Performance Instrumentation

Lecture 11

Motivation

- “My code is slow / uses lots of memory / is SIGKILLED. I implemented X, Y, and Z.”
  - Are those good? What should I do next?
- Measure (besides speedup)
  - Response time in normal and stressing scenarios
  - Energy usage
  - …

Overview of Performance Instrumentation

- Look inside the different processing tasks, and analyze their timing, and energy usage!
- Our approach for analyzing timing
  - Analyze performance bottlenecks and tune codes to improve timing in normal/stressing scenarios

Aspects

- Measures the quality attributes of the system
  - Scalability
  - Reliability
  - Resource usage
- Workload: normal (loaded) and overloaded
- Timeline
  - Not time sensitive: testing before release
  - Time sensitive: incident performance response (Software Reliability Engineering - SRE)

Understanding System's Performance

- Methodologies
- Instrumentation Tools
  - Observability Tools and Profiling
  - Debugging Tools
- Benchmarking

Anti-methodologies

- The lack of a deliberate methodology…
  - Tune things too early
- Street Light Anti-Method
  - Look for obvious issues in tools that are
    - Familiar
    - Found on the Internet
    - Found at random
- Drunk Man Anti-Method
  - Tune things at random until the problem goes away
  - Tune the wrong software (OS instead of application)

Problem Statement Method

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

Methodologies

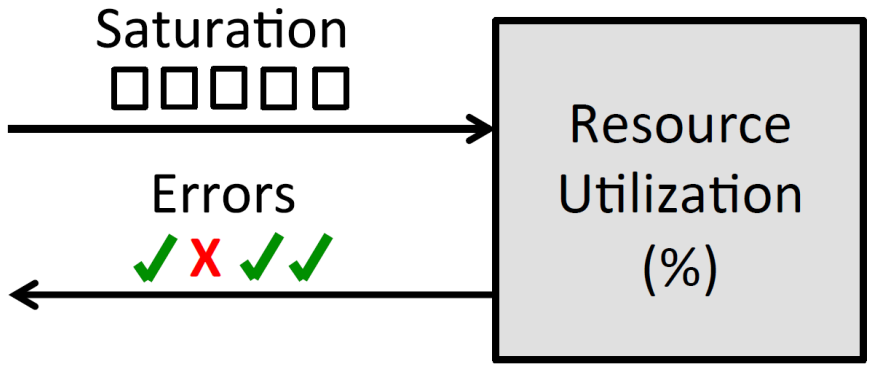
- Performance analysis in 60 seconds
- USE method
- CPU Profile Method
- Resource Analysis
- Others
  - Drill-down analysis
  - Off-CPU analysis
  - Static performance tuning
  - …

Performance Analysis in 60 Sec (Linux)

Load averages	1. uptime
Kernel errors	2. dmesg | tail
Overall stats by time	3. vmstat 1
CPU balance	4. mpstat -P ALL 1
Process usage	5. pidstat 1
Disk I/O	6. iostat -xz 1
Memory usage	7. free -m
Network I/O	8. sar -n DEV 1
TCP stats	9. sar -n TCP,ETCP 1
Check overview	10. top

USE Method

- For every resource (CPU, memory, bus, etc.), check
  - **U**tilization: busy time
  - **S**aturation: queue length or queued time
  - **E**rrors: 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

Instrumentation Tools

- Modify the source code, executable or runtime environment to understand the performance
  - Manual: Performed by the programmer
  - Automatic source level: instrumentation added to the source code by an automatic tool
  - Intermediate language/Compiler assisted: instrumentation added to assembly or decompiled bytecodes
  - Binary translation: adds instrumentation to a compiled executable.
  - Runtime instrumentation: program run is fully supervised and controlled by the tool.
  - Runtime injection: code is modified at runtime to have jumps to helper functions.

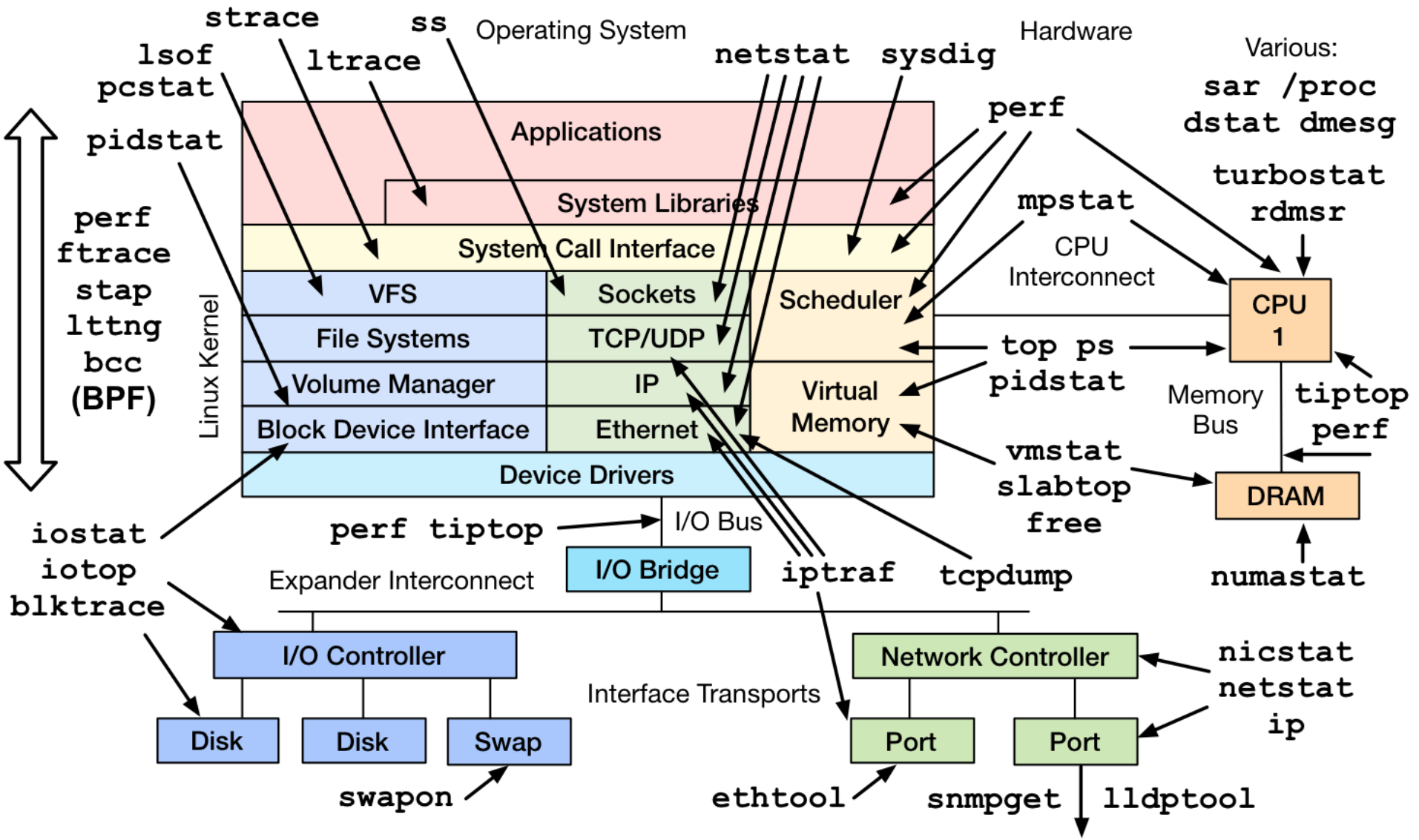
Overhead

- The execution is slowed down when instrumentation is added
- Ideally, minimal impact is desired
  - Types of tools
- Time measurements might not be accurate, but other measurements should be useful
  - Finding bottlenecks
  - Debugging

