Linux Performance Tools

A Rapid Overview of tools

Agenda

- Methodologies
- Tools
- Tool Types:
  - Observability
  - Benchmarking
  - Tuning
  - Static
- Profiling
- Tracing

Methodologies

- There are dozens of performance tools for Linux
  - Packages: sysstat, procps, coreutils, …
  - Commercial products
- Methodologies can provide guidance for choosing and using tools effectively
  - A starting point, a process, and an ending point

Street Light Anti Method

- Pick observability tools that are:
  - Familiar
  - Found on the Internet
  - Found at random
- Run tools
- Look for obvious issues

Drunken Man Anti-Method

- Tune Things at Random until problem goes away

Blame Others Anti-Method

- Find a system or environment component you are not responsible for
- Hypothesize that the issue is with that component
- Redirect the issue to the responsible team
- When proven wrong, go to 1

Actual Methodologies

- Problem Statement Method
- Workload Characterization Method
- USE Method
- Off-CPU Analysis
- CPU Profile Method
- RTFM Method
- Active Benchmarking
- Static Performance Tuning
- …

Methodologies – Problem Statement

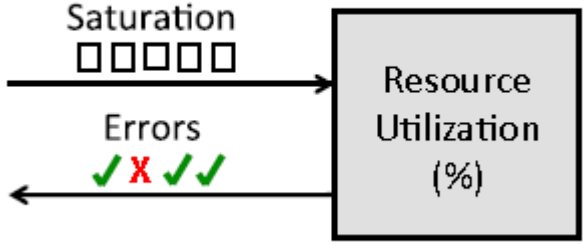
- What makes you **think** there is a performance problem?
- Has this system **ever** performed well?
- What has **changed** recently? (Software? Hardware? Load?)
- Can the performance degradation be expressed in terms of **latency** or run time?
- Does the problem affect **other** people or applications (or is it just you)?
- What is the **environment**? Software, hardware, instance types? Versions? Configuration?

Methodologies – Workload Characterization

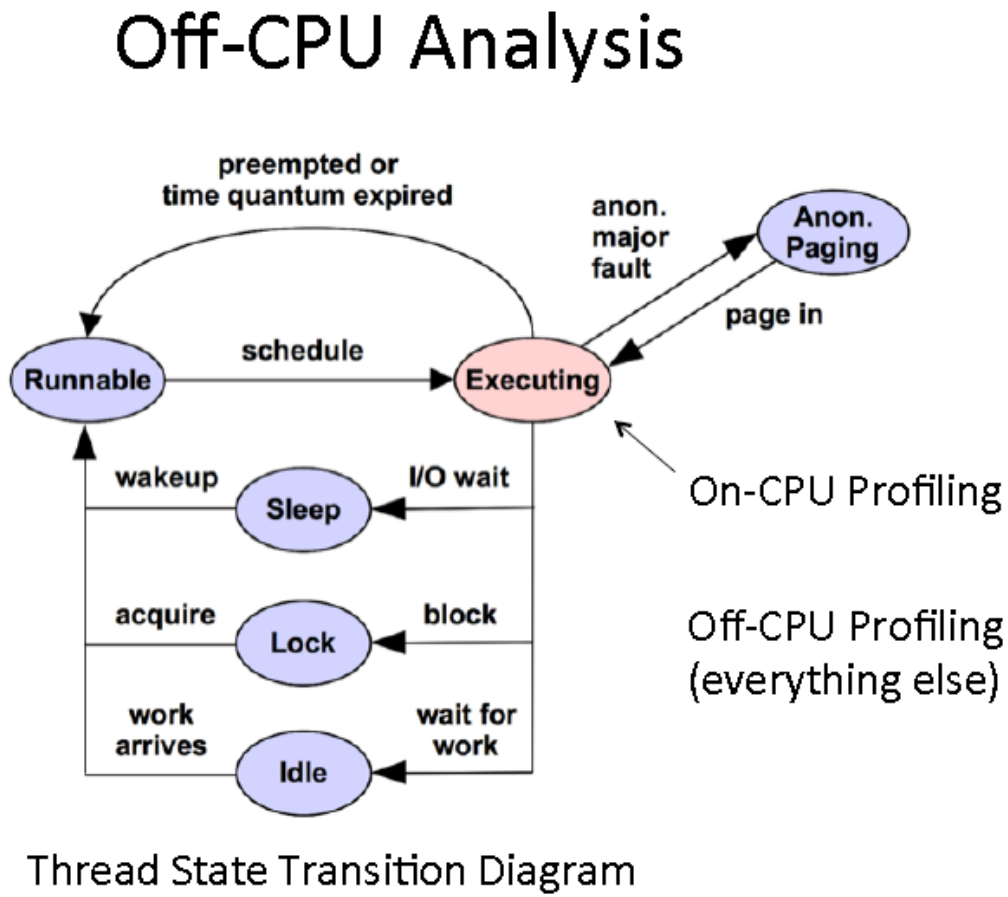
- **Who** is causing the load? PID, UID, IP addr, ...
- **Why** is the load called? code path, stack trace
- **What** is the load? IOPS, tput, type, r/w
- **How** is the load changing over time?

