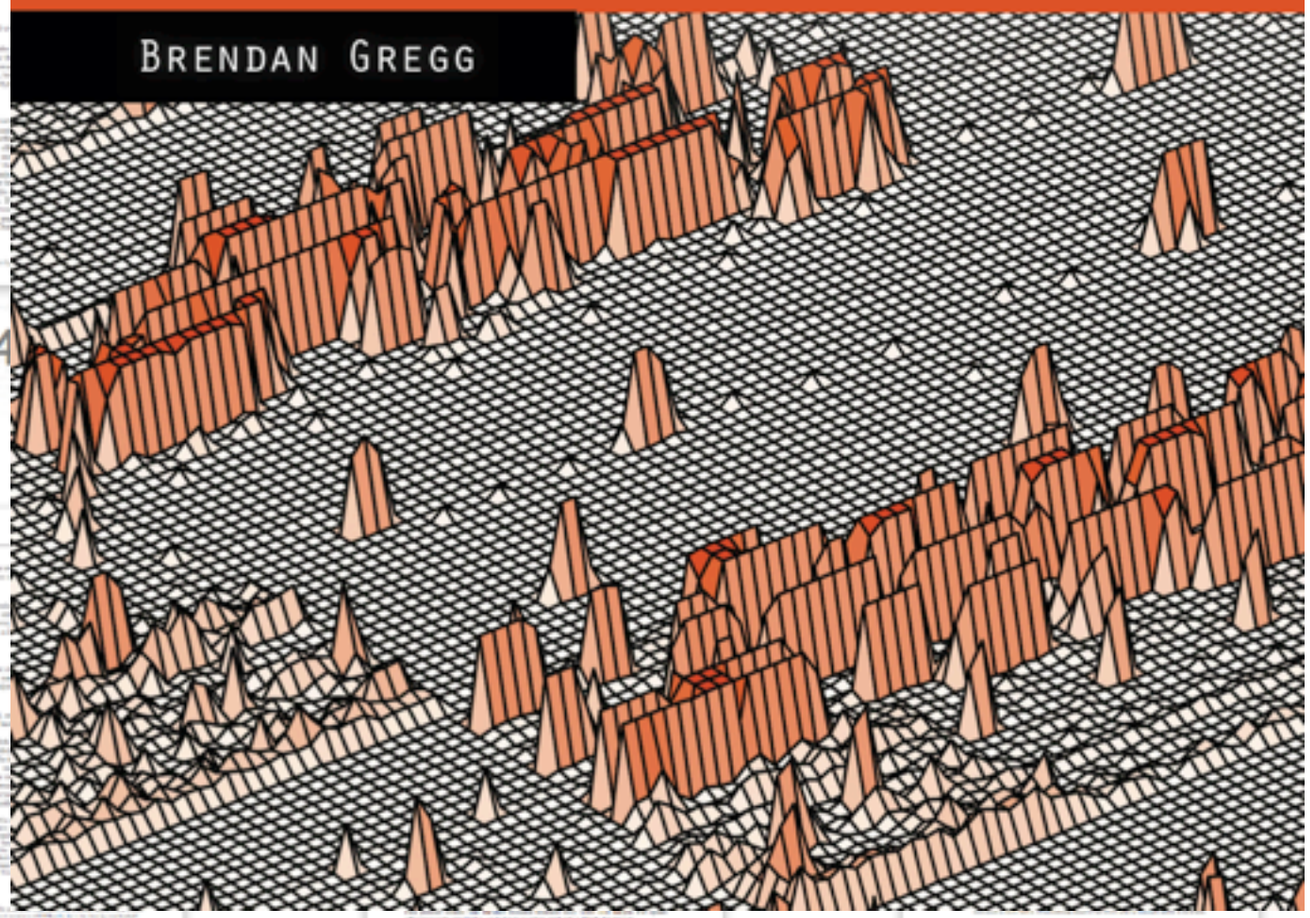
PRENTICE HALL

Systems Performance

Enterprise and the Cloud

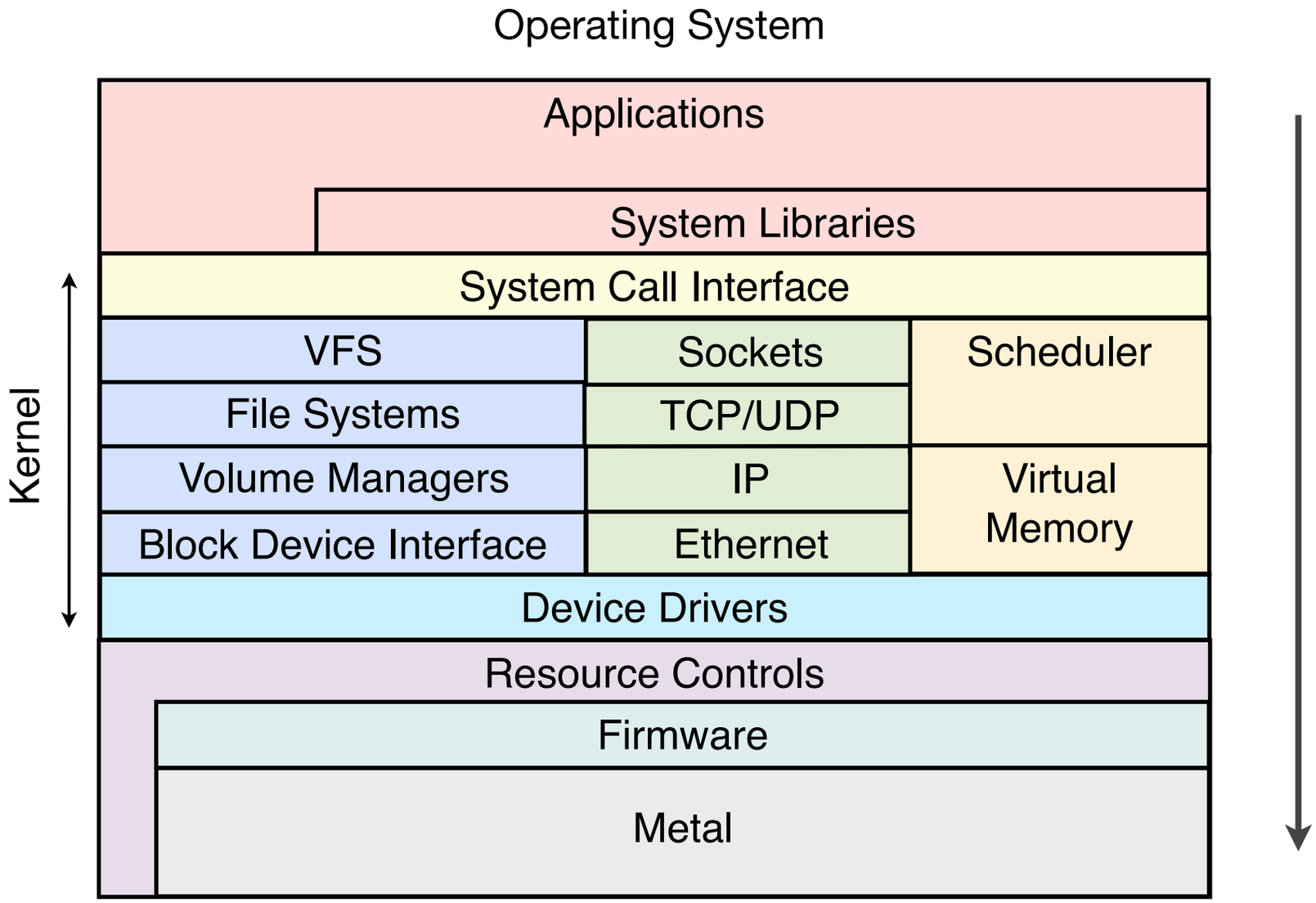
Brendan Gregg



BayLISA, Oct 2013

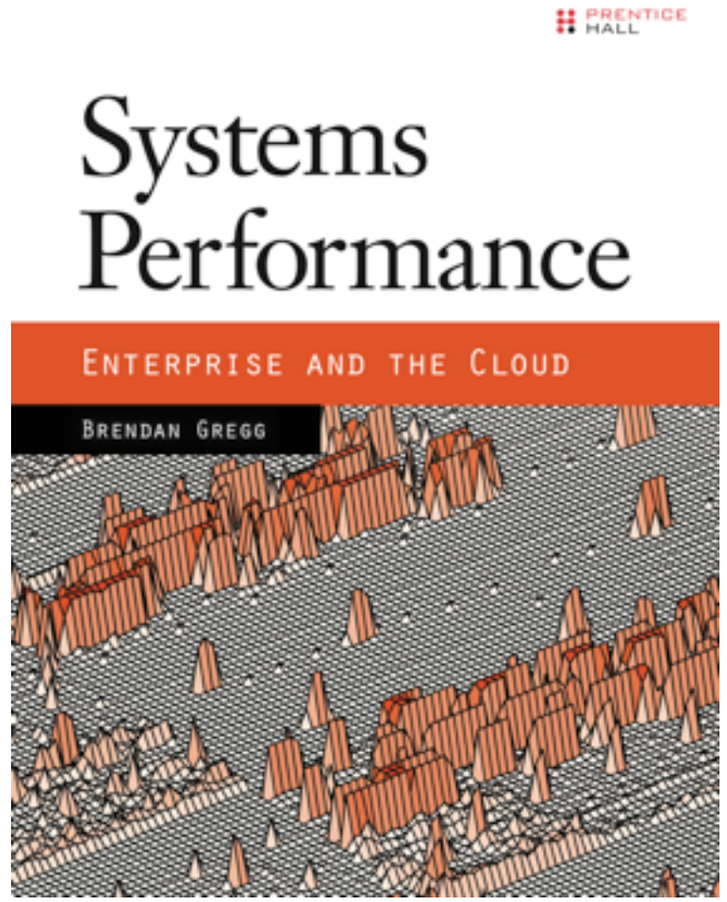
Systems Performance

- Analysis of apps to metal. Think LAMP not AMP.
- An activity for everyone: from casual to full time.
- The *basis* is the system
- The *target* is everything
- All software can cause performance problems



Systems Performance: Enterprise and the Cloud

- Brendan Gregg (and many others); Prentice Hall, 2013
- 635 pages of chapters, plus appendices, etc
- Background, methodologies, examples
- Examples from:
  - Linux (Ubuntu, Fedora, CentOS)
  - illumos (SmartOS, OmniOS)
- Audience:
  - Sysadmins, developers, everyone
  - Enterprise and cloud environments



The Author: Brendan Gregg

- Currently at Joyent, previously Brendan@Sun, then Oracle
- Lead Performance Engineer: debugs perf on SmartOS/Linux/Windows daily, small to large cloud environments, any layer of the software stack, down to firmware and metal. Previously a kernel engineer, performance consultant, trainer.
- Written hundreds of published perf tools (too many), including the original iosnoop, iotop, execsnoop, nicstat, psio, etc.
- Created visualizations: heat maps for various uses, flame graphs, frequency trails, cloud process graphs
- Developed methodologies: USE method, TSA method
- Co-authored books: DTrace, Solaris Performance and Tools

Goals

- Modern systems performance: including cloud computing, dynamic tracing, visualizations, open source
- Accessible to a wide audience
- Help you maximize system and application performance
- Quickly diagnose performance issues: eg, outilers
- Turn unknown unknowns into known unknowns – actionable
- 10+ year shelf life: document concepts and methodology first, with tools and tunables of the day as examples of application

Personal Motivation

- The need for a good reference for:
  - Internal Joyent staff
  - External customers
  - IT at large
- As a reference for classes
  - I’ve been teaching professional classes in system administration and performance on and off since 2001
  - I’ve learned a lot from teaching students to solve real performance problems, to see what works, what doesn’t
  - I’ve been using this book already for teaching the Joyent cloud performance class: http://joyent.com/training, next class Nov 18th 2013

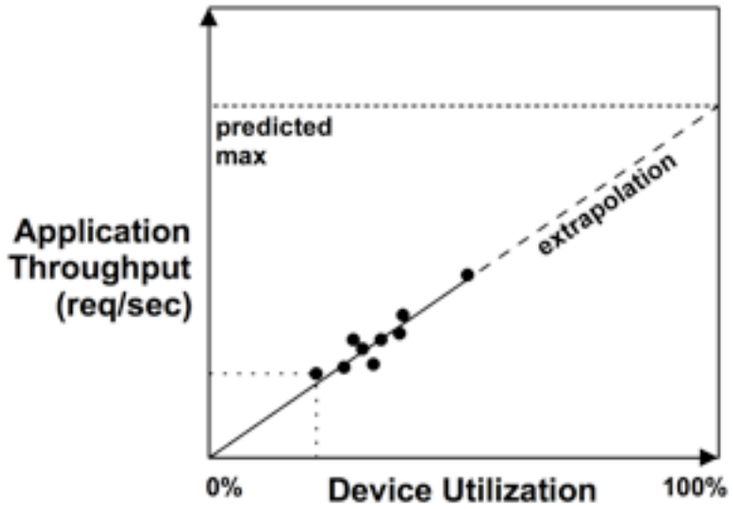
Table of Contents

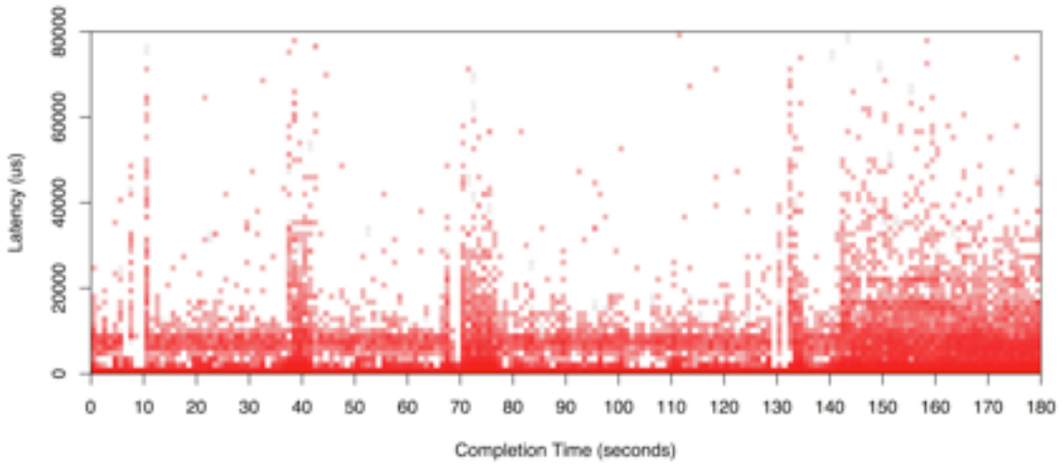
Highlights:

