

Cloud Performance Root Cause Analysis at Netflix

Brendan Gregg

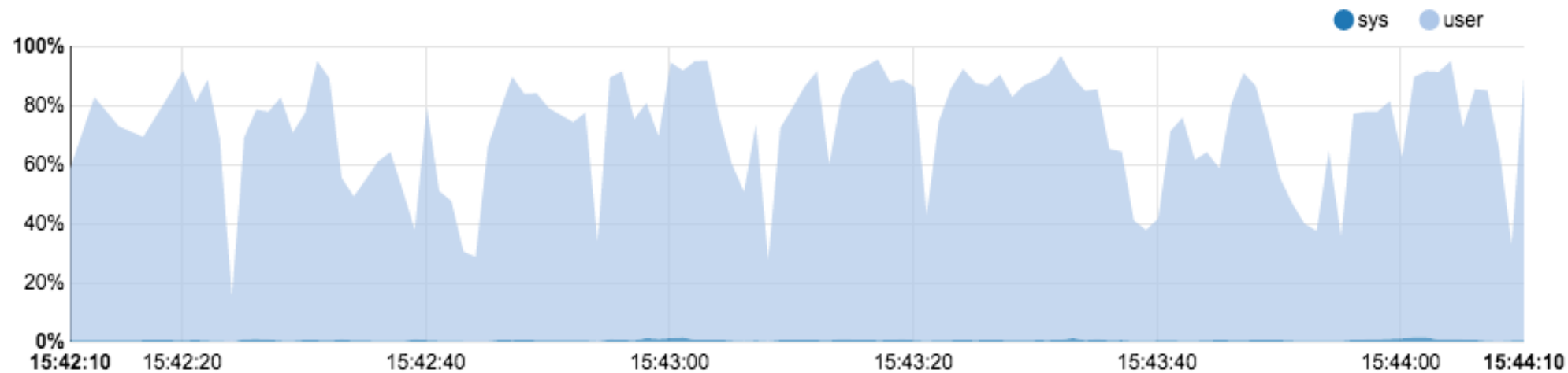
Senior Performance Architect
Cloud and Platform Engineering

YOW! Conference Australia
Nov-Dec 2018

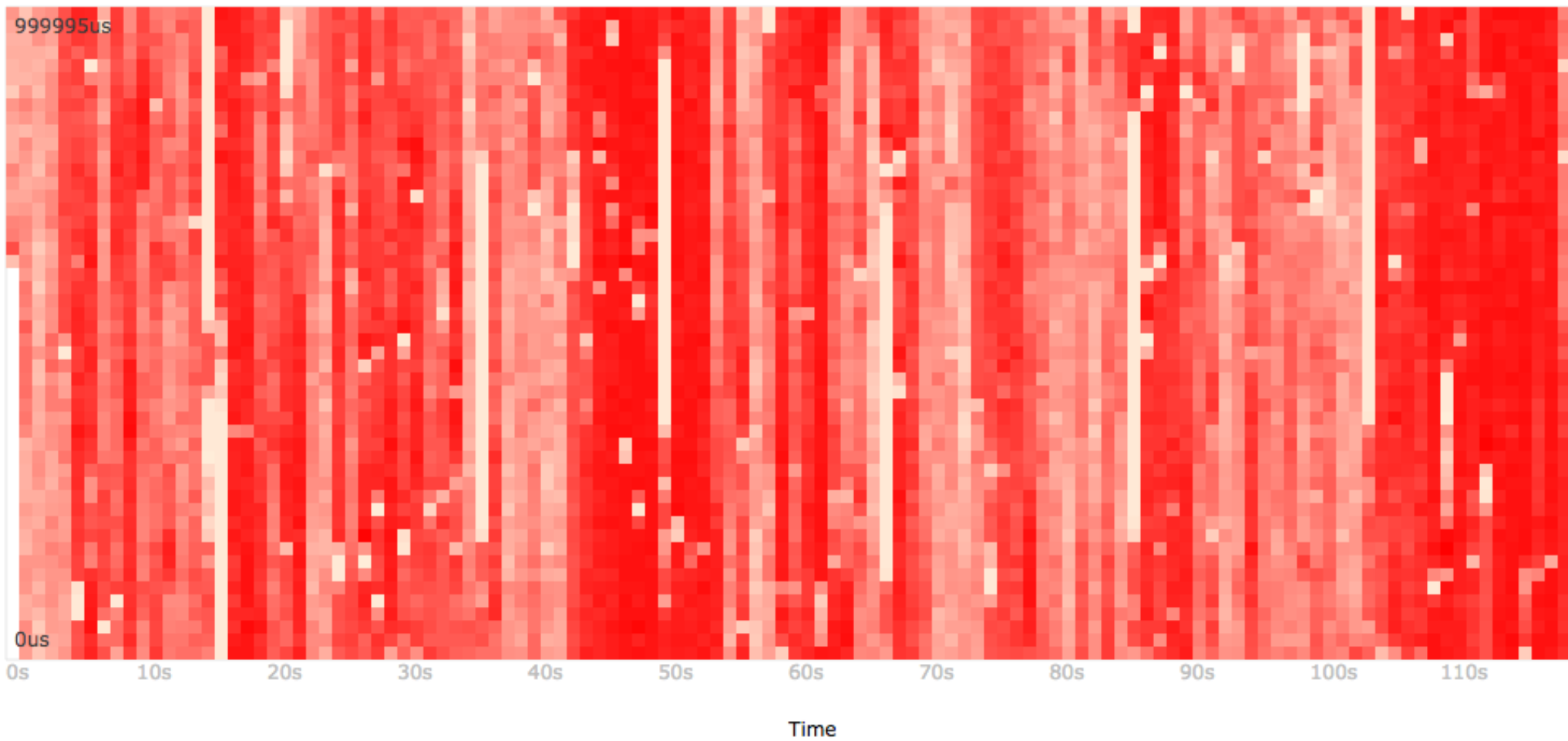
NETFLIX

Experience: CPU Dips

CPU Utilization



CPU Subsecond-Offset Heat Map



```
# perf record -F99 -a
```

```
# perf script
```

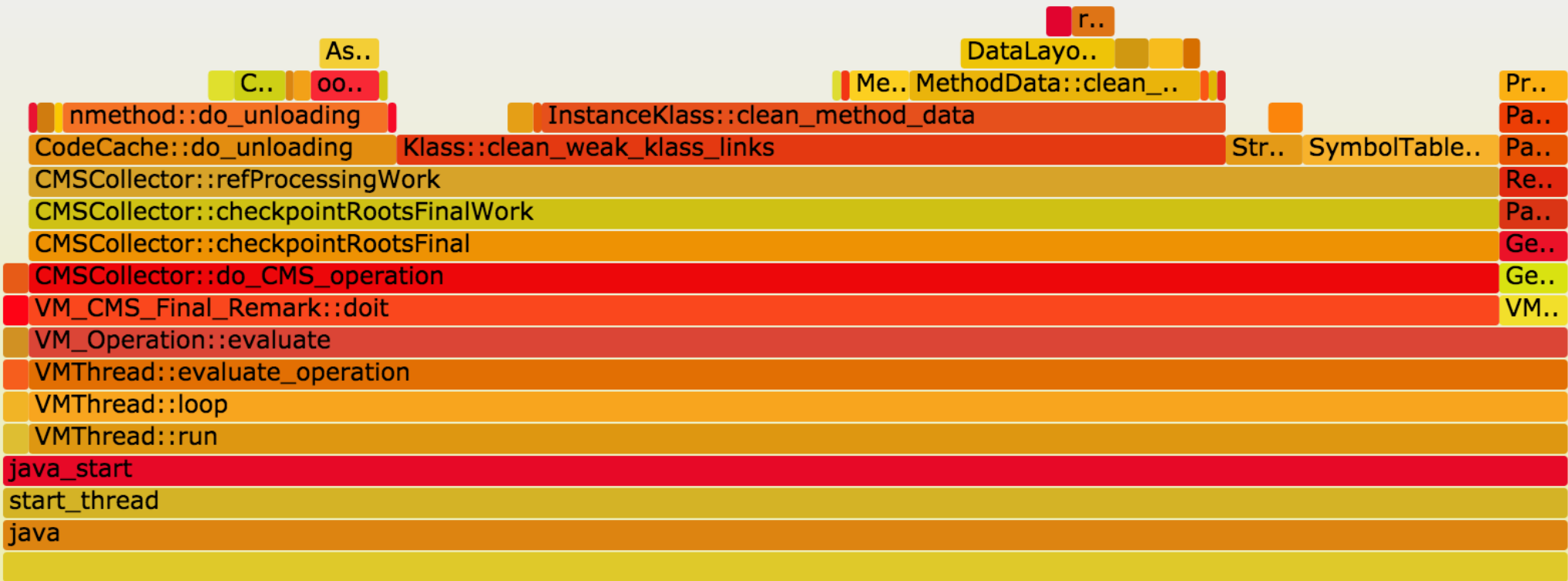
```
[...]
```

```
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(unsig
java 14341 [019] 252764.225141: cycles: 7f36571922e8 nmethod::do_unloading(BoolObje
java 14341 [019] 252764.234578: cycles: 7f3656ec4960 CodeHeap::block_start(void*) c
```

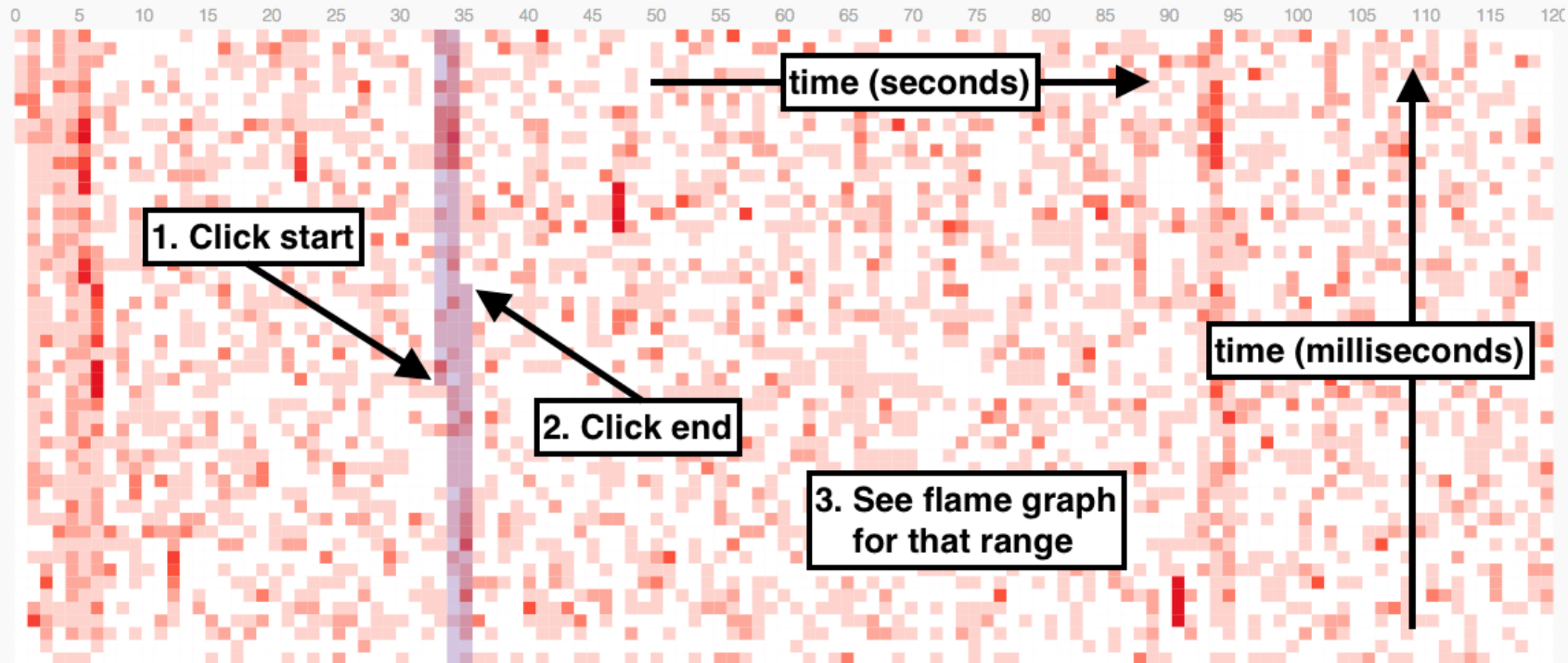
```
[...]
```

Single-CPU runs Flame Graph

Search



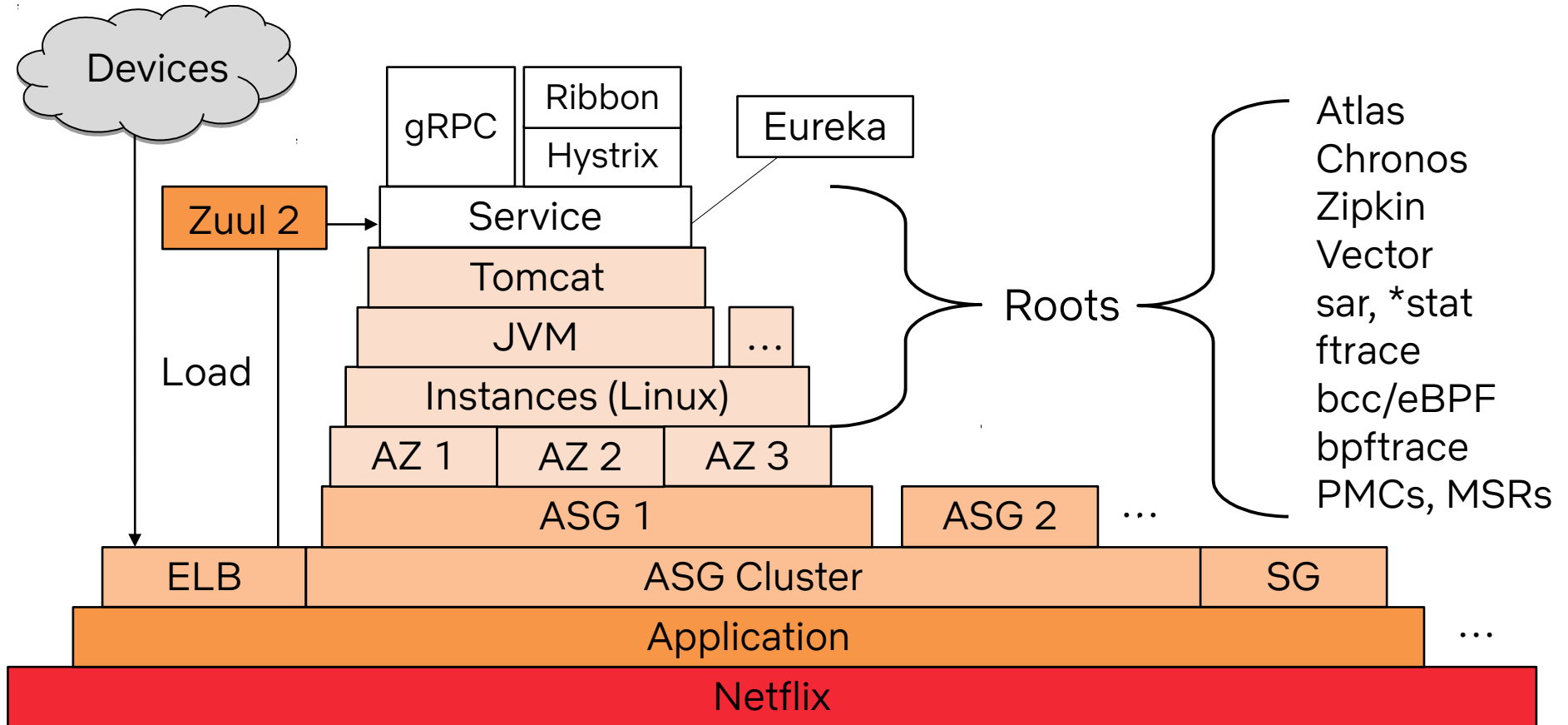
Rows 50



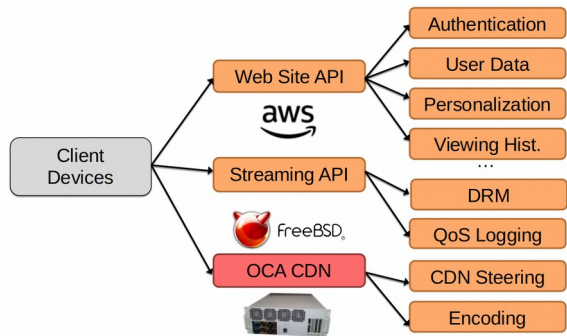
second: 35, millisecond: 580, count: 1

Observability
Methodology
Velocity

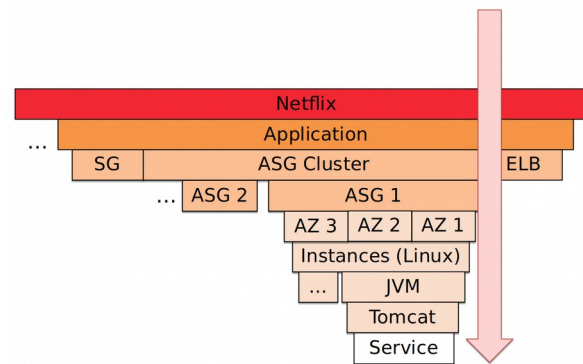
Root Cause Analysis at Netflix



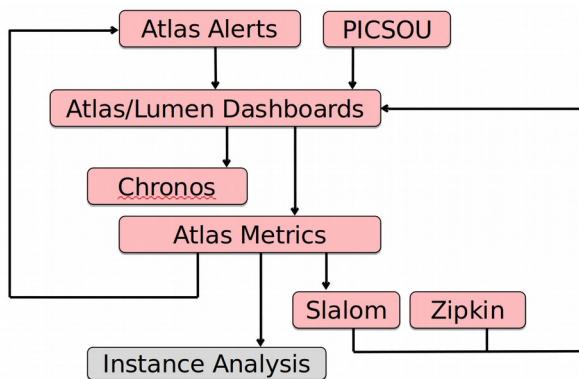
Agenda



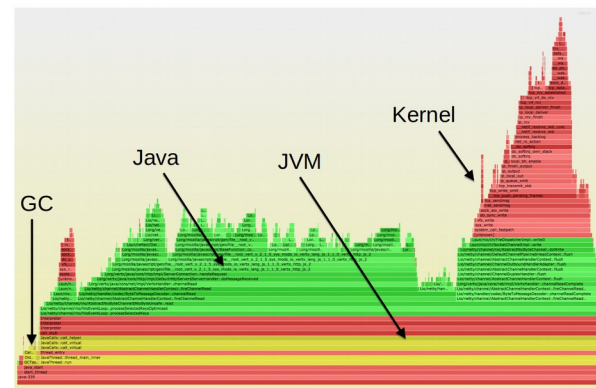
1. The Netflix Cloud



2. Methodology



3. Cloud Analysis



4. Instance Analysis

Since 2014

Asgard → Spinnaker

Salp → Zipkin

gRPC adoption

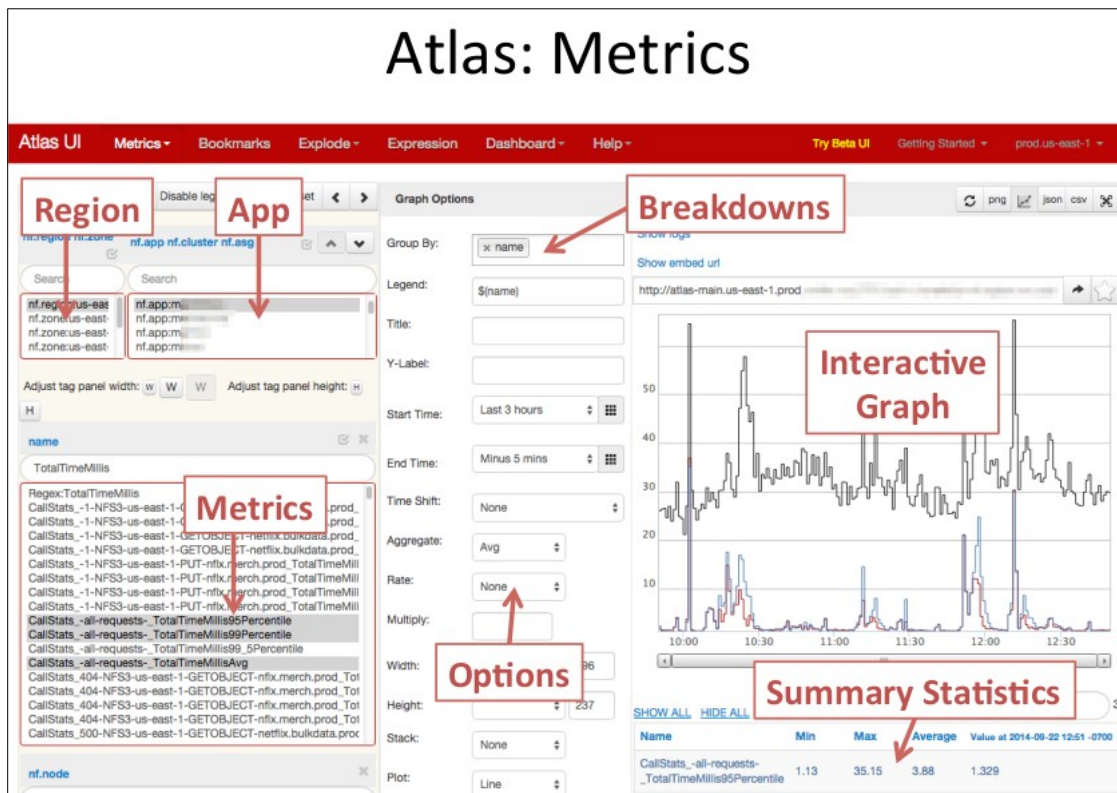
New Atlas UI & Lumen

Java frame pointer

eBPF bcc & bpftrace

PMCs in EC2

...