Methodologies - USE

- For every resource, check:
  - 1. **Utilization**
  - 2. **Saturation**
  - 3. **Errors**
- Definitions:
  - Utilization: busy time
  - Saturation: queue length or queued time
  - Errors: easy to interpret (objective)
- Helps if you have a functional (block) diagram of your system / software / environment, showing all resources
- Start with the questions, then find the tools



Off-CPU Analysis



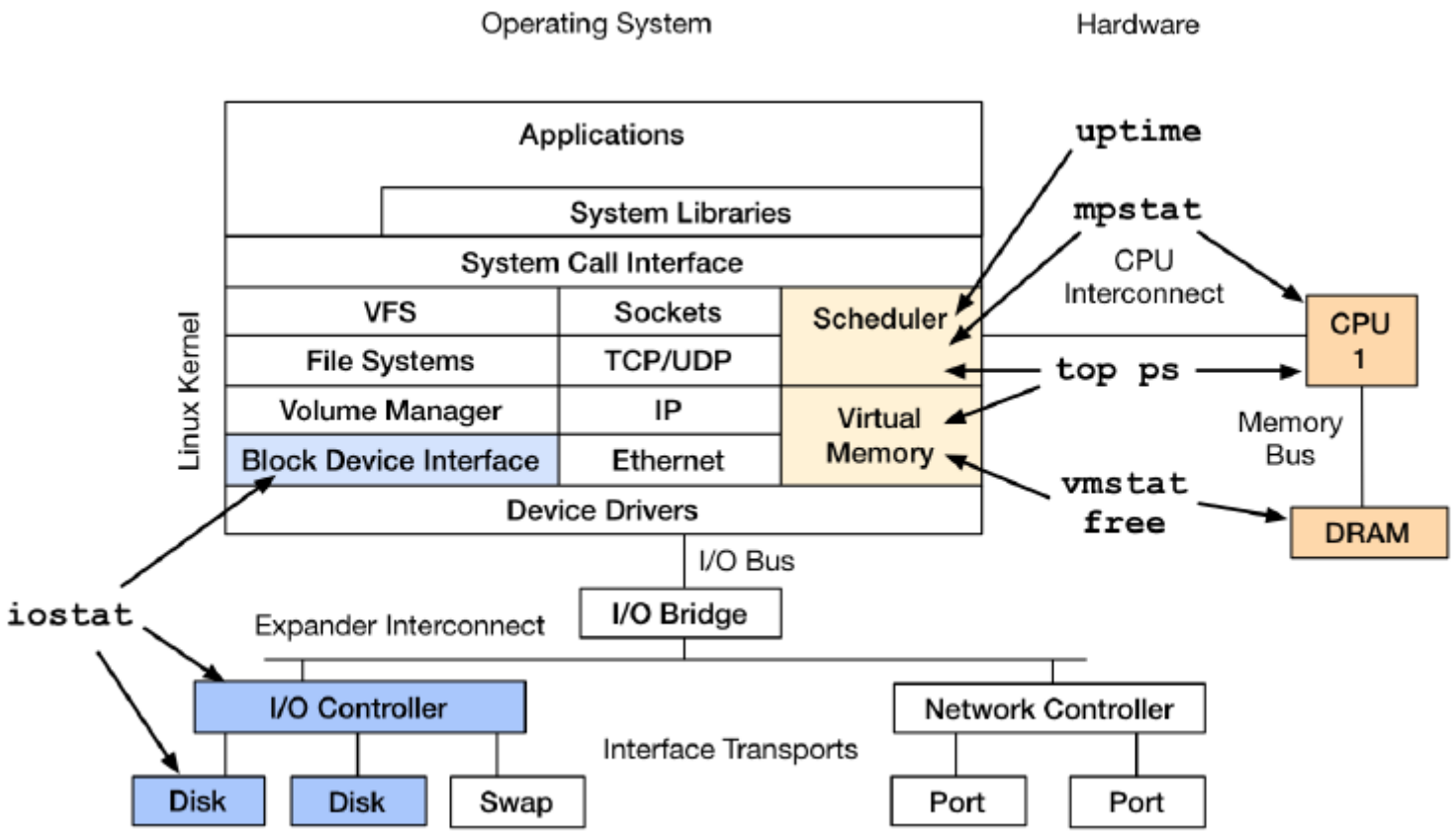
CPU Profile Method

- Take a CPU profile
- Understand all software in profile > 1%
- Discovers a wide range of performance issues by their CPU usage
- Narrows software to study

Tool Types

- **Observability:** Watch activity. Safe, usually, depending on resource overhead.
- **Benchmarking:** Load test. Caution: production tests can cause issues due to contention.
- **Tuning:** Change. Danger: changes could hurt performance, now or later with load.
- **Static:** Check configuration. Should be safe.

