



Introduction to Optimizing and Measuring performance on AWS Graviton Processors

Arthur Petitpierre

Principal Graviton Specialist SA
Amazon Web Services

Part 1 – Know about the architecture

Graviton3 CPU enhancements



AWS Graviton2

4–8 wide Fetch

4 wide Decode

8 wide issue



AWS Graviton3

8 wide Fetch

5–8 wide Decode

15 wide issue & 2x larger instruction window



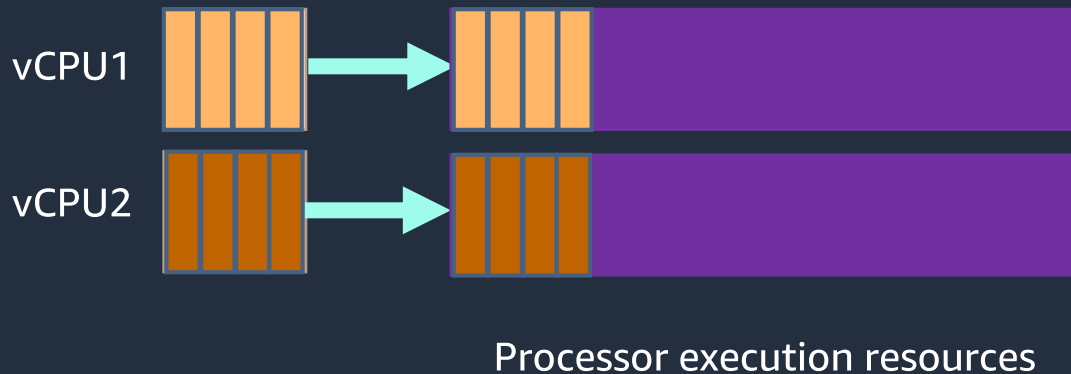
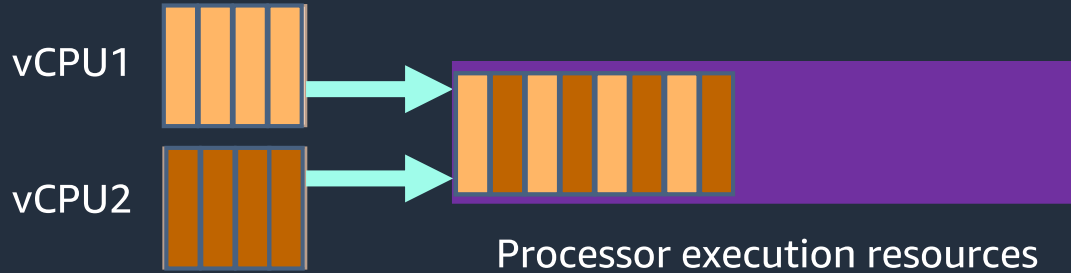
**bfloat16
256b
SVE**