Types of Tools

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

Linux Performance Observability Tools



http://www.brendangregg.com/linuxperf.html 2016

Profiling

- Objectives:
  - Profile CPU usage by stack sampling
  - Generate CPU flame graphs
- Use perf/perf_events
- Other tools:
  - Gprof
  - Vtune

Perf

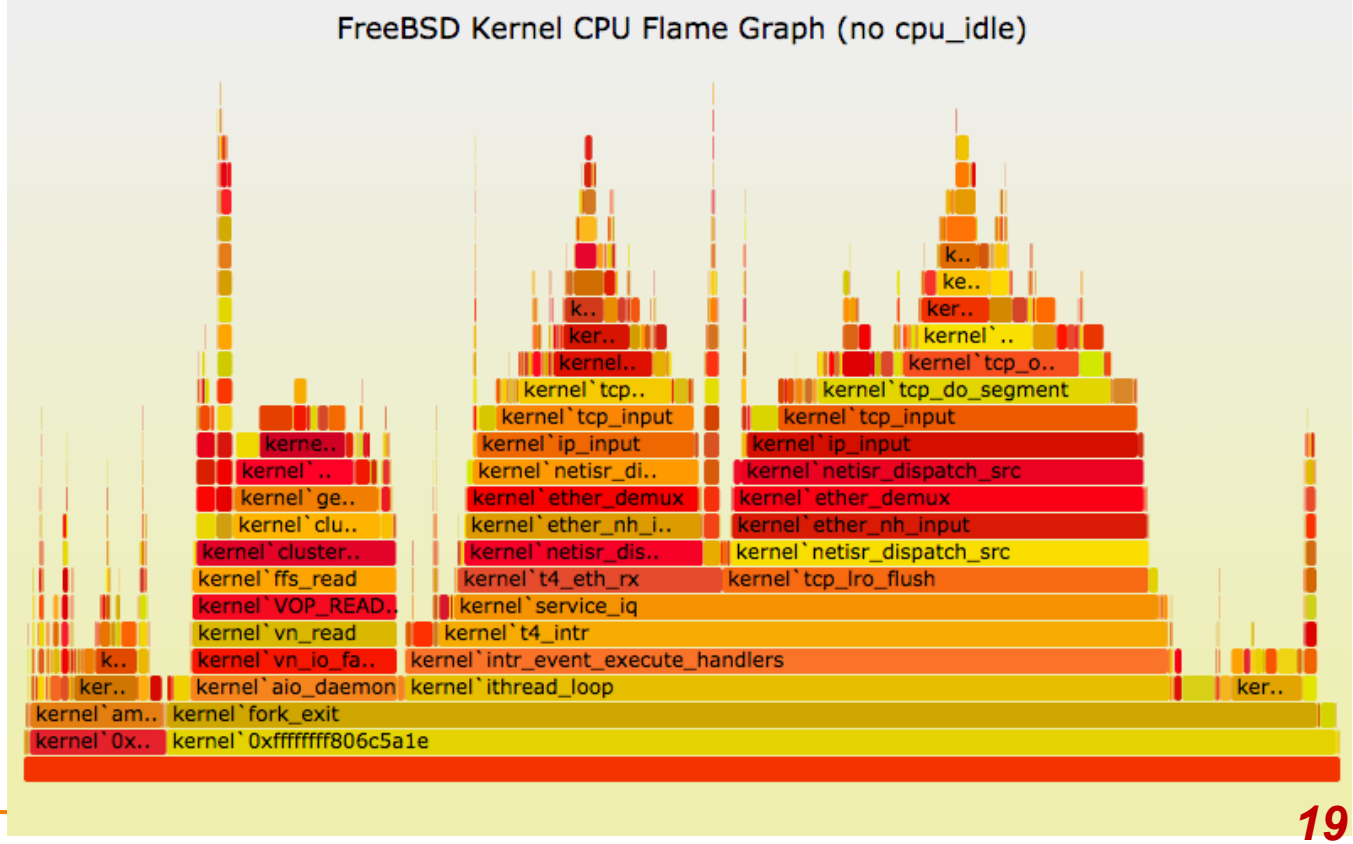
- Modern architectures expose performance counters
  - Cache misses, branch mispredicts, IPC, etc
- Perf tool provides easy access to these counters
  - perf list – list counters available on the system
  - perf stat – count the total events
  - perf record – profile using one event
  - perf report – Browse results of perf record

perf_events

- Provides the "perf" command
- In Linux source code: tools/perf
  - Usually pkg added by linux-tools-common, etc.
- **Multi-tool** with many capabilities
  - CPU profiling
  - Cache profiling
  - Static & dynamic tracing