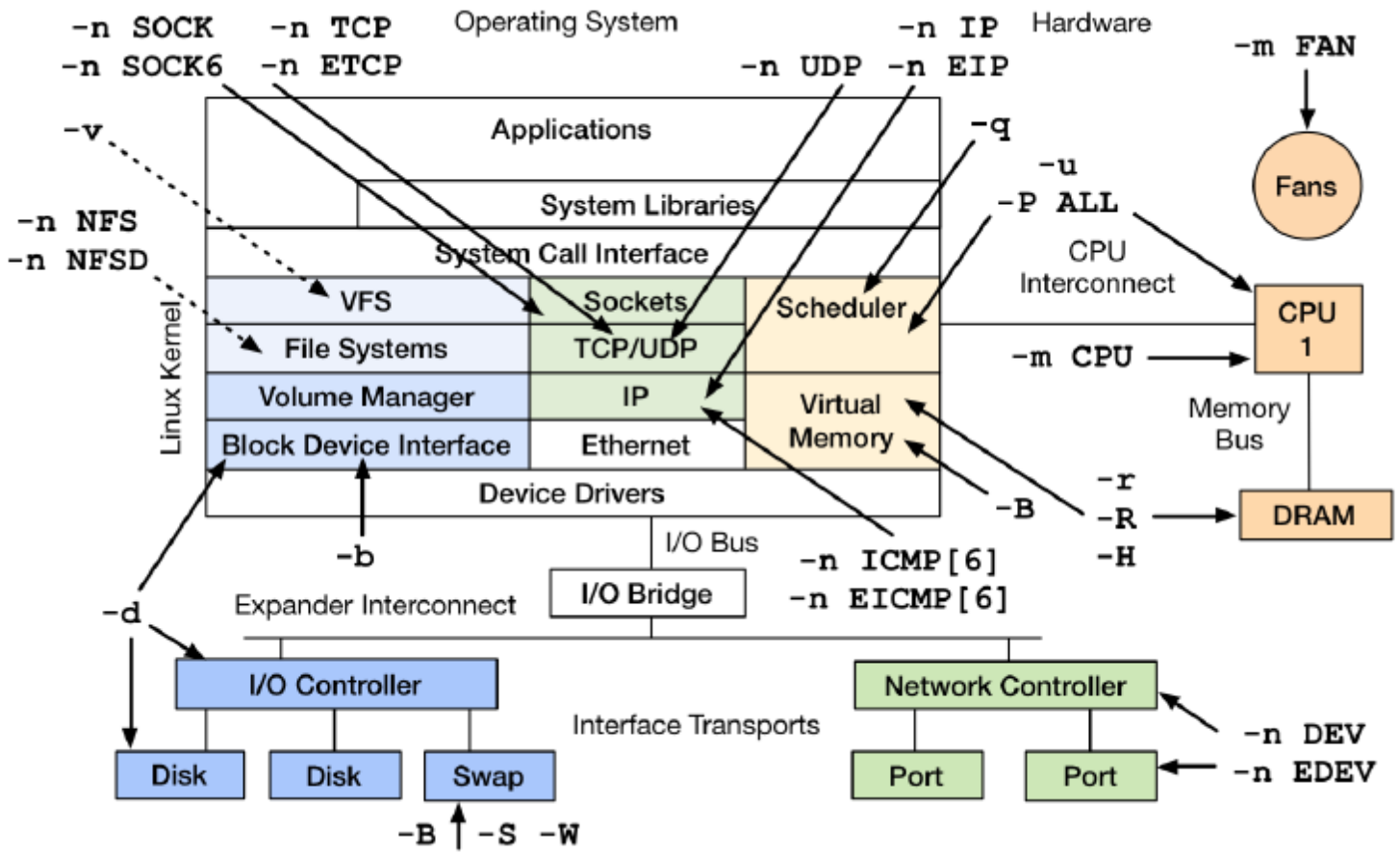
Observability



Observability

- uptime
- top (or htop)
- ps
- vmstat
- iostat
- mpstat
- free

Observability - SAR



Other tools

- Tools
  - strace
  - tcpdump
  - netstat
  - nicstat
  - pidstat
  - swapon
  - lsof
- You may also use collectl, atop, dstat, or another measure-all tool
- **The tool isn't important – it's important to have *a* way to measure everything**
- In cloud environments, you are probably using a monitoring product, developed in-house or commercial.
  - Atlas for cloud-wide monitoring, and Vector for instance-level analysis @NetFlix....