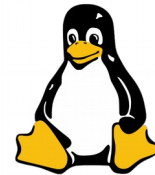
From Clouds to Roots (2014 presentation): Old Atlas UI

NETFLIX

>150k AWS EC2 server instances



~34% US Internet traffic at night



ubuntu

>130M members



FreeBSD®

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)

PMC: Performance Monitoring Counter (CPU perf counter)

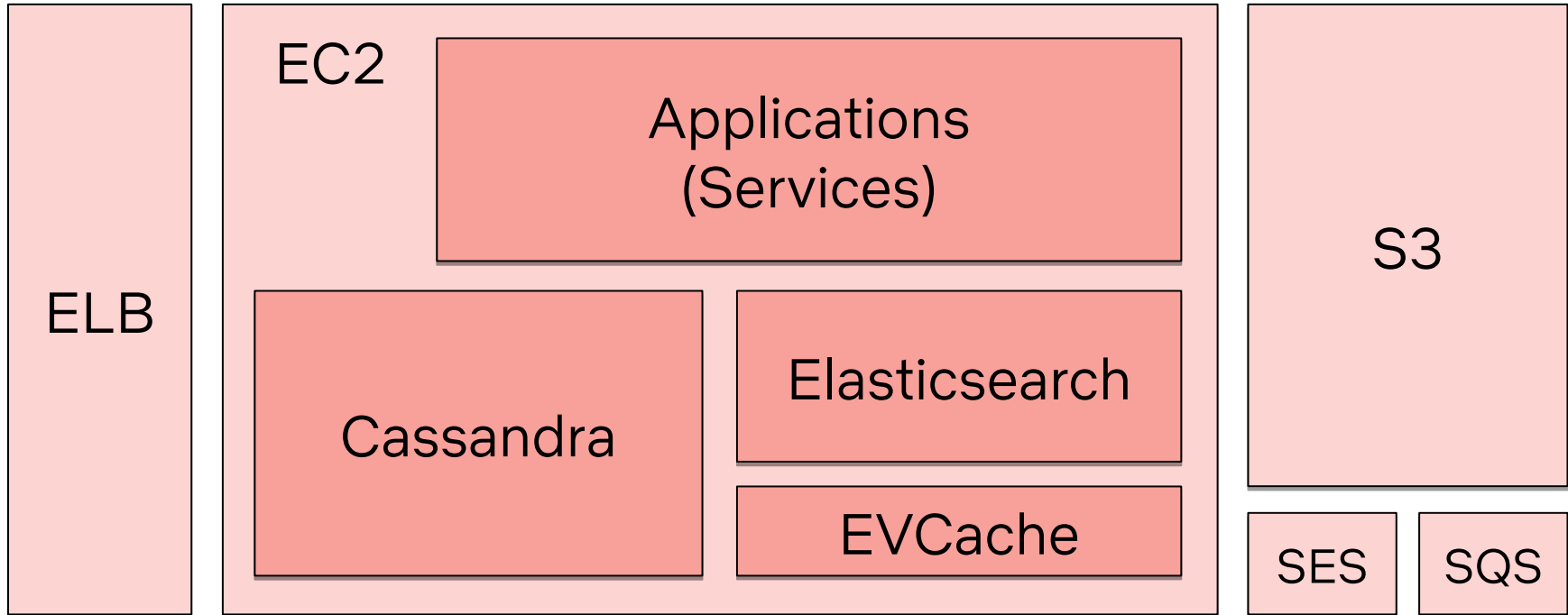
eBPF: extended Berkeley Packet Filter (kernel VM)

1. The Netflix Cloud

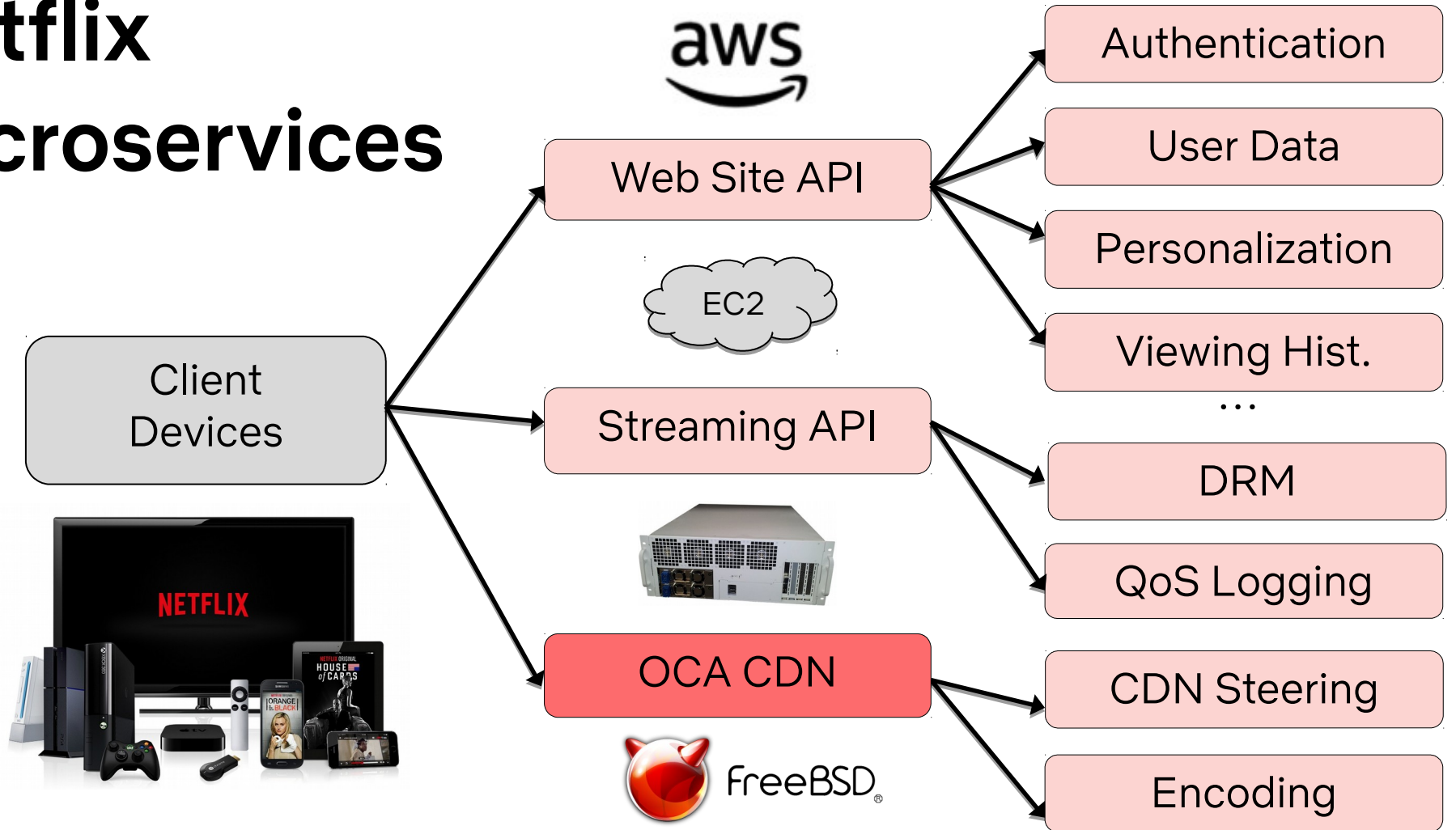
Overview

NETFLIX

The Netflix Cloud



Netflix Microservices



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/>

NETFLIX
OSS

Netflix Open Source Software Center

Netflix is committed to open source. Netflix both leverages and provides open source technology focused on providing the leading Internet television network. Our technology focuses on providing immersive experiences across all internet-connected screens. Netflix's deployment technology allows for continuous build and integration into our worldwide deployments serving members in over 50 countries. Our focus on reliability defined the bar for cloud based elastic deployments with several layers of failover. Netflix also provides the technology to operate services responsibly with operational insight, peak performance, and security. We provide technologies for data (persistent & semi-persistent) that serve the real-time load to our 62 million members, as well as power the big data analytics that allow us to make informed decisions on how to improve our service. If you want to learn more, jump into any of the functional areas below to learn more.

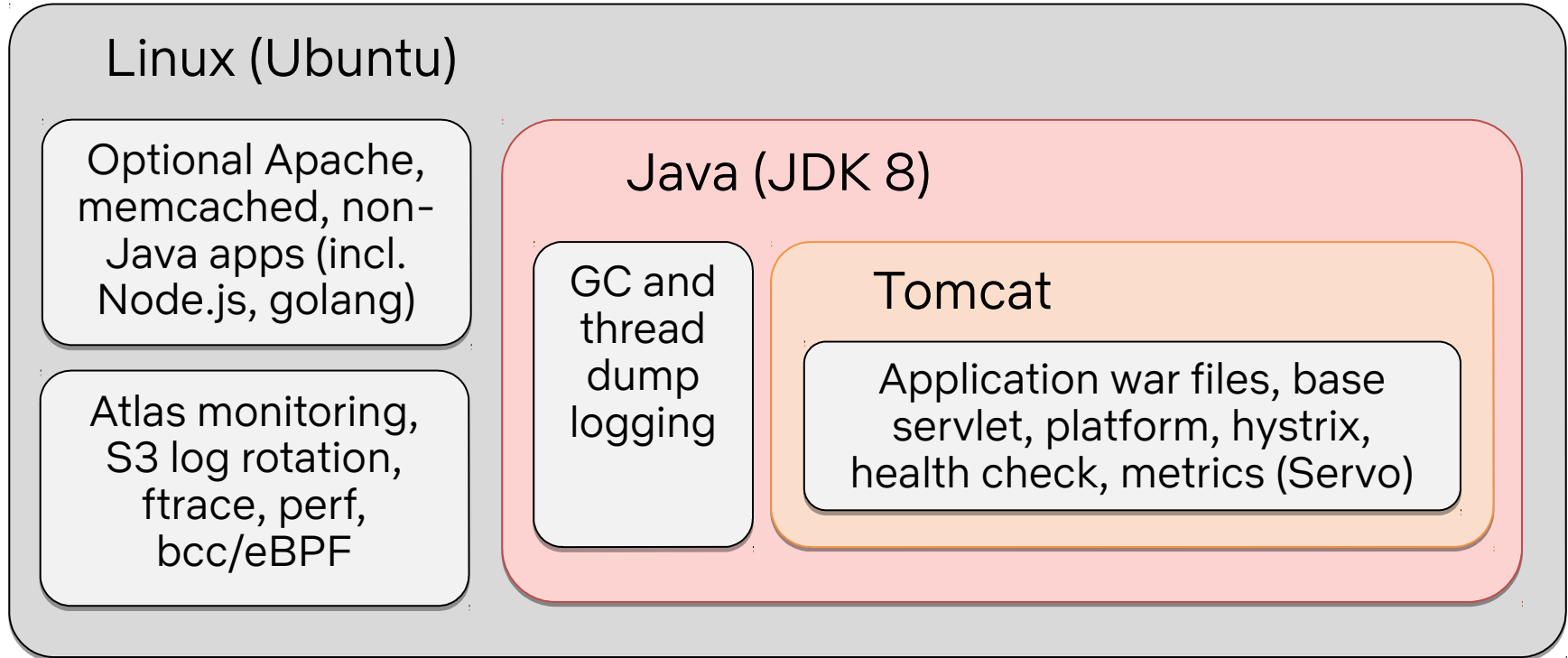


Big Data

Tools and services to get the most out of your (big) data

Data is invaluable in making Netflix such an exceptional service for our customers. Behind the scenes, we have a rich ecosystem of (big) data technologies facilitating our algorithms and analytics. We use and contribute to broadly-adopted open source technologies including Hadoop, Hive, Pig, Parquet, Presto, and Spark. In addition, we've developed and contributed some additional tools and services, which have further elevated our data platform. [Genie](#) is a powerful, REST-based abstraction to our various data processing frameworks, notably Hadoop. [Inviso](#) provides detailed insights into the performance of our Hadoop jobs and clusters. [Lipstick](#) shows the workflow of Pig jobs in a clear, visual fashion. And [Aegisthus](#) enables the bulk abstraction of data out of Cassandra for downstream analytic processing.