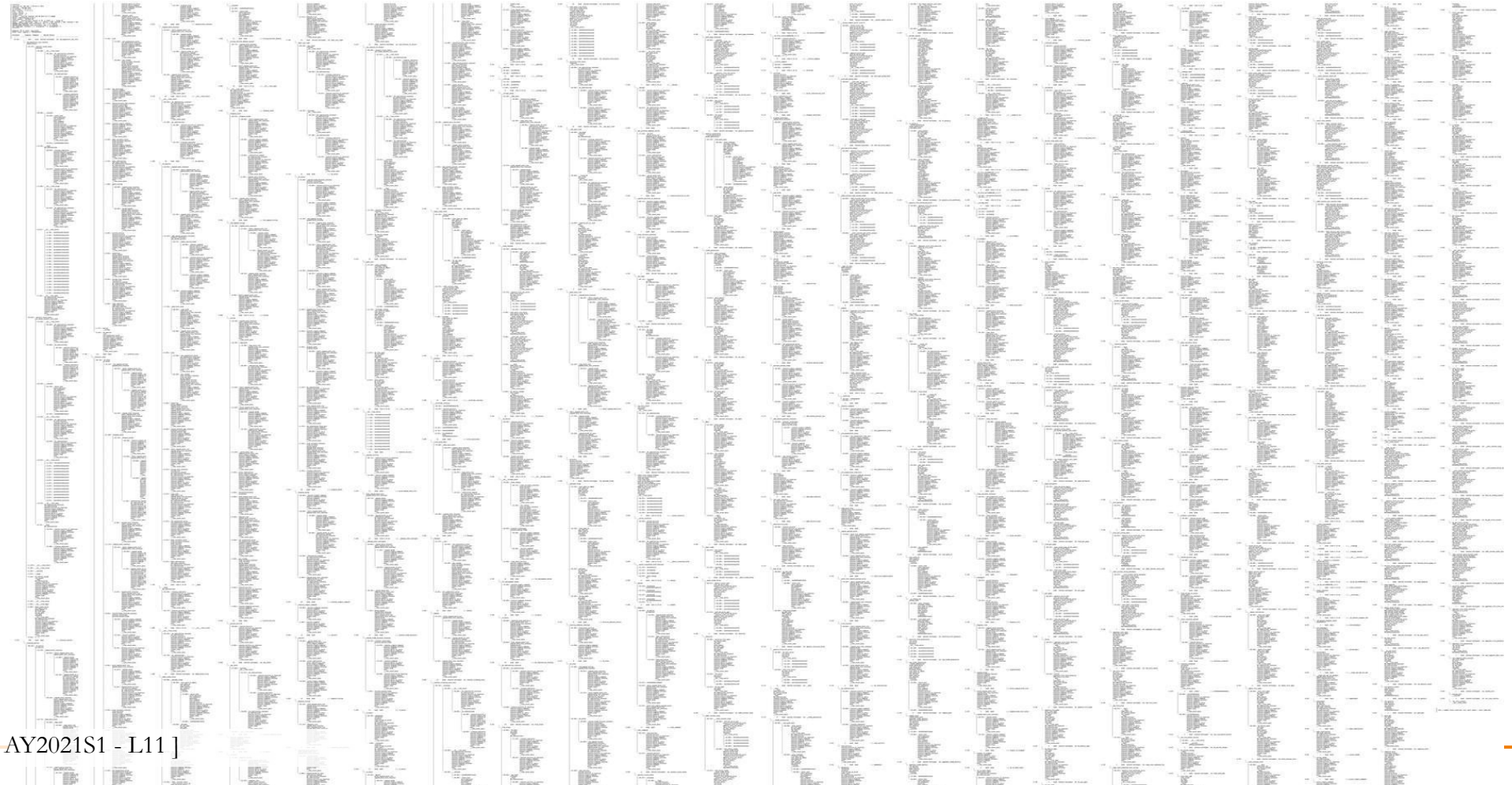
CPU Profile Method

- Understand all software in CPU profile > 1%
- Discovers a wide range of issues by CPU usage
  - Directly: CPU consumers
  - Indirectly: initialization of I/O, locks, times, ...
- Narrows down software to study
- Flame graphs →