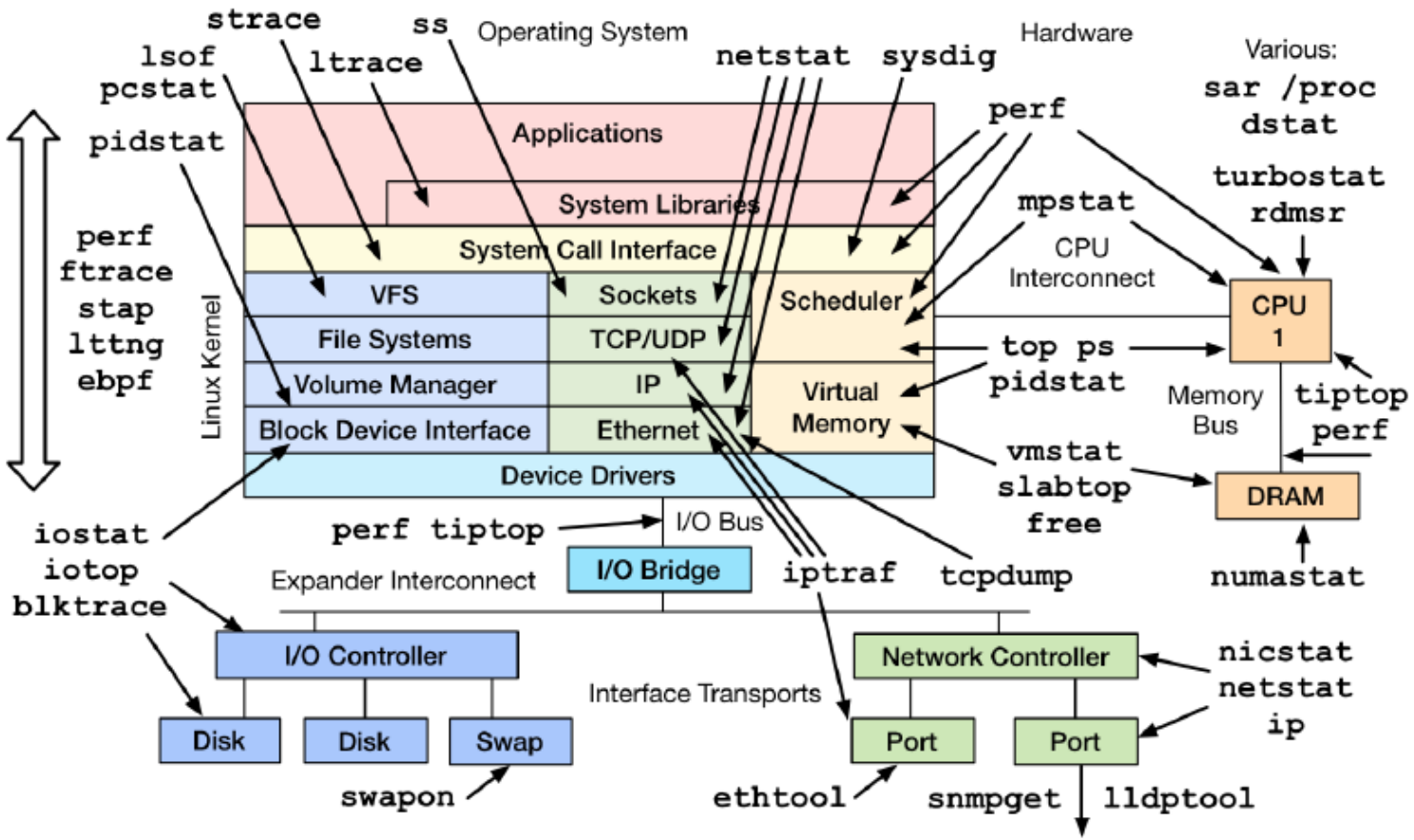
Advanced Observability Tools

- Misc:
  - ltrace, ss, iptraf, ethtool, snmpget, lldptool, iotop, blktrace, slabtop, /proc, pcstat
- CPU Performance Counters:
  - perf_events, tiptop, rdmsr
- Advanced Tracers:
  - perf_events, ftrace, **eBPF**, SystemTap, ktap, LTTng, dtrace4linux, sysdig

Even More Advanced tools

Tool	Description
ltrace	Library call tracer
ethtool	Mostly interface tuning; some stats
snmpget	SNMP network host statistics
lldptool	Can get LLDP broadcast stats
blktrace	Block I/O event tracer
/proc	Many raw kernel counters
pmu-tools	On- and off-core CPU counter tools