Cloud Instances



Typical BaseAMI

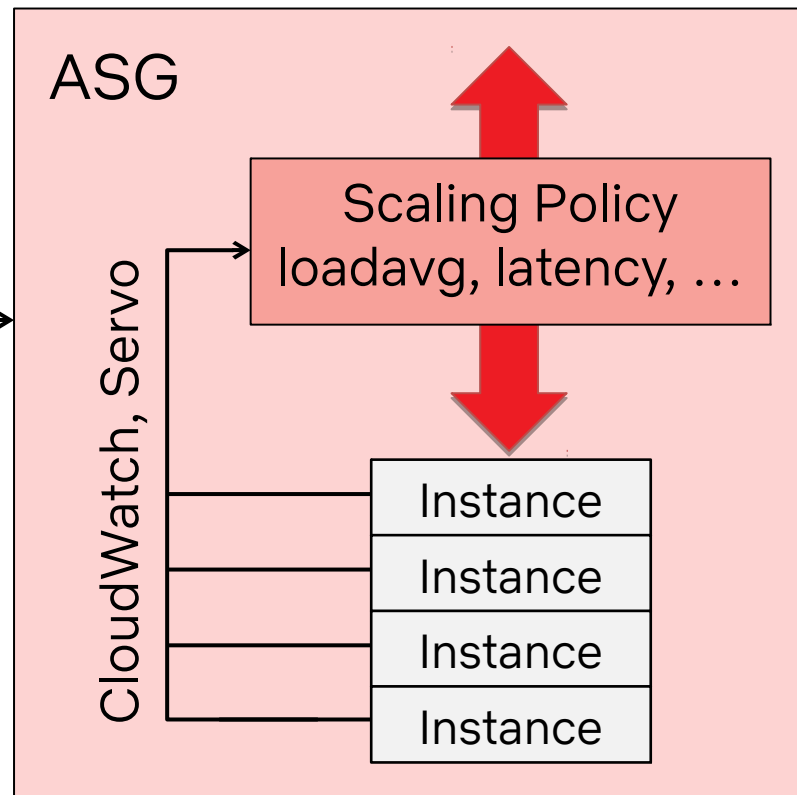
5 Key Issues

And How the Netflix Cloud is
Architected to Solve Them



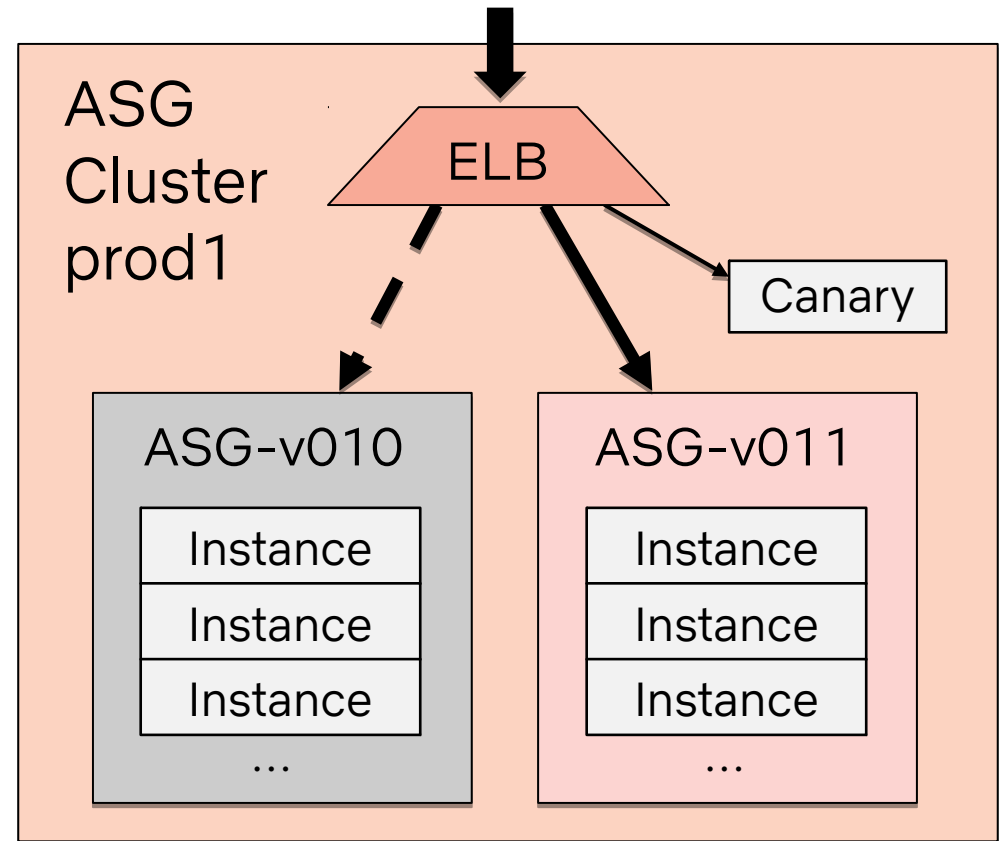
1. Load Increases → Auto Scaling Groups

- 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



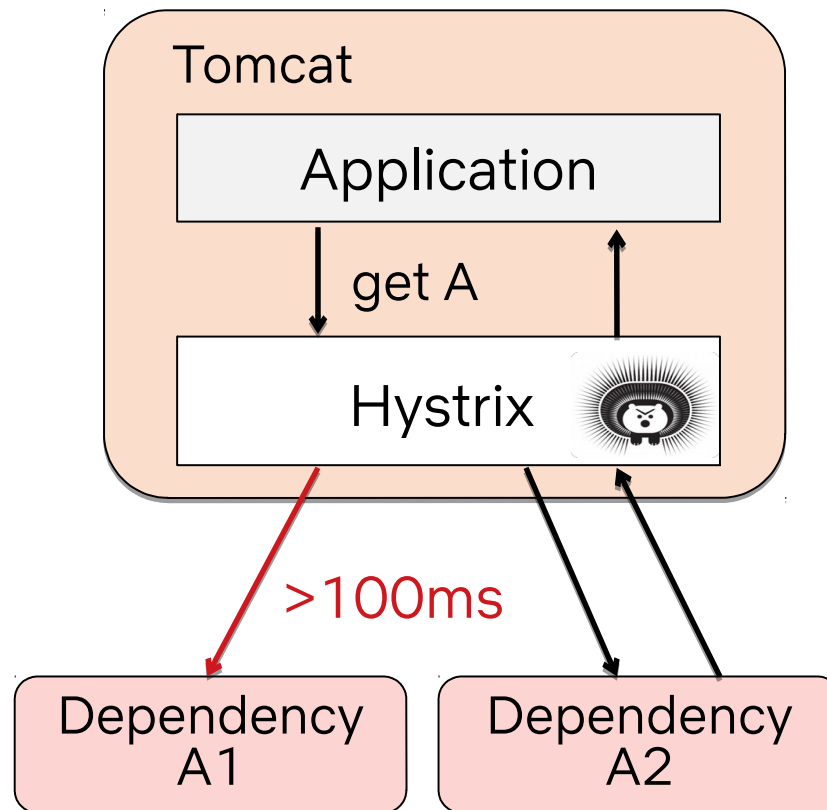
2. Bad Push → ASG Cluster Rollback

- 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



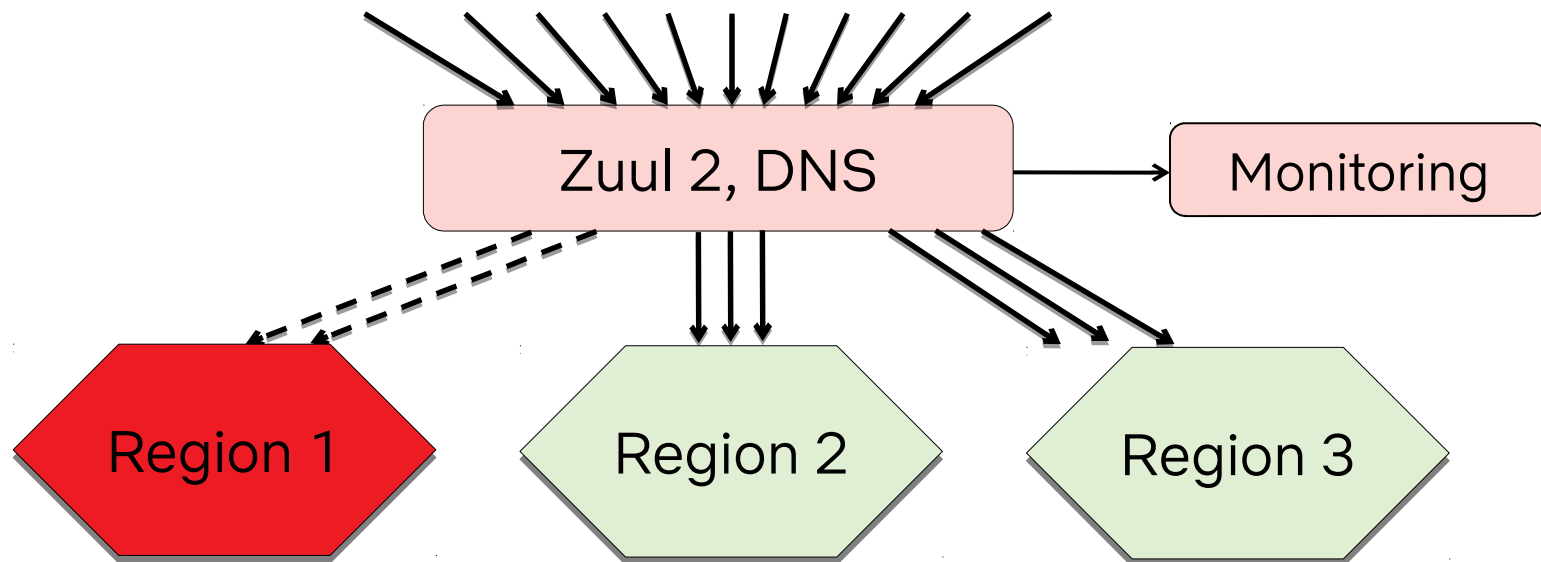
3. Instance Failure → Hystrix Timeouts

- 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

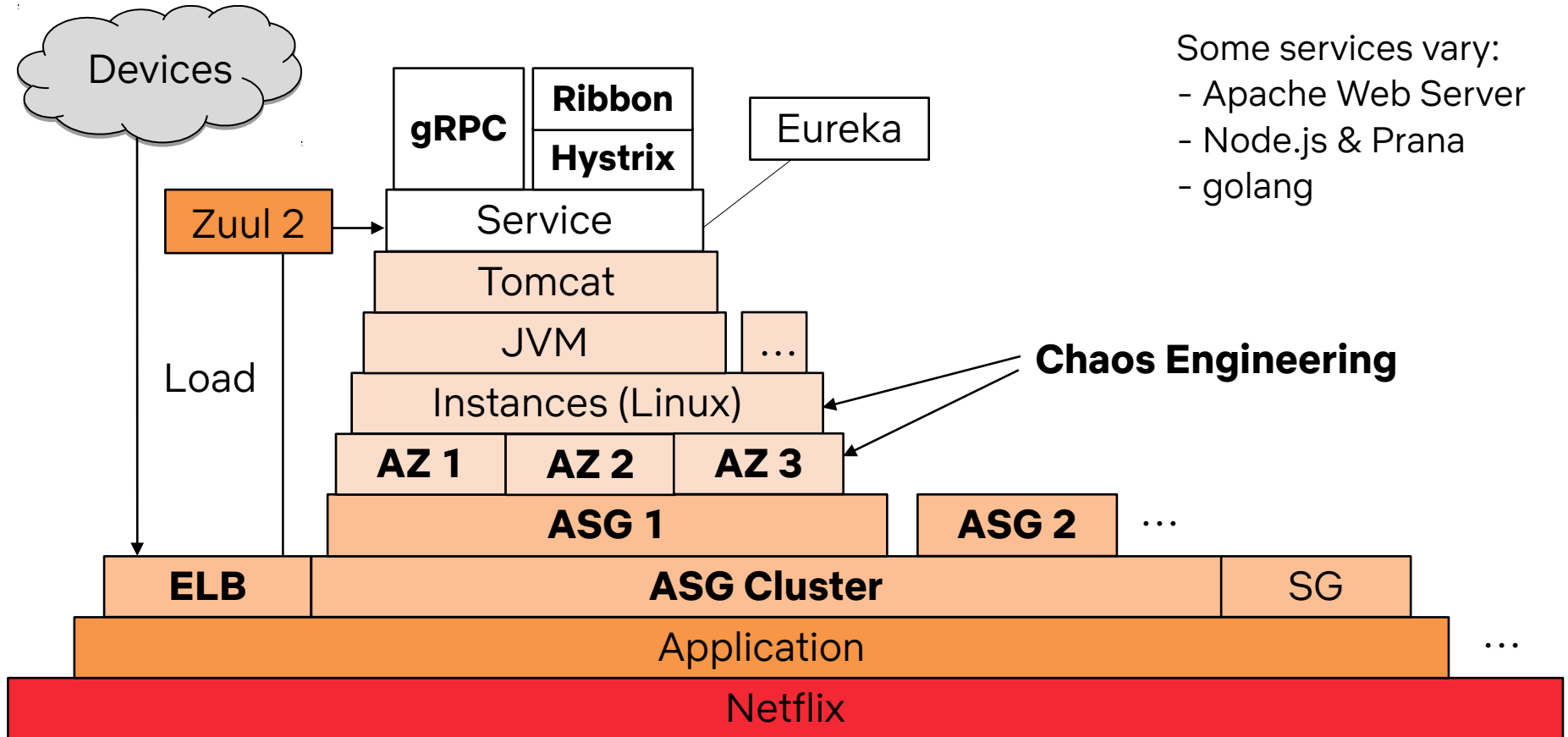
Janitor: kills unused instances

Security: checks violations

10-18: geographic issues



A Resilient Architecture

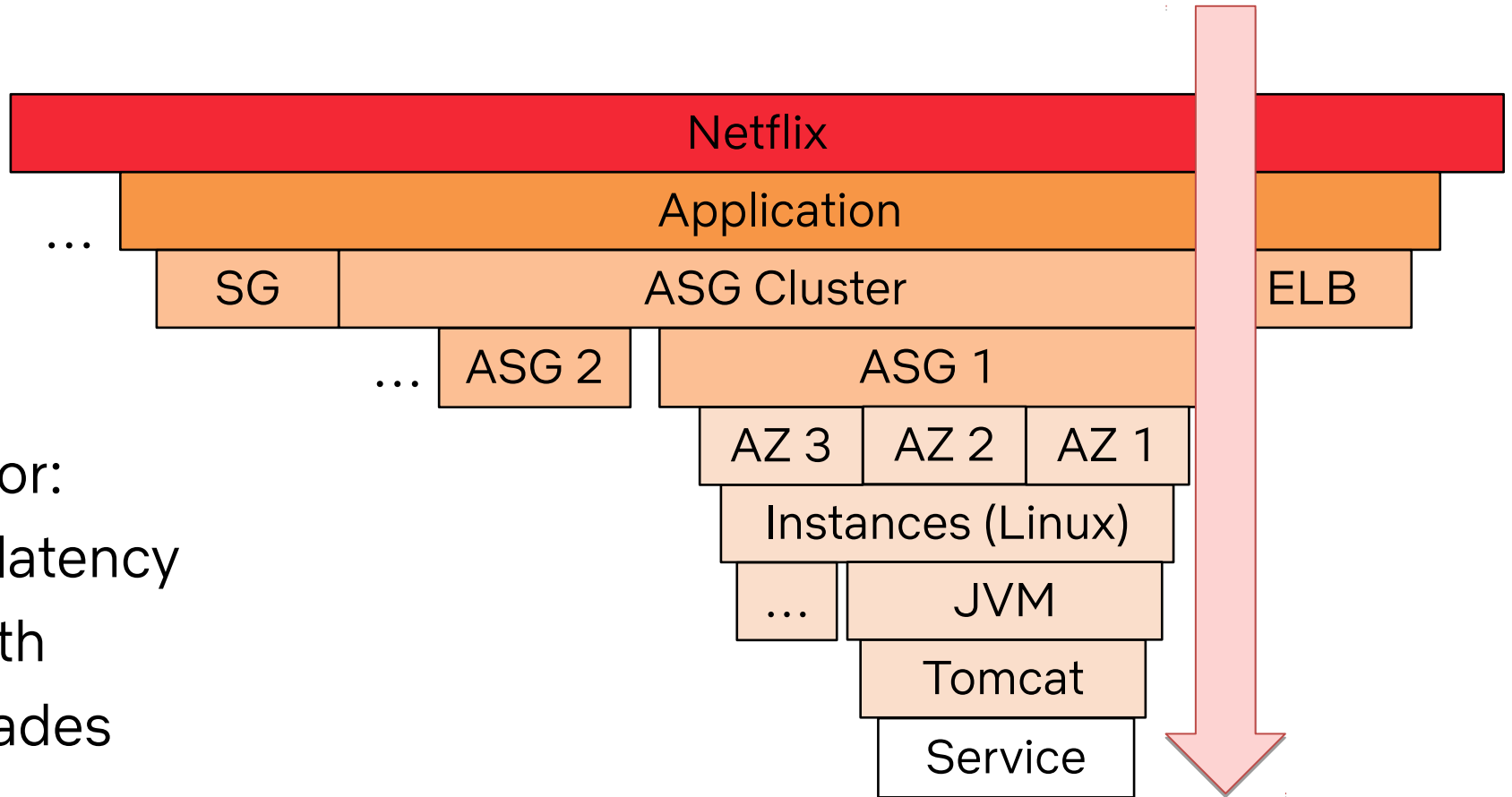


2. Methodology

Cloud & Instance

NETFLIX

Why Do Root Cause Perf Analysis?



Often for:

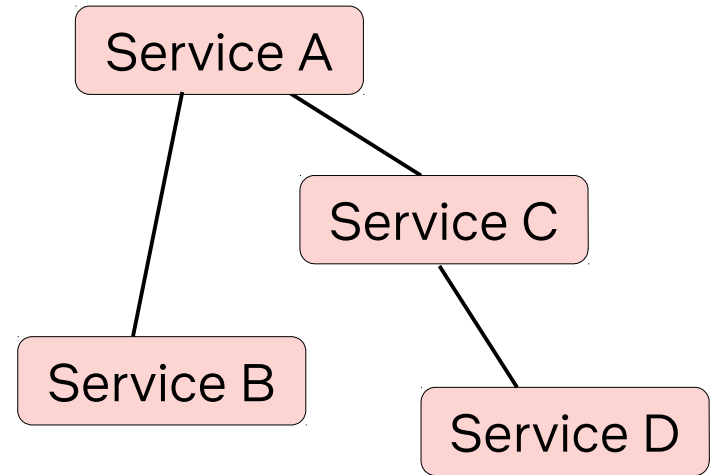
- High latency
- Growth
- Upgrades

Cloud Methodologies

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

For each microservice, check:

- Rate
- Errors
- Duration

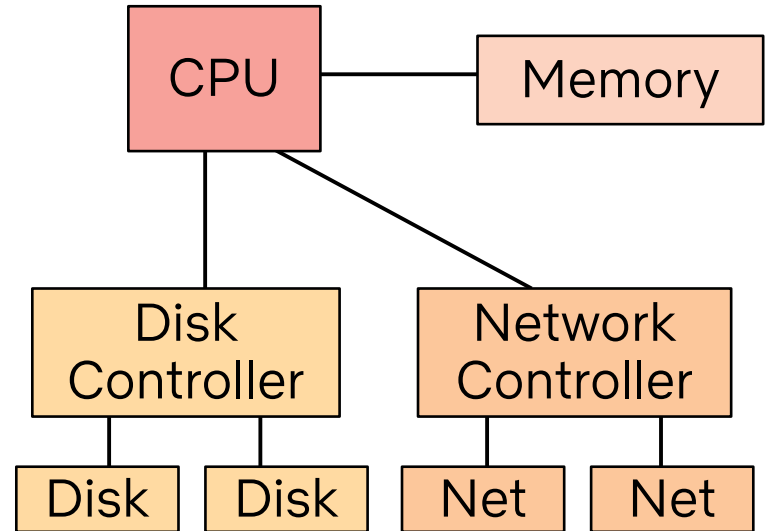


Instance Methodologies

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

For each resource, check:

- Utilization
- Saturation
- Errors

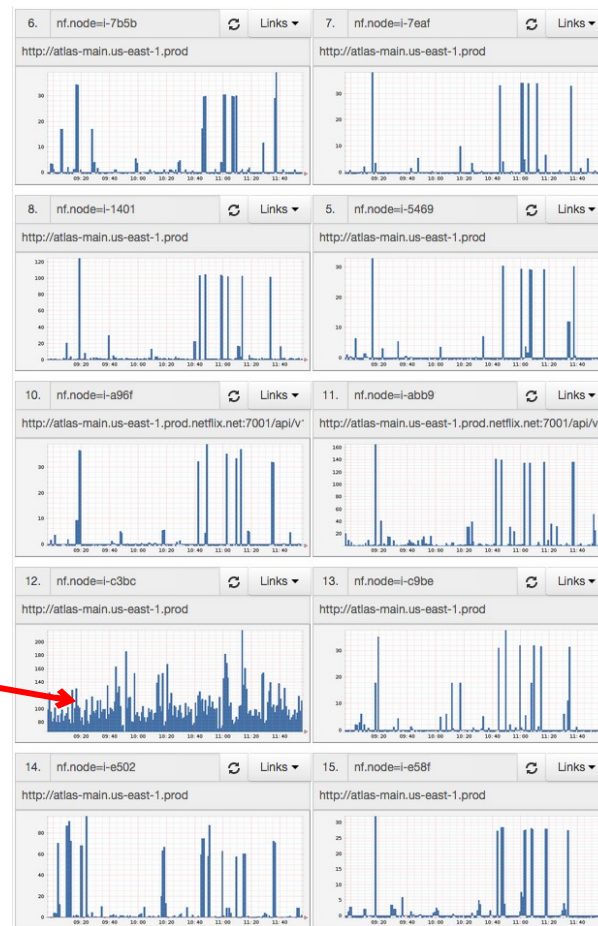


Bad Instance Anti-Method

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



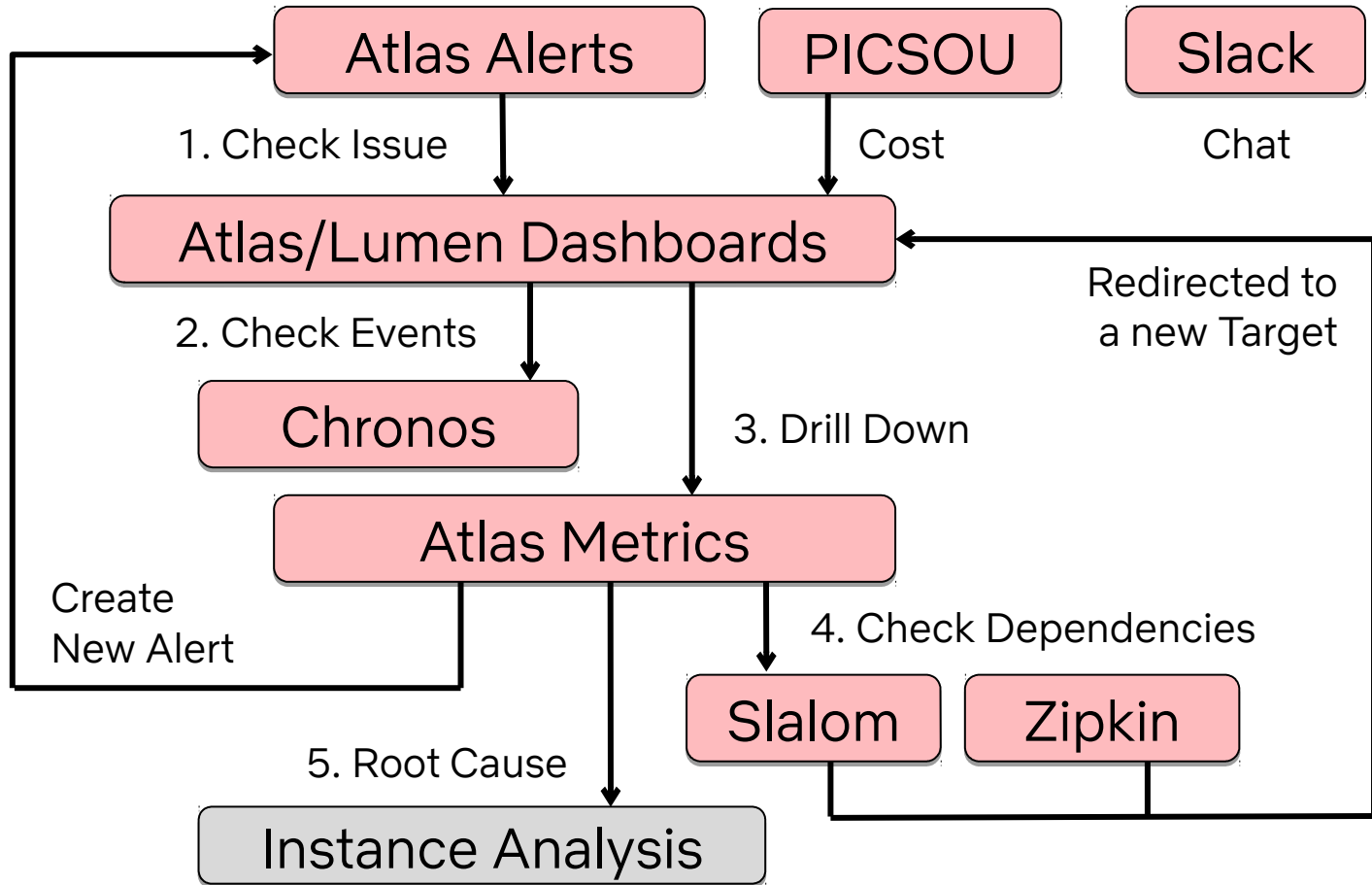
3. Cloud Analysis

Atlas, Lumen, Chronos, ...

NETFLIX

Netflix Cloud Analysis Process

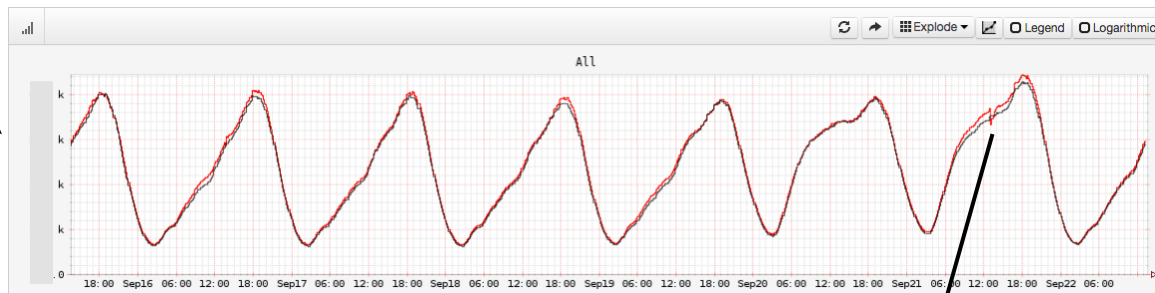
Example path
enumerated



Plus some other
tools not pictured

Atlas: Alerts

Custom alerts on streams per second (SPS) changes, CPU usage, latency, ASG growth, client errors, ...



The screenshot shows a Gmail inbox interface. At the top left is the Gmail logo and a search bar. On the left side, there are navigation buttons for "Compose", "Inbox" (with 34,122 items), "Starred", "Snoozed", "Important", and "Sent". On the right side, there are icons for "NETFLIX" and a user profile picture. The main area shows a list of five alert emails, all from "alert-do-not-reply". The first email is titled "[core-alert] <prod> [eu-west-1] Analysis of SPS_by_device.operationaN..." and is dated 9:24 AM. The second email is titled "[core-alert] Email-only Alert: eu-west-1 SPS Alert for [streaming_stick] - F" and is dated 9:22 AM. The third email is titled "[core-alert] <prod> [eu-west-1] Analysis of SPS_by_device.operationaN..." and is dated 9:21 AM. The fourth email is titled "[core-alert] Alert: eu-west-1 SPS Alert for [ce] - Prod eu-west-1 SPS_by_..." and is dated 9:20 AM. The fifth email is titled "[core-alert] Alert: eu-west-1 SPS All - Prod eu-west-1 SPS_ALL Summar..." and is dated 9:19 AM. At the top right of the email list, it says "1-50 of 43,548".

[core-alert] <prod> [eu-west-1] Analysis of SPS_by_device.operationName [redacted]
[streaming_stick] [redacted] >> inbox x

alert-do-not-reply@ [redacted]
to core-alerts

9:24 AM (1 hour ago) ☆ ↵ ⋮

Italian > English Translate message Turn off for: Italian x

Prod eu-west-1 SPS_by_device.operationName [redacted]

Summary: [eu-west-1] Analysis of SPS_by_device.operationName [redacted]

Check time: 2018-11-28 14:21:59
Time of alert: 2018-11-28 14:17:00

Environment: prod
Region: eu-west-1

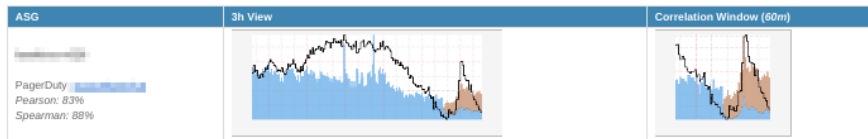
Match set: streaming_stick

Incident Key: [redacted]

Winston Diagnostics and Remediation

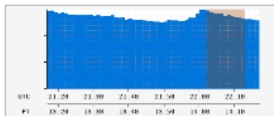
Did this correlate with a new or dying ASG?

Note: ASGs are listed in order of correlation confidence (starting with the highest) for at least one of the ASGs in the cluster.



Related to a Fast Property Change?

The highlighted portion of this graph shows the range of time evaluated for Fast Property changes. The query performed filters things which we believe are NOT related to typical streaming issues. Here's an [unfiltered list of Fast Properties in Chronos UI](#).



time	region	action	name	app	cluster	stack
14:10:56	eu-west-1	delete	[redacted]	central-frontend	central-frontend-prod central-frontend-prod central-frontend-prod	awscli, aws, Git, aws, awscli
14:10:19		delete	[redacted]	central-frontend	central-frontend-prod	awscli
14:09:19		update	[redacted]	central-frontend	central-frontend-prod	
14:06:00		delete	[redacted]	central-frontend	central-frontend-prod	
14:05:49		delete	[redacted]	central-frontend	central-frontend-prod	
14:04:19		delete	[redacted]	central-frontend	central-frontend-prod	
14:03:58		delete	[redacted]	central-frontend	central-frontend-prod	

Metrics

View the status of key metrics below. These can be changed or redefined in the [Alert configuration](#).

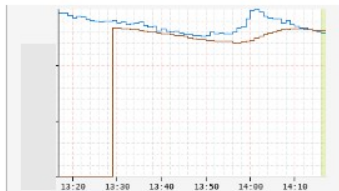
Alert Signal (as of 2018-11-28 14:17:00) (PNG | UI)



streaming_stick
Max : 1.000 Min : 0.000
Avg : 16.667m Lest : 1.000
Tot : 1.000 Cnt : 60.000
... 1 of 1 lines matched filter ...

Frame: 1h, End: 2018-11-28T14:18:00:00[US/Pacific], Step: 1m
Fetch: 100ms (L: 0.7, 2h, 5.2h, 2.0, D: 5.2h, 305.2h, 120.0h)

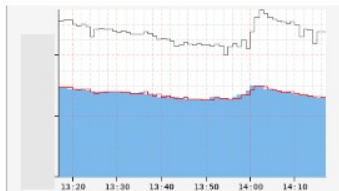
REGIONAL: Alert Visualization (as of 2018-11-28 14:17:00) (PNG | UI)



streaming_stick 80%
[redacted]
[redacted]

Frame: 1h, End: 2018-11-28T14:18:00:00[US/Pacific], Step: 1m
Fetch: 100ms (L: 0.9, 2h, 2.2h, 3.0, D: 2.0h, 122.2h, 100.0h)

REGIONAL: SPS by Data Source (as of 2018-11-28 14:17:00) (PNG | UI)



Description / Instructions

This is the signal the alert is based on.

[View Current Graph \(PNG | UI\)](#)

Description / Instructions

[View Current Graph \(PNG | UI\)](#)

Description / Instructions

The blue area is the signal we use for alerting. If the other signals look good and the blue drops, it's likely just a metrics issue of some sort.

[View Current Graph \(PNG | UI\)](#)

[core-alert] <prod> [eu-west-1] Analysis of SPS_by_device.operationName [redacted]
 [streaming_stick] [redacted] >> inbox x

alert-do-not-reply@ [redacted]
 to core-alerts

9:24 AM (1 hour ago) ☆ ↵ ⋮

Italian > English Translate message Turn off for: Italian x

Prod eu-west-1 SPS_by_device.operationName [redacted]

Summary: [eu-west-1] Analysis of SPS_by_device.operationName [redacted]

Check time: 2018-11-28 14:21:59
 Time of alert: 2018-11-28 14:17:00

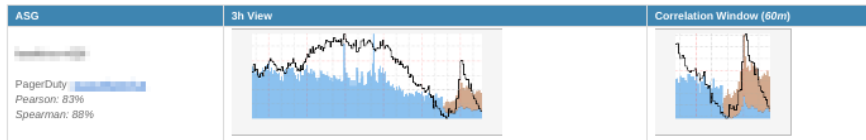
Environment: prod
 Region: eu-west-1

Winston: Automated Diagnostics & Remediation

Winston Diagnostics and Remediation

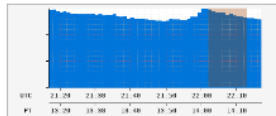
Did this correlate with a new or dying ASG?

Note: ASGs are listed in order of correlation confidence (starting with the highest) for at least one of the ASGs in the cluster.



Related to a Fast Property Change?

The highlighted portion of this graph shows the range of time evaluated for Fast Property changes. The query performed filters things which we believe are NOT related to typical streaming issues. Here's an [unfiltered list of Fast Properties in Chronos UI](#).



Chronos: Possible Related Events

time	region	action	name	app	cluster	stack
14:10:56	eu-west-1	delete	[redacted]	streaming_stick	[redacted]	streaming_stick, SPS, SPS, SPS
14:10:19		delete	[redacted]	[redacted]	[redacted]	[redacted]
14:09:19		update	[redacted]	[redacted]	[redacted]	[redacted]
14:06:00		delete	[redacted]	[redacted]	[redacted]	[redacted]
14:05:49		delete	[redacted]	[redacted]	[redacted]	[redacted]
14:04:19		delete	[redacted]	[redacted]	[redacted]	[redacted]
14:03:58		delete	[redacted]	[redacted]	[redacted]	[redacted]

Metrics

View the status of key metrics below. These can be changed or redefined in the [Alert configuration](#).

Alert Signal (as of 2018-11-28 14:17:00) (PNG | UI)



streaming_stick
 Max : 1.000 Min : 0.000
 Avg : 16.667m Lmt : 1.000
 Tot : 1.000 Cnt : 60.000
 ... 1 of 1 lines matched filter ...