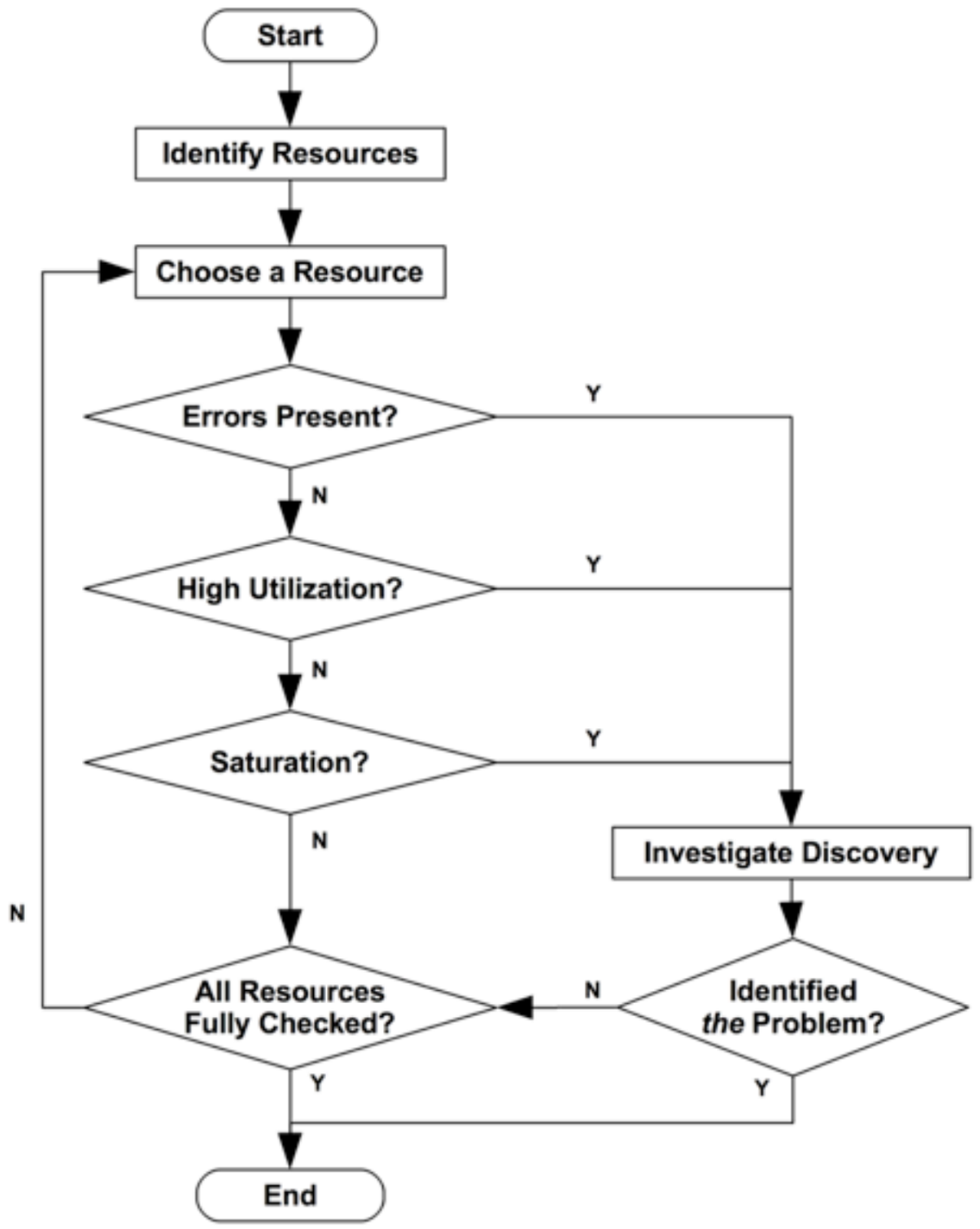
- Chapter 2 Methodologies:
  - Many documented for the first time; some created by me
- Chapter 3 Operating Systems:
  - 30 page summary of OS internals
- Chapter 6-10: CPUs, Memory, FS, Disks, Network
  - Background, methodology, tools
- Chapter 11: Cloud Computing
  - Different technologies and their performance
- Chapter 12: Benchmarking
  - For the good of the industry. Please, everyone, read this.

Chapter 2 Methodologies

- Documenting the black art of systems performance
- Also summarizes concepts, statistics, visualizations