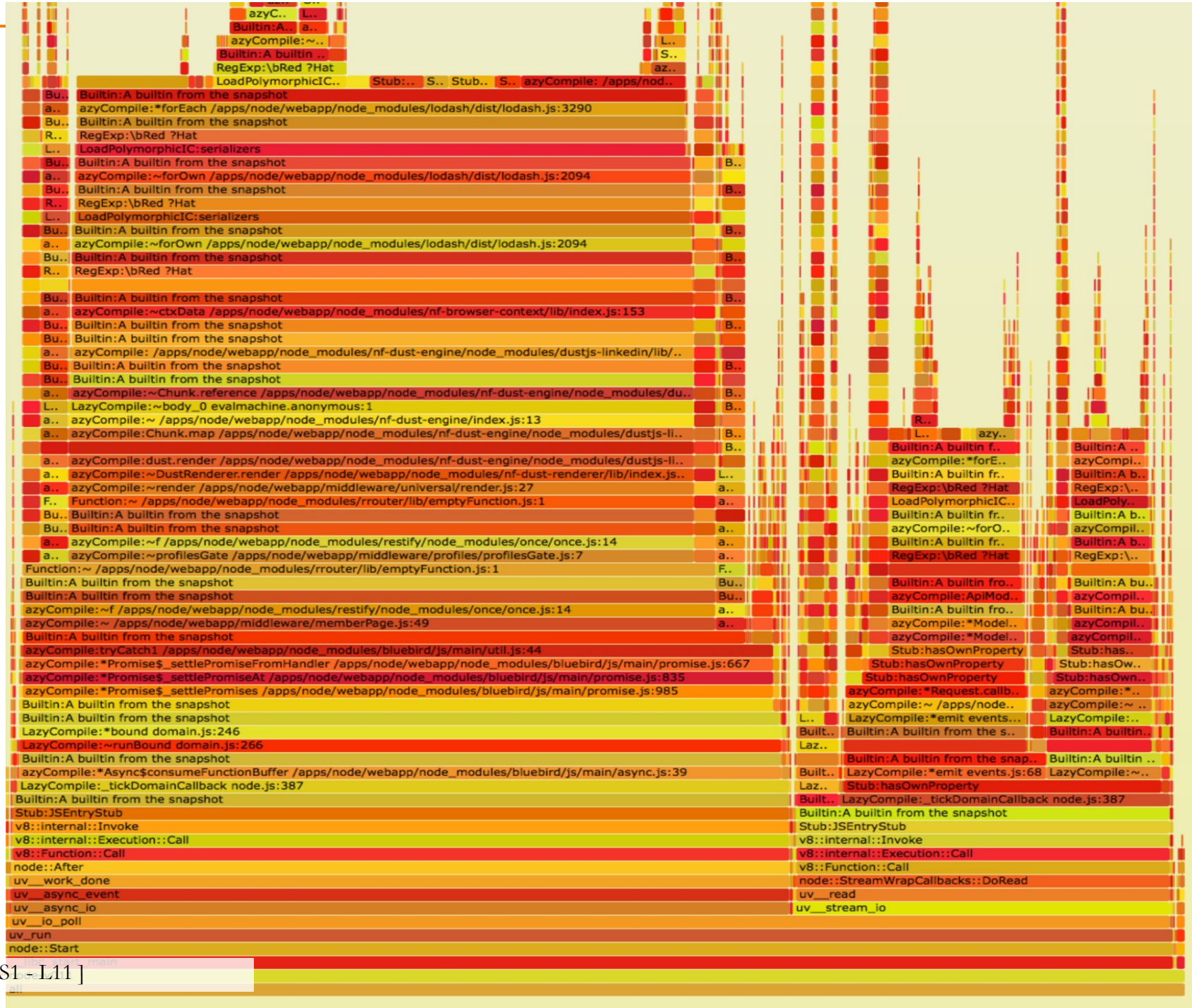
perf_events: Full "Report" Output

- `perf record -F 99 -p 13204 -g -- sleep 30`
- `perf report -n --stdio`

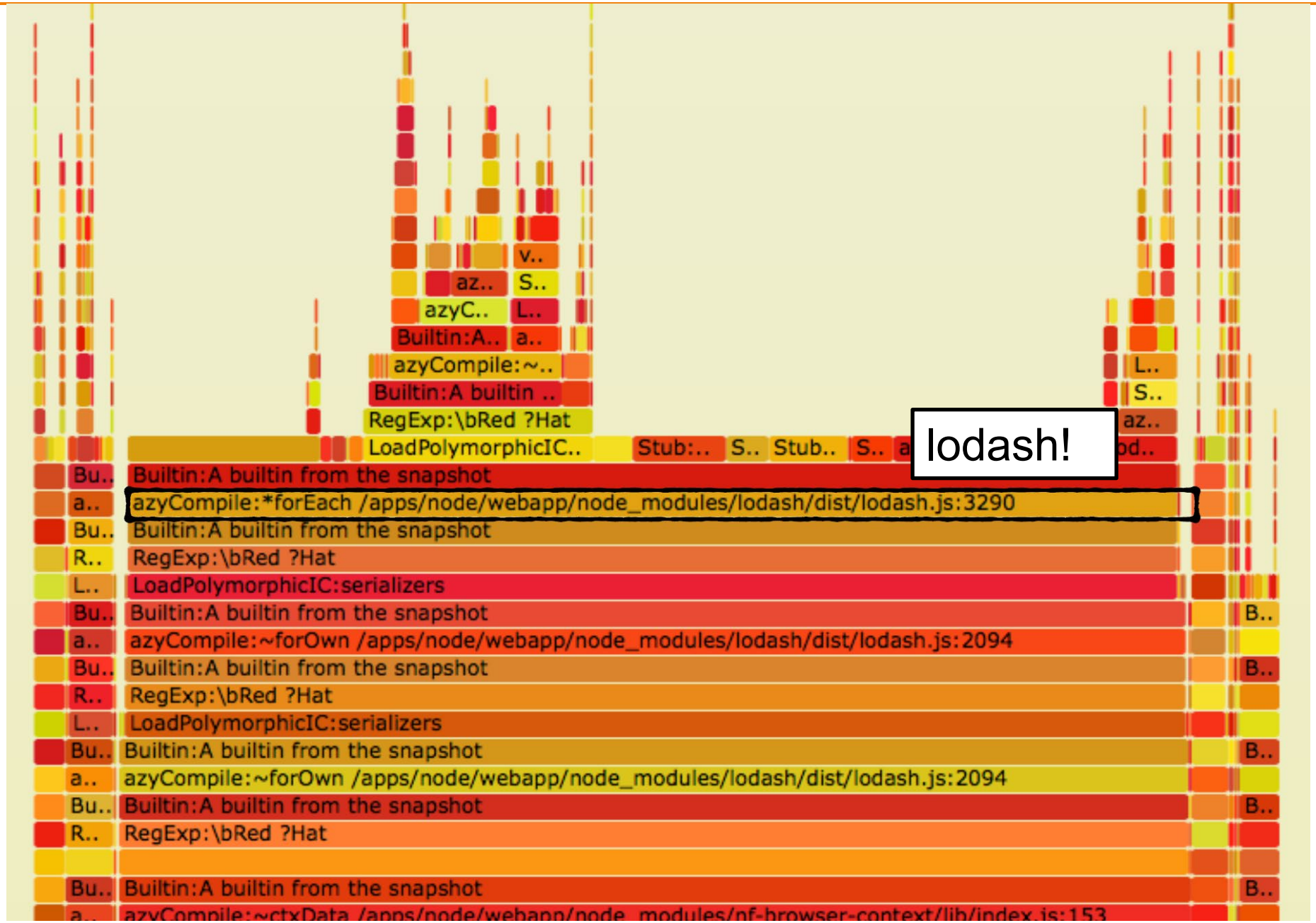


Profiling with Flame Graphs

- Each box presents a function in the stack (stack frame)
- x-axis: percent of time on CPU
- y-axis: stack depth
- colors: random, or can be a dimension


azyC.. L..
Builtin:A.. a..
azyCompile:~..
Builtin:A builtin ..
RegExp:\bRed ?Hat
LoadPolymorphicIC.. Stub:.. S.. Stub.. S.. azyCompile: /apps/nod..
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:*forEach /apps/node/webapp/node_modules/lodash/dist/lodash.js:3290
Bu.. Builtin:A builtin from the snapshot
R.. RegExp:\bRed ?Hat
L.. LoadPolymorphicIC:serializers
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~forOwn /apps/node/webapp/node_modules/lodash/dist/lodash.js:2094
Bu.. Builtin:A builtin from the snapshot
R.. RegExp:\bRed ?Hat
L.. LoadPolymorphicIC:serializers
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~forOwn /apps/node/webapp/node_modules/lodash/dist/lodash.js:2094
Bu.. Builtin:A builtin from the snapshot
R.. RegExp:\bRed ?Hat
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~ctxData /apps/node/webapp/node_modules/nf-browser-context/lib/index.js:153
Bu.. Builtin:A builtin from the snapshot
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile: /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-linkedin/lib/..
Bu.. Builtin:A builtin from the snapshot
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~Chunk.reference /apps/node/webapp/node_modules/nf-dust-engine/node_modules/du..
L.. LazyCompile:~body_0 evalmachine.anonymous:1
a.. azyCompile:~ /apps/node/webapp/node_modules/nf-dust-engine/index.js:13
a.. azyCompile:Chunk.map /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-li..
a.. azyCompile:dust.render /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-li..
a.. azyCompile:~DustRenderer.render /apps/node/webapp/node_modules/nf-dust-renderer/lib/index.js..
a.. azyCompile:~render /apps/node/webapp/middleware/universal/render.js:27
F.. Function:~ /apps/node/webapp/node_modules/rrouter/lib/emptyFunction.js:1
Bu.. Builtin:A builtin from the snapshot
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~f /apps/node/webapp/node_modules/restify/node_modules/once/once.js:14
a.. azyCompile:~profilesGate /apps/node/webapp/middleware/profiles/profilesGate.js:7
Function:~ /apps/node/webapp/node_modules/rrouter/lib/emptyFunction.js:1
Builtin:A builtin from the snapshot
Builtin:A builtin from the snapshot
azyCompile:~f /apps/node/webapp/node_modules/restify/node_modules/once/once.js:14
azyCompile:~ /apps/node/webapp/middleware/memberPage.js:49
Builtin:A builtin from the snapshot
azyCompile:tryCatch1 /apps/node/webapp/node_modules/bluebird/js/main/util.js:44
azyCompile:*Promise$_settlePromiseFromHandler /apps/node/webapp/node_modules/bluebird/js/main/promise.js:667
azyCompile:*Promise$_settlePromiseAt /apps/node/webapp/node_modules/bluebird/js/main/promise.js:835
azyCompile:*Promise$_settlePromises /apps/node/webapp/node_modules/bluebird/js/main/promise.js:985
Builtin:A builtin from the snapshot
Builtin:A builtin from the snapshot
LazyCompile:*bound domain.js:246
LazyCompile:~runBound domain.js:266
Builtin:A builtin from the snapshot
azyCompile:*Async$consumeFunctionBuffer /apps/node/webapp/node_modules/bluebird/js/main/async.js:39
LazyCompile:_tickDomainCallback node.js:387
Builtin:A builtin from the snapshot
Stub:JSEntryStub
v8::internal::Invoke
v8::internal::Execution::Call
v8::Function::Call
node::After
uv__work_done
uv__async_event
uv__async_io
uv__io_poll
uv_run
node::Start
Builtin:A builtin f..
azyCompile:*forE..
Builtin:A builtin fr..
RegExp:\bRed ?Hat
LoadPolymorphicIC..
Builtin:A builtin fr..
azyCompile:~forO..
Builtin:A builtin fr..
RegExp:\bRed ?Hat
Builtin:A builtin fro..
azyCompile:ApiMod..
Builtin:A builtin fro..
azyCompile:*Model..
azyCompile:*Model..
Stub:hasOwnProperty
Stub:hasOwnProperty
azyCompile:*Request.callb..
azyCompile:~ /apps/node..
LazyCompile:*emit events...
Builtin:A builtin from the s..
Builtin:A builtin from the snap..
Builtin:A builtin from the snapshot
Stub:JSEntryStub
v8::internal::Invoke
v8::internal::Execution::Call
v8::Function::Call
node::StreamWrapCallbacks::DoRead
uv__read
uv__stream_io
Builtin:A bu..
azyCompil..
Builtin:A bu..
azyCompil..
azyCompil..
Stub:has..
Stub:hasOw..
Stub:hasOwn..
azyCompile:*..
azyCompile:~ ..
LazyCompile:..
Builtin:A builtin..
Builtin:A builtin ..
LazyCompile:~..
LazyCompile:*emit events.js:68
Stub:hasOwnProperty
LazyCompile:_tickDomainCallback node.js:387
Builtin:A builtin from the snapshot