**2x Mem ops
enhanced
prefetching**

**~2x
TLS**

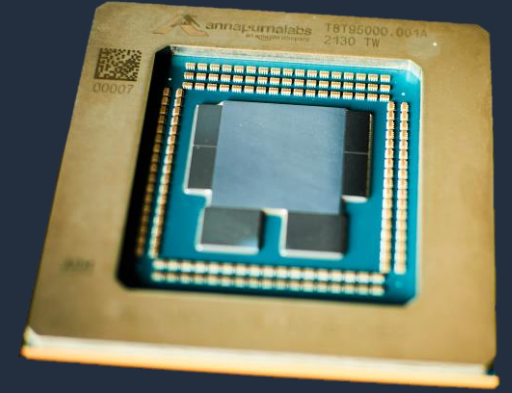
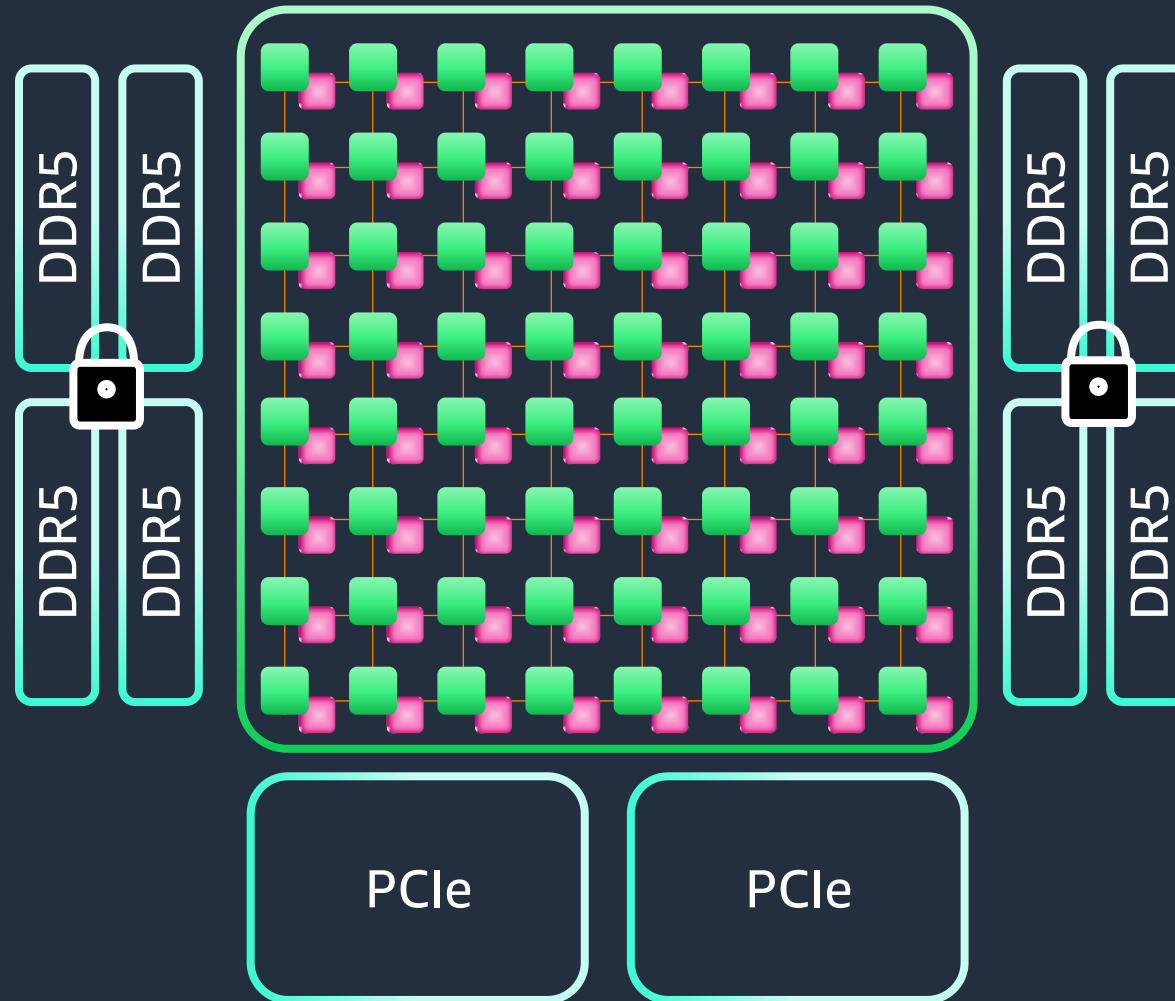


Graviton2/3 – No Simultaneous Multi Threading



Every vCPU is a physical core
No simultaneous multi
threading (SMT)

Graviton3 – Interconnect & system



Graviton features

Processor	Graviton2	Graviton3
Instances	M6g/M6gd, C6g/C6gd/C6gn, R6g/R6gd, T4g, X2gd, G5g, and Im4gn/Is4gen	C7g, M7g, R7g. In preview: C7gn
Core	Neoverse-N1	Neoverse-V1
cache coherent mesh interconnect	CMN-600	CMN-700
Architecture revision	ARMv8.2-a	ARMv8.4-a
Additional features	fp16, rcpc, dotprod, crypto	sve, rng, bf16, int8, crypto
Recommended -mcpu flag	neoverse-n1	neoverse-512tvb
RNG Instructions	Yes	Yes
SIMD instructions	2x Neon 128bit vectors	4x Neon 128bit vectors / 2x SVE 256bit vectors
LSE (atomic mem operations)	yes	yes
Pointer Authentication	no	yes
Cores	64	64
L1 cache (per core)	64KB inst / 64KB data	64KB inst / 64KB data
L2 cache (per core)	1MB	1MB
LLC (shared)	32MB	32MB
DRAM	8x DDR4	8x DDR5
DDR Encryption	yes	yes