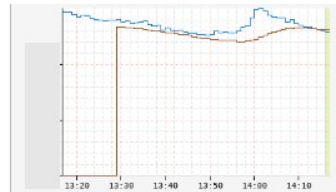
Frame: 1h, End: 2018-11-28T14:18:00:00[US/Pacific], Step: 1m
 Fetch: 1000ms (L: 87.3h, 5.5h, 2.0, D: 9.2M, 305.0h, 120.0h)

Description / Instructions

This is the signal the alert is based on.

[View Current Graph \(PNG | UI\)](#)

REGIONAL: Alert Visualization (as of 2018-11-28 14:17:00) (PNG | UI)

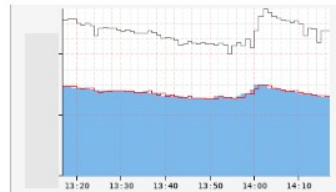


Description / Instructions

[View Current Graph \(PNG | UI\)](#)



REGIONAL: SPS by Data Source (as of 2018-11-28 14:17:00) (PNG | UI)



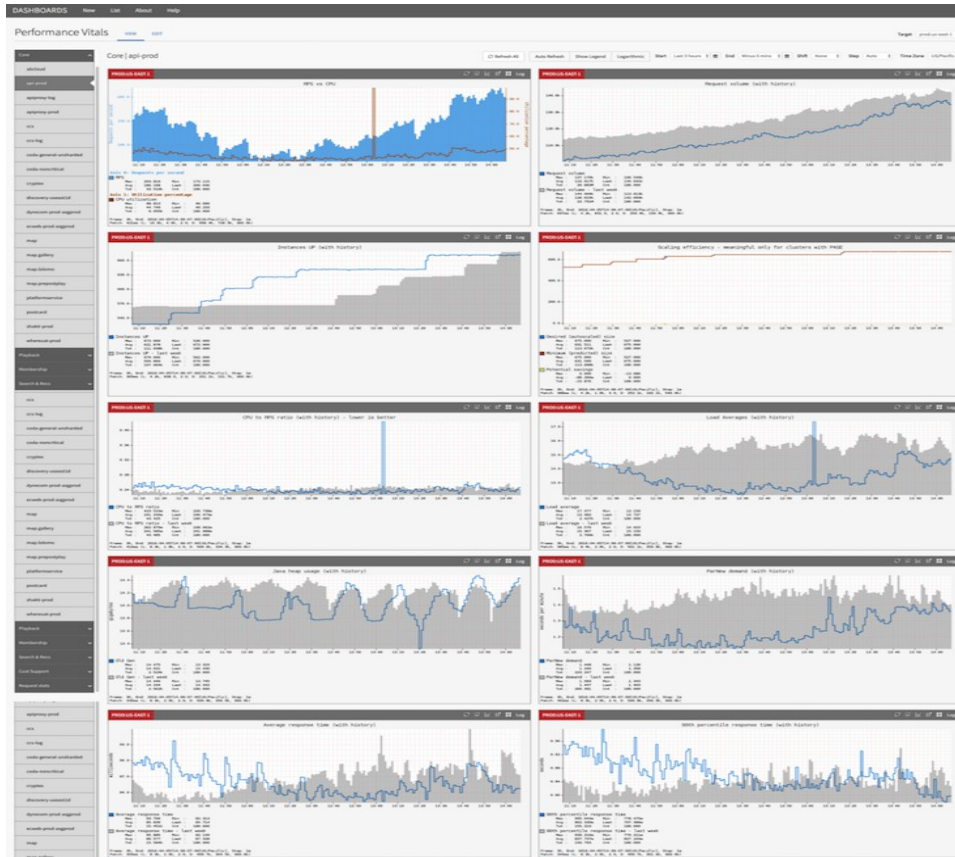
Description / Instructions

The blue area is the signal we use for alerting. If the other signals look good and the blue drops, it's likely just a metrics issue of some sort.

[View Current Graph \(PNG | UI\)](#)

Links to Atlas
Dashboards & Metrics

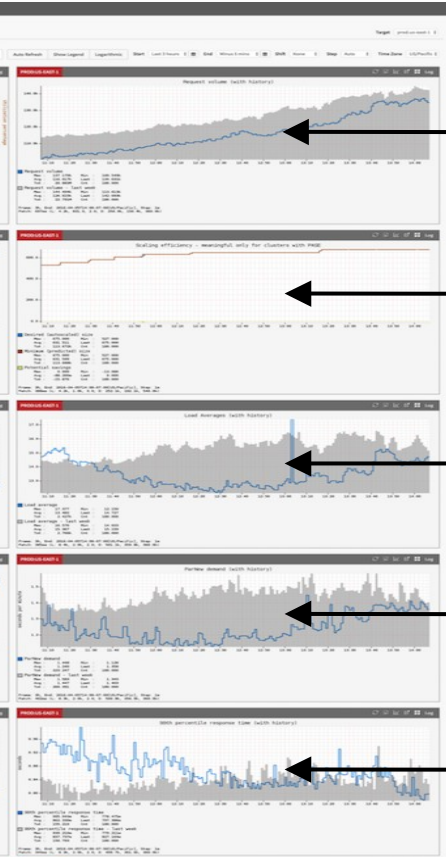
Atlas: Dashboards



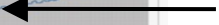
Atlas: Dashboards

Netflix perf vitals dashboard

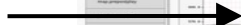
1. RPS, CPU



2. Volume



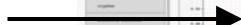
3. Instances



4. Scaling



5. CPU/RPS



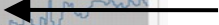
6. Load avg



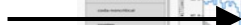
7. Java heap



8. ParNew



9. Latency

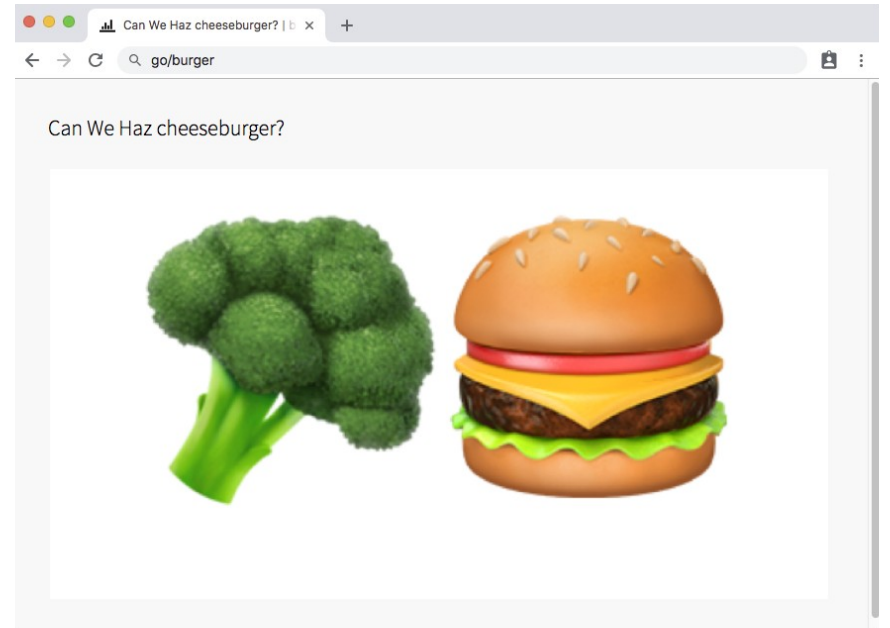


10. 99th tile



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 screenshot displays the Atlas Metrics interface. The top navigation bar includes the Atlas logo, 'Edit View' buttons, 'Atlas UI stable build: Switch 24 hours', and utility buttons like 'Clear', 'Help', 'What's New', and 'Support'. The main content area is split into a configuration sidebar on the left and a visualization area on the right.

Configuration Sidebar:

- Scope:** Search for 'api'. Results include 'nf.cluster:api', 'nf.app', and 'nf.asg'.
- name:** Search for 'cpu'. Results include 'cpu', 'CpuRawUser', 'CpuRawSystem', 'ssCpuUtilization', 'ssCpuUser', and 'ssCpuSystem'.
- Add tag:** A dropdown menu with 'Select...' and a 'show all' checkbox.
- Aggregation:** Radio buttons for 'sum' (selected), 'max', 'min', 'node avg', and 'UP avg'. A 'per step' checkbox is also present.
- Grouping:** A search filter input field.

Visualization Area:

- Env:** prod test | **Region:** us-east-1
- Dashboard target:** prod.us-east-1
- Graph:** A line chart showing two metrics over time. The y-axis ranges from 50.0k to 350k. The x-axis shows dates from Oct 20 to Oct 21. A blue line (CpuRawSystem) remains near zero. An orange line (CpuRawUser) peaks at approximately 350k around Oct 20 20:00 and then declines.
- Table:** A summary table with columns: Min, Max, Average, Total, Name, and Visibility.

Min	Max	Average	Total	Name	Visibility
4.80k	21.9k	12.3k	3.55M	(name=CpuRawSystem)	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
68.0k	382k	198k	57.0M	(name=CpuRawUser)	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

Time Zone: PT UTC PT+UTC UTC+PT

Start: Prior day | **End:** 5 minutes ago

Export options: JSON, CSV, CSV+, Stats, Send to

Atlas: Metrics

The screenshot shows the Atlas Metrics interface. On the left, the configuration panel includes sections for Scope, name, Add tag, Aggregation, and Grouping. On the right, the main dashboard displays a line graph, a summary statistics table, and time range controls. Annotations with arrows point to various elements: 'Application' points to the 'nf.app' entries in the Scope list; 'Metrics' points to the 'cpu' entries in the name list; 'Presentation' points to the 'sum' aggregation option; 'Region' points to the 'us-east-1' dropdown; 'Interactive graph' points to the line graph; 'Summary statistics' points to the table below the graph; and 'Time range' points to the 'Start' and 'End' controls.

Application →

Metrics →

Presentation →

Region →

Interactive graph →

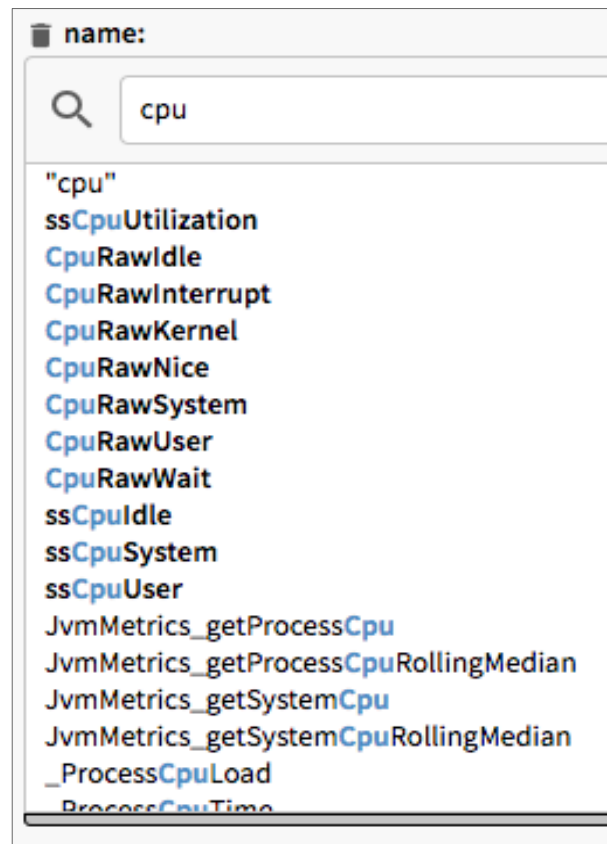
Summary statistics →

Time range →

Min	Max	Average	Total	Name
4.80k	21.9k	12.3k	3.55M	(name=CpuRawSystem)
68.0k	382k	198k	57.0M	(name=CpuRawUser)

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

CHRONOS Prod Events Fast Property Subscriptions Miss Event Alerts Other Events More Insight Support

2018-11-27T10:09

Start: e-1h End: now
yyyy-MM-ddTHH:mm

500 new events

10 of 42 selected

Show Statistics

Start Time	Region	Application	Cluster	Stack	Source App	Action	Event Type	Name	Description	Detail
2018-11-27 10:08:54	us-east-1	delete	object	
2018-11-27 10:08:54	us-east-1	package	...	create	scheduled Action	
2018-11-27 10:08:54	us-east-1	package	...	create	scheduled Action	
2018-11-27 10:08:53		publish	gutenberg	
2018-11-27 10:08:53	us-east-1	package	...	create	scheduled Action	
2018-11-27 10:08:53	us-east-1	package	...	create	scheduled Action	
2018-11-27 10:08:53	us-east-1	main	...	update	cluster	
2018-11-27 10:08:53	us-east-1	package	...	create	scheduled Action	
2018-11-27 10:08:53		alpha	...	start	job	
2018-11-27 10:08:52	us-east-1	package	...	create	scheduled Action	
2018-11-27 10:08:52		end	workflow	
2018-11-27 10:08:52	{ 2 values }	{ 2 values }	...	{ 3 values }	{ 2 values }	

Chronos: Change Tracking

Predefined Queries

All events except low criticality

Save

Search

Search chronos events X

Region

UNDEFINED us-east-1 us-west-1

us-west-2 eu-west-1

Applied Filters

Criticality

exclude: low

Add filter include exclude

Source App

Start: e-1h

2018-11-27T10:09

End: now

yyyy-MM-ddTHH:mm

500 new events

Scope

Time Range

Event Log

Show Statistics

10 of 42 selected

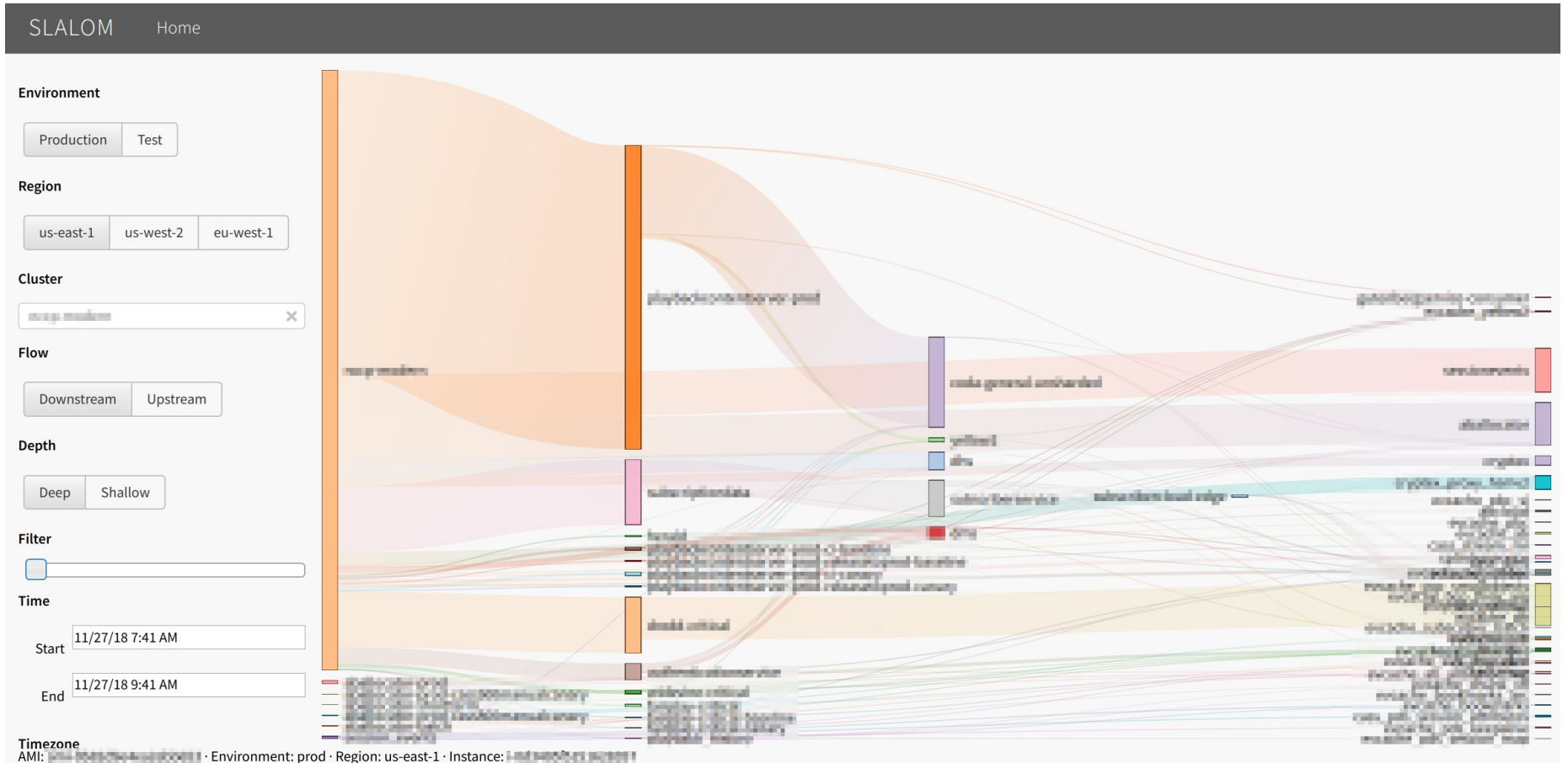
Start Time	Region	Application	Cluster	Stack	Source App	Action	Event Type	Name	Description	Detail
2018-11-27 10:08:54	us-east-1	chronos_backend-main	chronos_backend-main	package	chronos_backend-main	delete	object	chronos_backend-main	Task status change: task status finished, new status: failed, reason: ResourceGroup (1...)	Q
2018-11-27 10:08:54	us-east-1	chronos_backend-main	chronos_backend-main	package	chronos_backend-main	create	scheduled Action	chronos_backend-main	chronos_backend-main	Q
2018-11-27 10:08:54	us-east-1	chronos_backend-main	chronos_backend-main	package	chronos_backend-main	create	scheduled Action	chronos_backend-main	chronos_backend-main	Q
2018-11-27 10:08:53						publish	gutenberg	chronos_backend-main	2018-11-27 10:08:53: chronos_backend-main	Q
2018-11-27 10:08:53	us-east-1	chronos_backend-main	chronos_backend-main	package	chronos_backend-main	create	scheduled Action	chronos_backend-main	chronos_backend-main	Q
2018-11-27 10:08:53	us-east-1	chronos_backend-main	chronos_backend-main	package	chronos_backend-main	create	scheduled Action	chronos_backend-main	chronos_backend-main	Q

CREATE FILTER

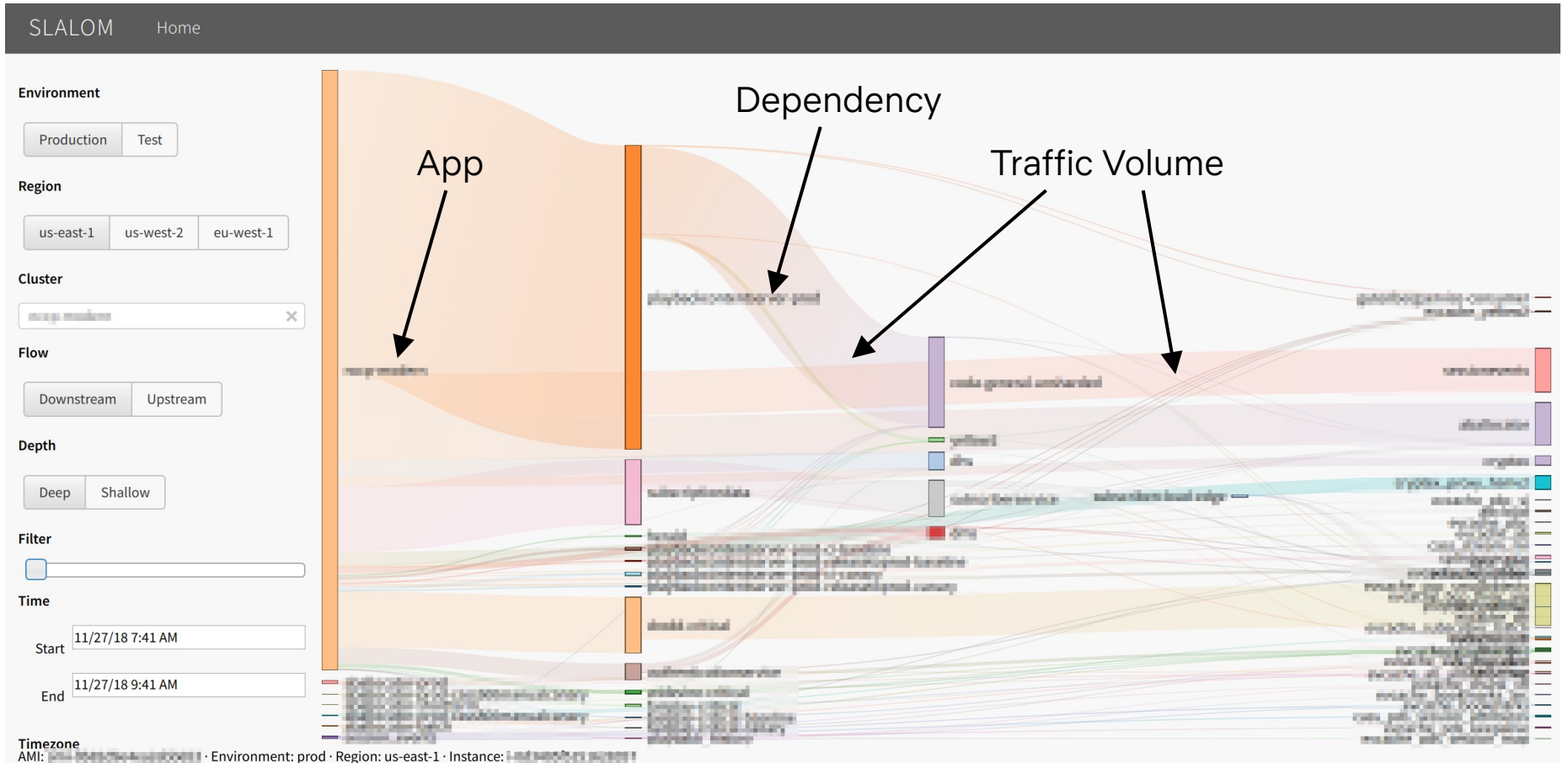
+ Include - Exclude

chronos_backend-main	package	chronos_backend-main	update	cluster	chronos_backend-main	Chronos backend is indexing 98 events per minute
----------------------	---------	----------------------	--------	---------	----------------------	--

Slalom: Dependency Graphing



Slalom: Dependency Graphing



Zipkin UI: Dependency Tracing

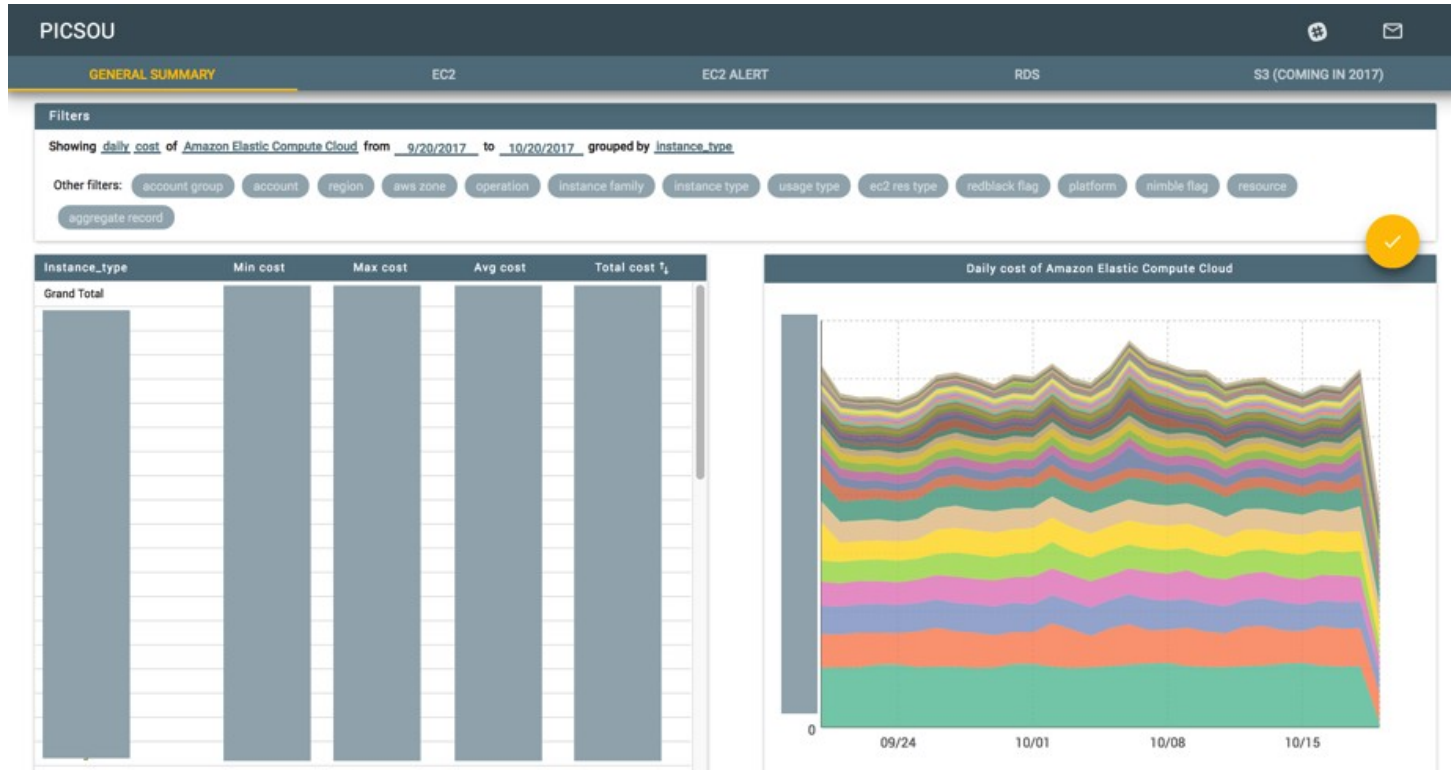
Duration: 896.000ms Services: 20 Depth: 9 Total Spans: 793 JSON

Expand All Collapse All Filter Service ...