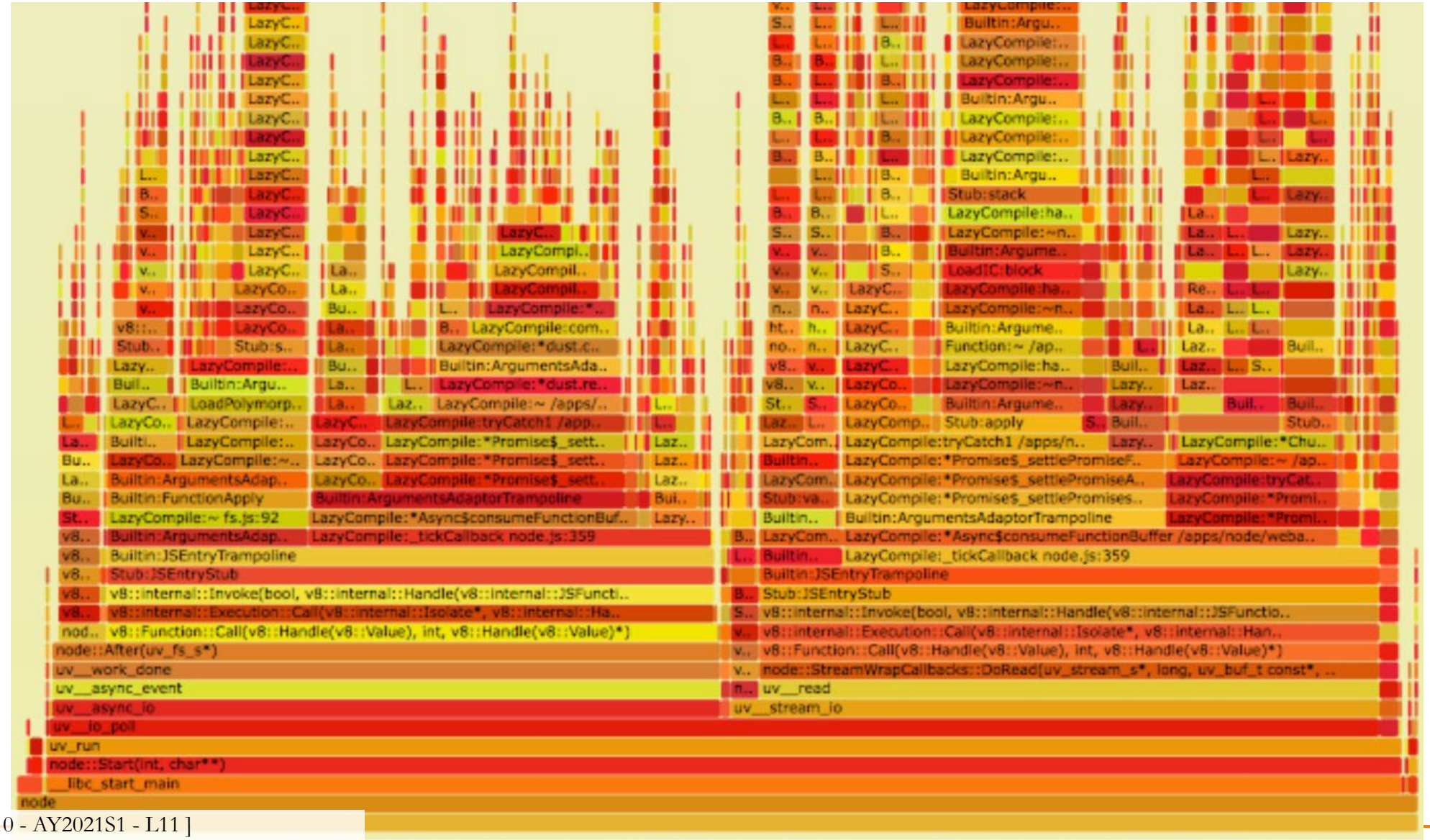

R.. RegExp:\bRed ?Hat
L.. LoadPolymorphicIC:serializers
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~forOwn /apps/node/webapp/node_modules/lodash/dist/lodash.js:2094
Bu.. Builtin:A builtin from the snapshot
R.. RegExp:\bRed ?Hat
L.. LoadPolymorphicIC:serializers
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~forOwn /apps/node/webapp/node_modules/lodash/dist/lodash.js:2094
Bu.. Builtin:A builtin from the snapshot
R.. RegExp:\bRed ?Hat
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~ctxData /apps/node/webapp/node_modules/nf-browser-context/lib/index.js:153
Bu.. Builtin:A builtin from the snapshot
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile: /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-linkedin/lib/..
Bu.. Builtin:A builtin from the snapshot
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~Chunk.reference /apps/node/webapp/node_modules/nf-dust-engine/node_modules/du..
L.. LazyCompile:~body_0 evalmachine.anonymous:1
a.. azyCompile:~ /apps/node/webapp/node_modules/nf-dust-engine/index.js:13
a.. azyCompile:Chunk.map /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-li..
a.. azyCompile:dust.render /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-li..
a.. azyCompile:~DustRenderer.render /apps/node/webapp/node_modules/nf-dust-renderer/lib/index.js..
a.. azyCompile:~render /apps/node/webapp/middleware/universal/render.js:27
F.. Function:~ /apps/node/webapp/node_modules/rrouter/lib/emptyFunction.js:1
Bu.. Builtin:A builtin from the snapshot
Bu.. Builtin:A builtin from the snapshot
a.. azyCompile:~profilesGate /apps/node/webapp/middleware/profiles/profilesGate.js:7
Function:~ /apps/node/webapp/node_modules/rrouter/lib/emptyFunction.js:1
Builtin:A builtin from the snapshot
azyCompile:~f /apps/node/webapp/node_modules/restify/node_modules/once/once.js:14
> 50% CPU time


lodash!
Builtin:A builtin from the snapshot
azyCompile:*forEach /apps/node/webapp/node_modules/lodash/dist/lodash.js:3290
Builtin:A builtin from the snapshot
RegExp:\bRed ?Hat
LoadPolymorphicIC:serializers
Builtin:A builtin from the snapshot
azyCompile:~forOwn /apps/node/webapp/node_modules/lodash/dist/lodash.js:2094
Builtin:A builtin from the snapshot
RegExp:\bRed ?Hat
LoadPolymorphicIC:serializers
Builtin:A builtin from the snapshot
azyCompile:~forOwn /apps/node/webapp/node_modules/lodash/dist/lodash.js:2094
Builtin:A builtin from the snapshot
RegExp:\bRed ?Hat
Builtin:A builtin from the snapshot