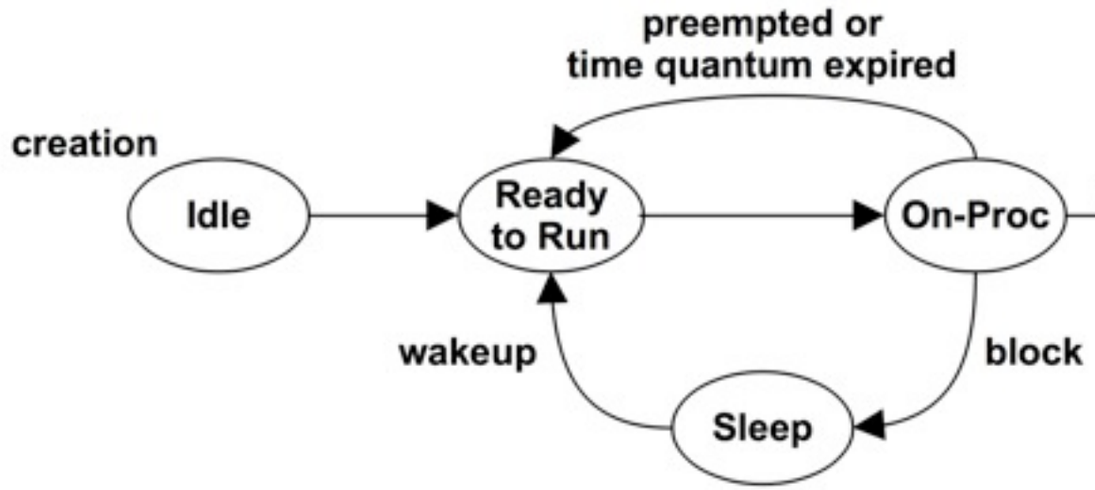


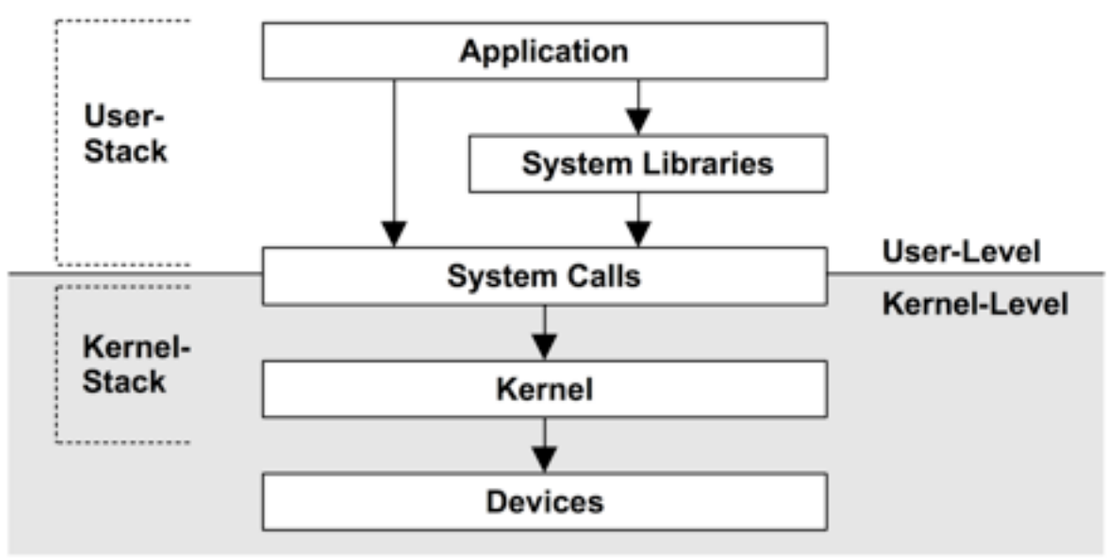


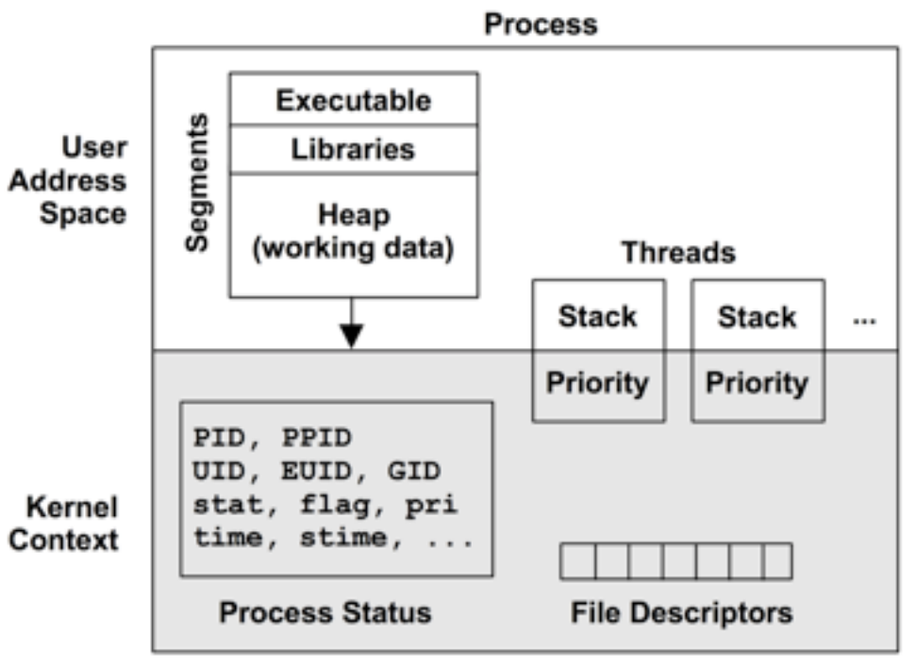


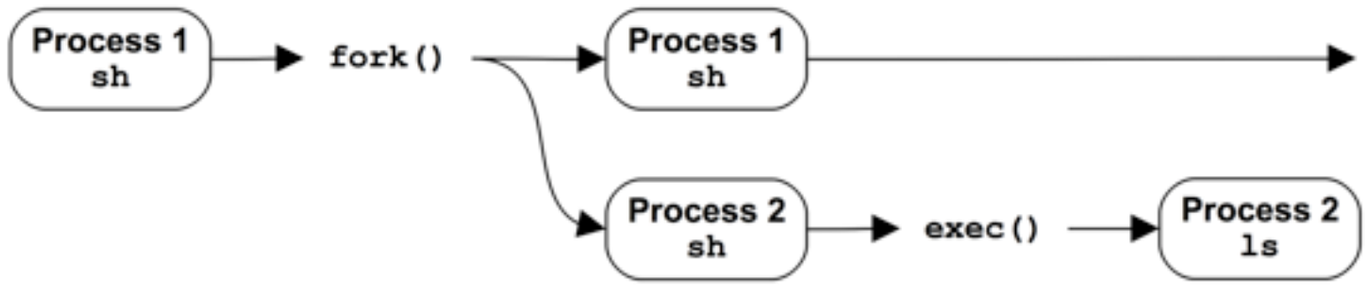
Chapter 3 Operating Systems

- The OS crash course you missed at University







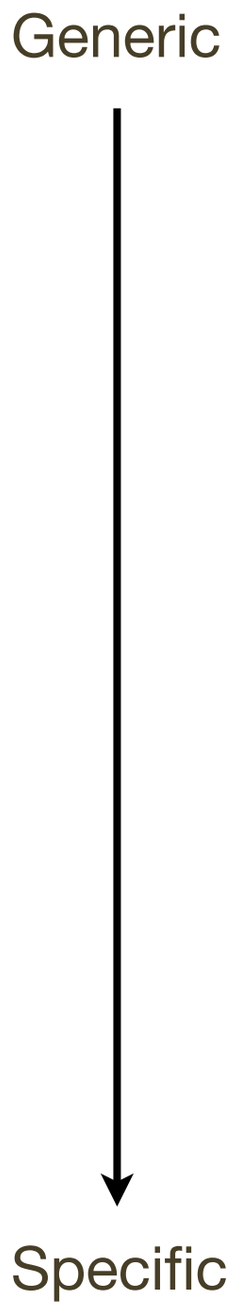


Chapter 6-10 Structure

- Background
  - Just enough OS and HW internals
- Methodologies
  - For beginners, casual users, experts
  - How to start, and steps to proceed