[Service 1] [Service 2] [Service 3] [Service 4] [Service 5] [Service 6] [Service 7] [Service 8] [Service 9] [Service 10] [Service 11] [Service 12] [Service 13] [Service 14] [Service 15] [Service 16] [Service 17] [Service 18] [Service 19] [Service 20]



PICSOU: AWS Usage



← Breakdowns

← Cost per hour

← Details (redacted)

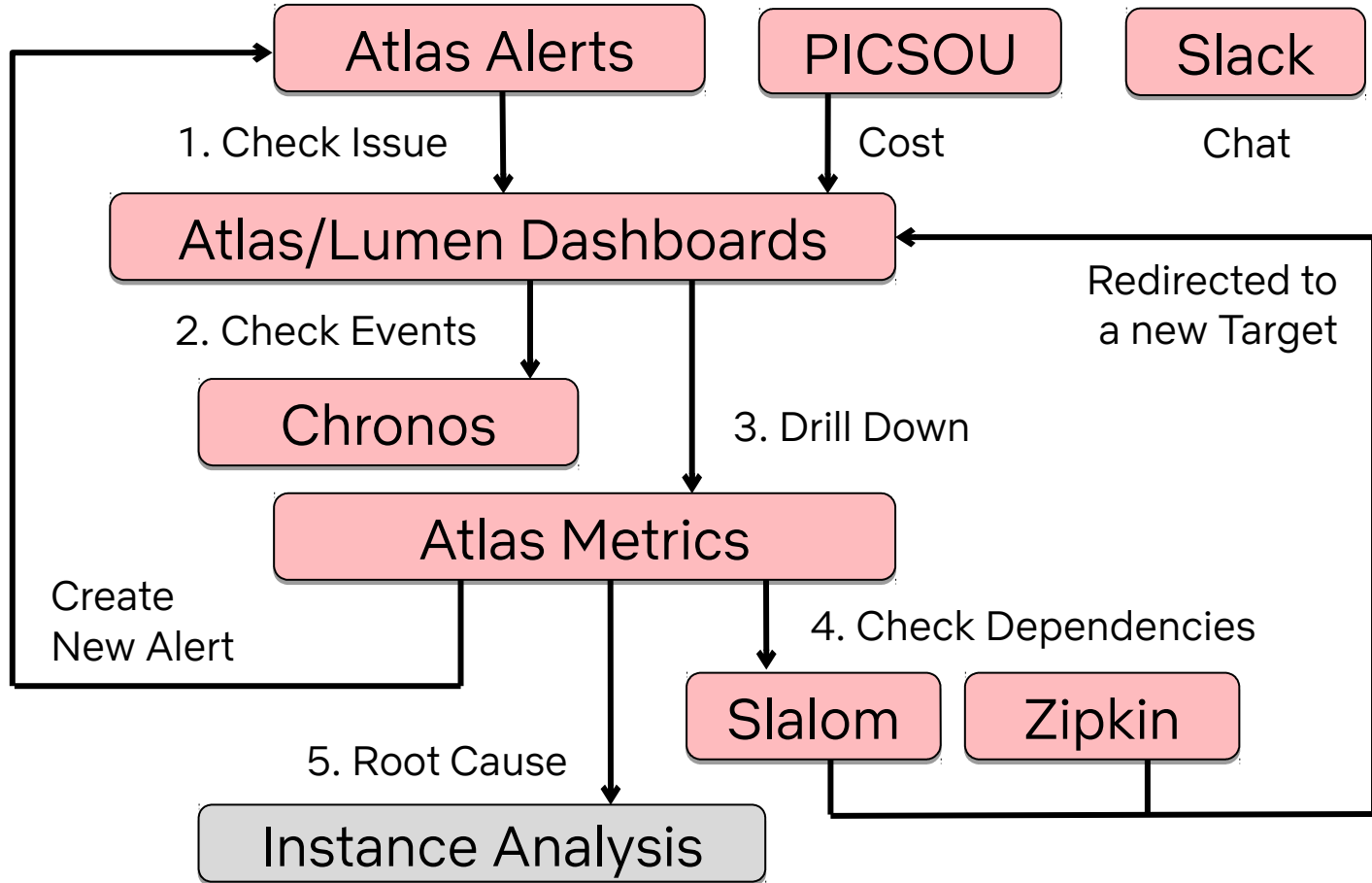
Slack: Chat

Latency is high in us-east-1

Sorry
We just did a bad push

Netflix Cloud Analysis Process

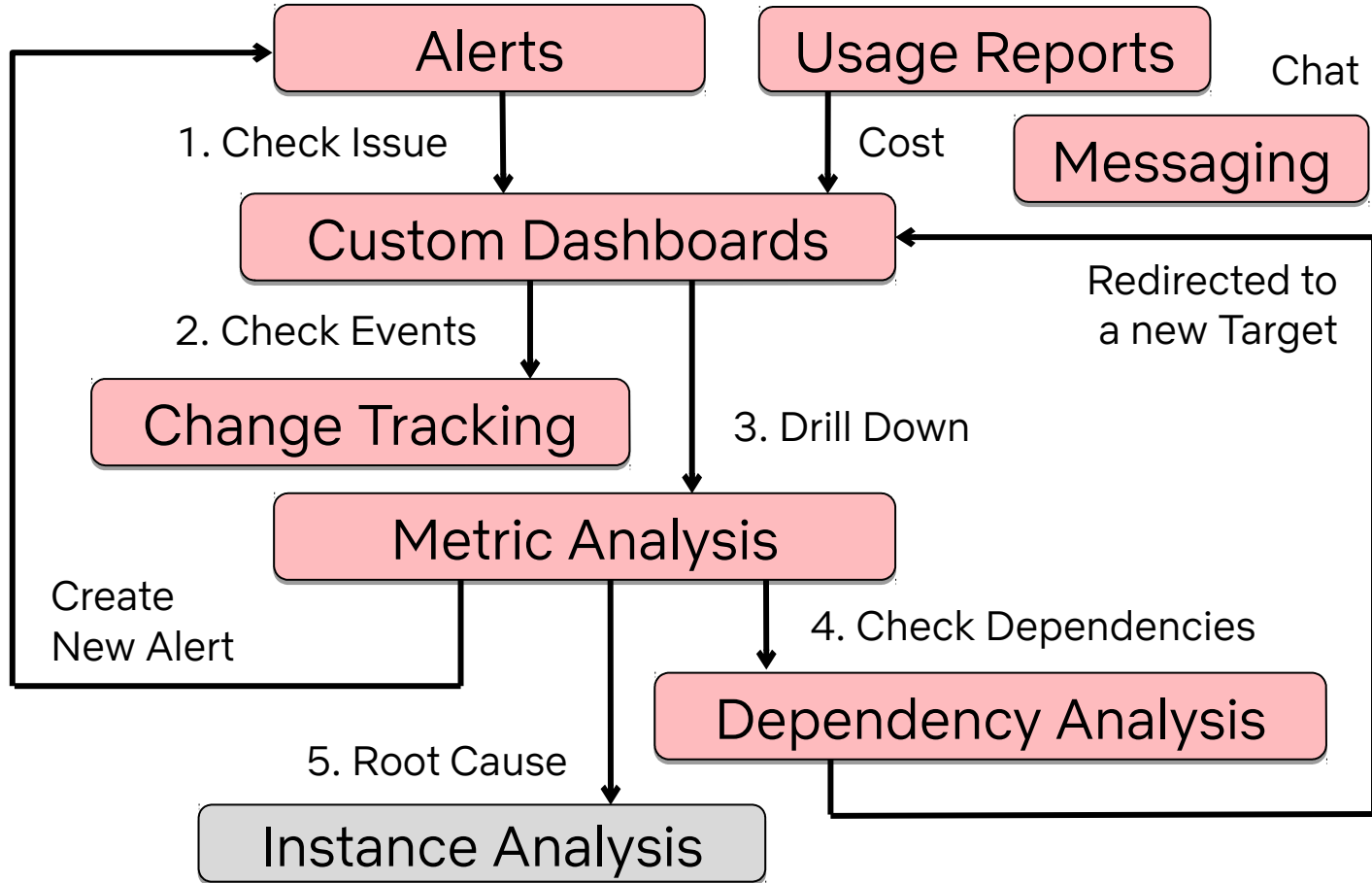
Example path
enumerated



Plus some other
tools not pictured

Generic Cloud Analysis Process

Example path
enumerated



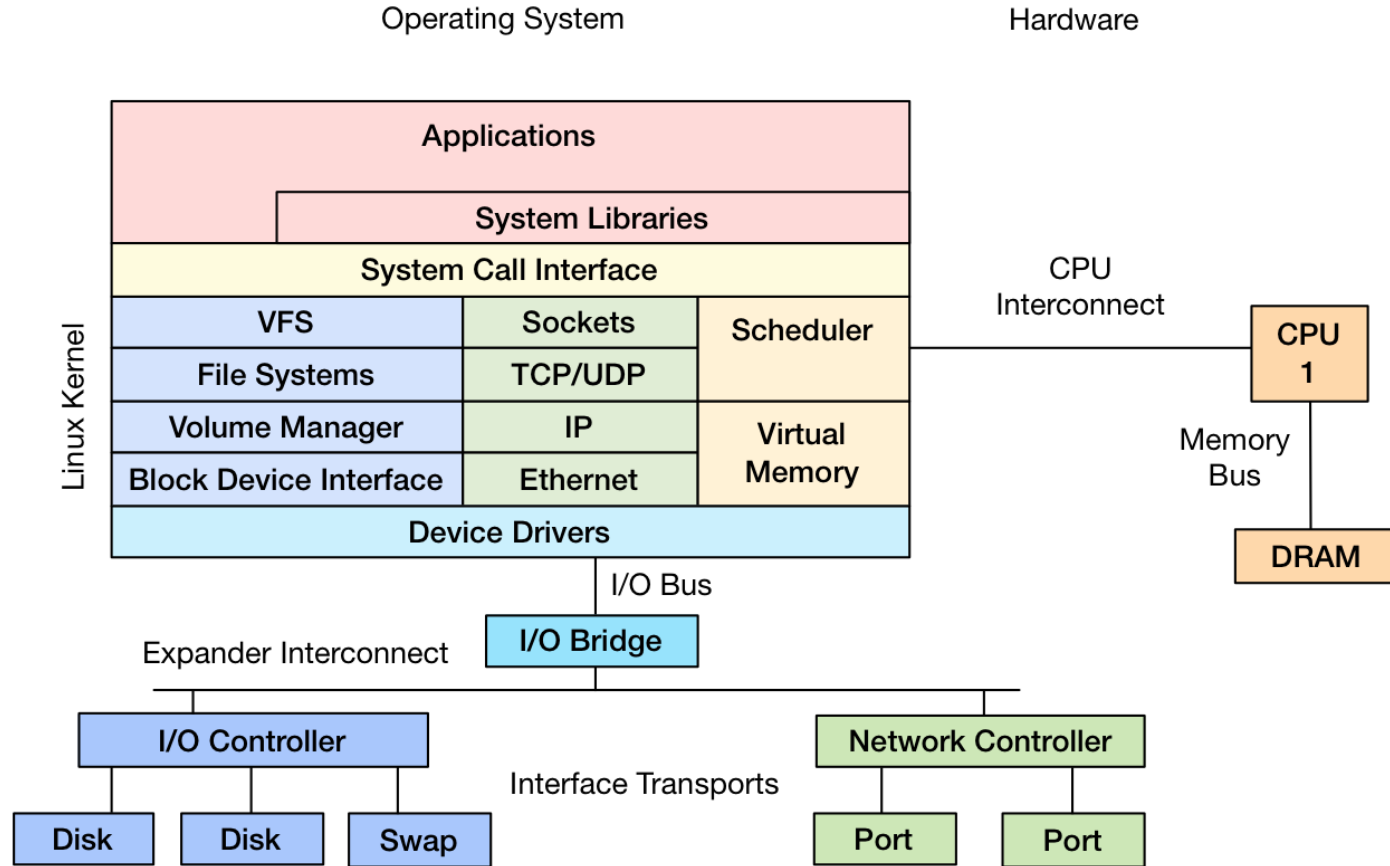
Plus other tools
as needed

4. Instance Analysis

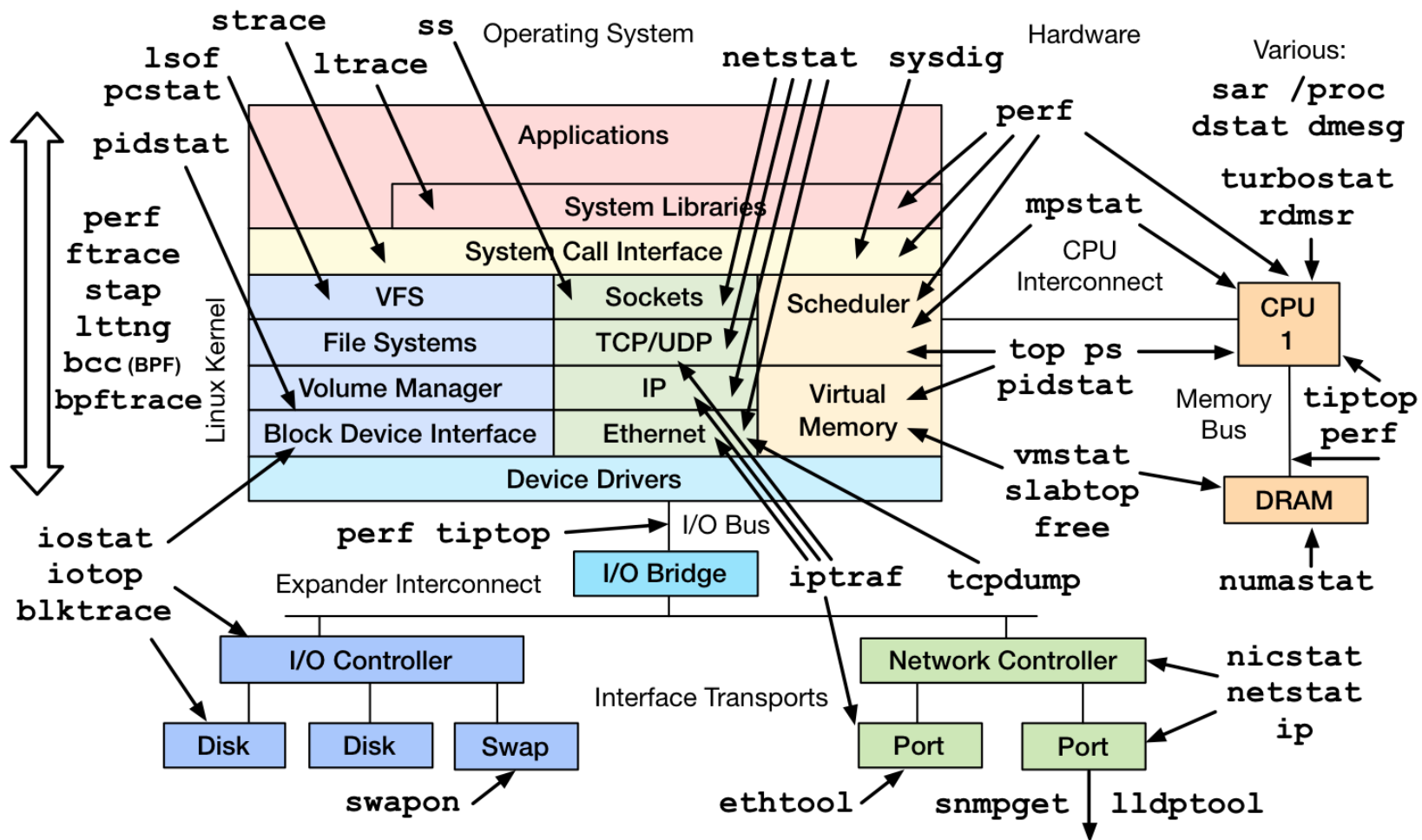
- 1. Statistics
- 2. Profiling
- 3. Tracing
- 4. Processor Analysis

NETFLIX

Linux Performance



Linux Performance Observability Tools



1. Statistics

Linux Tools

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

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

09:43:53 PM IFACE  rxpck/s   txpck/s   rxkB/s   txkB/s   rxcmp/s   txcmp/s  rxmcst/s %ifutil
09:43:54 PM   lo      15.00    15.00     1.31     1.31     0.00     0.00     0.00     0.00
09:43:54 PM  eth0 26392.00 33744.00 19361.43 28065.36  0.00     0.00     0.00     0.00

09:43:53 PM active/s passive/s   iseg/s   oseg/s
09:43:54 PM   18.00   132.00 17512.00 33760.00