Graviton cache hierarchy and processor features

```
$ lscpu
Architecture:          aarch64
  CPU op-mode(s):      32-bit, 64-bit
  Byte Order:           Little Endian
CPU(s):                1
  On-line CPU(s) list: 0
Vendor ID:             ARM
  Model:               1
  Thread(s) per core:  1
  Core(s) per socket:  1
  Socket(s):           1
  Stepping:            r1p1
  BogomIPS:            2100.00
  Flags:               fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrdm jscvt fcma lrcpc dcpop sha3 sm3
sm4 asimddp sha512 sve asimdfhm dit uscat ilrcpc flagm ssbs paca pacg dcpodp sve
                      i8mm svebf16 i8mm bf16 dgh rng
Caches (sum of all):
  L1d:                64 KiB (1 instance)
  L1i:                64 KiB (1 instance)
  L2:                 1 MiB (1 instance)
  L3:                 32 MiB (1 instance)
NUMA:
  NUMA node(s):       1
  NUMA node0 CPU(s):  0
```

Part 2 – Performance best practices

Graviton2 was announced on ?

December the 3rd, 2019

Modernize your software stack

Graviton2 was announced on December the 3rd, 2019

But:

Software	Release date
RHEL 7	June 10 th , 2014
OpenJDK 8	March 18 th , 2014
GCC 5.4	June 3 rd , 2016
Python 2.7	July 4 th , 2010

Sorry, those versions aren't and can't be optimized for Graviton processors

AWS Graviton getting started guide on Github

<https://github.com/aws/aws-graviton-getting-started>

- This guide has been assembled by our Graviton team and is designed to help customers transition and optimize their applications.
- It covers various languages and libraries, and includes tips and tricks for each.
- In general, using latest versions of operating systems, compilers, and language runtimes will provide access to latest Arm64 improvements and optimizations.

Java version(s) recommendation for Graviton

Minimum version: 8

Recommended version: at least 11

However: **the more recent the better!**

And Amazon Corretto is even better ! <https://aws.amazon.com/corretto/>

<https://github.com/aws/aws-graviton-getting-started/blob/main/java.md>

C/C++ on Graviton

CPU	Flag	GCC version	LLVM version
Graviton2	-mcpu=neoverse-n1*	GCC-9^	Clang/LLVM 10+
Graviton3(E)	-mcpu=neoverse-512tvb%	GCC 11+	Clang/LLVM 14+

<https://github.com/aws/aws-graviton-getting-started/blob/main/c-c%2B%2B.md>

Part 3 – Performance analysis

Micro-benchmarking / Passive benchmarking

Why it won't help you ... At least not initially.

Passive benchmarks: collection of benchmark data without analysis.

Active benchmarking: analyzing performance why benchmarks are running

“How do Graviton instances compare to x86 based instances on Java?”