Before


Builtin:A builtin from the snapshot
azyCompile:~ctxData /apps/node/webapp/node_modules/nf-browser-context/lib/index.js:153
azyCompile: /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-linkedin/lib/..
azyCompile:~Chunk.reference /apps/node/webapp/node_modules/nf-dust-engine/node_modules/du..
LazyCompile:~body_0 evalmachine.anonymous:1
azyCompile:~ /apps/node/webapp/node_modules/nf-dust-engine/index.js:13
azyCompile:Chunk.map /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-li..
azyCompile:dust.render /apps/node/webapp/node_modules/nf-dust-engine/node_modules/dustjs-li..
azyCompile:~DustRenderer.render /apps/node/webapp/node_modules/nf-dust-renderer/lib/index.js..
azyCompile:~render /apps/node/webapp/middleware/universal/render.js:27
Function:~ /apps/node/webapp/node_modules/rrouter/lib/emptyFunction.js:1
azyCompile:~f /apps/node/webapp/node_modules/restify/node_modules/once/once.js:14
azyCompile:~profilesGate /apps/node/webapp/middleware/profiles/profilesGate.js:7
azyCompile:~ /apps/node/webapp/middleware/memberPage.js:49
azyCompile:tryCatch1 /apps/node/webapp/node_modules/bluebird/js/main/util.js:44
azyCompile:*Promise$_settlePromiseFromHandler /apps/node/webapp/node_modules/bluebird/js/main/promise.js:667
azyCompile:*Promise$_settlePromiseAt /apps/node/webapp/node_modules/bluebird/js/main/promise.js:835
azyCompile:*Promise$_settlePromises /apps/node/webapp/node_modules/bluebird/js/main/promise.js:985
LazyCompile:*bound domain.js:246
LazyCompile:~runBound domain.js:266
azyCompile:*Async$consumeFunctionBuffer /apps/node/webapp/node_modules/bluebird/js/main/async.js:39
LazyCompile:_tickDomainCallback node.js:387
Stub:JSEntryStub
v8::internal::Invoke
v8::internal::Execution::Call
v8::Function::Call
node::After
uv__work_done
uv__async_event
uv__async_io
uv__io_poll
uv_run
node::Start
__libc_start_main
Builtin:A builtin f..
azyCompile:*forE..
Builtin:A builtin fr..
RegExp:\bRed ?Hat
LoadPolymorphicIC..
azyCompile:~forO..
azyCompile:ApiMod..
azyCompile:*Model..
Stub:hasOwnProperty
azyCompile:*Request.callb..
azyCompile:~ /apps/node..
LazyCompile:*emit events...
Builtin:A builtin from the s..
Builtin:A builtin from the snap..
LazyCompile:*emit events.js:68
Stub:hasOwnProperty
LazyCompile:_tickDomainCallback node.js:387
Builtin:A builtin from the snapshot
Stub:JSEntryStub
v8::internal::Invoke
v8::internal::Execution::Call
v8::Function::Call
node::StreamWrapCallbacks::DoRead
uv__read
uv__stream_io

After


node::After(uv_fs_s*)
uv__work_done
uv__async_event
uv__async_io
uv__io_poll
uv_run
node::Start(int, char**)
__libc_start_main
node
uv__read
uv__stream_io

Debugging Tools

- Help in identifying bugs
  - Valgrind
    - Heavy-weight binary instrumentation: >4x overhead
    - Designed to shadow all program values: registers and memory
      - Shadowing requires serializing threads
    - Usually used for memcheck
  - Sanitizers
    - Compilation-based approach

Valgrind memcheck

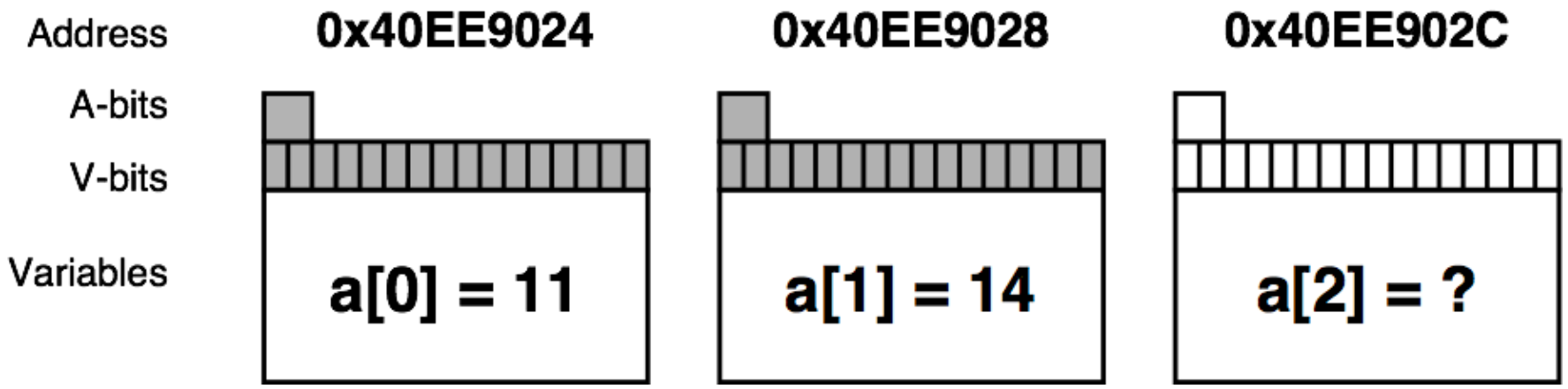
- Validates memory operations in a program
  - Each allocation is freed only once
  - Each access is to a currently allocated space
  - All reads are to locations already written
  - 10 – 20x overhead

```
valgrind --tool=memcheck <prog …>
```

```
...
==29991== HEAP SUMMARY:
==29991==     in use at exit: 2,694,466,576 bytes in 2,596 blocks
==29991==   total heap usage: 16,106 allocs, 13,510 frees, 3,001,172,305 bytes allocated
==29991==
==29991== LEAK SUMMARY:
==29991==    definitely lost: 112 bytes in 1 blocks
==29991==    indirectly lost: 0 bytes in 0 blocks
==29991==      possibly lost: 7,340,200 bytes in 7 blocks
==29991==    still reachable: 2,687,126,264 bytes in 2,588 blocks
==29991==         suppressed: 0 bytes in 0 blocks
```

Valgrind – How to?