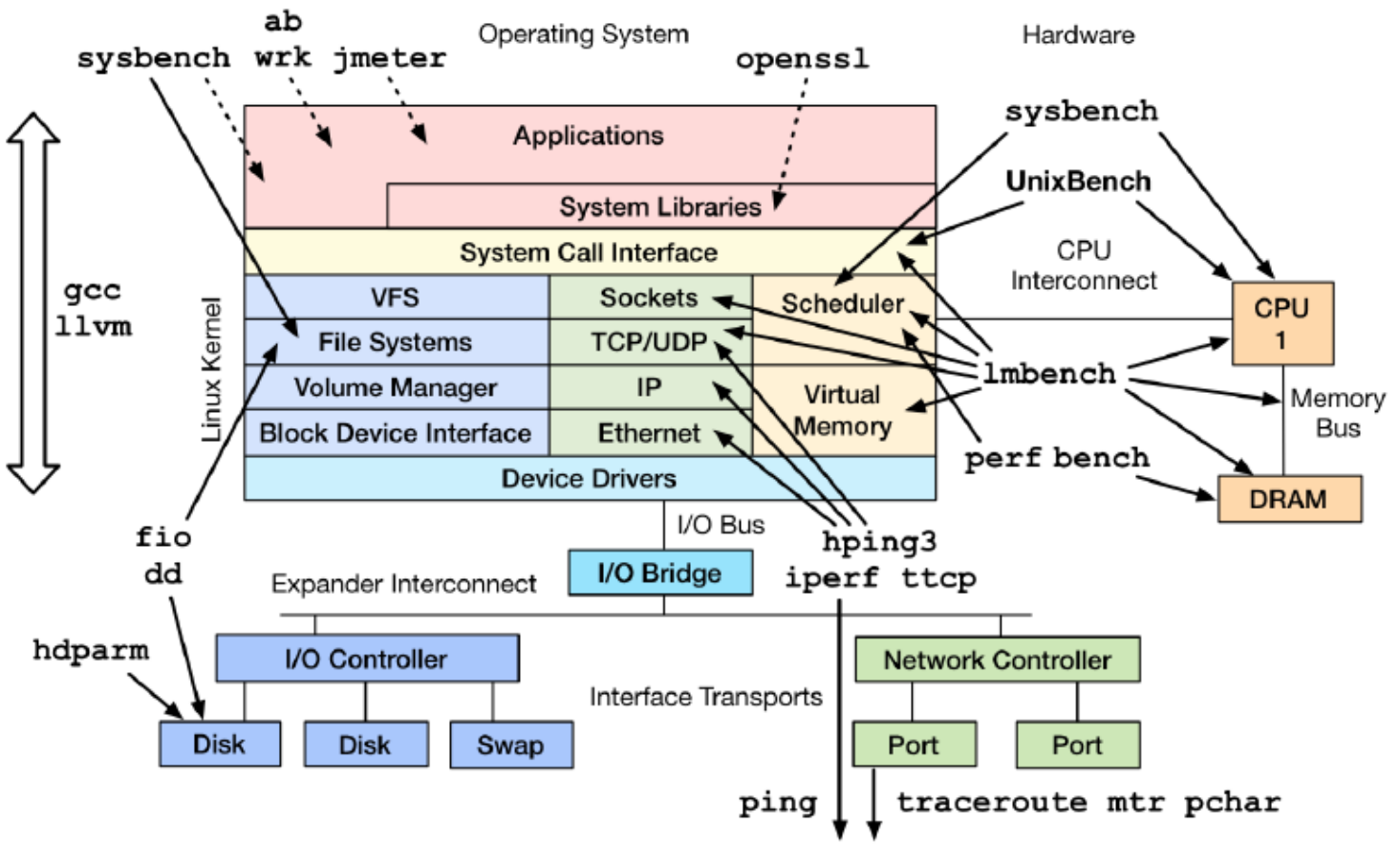
In Summary


strace
ss
Operating System
Hardware
Various:
lsof
ltrace
netstat
sysdig
pcstat
perf
sar /proc
dstat
pidstat
turbostat
rdmsr
Applications
System Libraries
mpstat
perf
ftrace
stap
lttng
ebpf
System Call Interface
CPU
Interconnect
VFS
Sockets
Scheduler
CPU
1
File Systems
TCP/UDP
top ps
pidstat
Linux Kernel
Volume Manager
IP
Virtual
Memory
Memory
Bus
tiptop
perf
Block Device Interface
Ethernet
vmstat
slabtop
free
Device Drivers
DRAM
iostat
iotop
blktrace
perf tiptop
I/O Bus
I/O Bridge
Expander Interconnect
iptraf
tcpdump
numastat
I/O Controller
Network Controller
nicstat
netstat
ip
Interface Transports
Disk
Disk
Swap
Port
Port
swapon
ethtool
snmpget
lldptool

Benchmarking tools

- Multi:
  - UnixBench, lmbench, sysbench, perf bench
- FS/disk:
  - dd, hdparm, fio
- App/lib:
  - ab, wrk, jmeter, openssl
- Networking:
  - ping, hping3, iperf, ttcp, traceroute, mtr, pchar