- Example Application
  - Linux, illumos
  - Tools, screenshots, case studies
  - Some tunables of the day

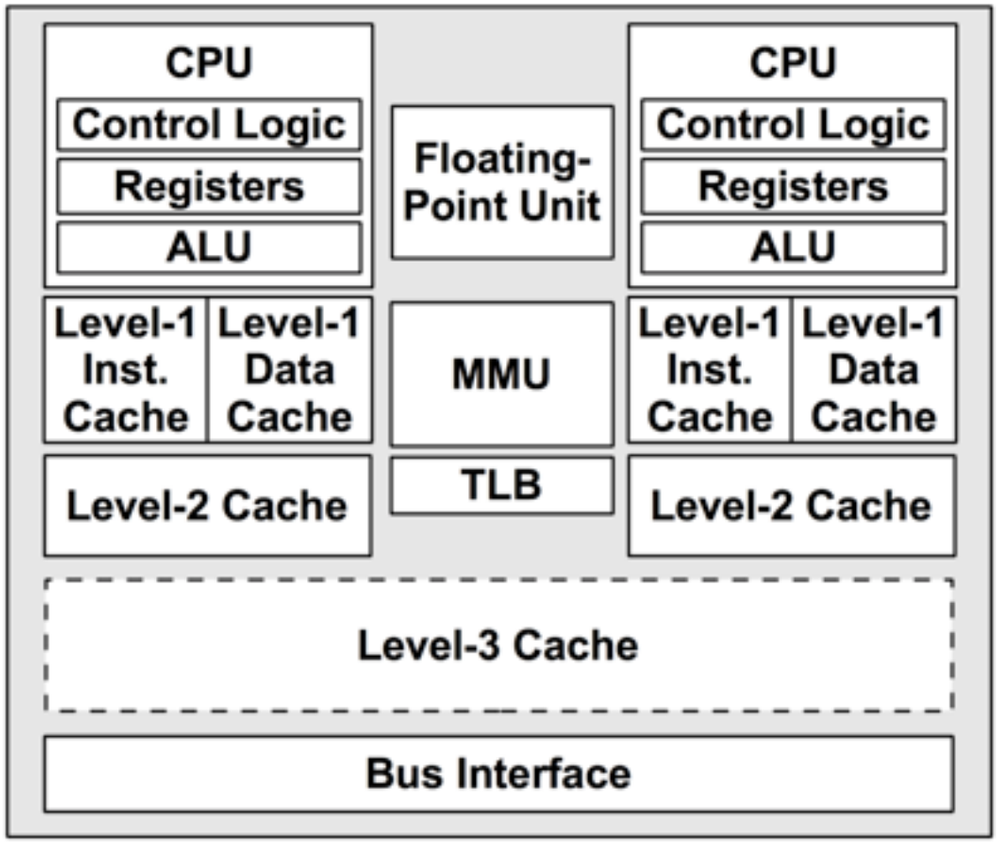
Chapter 6-10 Structure

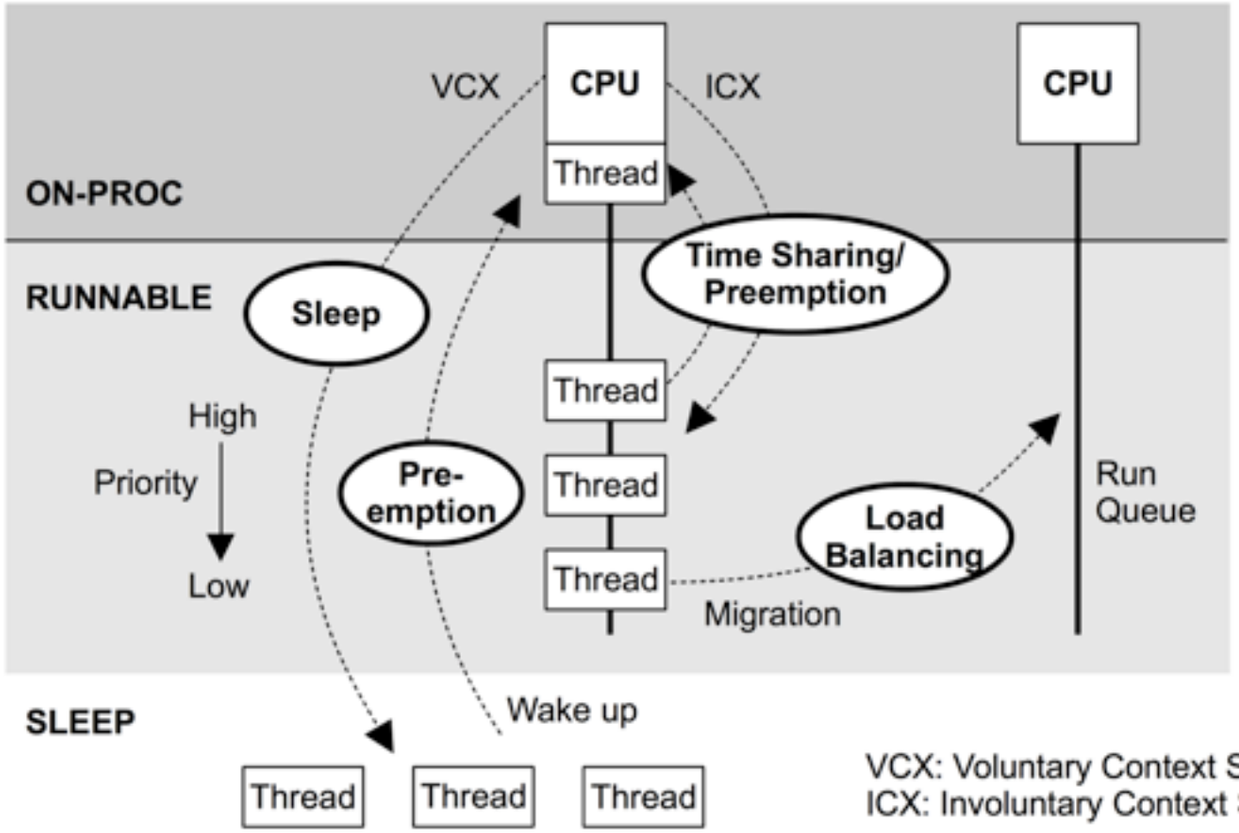
- Background
  - Just enough OS and HW internals
- Methodologies
  - For beginners, casual users, experts
  - How to start, and steps to proceed
- Example Application
  - Linux, illumos
  - Tools, screenshots, case studies
  - Some tunables of the day