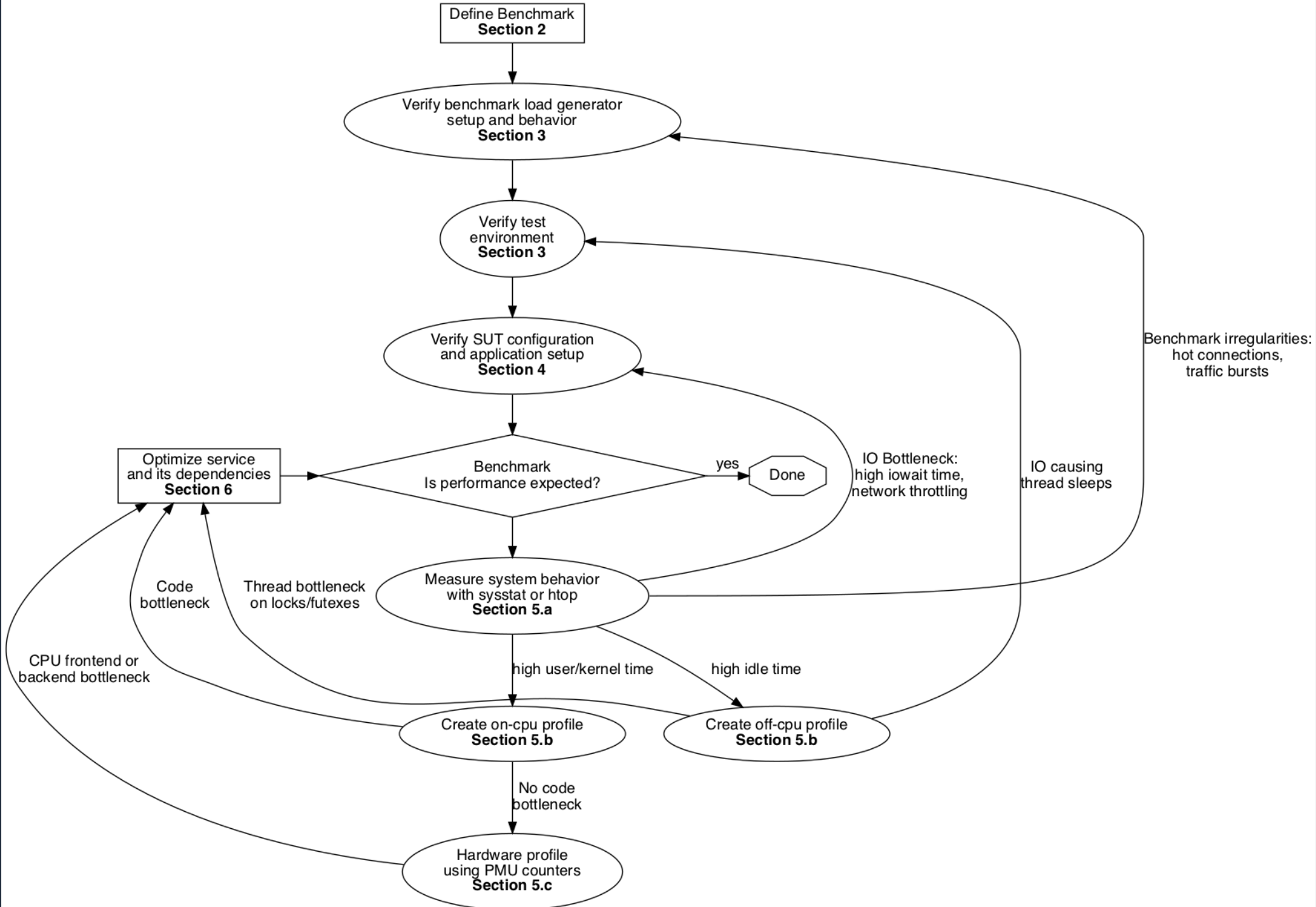
“How does a Graviton instance’s request throughput compare to current instances on my Java application at a P99 of 10ms for a mix of 60% GETS and 40% PUTS on Ubuntu 20.04LTS?”

Graviton Performance Runbook

https://github.com/aws/aws-graviton-getting-started/blob/main/perfrunbook/graviton_perfrunbook.md

Written by the AWS Graviton Performance team

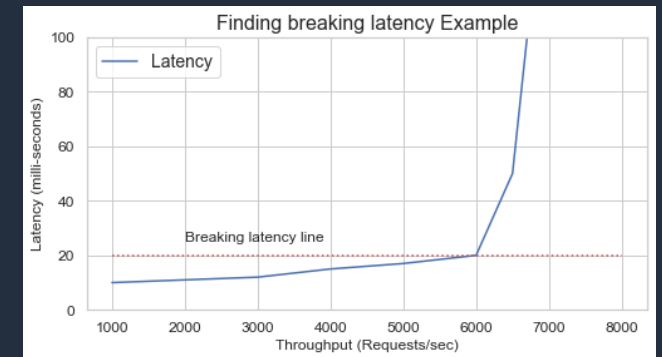
Covers **tools and best practices** when doing **performance analysis on Graviton based instances.**



Step 1 – Define your benchmark

Benchmark definition:

- System under test (SUT)
- How to drive the load
- Load point
 - Maximum Throughput
 - Throughput at breaking latency