09:43:53 PM atptf/s  estres/s retrans/s isegerr/s   orsts/s
09:43:54 PM   0.00    0.00    11.00    0.00    0.00
[...]
```

- 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



SPINNAKER Applications Search Properties Analytics Search

bgregg PIPELINES INFRASTRUCTURE TASKS

Infrastructure CLUSTERS LOAD BALANCERS SECURITY GROUPS PROPERTIES

SEARCH

ACCOUNT test

REGION us-east-1 us-west-1 us-west-2

STACK (none) amazon bgregg bionic build c5.18xl

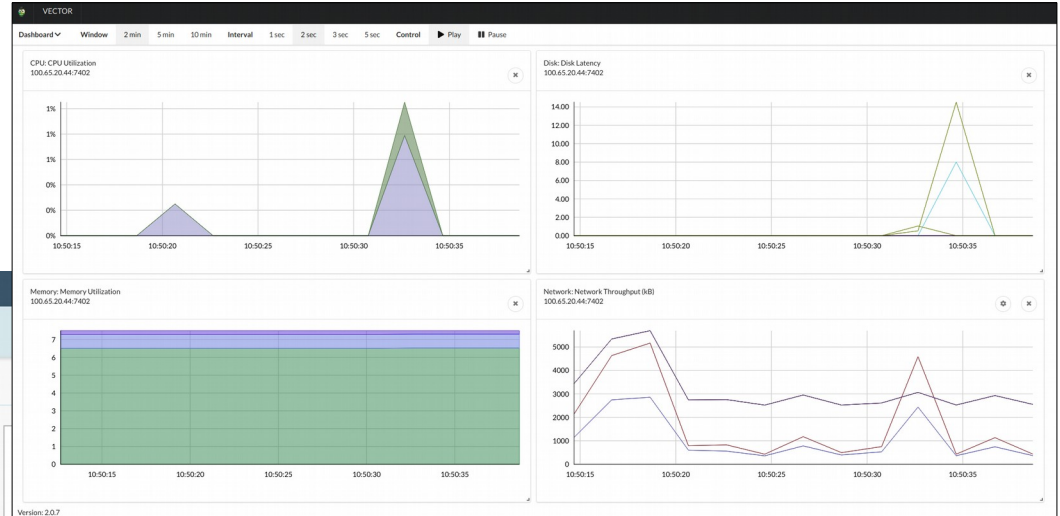
TEST bgregg-bionic-test 5

US-EAST-1 V001 3 V000 1

US-WEST-1 V001 1

TEST bgregg-build-hvm 2

US-EAST-1 V005 2



Instance Actions Insight

INSTANCE INFO

Launched 20:14:1c

In TE 1c

Type m5.large

Server Group bgregg-bionic-test-v001

VPC

Subnet

Image ID

Perf Vitals Dashboard
Generic App Dashboard
Atlas (Metrics)
Base Server (Metrics)
Vector



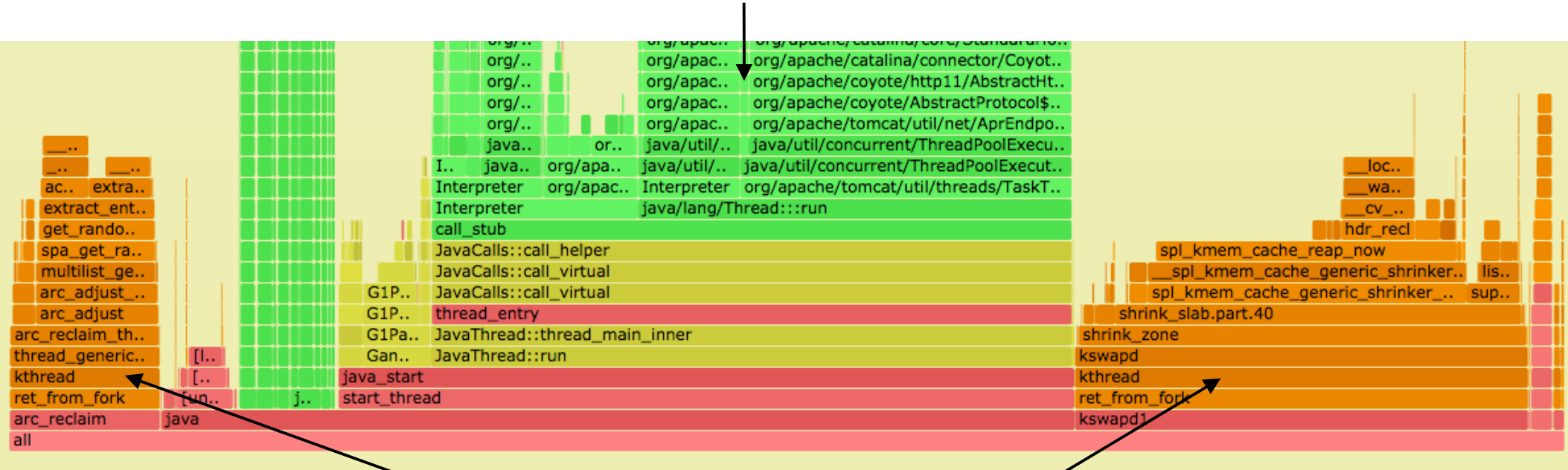
2. Profiling

Experience:

“ZFS is eating my CPUs”

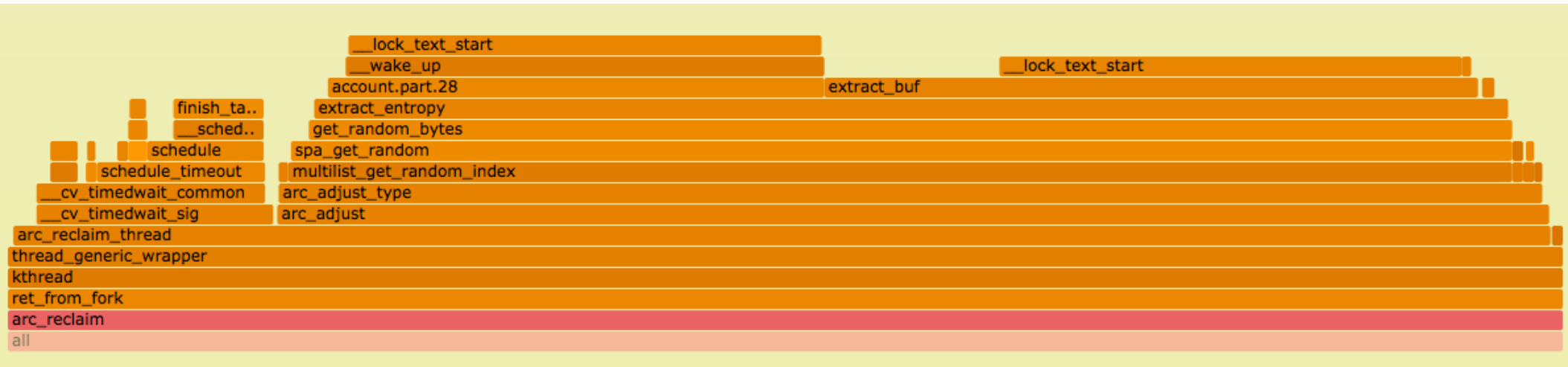
CPU Mixed-Mode Flame Graph

Application (truncated)

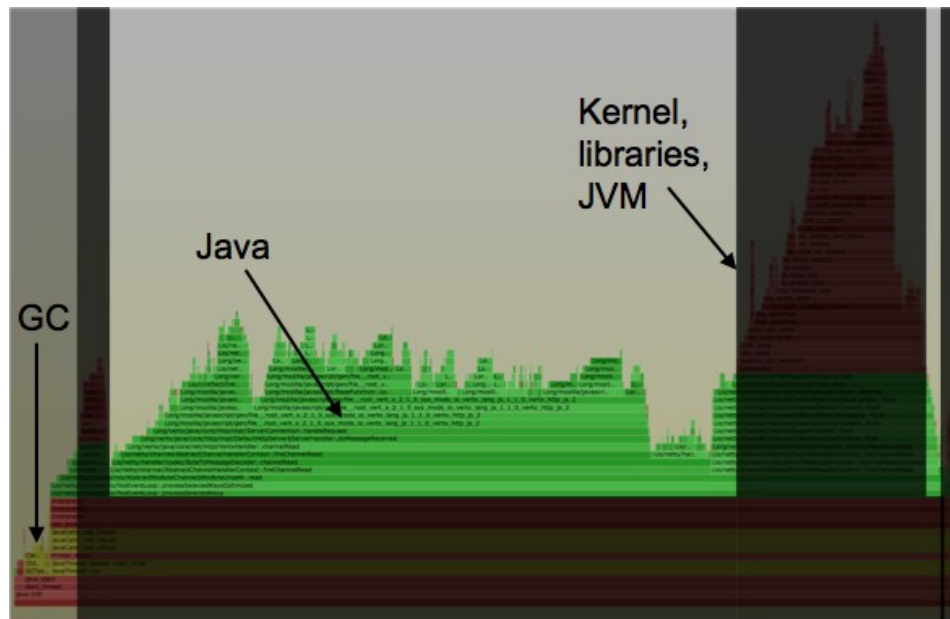


38% kernel time (why?)

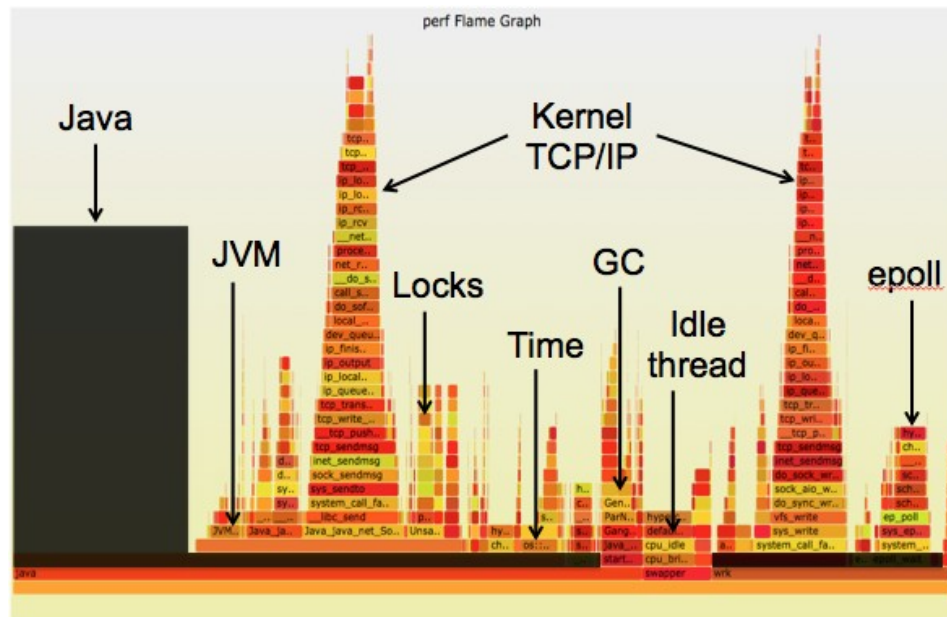
Zoomed



2014: Java Profiling

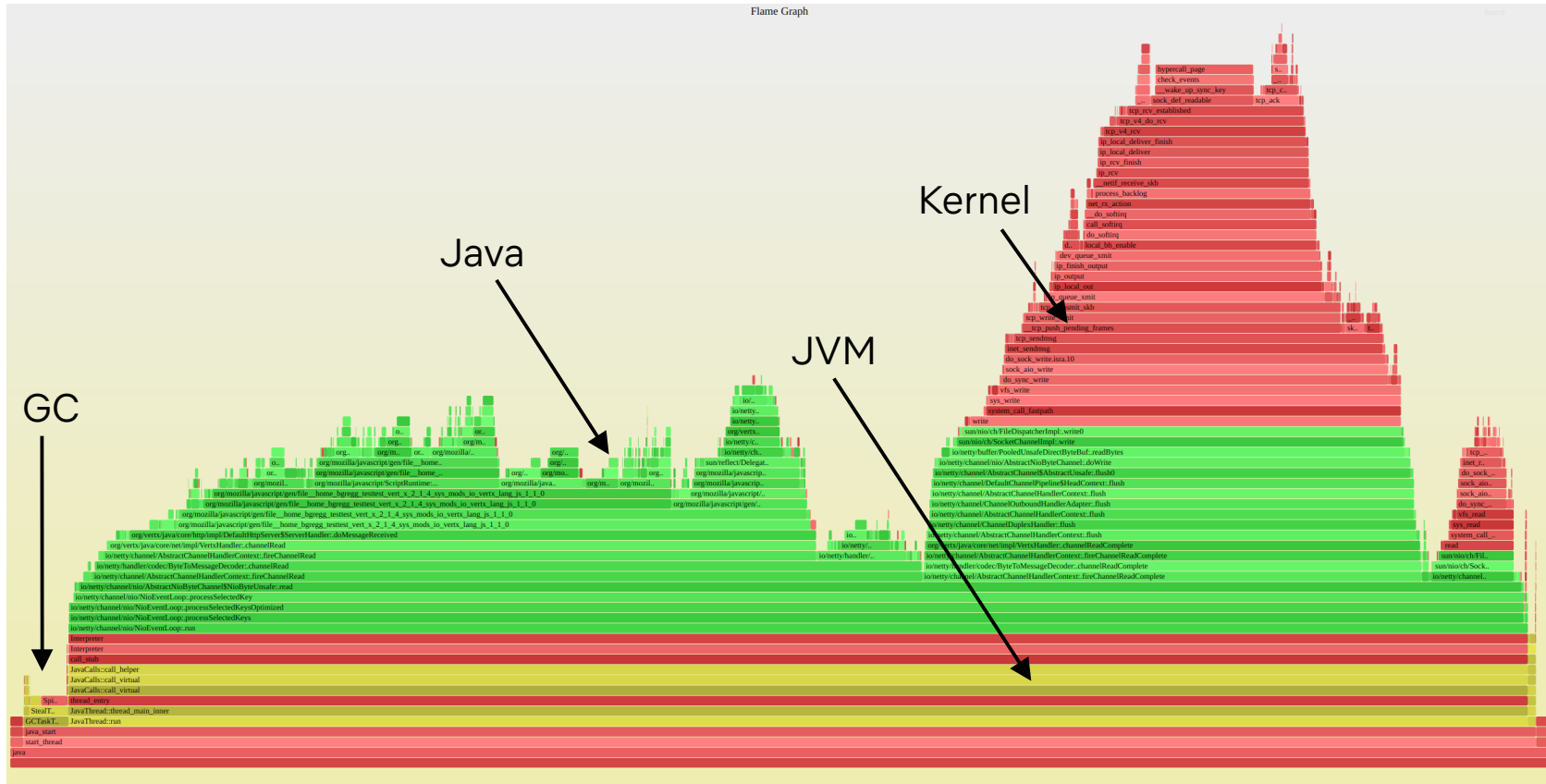


Java Profilers



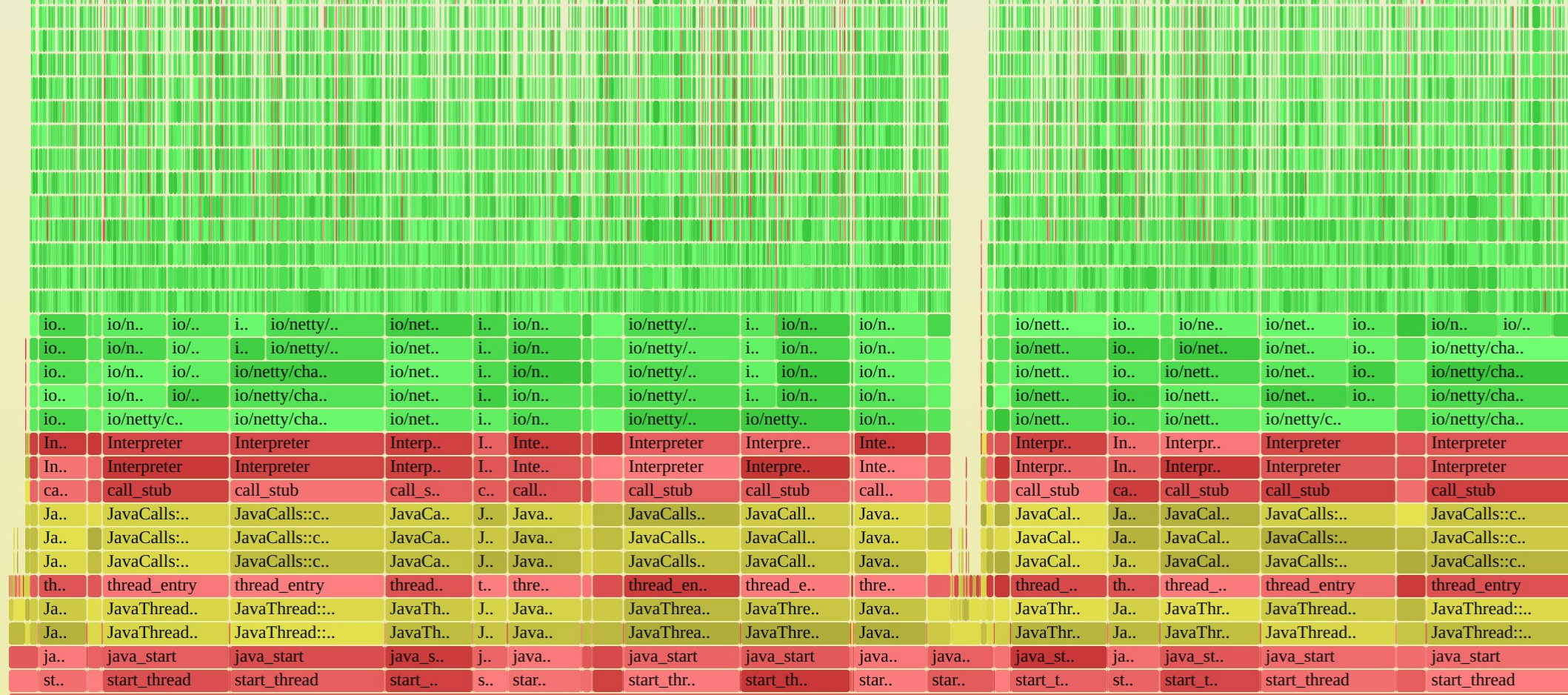
System Profilers

2018: Java Profiling

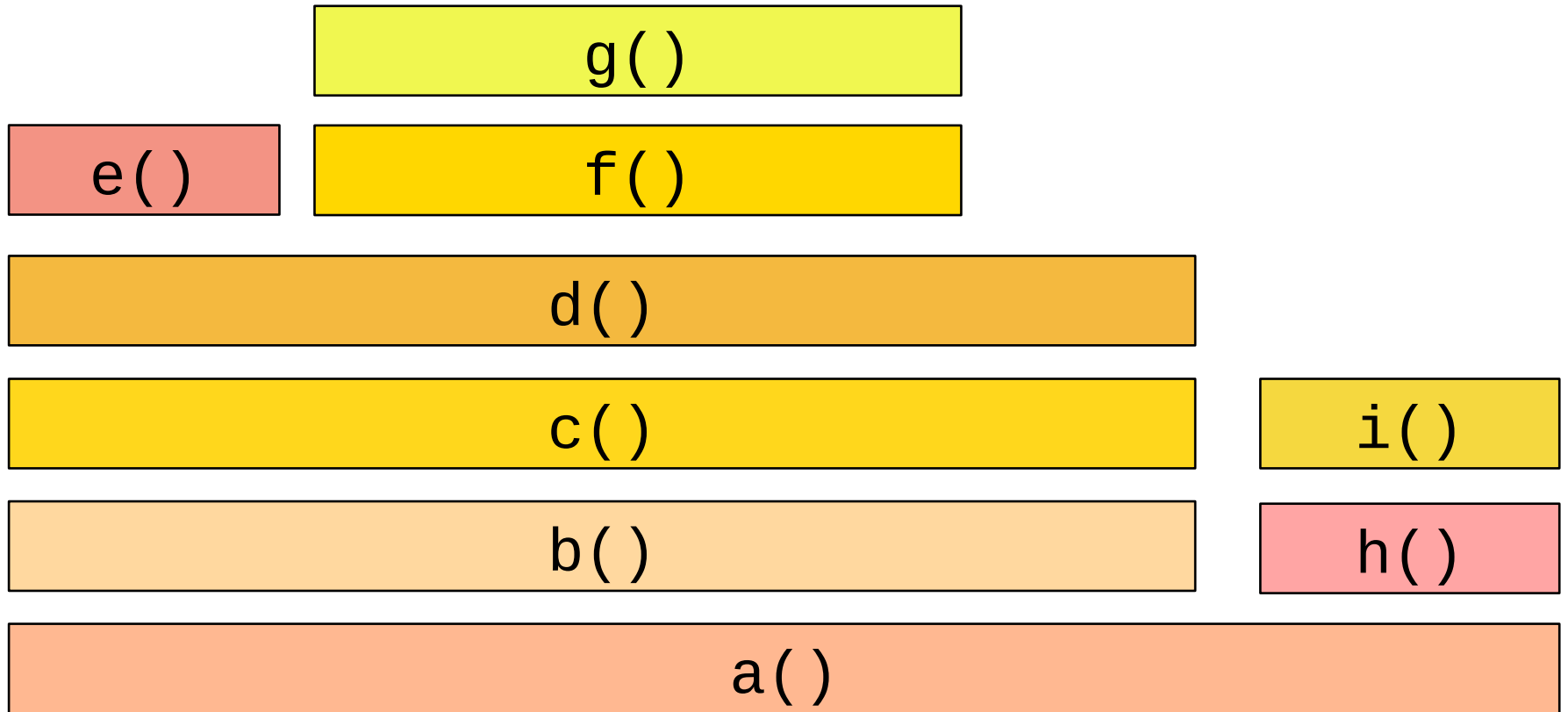


CPU Mixed-mode Flame Graph

CPU Flame Chart (same data)

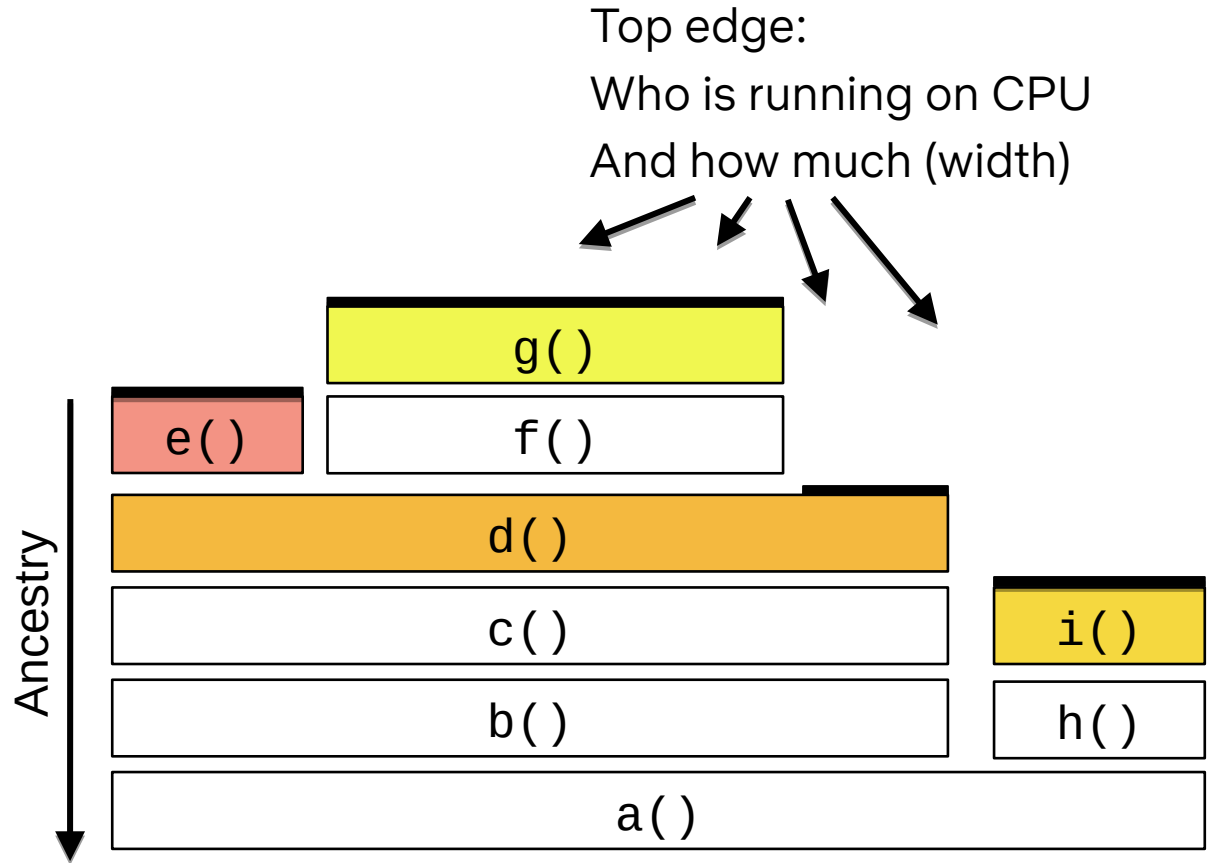


CPU Flame Graphs



CPU Flame Graphs

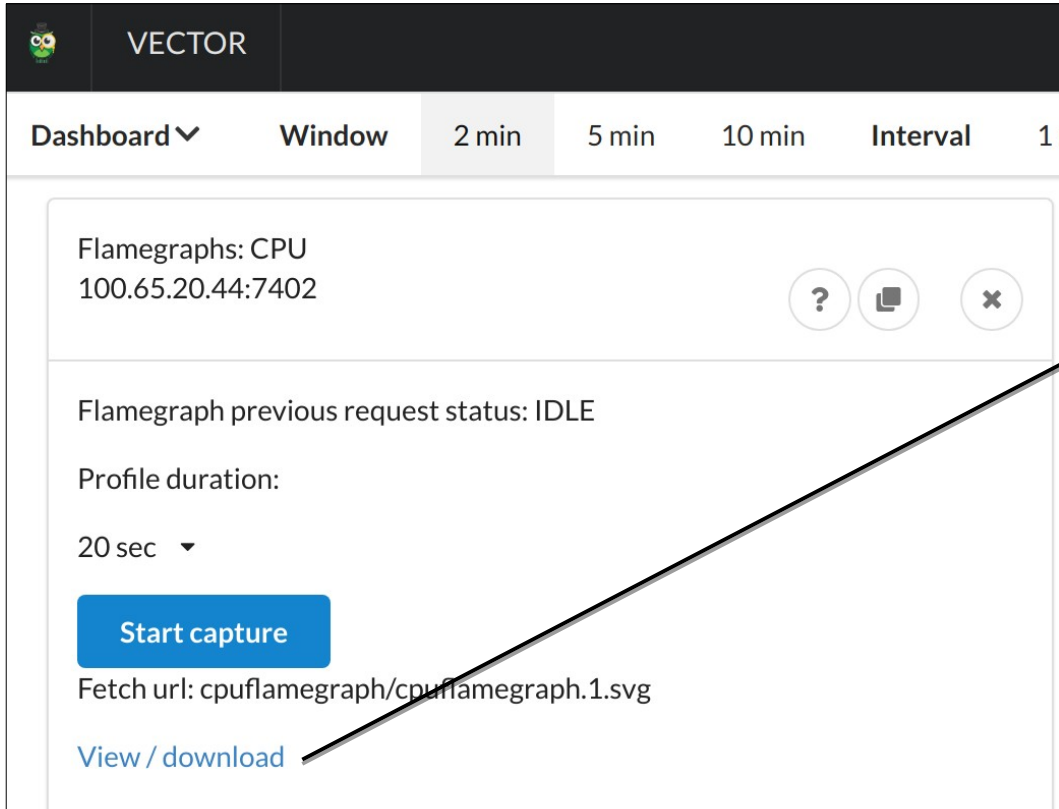
- Y-axis: **stack depth**
 - 0 at bottom
 - 0 at top == icicle graph
- X-axis: **alphabet**
 - Time == flame chart
- Color: random
 - Hues often used for language types
 - Can be a dimension eg, CPI



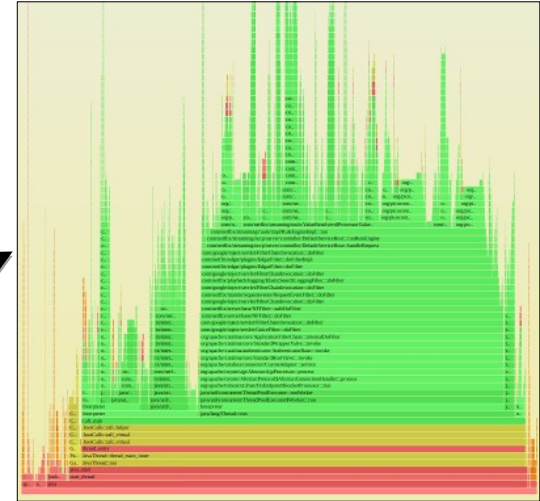
Application Profiling

- 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

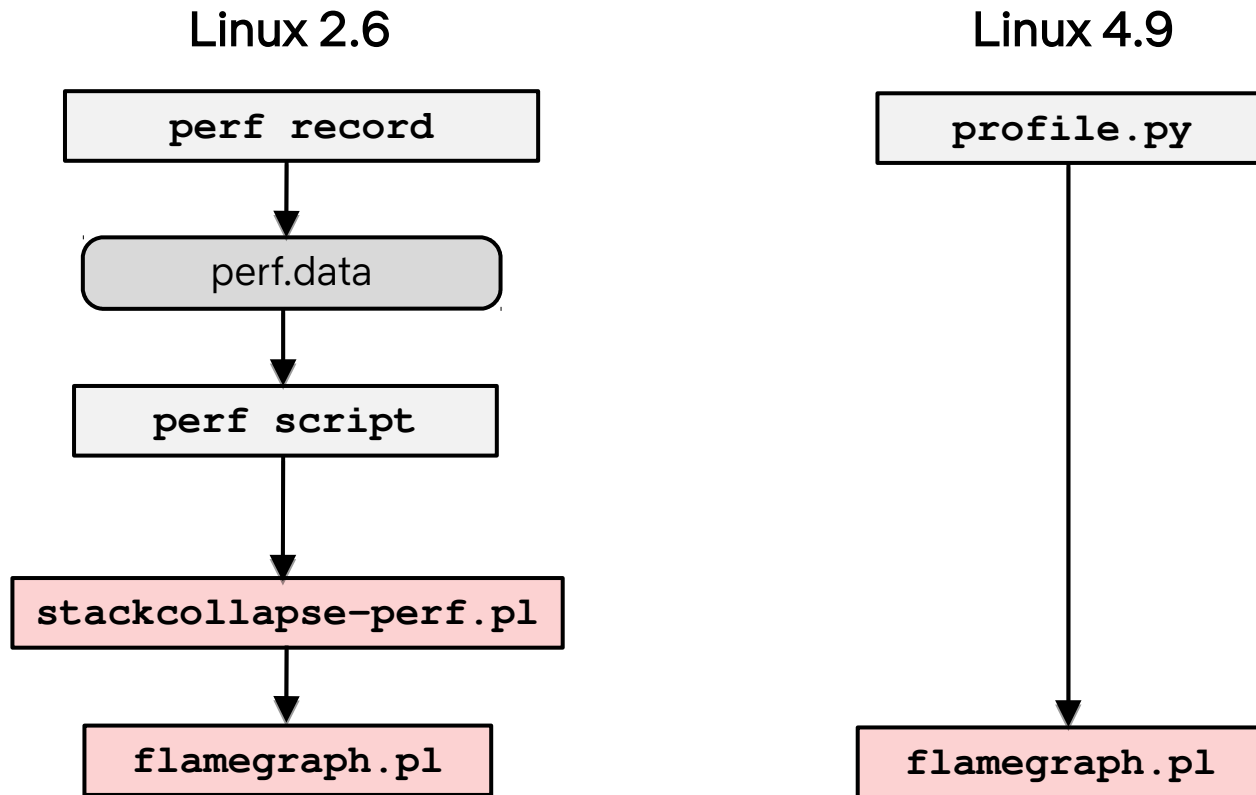
Vector: Push-button Flame Graphs



The screenshot shows the Vector web interface. At the top left is the Vector logo and the name 'VECTOR'. Below that is a navigation bar with 'Dashboard' (dropdown), 'Window', and a time selection menu with options '2 min', '5 min', '10 min', and 'Interval'. The main content area displays 'Flamegraphs: CPU' and the IP address '100.65.20.44:7402'. There are three circular icons: a question mark, a copy icon, and a close icon. Below this, it says 'Flamegraph previous request status: IDLE'. The 'Profile duration:' is set to '20 sec'. A blue 'Start capture' button is present. At the bottom, the 'Fetch url:' is 'cpuflamegraph/cpuflamegraph.1.svg' and there is a 'View / download' link. An arrow points from the 'View / download' link to the flamegraph image on the right.



Future: eBPF-based Profiling



3. Tracing



Core Linux Tracers



Ftrace 2.6.27+ **Tracing views**

Plus other kernel tech:
kprobes, uprobes



perf 2.6.31+ **Official profiler & tracer**

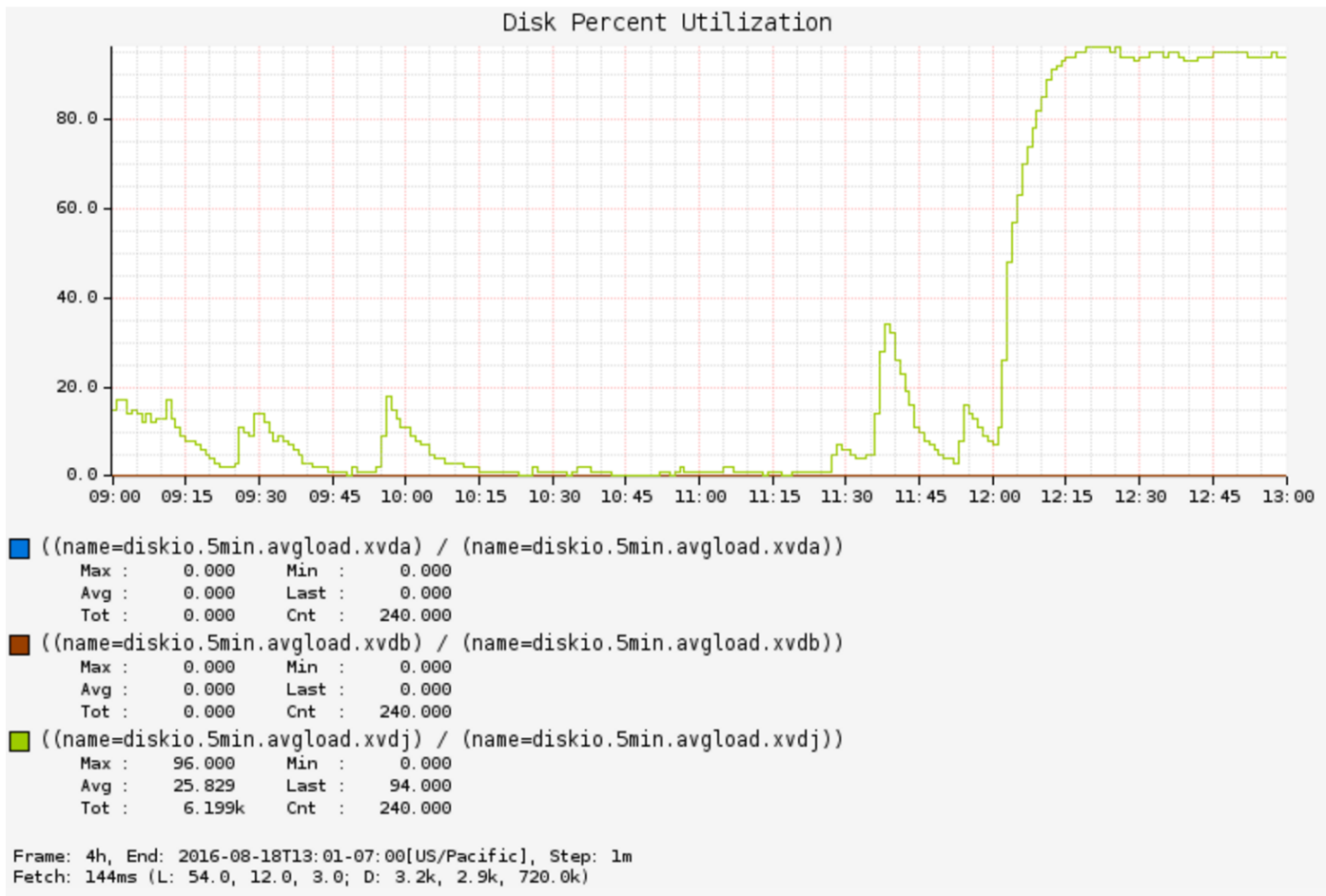


eBPF 4.9+ **Programmatic engine**

—
bcc - **Complex tools**

—
bpfttrace - **Short scripts**

Experience: Disk %Busy