Example: Chapter 6 CPUs

Hardware



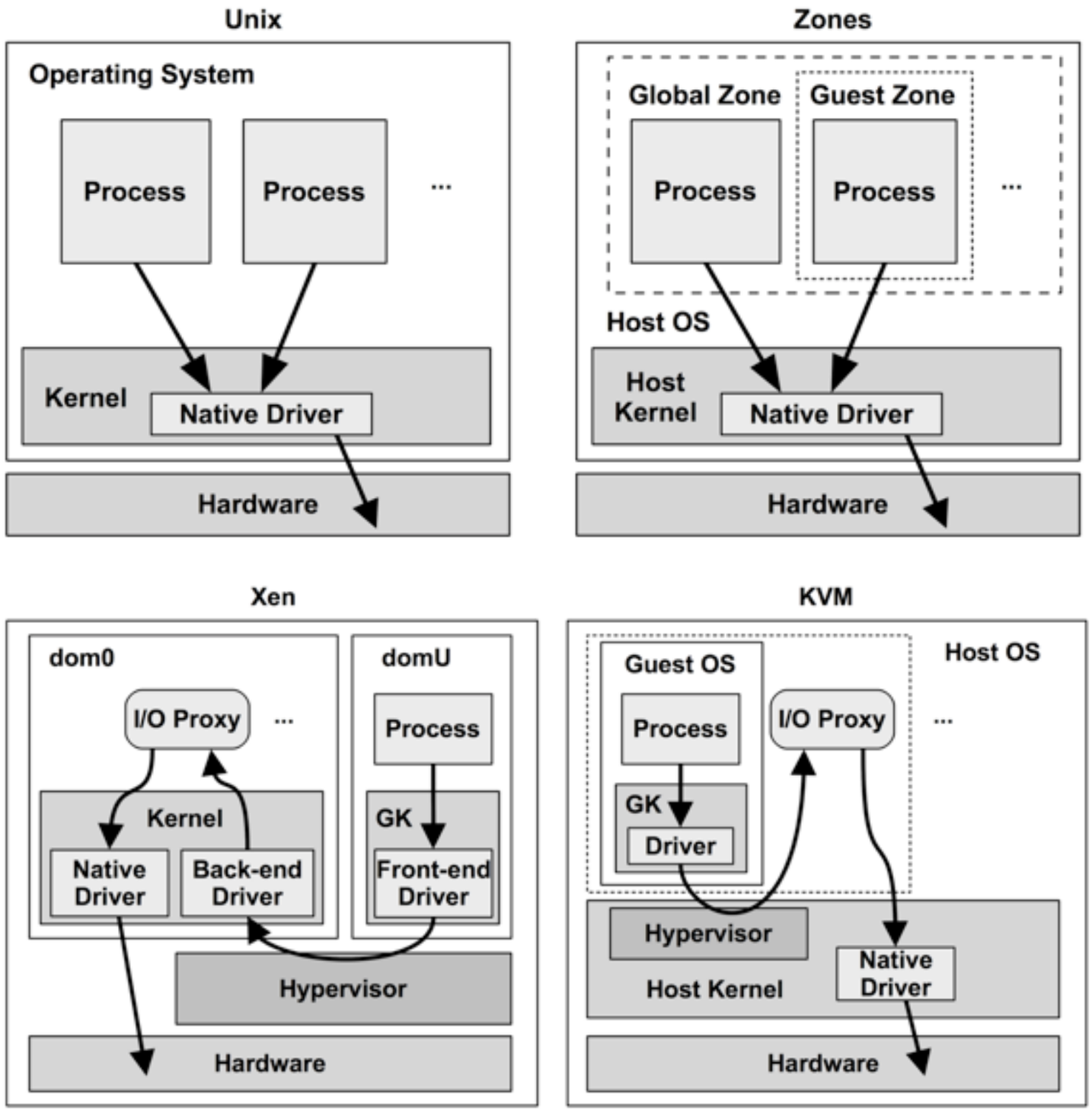


VCX: Voluntary Context Switch
ICX: Involuntary Context Switch

Software

Chapter 11 Cloud Computing

- OS Virtualization
- HW Virtualization
- Observability
- Performance
- Resource controls



Modern Systems Performance

- Comparing 1990's to 2010's

1990's Systems Performance

* Proprietary Unix, closed source, static tools

```
$ vmstat 1
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr cd cd s0 s5   in   sy   cs us sy id
 0 0 0 8475356 565176 2   8  0  0  0  0  1  0  0 -0 13  378  101  142  0  0 99
 1 0 0 7983772 119164 0   0  0  0  0  0  0 224 0  0  0 1175 5654 1196  1 15 84
 0 0 0 8046208 181600 0   0  0  0  0  0  0 322 0  0  0 1473 6931 1360  1  7 92
[...]
```

* Limited metrics and documentation
* Some perf issues could not be solved
* Analysis methodology constrained by tools
* Perf experts used inference and experimentation
* Literature is still around