Step 2 – Configure Load-generator and SUT

Load-generator:

- Ensure that it can drive enough traffic
- Ensure that it's far enough from its own limits

System under test:

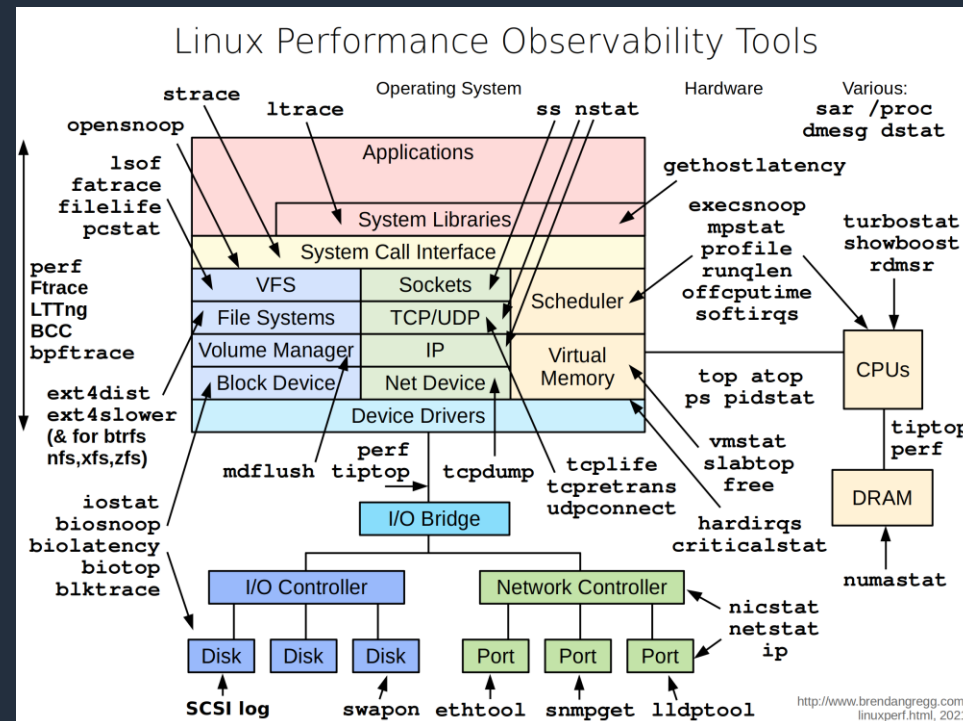
- Ensure you are comparing the same setups
- Check and fix errors and limits before testing performance

Step 3 – Analyzing performance (1)

1. What part of the system is slow or saturated?

For every resource, check utilization, saturation, and errors (USE method):

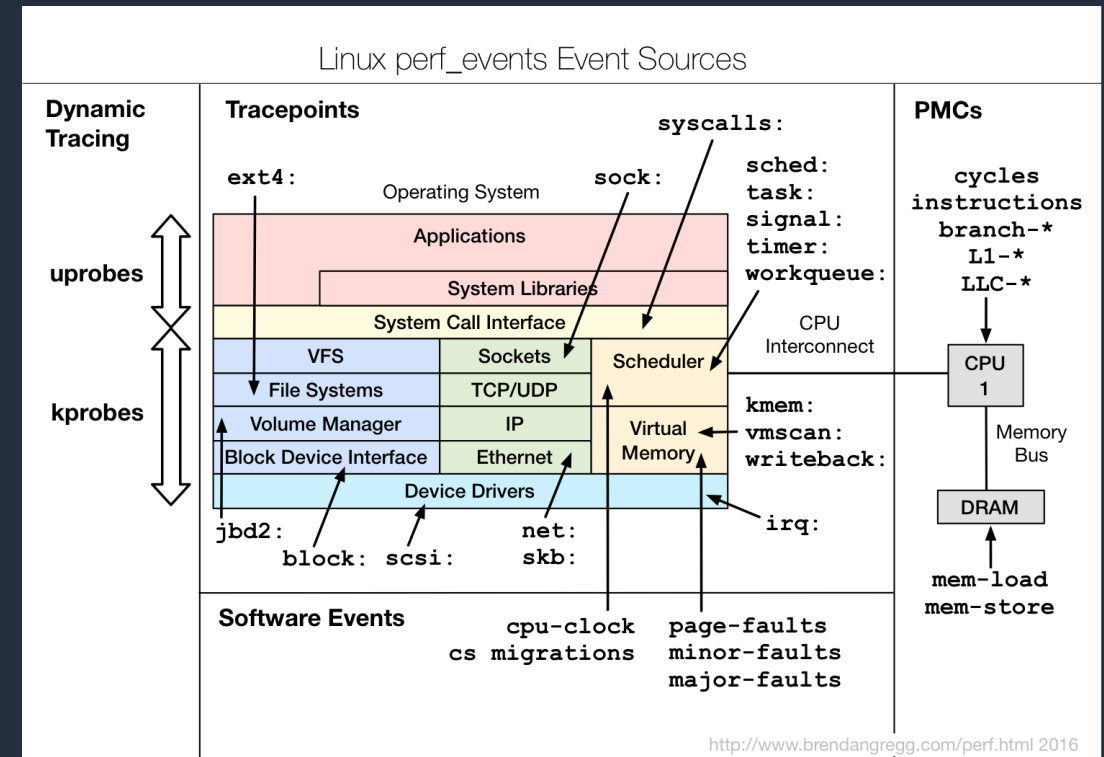
<https://www.brendangregg.com/USEmethod/use-linux.html>