```
# 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.
```

<code>>=(ms)</code>	<code>..</code>	<code><(ms)</code>	<code>: I/O</code>	<code> Distribution</code>
0	->	1	: 421	#####
1	->	2	: 95	#####
2	->	4	: 48	#####
4	->	8	: 108	#####
8	->	16	: 363	#####
16	->	32	: 66	#####
32	->	64	: 3	#
64	->	128	: 7	#



^C

```
# /apps/perf-tools/bin/iosnoop
```

```
Tracing block I/O. Ctrl-C to end.
```

COMM	PID	TYPE	DEV	BLOCK	BYTES	LATms
java	30603	RM	202,144	1670768496	8192	0.28
cat	6587	R	202,0	1727096	4096	10.07
cat	6587	R	202,0	1727120	8192	10.21
cat	6587	R	202,0	1727152	8192	10.43
java	30603	RM	202,144	620864512	4096	7.69
java	30603	RM	202,144	584767616	8192	16.12
java	30603	RM	202,144	601721984	8192	9.28
java	30603	RM	202,144	603721568	8192	9.06
java	30603	RM	202,144	61067936	8192	0.97
java	30603	RM	202,144	1678557024	8192	0.34
java	30603	RM	202,144	55299456	8192	0.61
java	30603	RM	202,144	1625084928	4096	12.00
java	30603	RM	202,144	618895408	8192	16.99
java	30603	RM	202,144	581318480	8192	13.39
java	30603	RM	202,144	1167348016	8192	9.92
java	30603	RM	202,144	51561280	8192	22.17
[...]						



```
# perf record -e block:block_rq_issue --filter rwbs ~ "*M*" -g -a
```

```
# perf report -n -stdio
```

```
[...]
```

```
# Overhead      Samples      Command      Shared Object      Symbol
```

```
# .....      .....      .....      .....      .....
```

```
# 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_iorequest
```

```
---88.84%--- _xfs_buf_read  
             xfs_buf_read_map
```

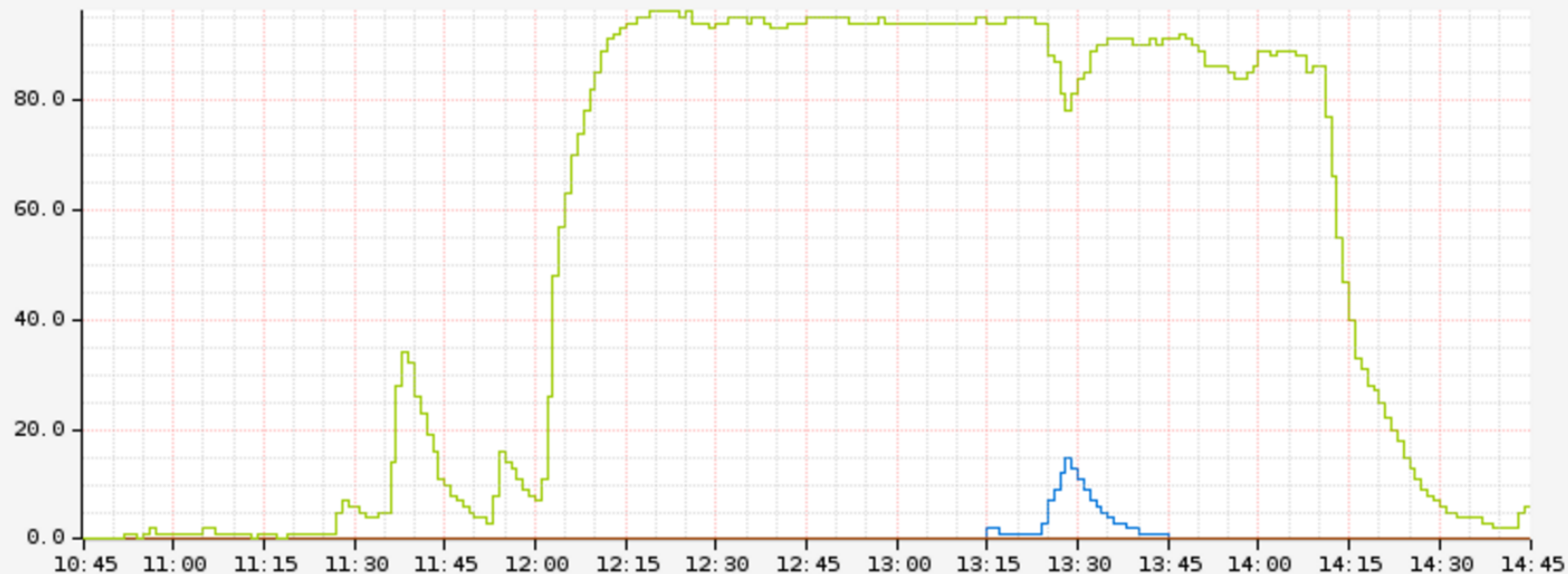
```
             ---87.89%--- xfs_trans_read_buf_map
```

```
             ---97.96%--- xfs_imap_to_bp  
                           xfs_iread  
                           xfs_iget  
                           xfs_lookup  
                           xfs_vn_lookup  
                           lookup_real  
                           _lookup_hash  
                           lookup_slow  
                           path_lookupat  
                           filename_lookup  
                           user_path_at_empty  
                           user_path_at  
                           vfs_fstatat
```

```
             ---99.48%--- SYSC_newlstat  
                           sys_newlstat  
                           system_call_fastpath  
                           lxstat64
```

```
             Lsun/nio/fs/UnixNativeDispatcher;.lstat0  
             0x7f8f963c847c
```

Disk Percent Utilization



■ ((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

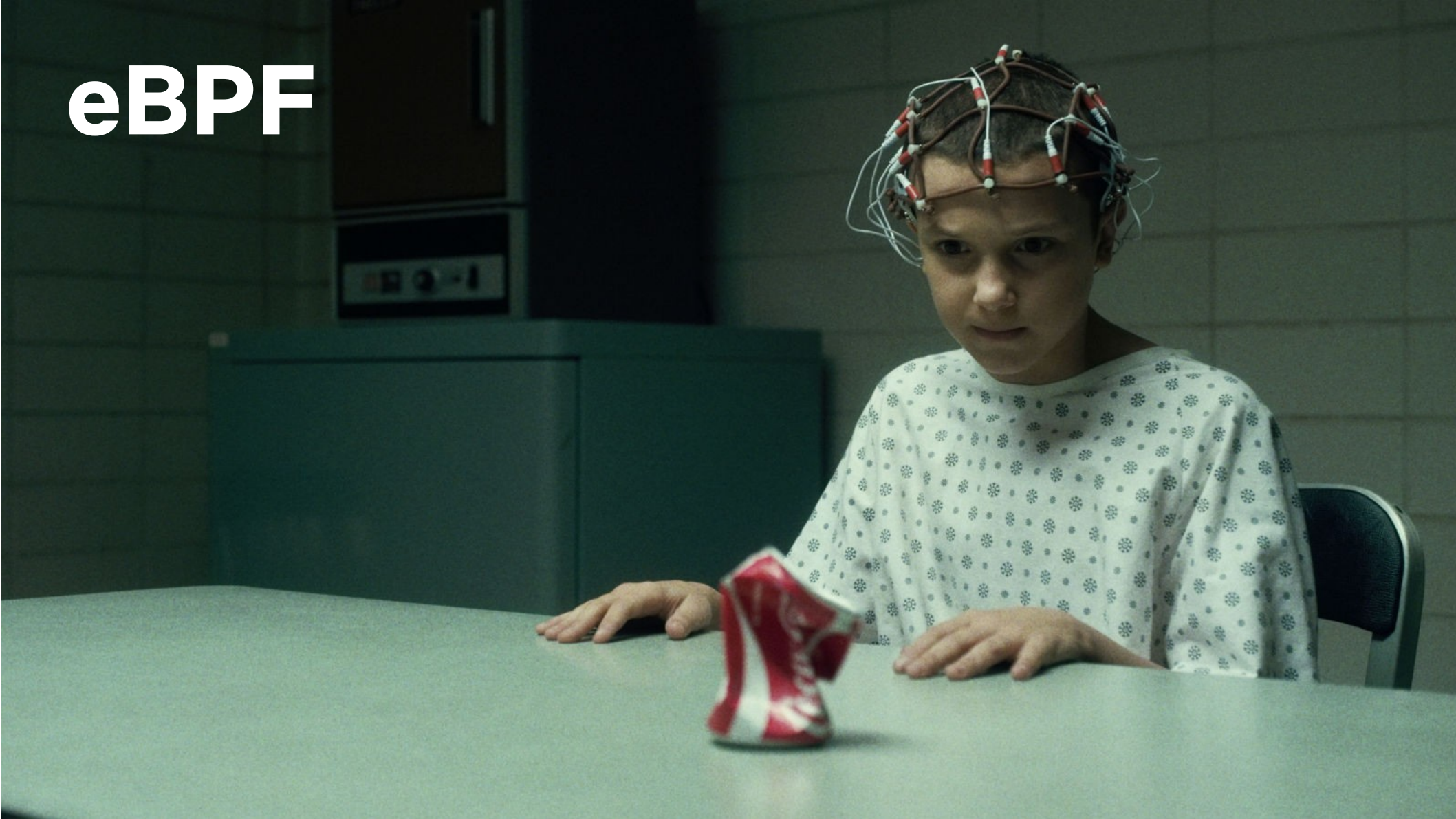
Frame: 4h, End: 2016-08-18T14:46:07:00[US/Pacific], Step: 1m
Fetch: 96ms (L: 30.0, 6.0, 3.0; D: 1.8k, 1.4k, 720.0k)