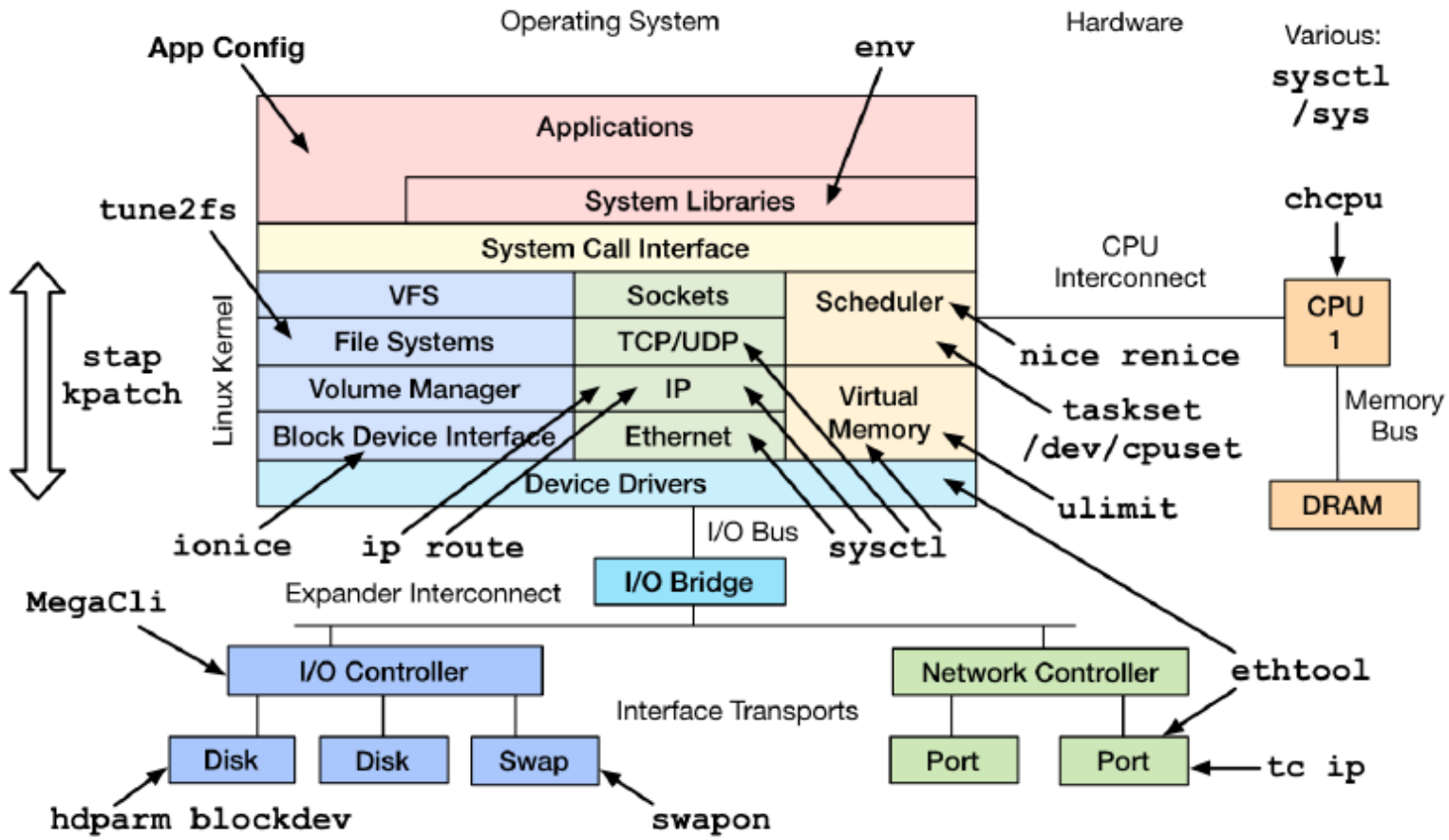
Benchmarking tools



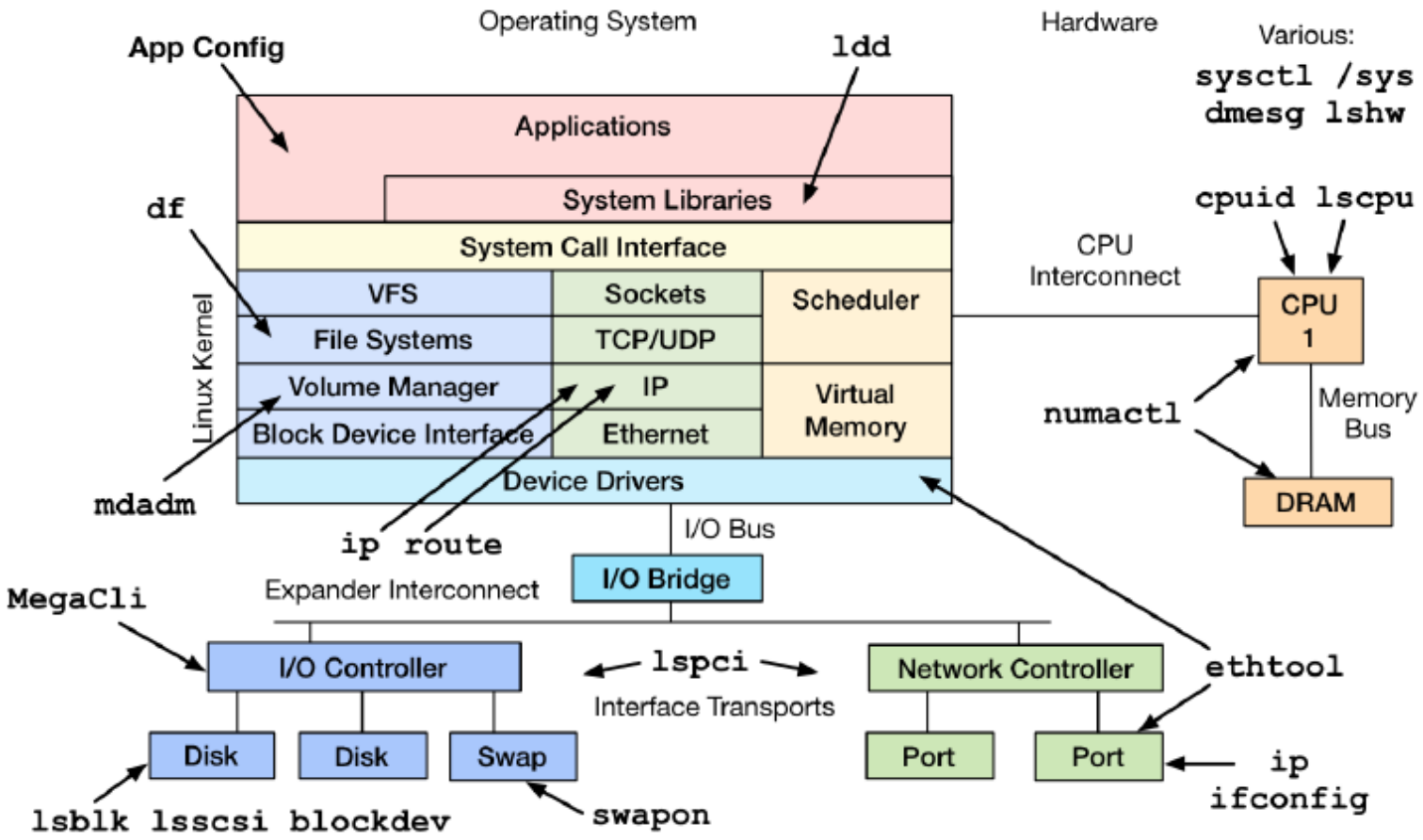
Tuning tools

- Generic interfaces:
  - sysctl, /sys
- Many areas have custom tuning tools:
  - Applications: their own config
  - CPU/scheduler: nice, renice, taskset, ulimit, chcpu
  - Storage I/O: tune2fs, ionice, hdparm, blockdev, …
  - Network: ethtool, tc, ip, route
  - Dynamic patching: stap, kpatch