- Shadow memory
  - Used to track and store information on the memory that is used by a program during its execution.
  - Used to detect and report incorrect accesses of memory



A-bit: corresponding byte accessible

V-bit: corresponding bit initialized

Sanitizers

- Compilation-based approach to detect issues
  - GCC and LLVM support
  - ~2x overhead
  - Add “-fsanitize=address”
- Examples:
  - ThreadSanitizer (e.g., data races)
  - MemorySanitizer (e.g., uninitialized reads)
  - UndefinedBehaviorSanitizer (e.g., Integer Overflow, Null Pointer)
  - LeakSanitizer (e.g., memory leaks)

-fsanitize=address Example

```
$ cat test.c
int main(int argc, char **argv) {
    int a[2] = {11, 14};
    return a[2];
}
$ gcc -fsanitize=address -o test test.c
$ ./test
=================================================================
==14504== ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7fff578920e8 at pc
0x400844 bp 0x7fff578920b0 sp 0x7fff578920a8
READ of size 4 at 0x7fff578920e8 thread T0
    #0 0x400843 (/home/ubuntu/test+0x400843)
    #1 0x7fbfceb57ec4 (/lib/x86_64-linux-gnu/libc-2.19.so+0x21ec4)
    #2 0x400688 (/home/ubuntu/test+0x400688)
Address 0x7fff578920e8 is located at offset 40 in frame <main> of T0's stack:
  This frame has 1 object(s):
    [32, 40) 'a'
Shadow bytes around the buggy address:
  0x10006af0a400: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x10006af0a410: 00 00 00 00 00 00 00 00 f1 f1 f1 f1 00[f4]f4 f4
  0x10006af0a420: f3 f3 f3 f3 00 00 00 00 00 00 00 00 00 00 00 00
```

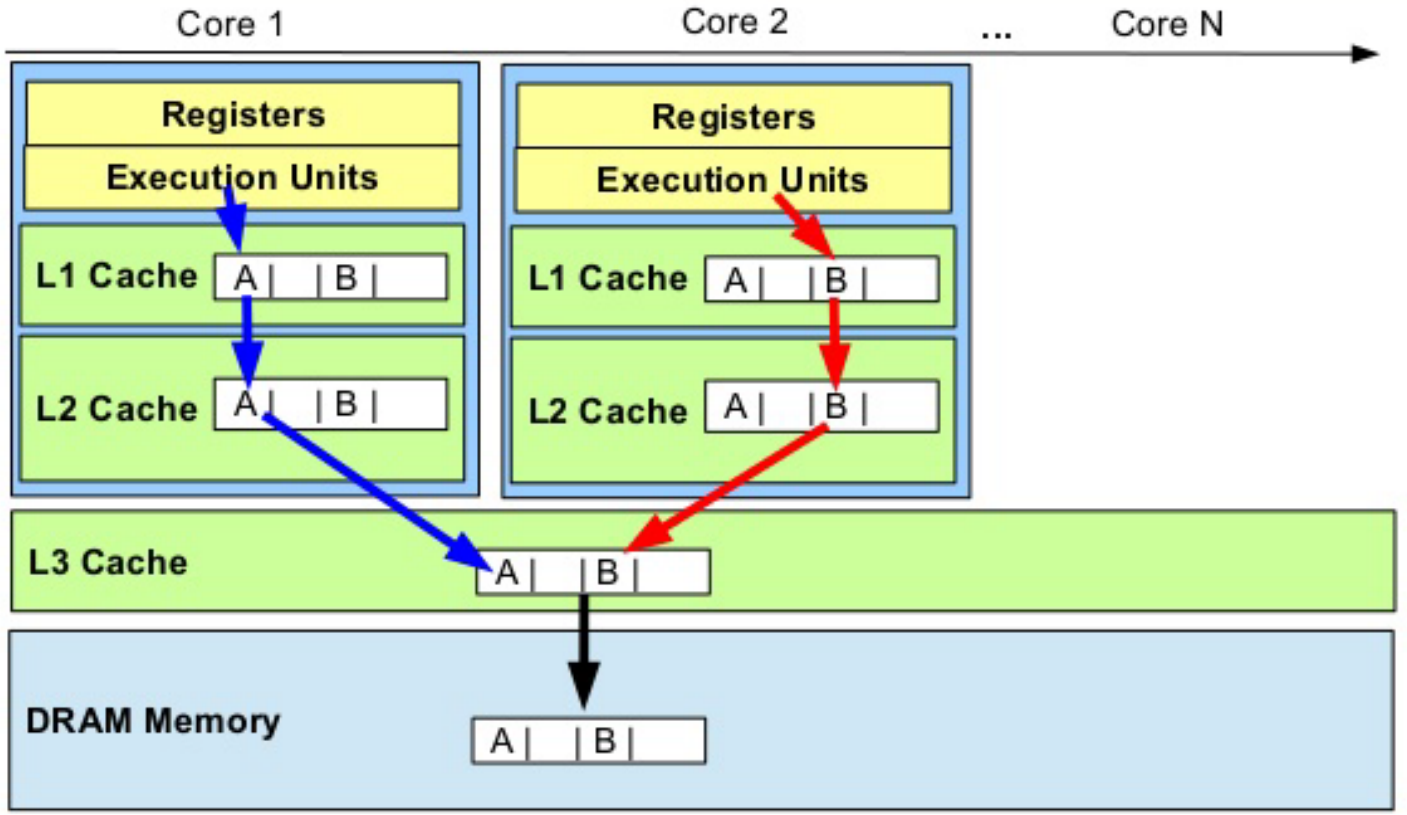
-fsanitize=thread Example

```
#include <pthread.h>
#include <stdio.h>
int global;
void *Thread1(void *x) {
  global++;
  return NULL;
}
void *Thread2(void *x) {
  global--;

  return NULL;
}
int main() {
  pthread_t t[2];
  pthread_create(&t[0], NULL, Thread1, NULL);
  pthread_create(&t[1], NULL, Thread2, NULL);
  pthread_join(t[0], NULL);
  pthread_join(t[1], NULL);
}
```

```
==================
WARNING: ThreadSanitizer: data race (pid=17631)
  Read of size 4 at 0x562a4b132014 by thread T2:
    #0 Thread2 <null> (thread_san+0xa42)
    #1 <null> <null> (libtsan.so.0+0x296ad)
  Previous write of size 4 at 0x562a4b132014 by
thread T1:
    #0 Thread1 <null> (thread_san+0xa03)
    #1 <null> <null> (libtsan.so.0+0x296ad)
  Location is global 'global' of size 4 at
0x562a4b132014 (thread_san+0x000000202014)
  Thread T2 (tid=17635, running) created by main
thread at:
    #0 pthread_create <null> (libtsan.so.0+0x2bcee)
    #1 main <null> (thread_san+0xad3)
  Thread T1 (tid=17634, finished) created by main
thread at:
    #0 pthread_create <null> (libtsan.so.0+0x2bcee)
    #1 main <null> (thread_san+0xab2)
SUMMARY: ThreadSanitizer: data race
(/home/ccris/GitRepos/CS3210-AY2021-
S1/Lectures/perf_inst/thread_san+0xa42) in Thread2
==================
ThreadSanitizer: reported 1 warnings
```

Problem: False Sharing




```
struct foo {
    int x;
    int y;
};

static struct foo f;
```

```
int sum_a(void)
{
    int s = 0;
    for (int i = 0; i < 1000000; ++i)
        s += f.x;
    return s;
}
```

```
void inc_b(void)
{
    for (int i = 0; i < 1000000; ++i)
        ++f.y;
}
```

Identifying False Sharing using Perf

- Use `perf c2c`
- At a high level it shows:
  - The cachelines where false sharing was detected.
  - The readers and writers to those cachelines, and the offsets where those accesses occurred.
  - The pid, tid, instruction addr, function name, binary object name for those readers and writers.
  - The source file and line number for each reader and writer.
  - The average load latency for the loads to those cachelines.