```
# /usr/share/bcc/tools/biosnoop
```

TIME(s)	COMM	PID	DISK	T	SECTOR	BYTES	LAT(ms)
0.000000000	tar	8519	xvda	R	110824	4096	6.50
0.004183000	tar	8519	xvda	R	111672	4096	4.08
0.016195000	tar	8519	xvda	R	4198424	4096	11.88
0.018716000	tar	8519	xvda	R	4201152	4096	2.43
0.019416000	tar	8519	xvda	R	4201160	4096	0.61
0.032645000	tar	8519	xvda	R	4207968	4096	13.16
0.033181000	tar	8519	xvda	R	4207976	4096	0.47
0.033524000	tar	8519	xvda	R	4208000	4096	0.27
0.033876000	tar	8519	xvda	R	4207992	4096	0.28
0.034840000	tar	8519	xvda	R	4208008	4096	0.89
0.035713000	tar	8519	xvda	R	4207984	4096	0.81
0.036165000	tar	8519	xvda	R	111720	4096	0.37
0.039969000	tar	8519	xvda	R	8427264	4096	3.69
0.051614000	tar	8519	xvda	R	8405640	4096	11.44
0.052310000	tar	8519	xvda	R	111696	4096	0.55
0.053044000	tar	8519	xvda	R	111712	4096	0.56
0.059583000	tar	8519	xvda	R	8411032	4096	6.40
0.068278000	tar	8519	xvda	R	4218672	4096	8.57
0.076717000	tar	8519	xvda	R	4218968	4096	8.33
0.077183000	tar	8519	xvda	R	4218984	4096	0.40
0.082188000	tar	8519	xvda	R	8393552	4096	4.94

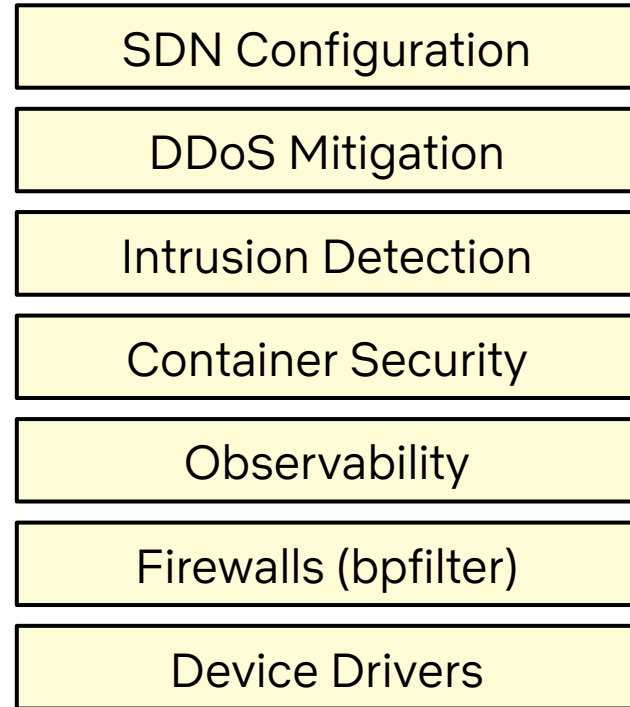
```
[...]
```


eBPF



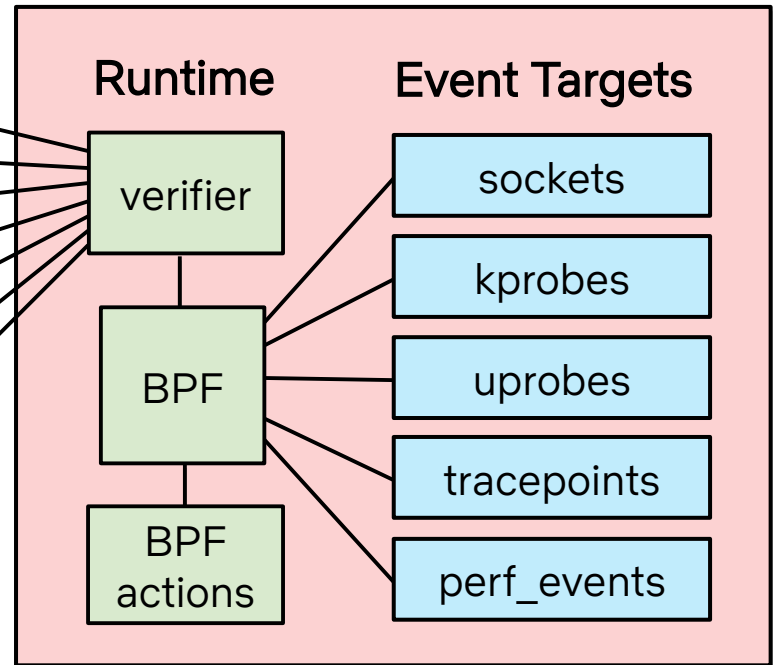
eBPF: extended Berkeley Packet Filter

User-Defined BPF Programs

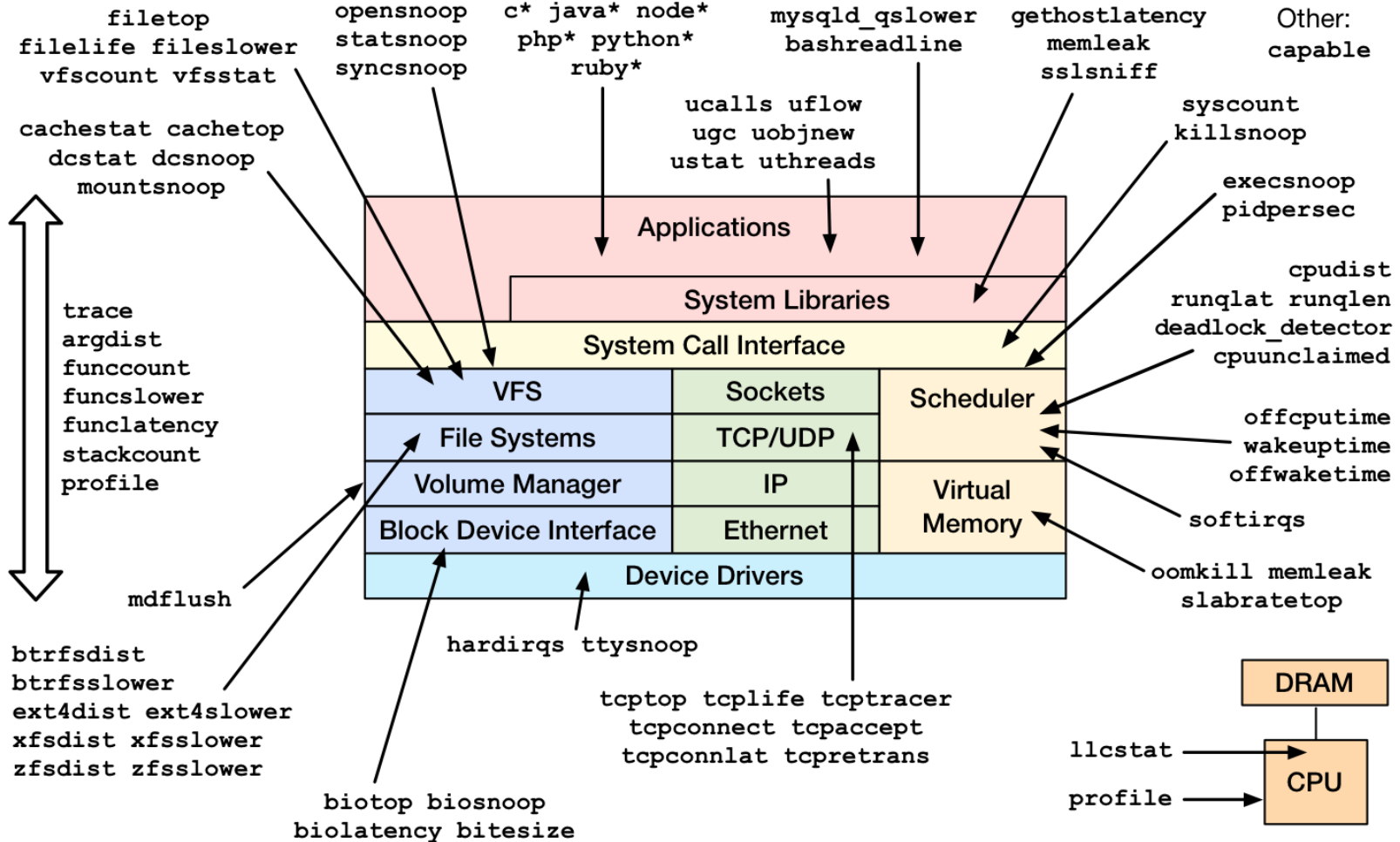


...

Kernel



Linux bcc/BPF Tracing Tools



bcc

```
# /usr/share/bcc/tools/tcplife
```

PID	COMM	LADDR	LPORT	RADDR	RPORT	TX_KB	RX_KB	MS
2509	java	100.82.34.63	8078	100.82.130.159	12410	0	0	5.44
2509	java	100.82.34.63	8078	100.82.78.215	55564	0	0	135.32
2509	java	100.82.34.63	60778	100.82.207.252	7001	0	13	15126.87
2509	java	100.82.34.63	38884	100.82.208.178	7001	0	0	15568.25
2509	java	127.0.0.1	4243	127.0.0.1	42166	0	0	0.61
12030	upload-mes	127.0.0.1	34020	127.0.0.1	8078	11	0	3.38
12030	upload-mes	127.0.0.1	21196	127.0.0.1	7101	0	0	12.61
3964	mesos-slav	127.0.0.1	7101	127.0.0.1	21196	0	0	12.64
12021	upload-sys	127.0.0.1	34022	127.0.0.1	8078	372	0	15.28
2509	java	127.0.0.1	8078	127.0.0.1	34022	0	372	15.31
2235	dockerd	100.82.34.63	13730	100.82.136.233	7002	0	4	18.50
2235	dockerd	100.82.34.63	34314	100.82.64.53	7002	0	8	56.73

```
[...]
```

bpftrace

```
# biolateness.bt
Attaching 3 probes...
Tracing block device I/O... Hit Ctrl-C to end.
^C
```

@usecs:

[256, 512)	2		
[512, 1K)	10	@	
[1K, 2K)	426	@@	
[2K, 4K)	230	@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@	
[4K, 8K)	9	@	
[8K, 16K)	128	@@@@@@@@@@@@@@@@	
[16K, 32K)	68	@@@@@@@@	
[32K, 64K)	0		
[64K, 128K)	0		
[128K, 256K)	10	@	

bpftrace: biolateness.bt

```
#!/usr/local/bin/bpftrace

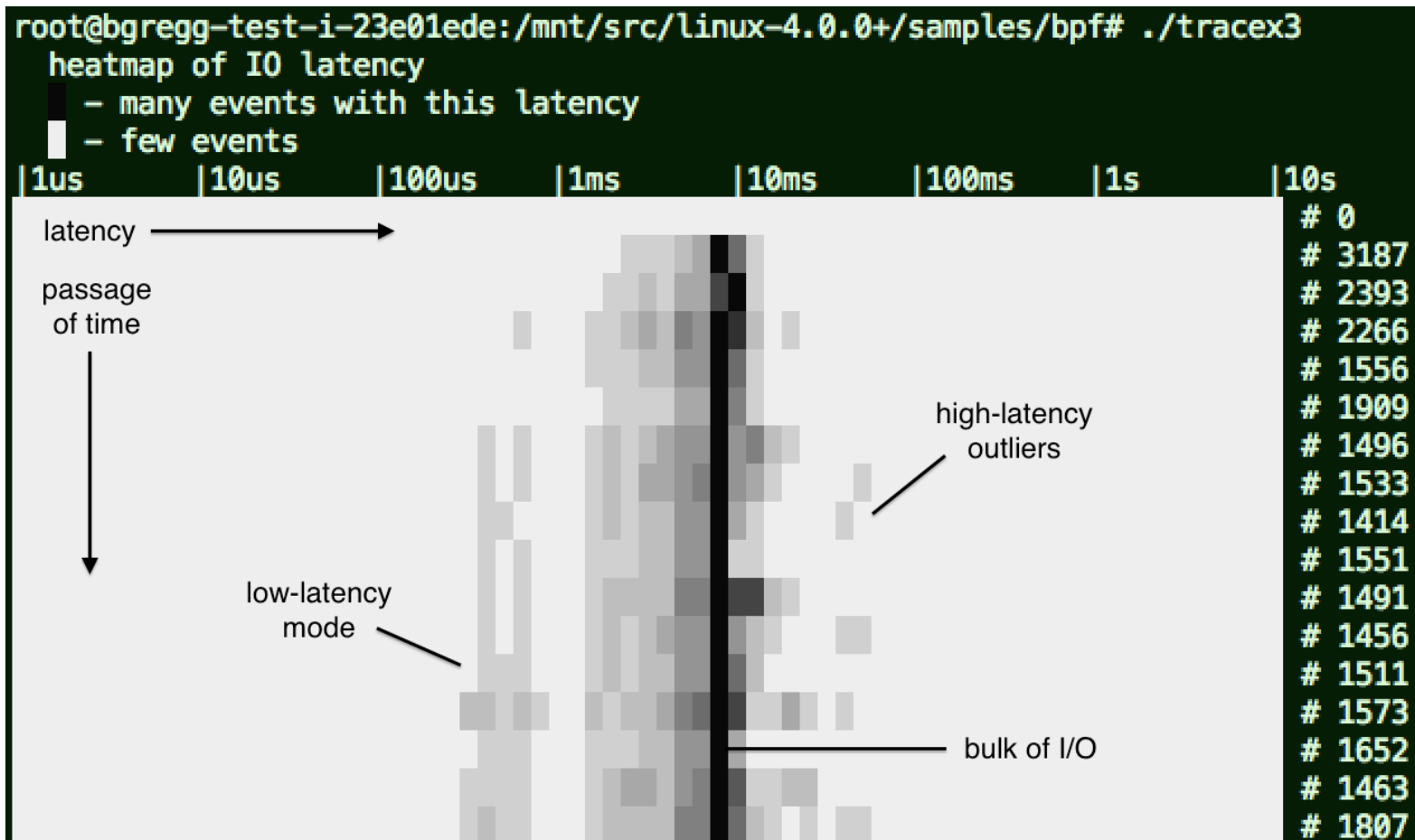
BEGIN
{
    printf("Tracing block device I/O... Hit Ctrl-C to end.\n");
}

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

kprobe:blk_account_io_completion
/@start[arg0]/

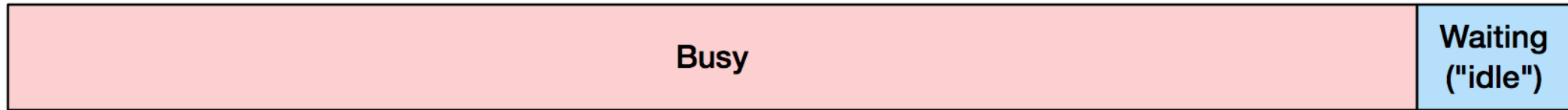
{
    @usecs = hist((nsecs - @start[arg0]) / 1000);
    delete(@start[arg0]);
}
```

Future: eBPF GUIs

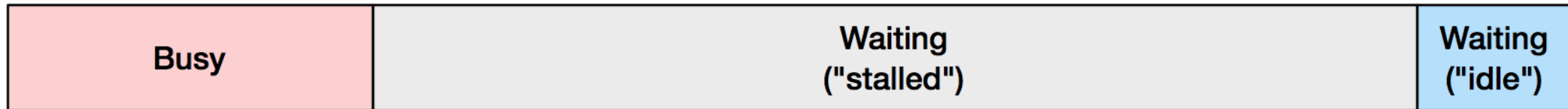


4. Processor Analysis

What “90% CPU Utilization” might suggest:



What it typically means on the Netflix cloud:

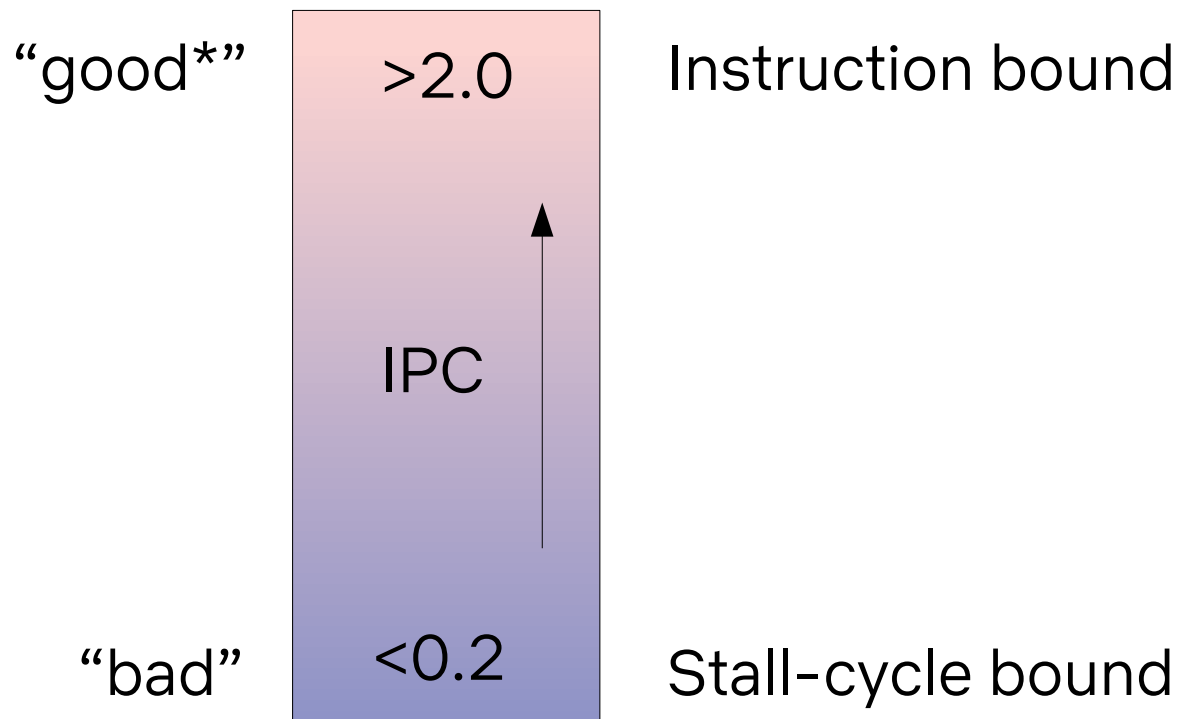


PMCs

- Performance Monitoring Counters help you analyze stalls
- Some instances (eg. Xen-based m4.16xl) have the architectural set:

Event Name	UMask	Event Select	Example Event Mask Mnemonic
UnHalted Core Cycles	00H	3CH	CPU_CLK_UNHALTED.THREAD_P
Instruction Retired	00H	C0H	INST_RETIRED.ANY_P
UnHalted Reference Cycles	01H	3CH	CPU_CLK_THREAD_UNHALTED.REF_XCLK
LLC Reference	4FH	2EH	LONGEST_LAT_CACHE.REFERENCE
LLC Misses	41H	2EH	LONGEST_LAT_CACHE.MISS
Branch Instruction Retired	00H	C4H	BR_INST_RETIRED.ALL_BRANCHES
Branch Misses Retired	00H	C5H	BR_MISP_RETIRED.ALL_BRANCHES

Instructions Per Cycle (IPC)



* probably; exception: spin locks

PMCs: EC2 Xen Hypervisor

```
# perf stat -a -- sleep 30
```

```
Performance counter stats for 'system wide':
```

1921101.773240	task-clock (msec)	#	64.034 CPUs utilized	(100.00%)
1,103,112	context-switches	#	0.574 K/sec	(100.00%)
189,173	cpu-migrations	#	0.098 K/sec	(100.00%)
4,044	page-faults	#	0.002 K/sec	
2,057,164,531,949	cycles	#	1.071 GHz	(75.00%)
<not supported>	stalled-cycles-frontend			
<not supported>	stalled-cycles-backend			
1,357,979,592,699	instructions	#	0.66 insns per cycle	(75.01%)
243,244,156,173	branches	#	126.617 M/sec	(74.99%)
4,391,259,112	branch-misses	#	1.81% of all branches	(75.00%)

```
30.001112466 seconds time elapsed
```

```
# ./pmcarch 1
```

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

```
[...]
```

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:

```
# 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...
```

TIME	C0_MCYC	C0_ACYC	UTIL	RATIO	MHz
23:39:07	1618910294	89419923	64%	5%	138
23:39:08	1774059258	97132588	70%	5%	136
23:39:09	2476365498	130869241	99%	5%	132

^C

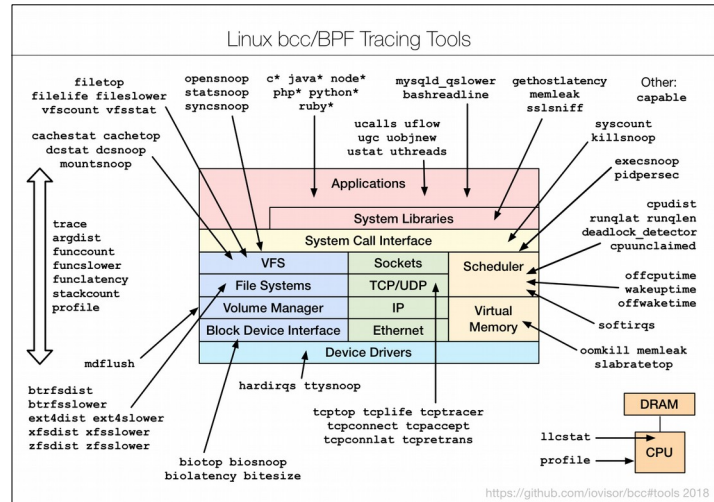
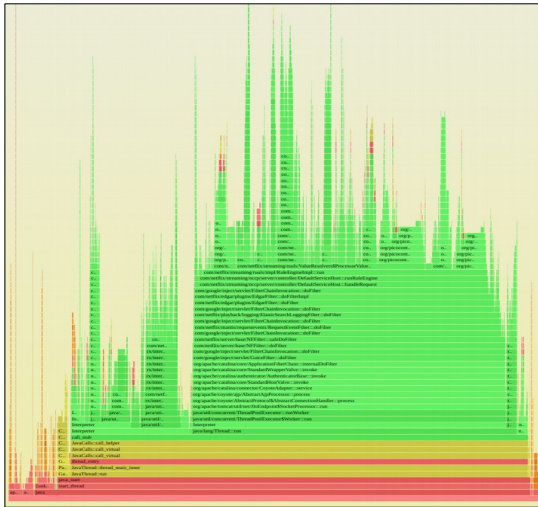
Summary

| Take-aways

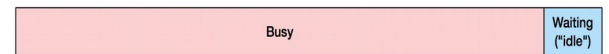
NETFLIX

Take Aways

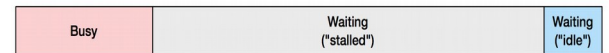
1. Get push-button **CPU flame graphs**: kernel & user
2. Check out **eBPF** perf tools: bcc, bpftrace
3. Measure **IPC** *as well as* CPU utilization using PMCs



90% CPU busy:



... really means:



Observability
Methodology
Velocity

Observability

Statistics, Flame Graphs, eBPF Tracing, Cloud PMCs

Methodology

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

Velocity

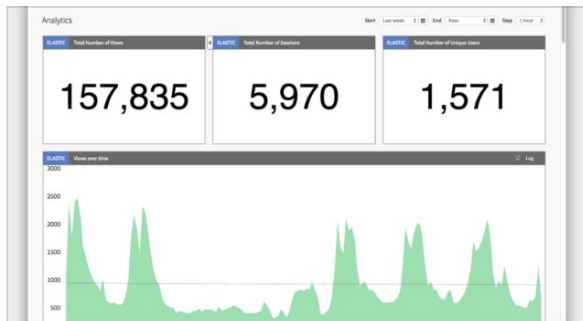
Self-service GUIs: Vector, FlameScope, ...

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>
- **Chaos:** <https://medium.com/netflix-techblog/chap-chaos-automation-platform-53e6d528371f> <https://principlesofchaos.org/>
- **Atlas:** <https://github.com/Netflix/Atlas>
- **Atlas:** <https://medium.com/netflix-techblog/introducing-atlas-netflixs-primary-telemetry-platform-bd31f4d8ed9a>
- **RED method:** <https://thenewstack.io/monitoring-microservices-red-method/>
- **USE method:** <https://queue.acm.org/detail.cfm?id=2413037>
- **Winston:** <https://medium.com/netflix-techblog/introducing-winston-event-driven-diagnostic-and-remediation-platform-46ce39aa81cc>
- **Lumen:** <https://medium.com/netflix-techblog/lumen-custom-self-service-dashboarding-for-netflix-8c56b541548c>
- **Flame graphs:** <http://www.brendangregg.com/flamegraphs.html>
- **Java flame graphs:** <https://medium.com/netflix-techblog/java-in-flames-e763b3d32166>
- **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

Netflix Tech Blog

🔒 A Medium Corporation [US] | <https://medium.com/netflix-techblog>



Lumen: Custom, Self-Service Dashboarding For Netflix

By Trent Willis



Netflix Technology Blog
Oct 18

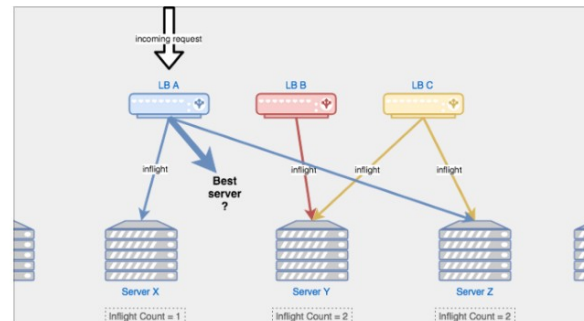


The Netflix Media Database (NMDB)

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



Netflix Technology Blog
Oct 16



Rethinking Netflix's Edge Load Balancing

The why's, how's and results from rethinking Netflix's edge load balancing



Netflix Technology Blog
Sep 29

Thank you.

Brendan Gregg
@brendangregg

NETFLIX