Tuning Tools



Static tools



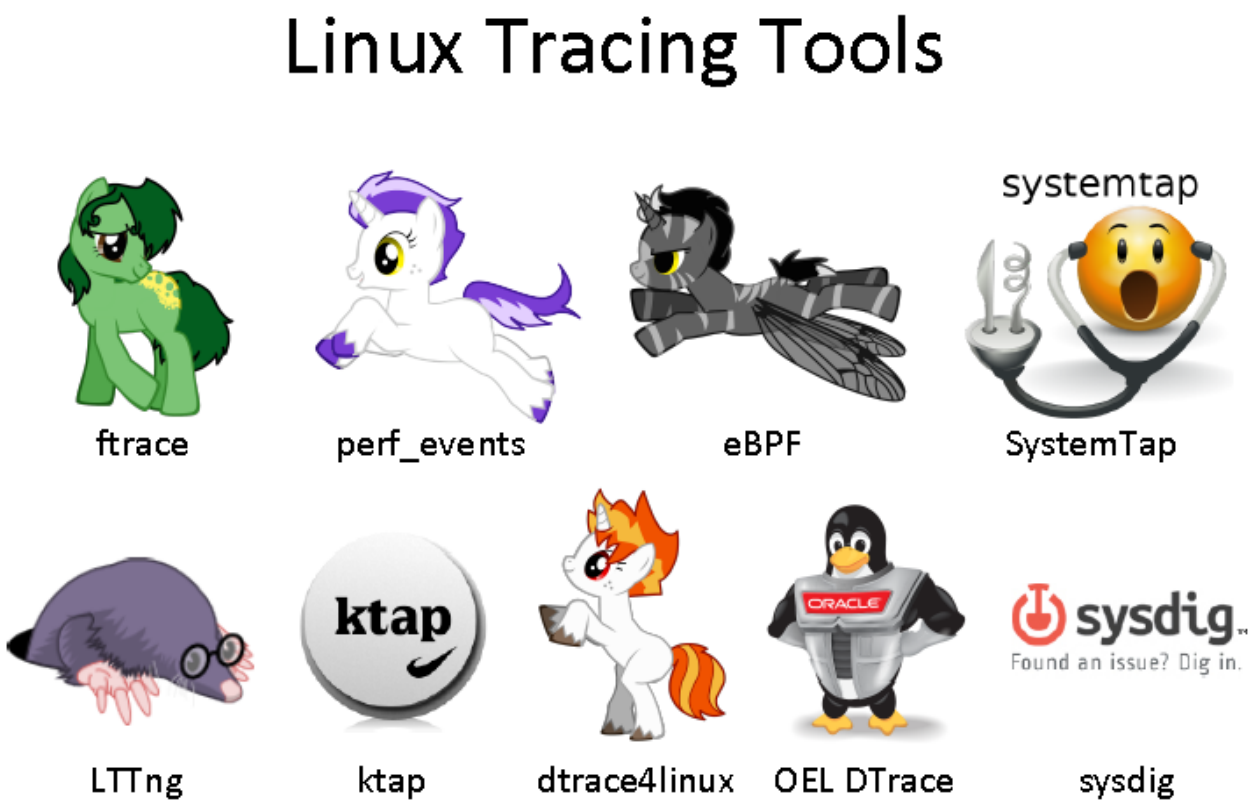
Profiling – perf_events

- **usage: perf [--version] [--help] [OPTIONS] COMMAND [ARGS]!**
- **!**
- **The most commonly used perf commands are:!**
- **annotate Read perf.data (created by perf record) and display annotated code!**
- **archive Create archive with object files with build-ids found in perf.data file!**
- **bench General framework for benchmark suites!**
- **buildid-cache Manage build-id cache.!**
- **buildid-list List the buildids in a perf.data file!**
- **data Data file related processing!**
- **diff Read perf.data files and display the differential profile!**
- **evlist List the event names in a perf.data file!**
- **inject Filter to augment the events stream with additional information!**

Profiling – perf_events

- **usage: perf [--version] [--help] [OPTIONS] COMMAND [ARGS]!**
- **kmem Tool to trace/measure kernel memory(slab) properties!**
- **kvm Tool to trace/measure kvm guest os!**
- **list List all symbolic event types!**
- **lock Analyze lock events!**
- **mem Profile memory accesses!**
- **record Run a command and record its profile into perf.data!**
- **report Read perf.data (created by perf record) and display the profile!**
- **sched Tool to trace/measure scheduler properties (latencies)!**
- **script Read perf.data (created by perf record) and display trace output!**
- **stat Run a command and gather performance counter statistics!**
- **test Runs sanity tests.!**
- **timechart Tool to visualize total system behavior during a workload!**
- **top System profiling tool.!**
- **trace strace inspired tool!**
- **probe Define new dynamic tracepoints!**
- **!**
- **See 'perf help COMMAND' for more information on a specific command.*!***