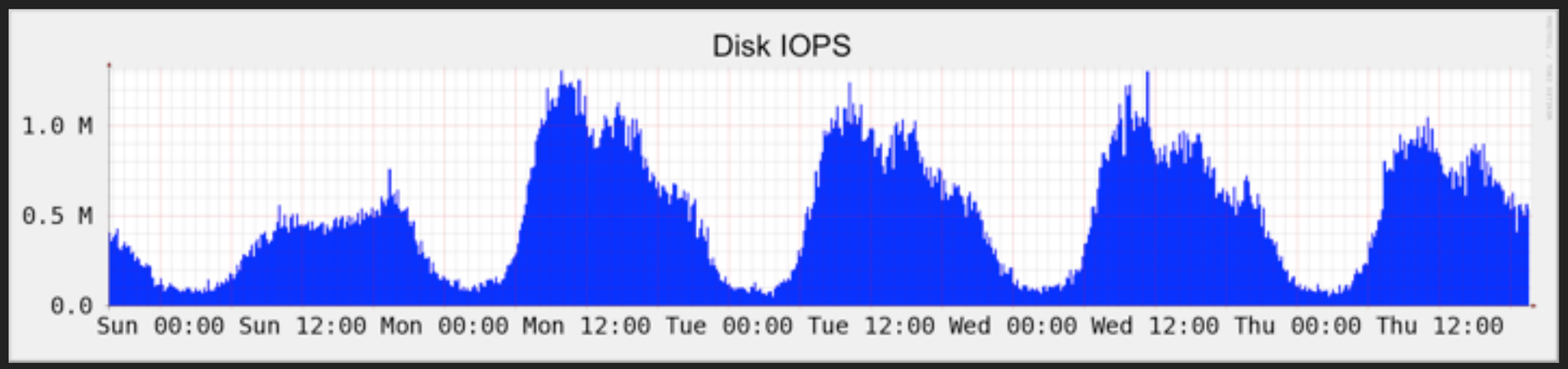
2010's Systems Performance

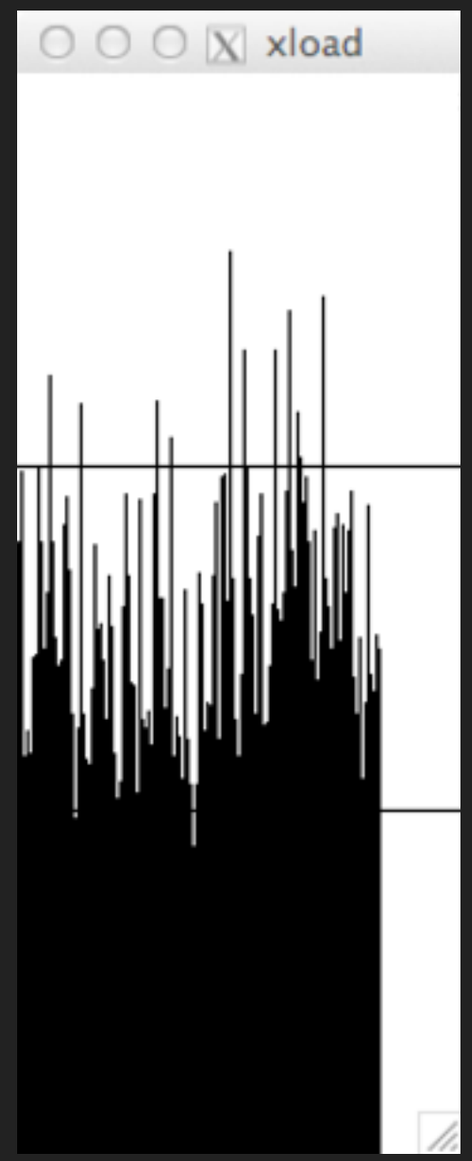
- Open source (the norm)
  - Ultimate documentation
- Dynamic tracing
  - Observe everything
- Visualizations
  - Comprehend many metrics
- Cloud computing
  - Resource controls can be the bottleneck!
- Methodologies
  - Where to begin, and steps to root cause

1990's Performance Visualizations

Text-based and line graphs

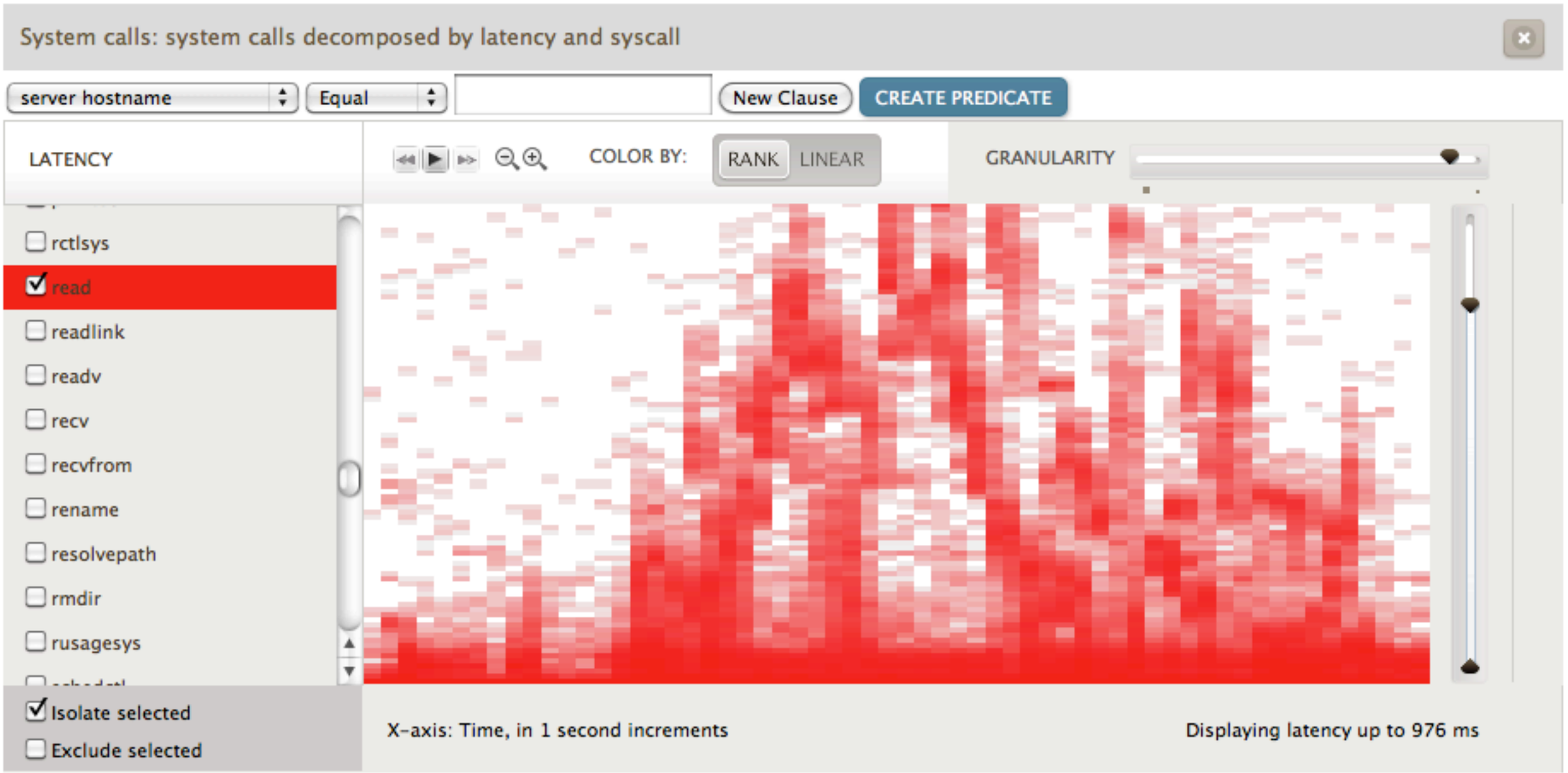
```
$ iostat -x 1
                    extended device statistics
device     r/s     w/s    kr/s    kw/s wait actv  svc_t  %w  %b
sd0        0.0     0.1     5.2     3.9  0.0  0.0   69.8   0   0
sd5        0.0     0.0     0.0     0.0  0.0  0.0    1.1   0   0
sd12       0.0     0.2     0.2     1.1  0.0  0.0    3.1   0   0
sd12       0.0     0.0     0.0     0.0  0.0  0.0    0.0   0   0
sd13       0.0     0.0     0.0     0.0  0.0  0.0    0.0   0   0
sd14       0.0     0.0     0.0     0.0  0.0  0.0    1.9   0   0
sd15       0.0     0.0     0.0     0.0  0.0  0.0    0.0   0   0
sd16       0.0     0.0     0.0     0.0  0.0  0.0    0.0   0   0
nfs6       0.0     0.0     0.0     0.0  0.0  0.0    0.0   0   0
[...]
```





