>
<th>UMask</th>
<th>Event Select</th>
<th>Example Event Mask Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnHalted Core Cycles</td>
<td>00H</td>
<td>3CH</td>
<td>CPU_CLK_UNHALTED.THREAD_P</td>
</tr>
<tr>
<td>Instruction Retired</td>
<td>00H</td>
<td>C0H</td>
<td>INST RETIRED.ANY_P</td>
</tr>
<tr>
<td>UnHalted Reference Cycles</td>
<td>01H</td>
<td>3CH</td>
<td>CPU_CLK_THREAD_UNHALTED.REF_XCLK</td>
</tr>
<tr>
<td>LLC Reference</td>
<td>4FH</td>
<td>2EH</td>
<td>LONGEST_LAT_CACHE.REFERENCE</td>
</tr>
<tr>
<td>LLC Misses</td>
<td>41H</td>
<td>2EH</td>
<td>LONGEST_LAT_CACHE.MISS</td>
</tr>
<tr>
<td>Branch Instruction Retired</td>
<td>00H</td>
<td>C4H</td>
<td>BR_INST RETIRED.ALL_BRANCHES</td>
</tr>
<tr>
<td>Branch Misses Retired</td>
<td>00H</td>
<td>C5H</td>
<td>BR_MISP RETIRED.ALL_BRANCHES</td>
</tr>
</tbody>
</table>
Instructions Per Cycle (IPC)

“good*”

>2.0

Instruction bound

IPC

<0.2

“bad”

Stall-cycle bound

* probably; exception: spin locks
PMCs: EC2 Xen Hypervisor

# perf stat -a -- sleep 30

Performance counter stats for 'system wide':

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>#</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>task-clock (msec)</td>
<td>1921101.773240</td>
<td></td>
<td>64.034 CPUs utilized</td>
</tr>
<tr>
<td>context-switches</td>
<td>1,103,112</td>
<td></td>
<td>0.574 K/sec</td>
</tr>
<tr>
<td>cpu-migrations</td>
<td>189,173</td>
<td></td>
<td>0.098 K/sec</td>
</tr>
<tr>
<td>page-faults</td>
<td>4,044</td>
<td></td>
<td>0.002 K/sec</td>
</tr>
<tr>
<td>cycles</td>
<td>2,057,164,531,949</td>
<td></td>
<td>1.071 GHz</td>
</tr>
<tr>
<td>instructions</td>
<td>1,357,979,592,699</td>
<td></td>
<td>0.66 insns per cycle</td>
</tr>
<tr>
<td>branches</td>
<td>243,244,156,173</td>
<td></td>
<td>126.617 M/sec</td>
</tr>
<tr>
<td>branch-misses</td>
<td>4,391,259,112</td>
<td></td>
<td>1.81% of all branches</td>
</tr>
</tbody>
</table>

30.001112466 seconds time elapsed

# ./pmcarch 1

<table>
<thead>
<tr>
<th>CYCLES</th>
<th>INSTRUCTIONS</th>
<th>IPC</th>
<th>BR_RETIRED</th>
<th>BR_MISPRED</th>
<th>BMR%</th>
<th>LLCREF</th>
<th>LLCMISS</th>
<th>LLC%</th>
</tr>
</thead>
<tbody>
<tr>
<td>38222881237</td>
<td>25412094046</td>
<td>0.66</td>
<td>4692322525</td>
<td>91505748</td>
<td>1.95</td>
<td>780435112</td>
<td>117058225</td>
<td>85.00</td>
</tr>
<tr>
<td>40754208291</td>
<td>26308406390</td>
<td>0.65</td>
<td>5286747667</td>
<td>95879771</td>
<td>1.81</td>
<td>751335355</td>
<td>123725560</td>
<td>83.53</td>
</tr>
<tr>
<td>35222264860</td>
<td>24681830086</td>
<td>0.70</td>
<td>4616980753</td>
<td>86190754</td>
<td>1.87</td>
<td>709841242</td>
<td>113254573</td>
<td>84.05</td>
</tr>
<tr>
<td>38176994942</td>
<td>26317856262</td>
<td>0.69</td>
<td>5055959631</td>
<td>92760370</td>
<td>1.83</td>
<td>787333902</td>
<td>119976728</td>
<td>84.76</td>
</tr>
</tbody>
</table>