Linux Perf

perf_events is an event-oriented observability tool, which can help you solve advanced performance and troubleshooting functions.

<https://perf.wiki.kernel.org/>

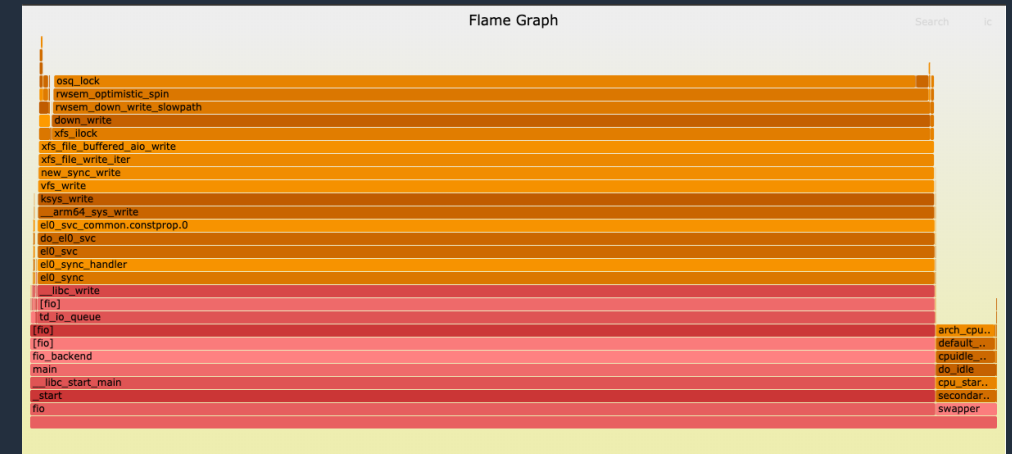


Step 3 – Analyzing performance (2)

2. What part of the **Code** is slow ?

- On-cpu profiling / FlameGraph

- Off-cpu Profiling / FlameGraph



https://github.com/aws/aws-graviton-getting-started/blob/main/perfrunbook/debug_code_perf.md

Step 3 – Analyzing performance (3)

3. What part of the **Hardware** is slow ?

- Sometimes applications will show as uniformly slightly slower on on-cpu profile, with no obvious hot-spot.
- Requires to measure counters in the CPU to understand the bottlenecks

https://github.com/aws/aws-graviton-getting-started/blob/main/perfrunbook/debug_hw_perf.md

Graviton Performance Runbook and Feedback

https://github.com/aws/aws-graviton-getting-started/blob/main/perfrunbook/graviton_perfrunbook.md

If you have used this guide and still can't find the root cause of your issues, what can you do next?

Please contact us at ec2-arm-dev-feedback@amazon.com or talk with your AWS account team representative to get additional help.



Thank you!

Arthur Petitpierre
arthurpt@amazon.com