2010's Performance Visualizations

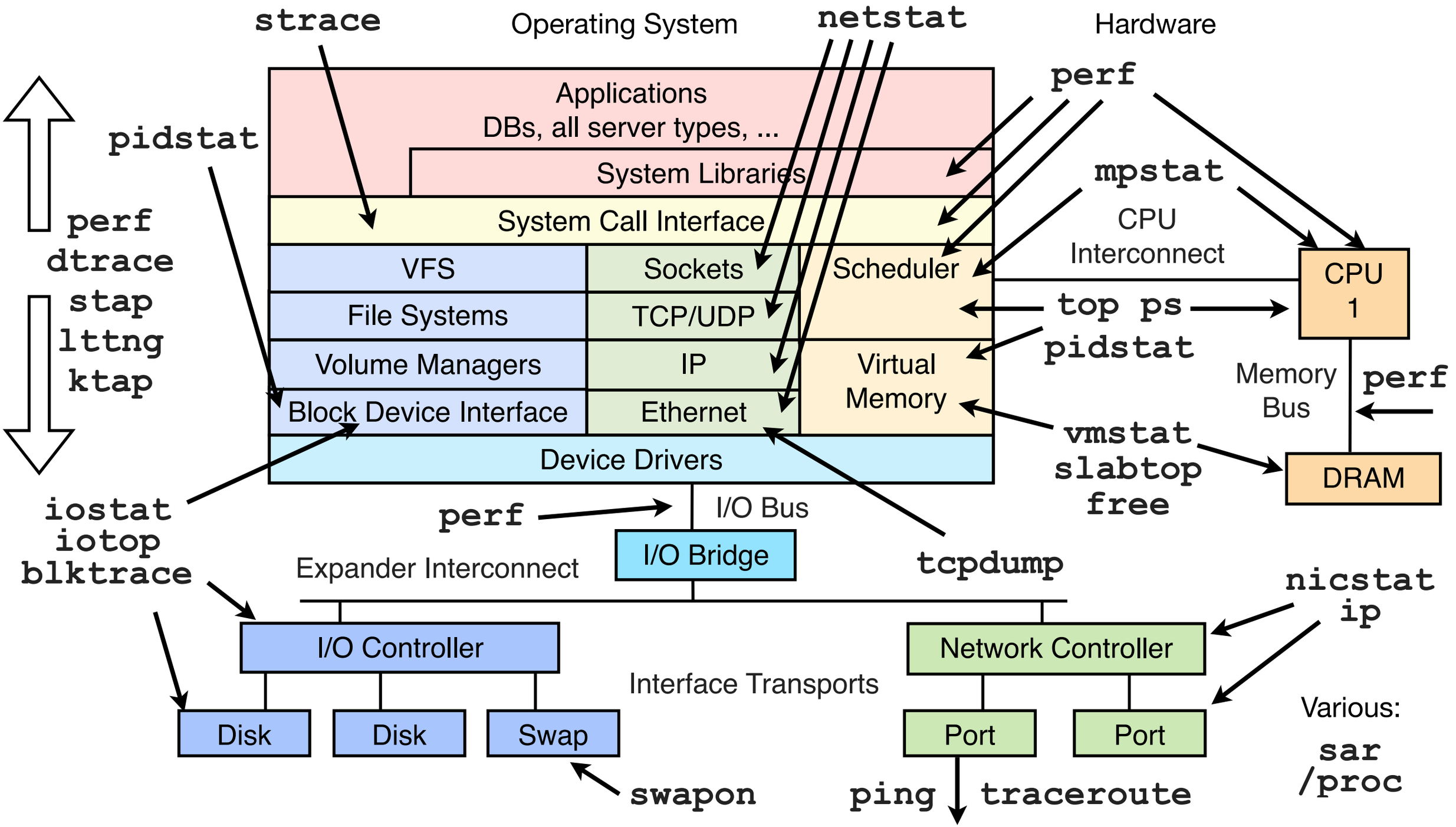
- Utilization and latency heat maps, flame graphs