[...]
PMCs: EC2 Nitro Hypervisor

- Some instance types (large, Nitro-based) support most PMCs!
- Meltdown KPTI patch TLB miss analysis on a c5.9xl:

```
nopti:
# tlbstat -C0 1
K_CYCLES   K_INSTR   IPC  DTLB_WALKS  ITLB_WALKS  K_DTLBCYC  K_ITLBCYC  DTLB%  ITLB%
2854768    2455917   0.86  565       2777        50         40          0.00  0.00
2884618    2478929   0.86  950       2756        6          38          0.00  0.00
2847354    2455187   0.86  396       297403      46         40          0.00  0.00
[...]

pti, nopcid:
# tlbstat -C0 1
K_CYCLES   K_INSTR   IPC  DTLB_WALKS  ITLB_WALKS  K_DTLBCYC  K_ITLBCYC  DTLB%  ITLB%
2875793    276051    0.10  89709496  65862302    787913      650834     27.40 22.63
2860557    273767    0.10  88829158  65213248    780301      644292     27.28 22.52
2885138    276533    0.10  89683045  65813992    787391      650494     27.29 22.55
2532843    243104    0.10  79055465  58023221    693910      573168     27.40 22.63  worst case
[...]
```
MSRs

• Model Specific Registers

• System config info, including current clock rate:

```plaintext
# showboost
Base CPU MHz : 2500
Set CPU MHz : 2500
Turbo MHz(s) : 3100 3200 3300 3500
Turbo Ratios : 124% 128% 132% 140%
CPU 0 summary every 1 seconds...

<table>
<thead>
<tr>
<th>TIME</th>
<th>C0_MCYC</th>
<th>C0_ACYC</th>
<th>UTIL</th>
<th>RATIO</th>
<th>MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>23:39:07</td>
<td>1618910294</td>
<td>89419923</td>
<td>64%</td>
<td>5%</td>
<td>138</td>
</tr>
<tr>
<td>23:39:08</td>
<td>1774059258</td>
<td>97132588</td>
<td>70%</td>
<td>5%</td>
<td>136</td>
</tr>
<tr>
<td>23:39:09</td>
<td>2476365498</td>
<td>130869241</td>
<td>99%</td>
<td>5%</td>
<td>132</td>
</tr>
</tbody>
</table>
^C
```
Summary
Take Aways

1. Get push-button **CPU flame graphs**: kernel & user

2. Check out **eBPF** perf tools: bcc, bpftrace

3. Measure **IPC as well as CPU utilization using PMCs**
Observability
Statistics, Flame Graphs, eBPF Tracing, Cloud PMCs

Methodology
USE method, RED method, Drill-down Analysis, ...

Velocity
Self-service GUIs: Vector, FlameScope, ...
Resources

- **2014 talk From Clouds to Roots**: http://www.slideshare.net/brendangregg/netflix-from-clouds-to-roots
  http://www.youtube.com/watch?v=H-E0MQTID0g
- **Atlas**: https://github.com/Netflix/Atlas
- **RED method**: https://thenewstack.io/monitoring-microservices-red-method/
- **USE method**: https://queue.acm.org/detail.cfm?id=2413037
- **Lumen**: https://medium.com/netflix-techblog/lumen-custom-self-service-dashboarding-for-netflix-8c56b541548c
- **Flame graphs**: http://www.brendangregg.com/flamegraphs.html
- **Vector**: http://vectoross.io https://github.com/Netflix/Vector
- **FlameScope**: https://github.com/Netflix/FlameScope
- **Tracing ponies**: thanks Deirdré Straughan & General Zoi's Pony Creator
- **ftrace**: http://lwn.net/Articles/608497/ - usually already in your kernel
- **perf**: http://www.brendangregg.com/perf.html - perf is usually packaged in linux-tools-common
- **tcplife**: https://github.com/iovisor/bcc - often available as a bcc or bcc-tools package
- **bpftrace**: https://github.com/iovisor/bpftrace
- **pmcarch**: https://github.com/brendangregg/pmc-cloud-tools
- **showboost**: https://github.com/brendangregg/msr-cloud-tools - also try turbostat
Lumen: Custom, Self-Service Dashboarding For Netflix

By Trent Willis

The Netflix Media Database (NMDB)

This blog post describes the Netflix Media DataBase (NMDB)—a highly queryable data system built on the Netflix micro-services platform...

Rethinking Netflix’s Edge Load Balancing

The why’s, how’s and results from rethinking Netflix’s edge load balancing
Thank you.