Not covering – Tracing Tools

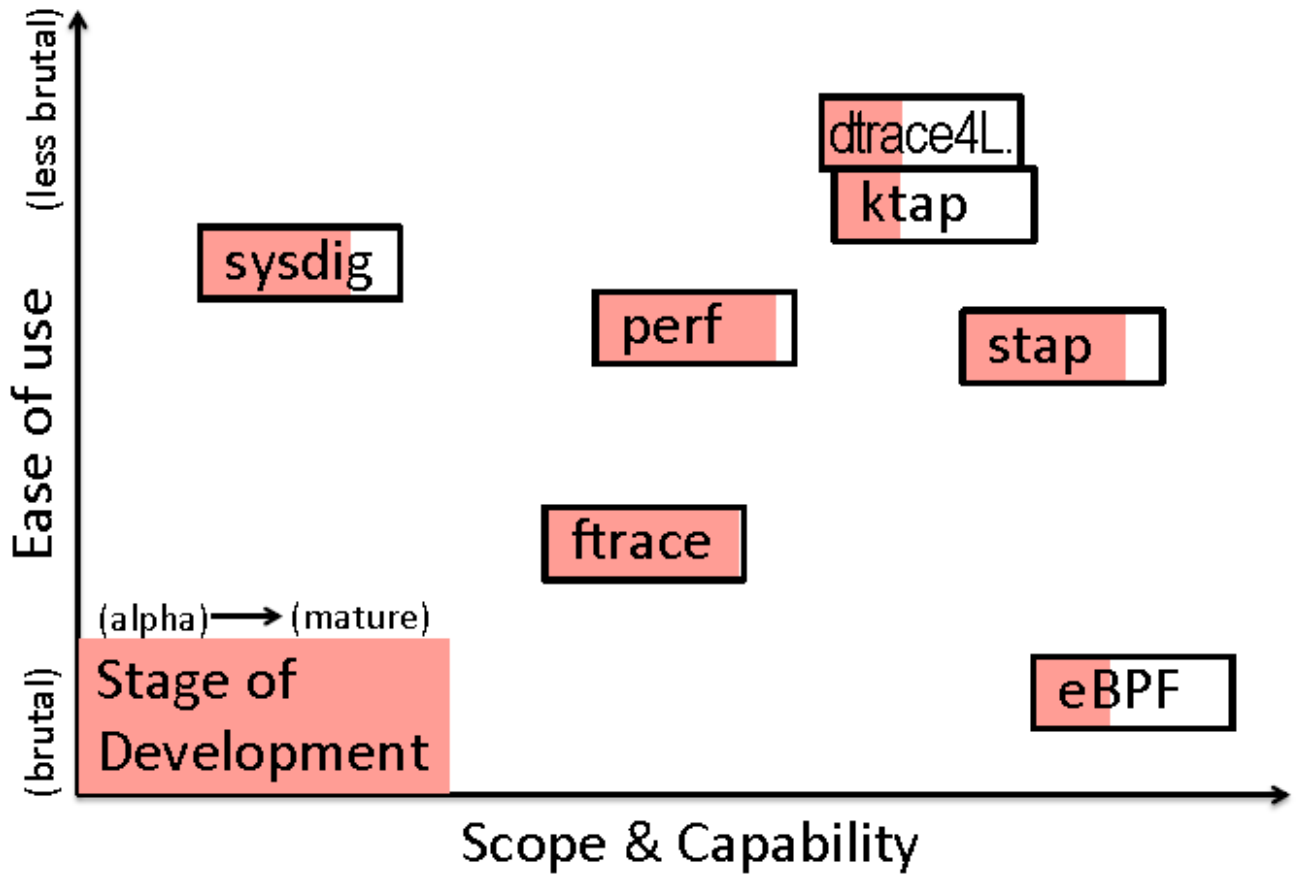


- Many choices (too many?), all still in development

Tracing Background

- Linux provides three tracing sources
  - **tracepoints**: kernel static tracing
  - **kprobes**: kernel dynamic tracing
  - **uprobes**: user-level dynamic tracing

The Future??


Ease of use
(less brutal)
(brutal)
dtrace4L.
ktap
sysdig
perf
stap
ftrace
(alpha) → (mature)
Stage of Development
eBPF
Scope & Capability

Notes….

- Major page faults on conventional (hard disk) computers can have a significant impact on performance. An average hard disk has an average rotational latency of 3ms, a seek-time of 5ms, and a transfer-time of 0.05 ms/page. So the total time for paging is near 8ms (8 000 µs). If the memory access time is 0.2 µs, then the page fault would make the operation about 40,000 times slower.

Notes

- Run Queue Notes
  - Run queues should not exceed number of cores (number of cores = max load)
  - It should be 70% of load
    - if consistently above – start investigating
  - it should not continually exceed 1/core
    - if yes, start fixing now
  - If above 5, serious trouble !!!
- One quadcore = 2 CPU x 2 Cores (and yes, we are simplifying matters)

Resource

- http://blog.sjas.de/posts/Linux-performance-observability-tools.html
- http://www.brendangregg.com/linuxperf.html , incl.tools diagrams as PNGs
- http://www.brendangregg.com/perf.html#FlameGraphs
- http://www.brendangregg.com/blog/2015-02-27/linux-profiling-at-neqlix.html
- http://www.brendangregg.com/blog/2015-03-17/linux-performance-analysis-perf-tools.html
- http://www.brendangregg.com/blog/2015-05-15/ebpf-one-small-step.html
- nicstat: http://sourceforge.net/projects/nicstat
- tiptop: http://tiptop.gforge.inria.fr

Resource

- ftrace and perf tools:
- • https://github.com/brendangregg/perf-tools
- • http://lwn.net/Articles/608497/
- MSR tools: https://github.com/brendangregg/msr-cloud-tools
- pcstat: https://github.com/tobert/pcstat
- eBPF: http://lwn.net/Articles/603983/
- ktap: http://www.ktap.org
- SystemTap: https://sourceware.org/systemtap/
- sysdig: http://www.sysdig.org