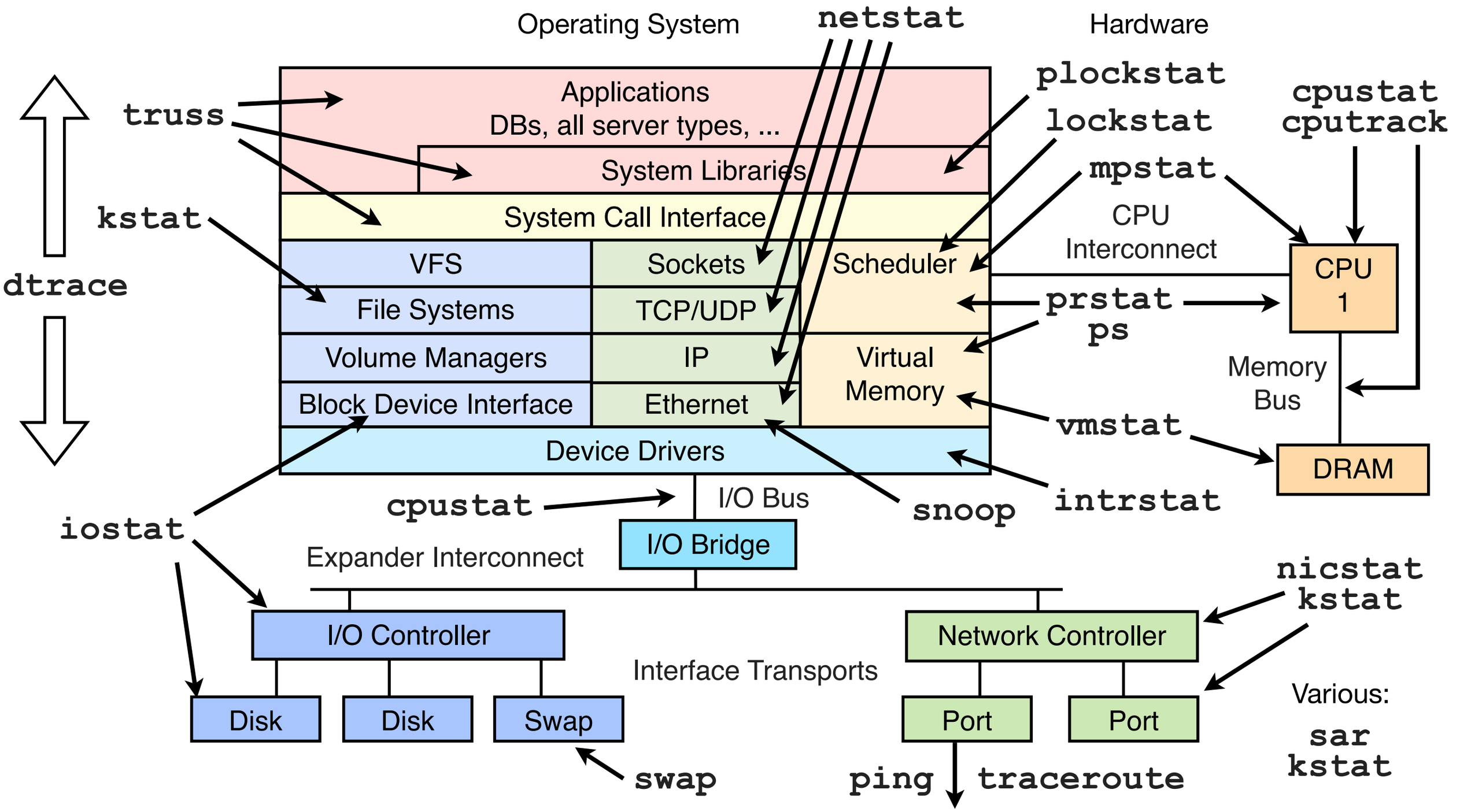
Modern Performance Analysis Tools

- Traditional tools
- Plus dynamic tracing to fill in gaps

Performance Analysis Tools: Linux

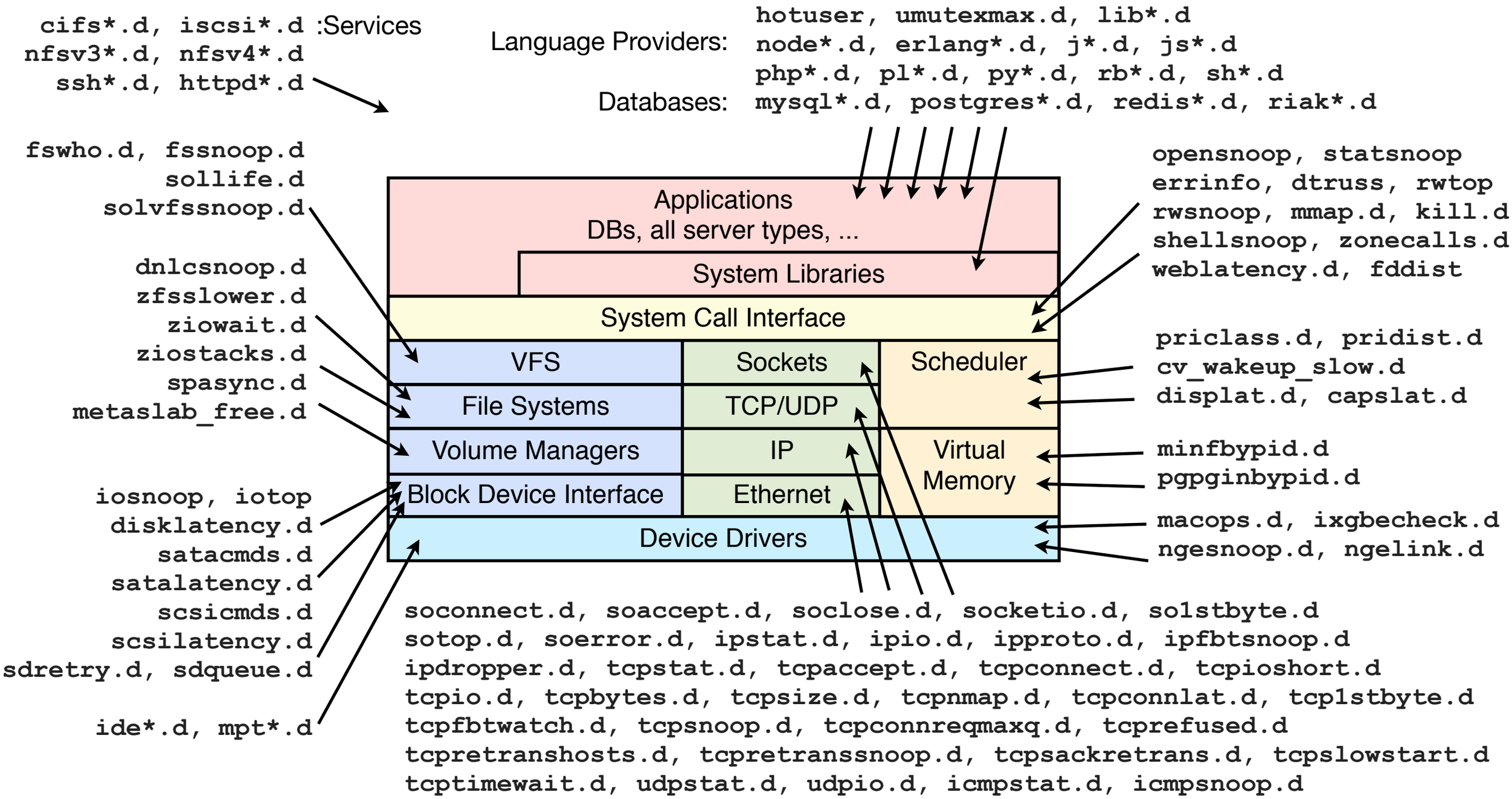


Performance Analysis Tools: illumos



Dynamic Tracing: DTrace

- Example DTrace scripts from the DTraceToolkit, DTrace book, ...



Too Many Tools

- It's not really about the tools
  - ... those previous diagrams aren't even in the book
- It's about what you need to accomplish, and then finding the tools to answer them
- This is documented as methodologies
- Tools are then used as examples

The following example uses the context switch software event to trace when applications leave the CPU and collects call stacks for 10 s:

```
# perf record -f -g -a -e context-switches sleep 10
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.417 MB perf.data (~18202 samples) ]
# perf report --stdio
# ========
# captured on: Wed Apr 10 19:52:19 2013
# hostname : 9d219ce8-cf52-409f-a14a-b210850f3231
[...]
#
# Events: 2K context-switches
#
# Overhead    Command      Shared Object      Symbol
# ........  .........  ..................  ..........
#
    47.60%       perl  [kernel.kallsyms]  [k] __schedule
                 |
                 --- __schedule
                     schedule
                     retint_careful
                     |
                     |--50.11%-- Perl_pp_unstack
                     |
                     |--26.40%-- Perl_pp_stub
                     |
                      --23.50%-- Perl_runops_standard

    25.66%        tar  [kernel.kallsyms]  [k] __schedule
                  |
                  --- __schedule
                      |
```

Modern Performance Methodologies

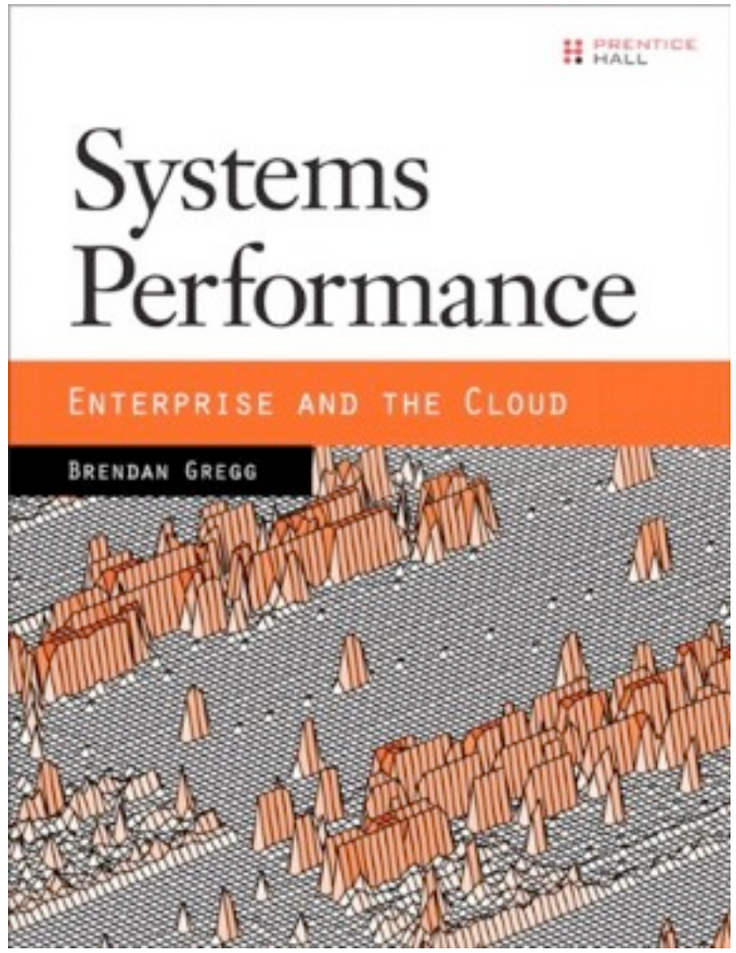
- Workload characterization
- USE Method
- TSA Method
- Drill-down Analysis
- Latency Analysis
- Event Tracing
- Static performance tuning
- ...
- Covered in Chapter 2 and later chapters

Table 2.4 Generic System Performance Methodologies (*Continued*)

Methodology	Type
Scientific method	observational analysis
Diagnosis cycle	analysis life cycle
Tools method	observational analysis
USE method	observational analysis
Workload characterization	observational analysis, capacity planning
Drill-down analysis	observational analysis
Latency analysis	observational analysis
Method R	observational analysis
Event tracing	observational analysis
Baseline statistics	observational analysis
Performance monitoring	observational analysis, capacity planning
Queueing theory	statistical analysis, capacity planning
Static performance tuning	observational analysis, capacity planning
Cache tuning	observational analysis, tuning
Micro-benchmarking	experimental analysis
Capacity planning	capacity planning, tuning

Systems Performance

- Really understand how systems work
- New observability, visualizations, methodologies
- Understand the challenges of cloud computing



- Brendan Gregg:
  - http://www.brendangregg.com
  - http://dtrace.org/blogs/brendan
  - twitter: @brendangregg

Sample Chapter →

http://dtrace.org/blogs/brendan/2013/06/21/systems-performance-enterprise-and-the-cloud/