Benchmark Tools

- ~100% of benchmarks are wrong
- Results are usually misleading
  - you benchmark A, but actually measure B, and conclude you measured C
- Common mistakes:
  - Testing the wrong target: eg, FS cache instead of disk
  - Choosing the wrong target: eg, disk instead of FS cache … doesn't resemble real world usage
  - Invalid results: bugs
- The energy needed to refute benchmarks is multiple orders of magnitude bigger than to run them

Benchmark Program: **Overview**

- It is **not easy** to evaluate and compare the performance between computer system
  - Our discussion focused only on processor and memory performance
  - ➔ Much more complicated if we bring in other aspects of a system

- We will look at a few well known benchmarks

Benchmarks: Industry Standards

- **SPEC benchmark suites:**
  - **SPECint**, **SPECfp**: For processor + memory + compiler
  - **SPECjvm2008**: For Java performance
  - Many others

- **EEMBC benchmark suites**
  - **ANDBench**: For Android performance
  - **DPIBench**: For system and network performance
  - etc

- **Numerical Aerodynamic Simulation (NAS):**
  - From NASA
  - Massively parallel benchmark: For computer cluster

Benchmarks: Simple Benchmark

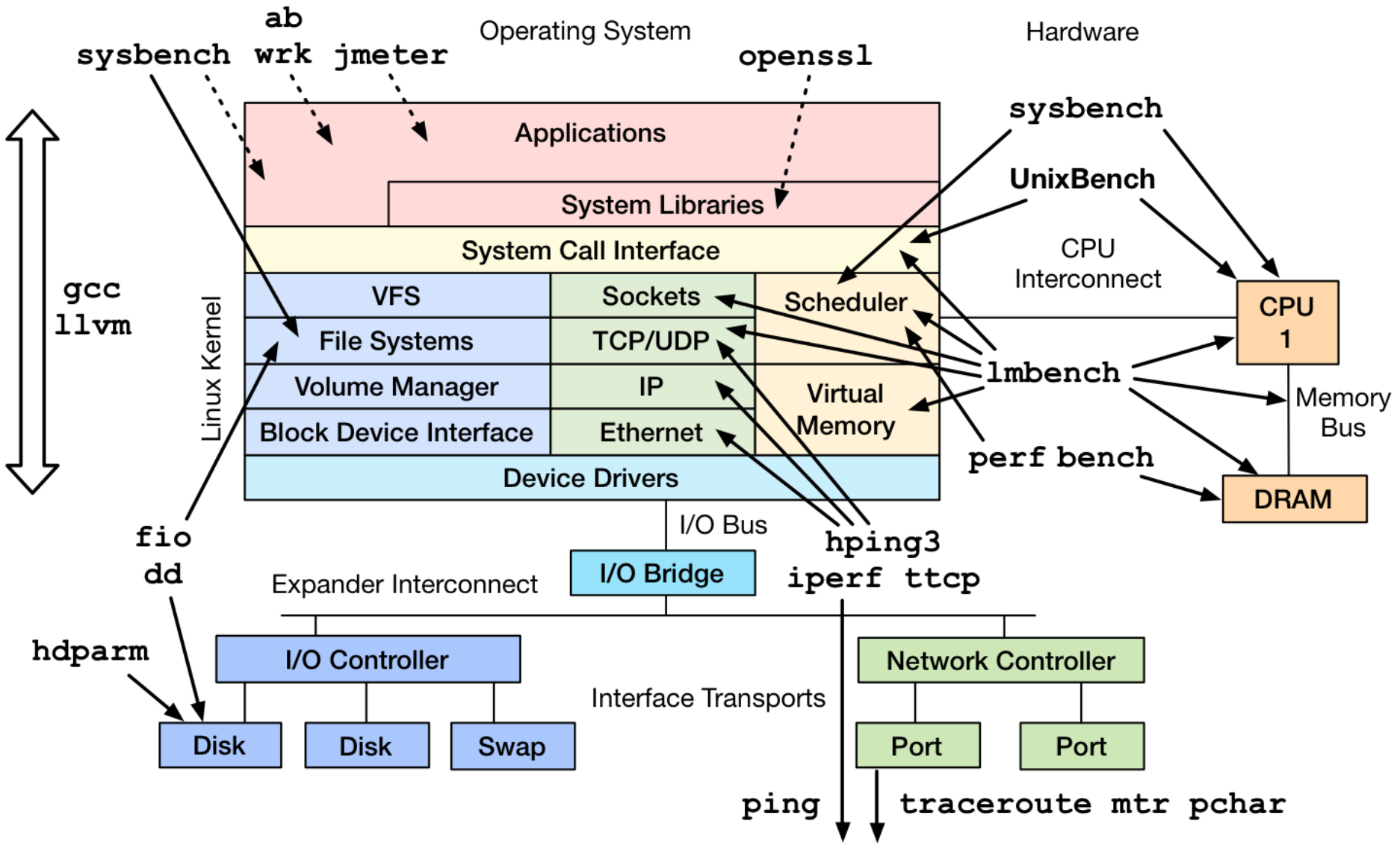
These benchmarks can be found easily on the web

- Linpack:
  - Linear Algebra Solver
  - Used in the SuperComputer ranking
- Dhrystone / Whetstone:
  - Small program to test integer / floating performance
- Tak Function:
  - To test recursion optimization

Active Benchmarking Method (Synthetic Performance Testing)

- Run the benchmark for hours
- While running, analyze and confirm the performance limiter using *observability tools*
  - Disk benchmark: run iostat, …
  - CPU benchmark: run pidstat, perf, flame graphs, …
- Answer the question: why isn't the result 10x better?

Linux Performance Benchmark Tools



http://www.brendangregg.com/linuxperf.html 2015

Summary

- Performance instrumentation needs understanding of resource utilization
  - Methodologies
  - Tools to observe performance
  - Benchmarks
- Profiling helps in avoiding performance problems

References

- Reading: Brendan Gregg, Netflix - http://www.brendangregg.com/linuxperf.html
- Further reading:
  - http://queue.acm.org/detail.cfm?id=1117403