challenges in

Google

Google-wide Profiling (`top` for Google)

from a random selection of machines and services, continuously collect:

- hardware counters
  - cycles, ins, br mispred, cache misses
- software profiles
  - heap, growth, lock contention, fragmentation

aggregate results, present top

- applications, libraries, functions

[1] http://research.google.com/pubs/pub36575.html

what works?

continuous testing

- tight integration with code, build system, compiler
- thread, memory checkers

loadtests (sometimes)

tracing

- kernel
- application

what doesn't work?

scale is a problem

- one in a million is commonplace
  - race conditions, memory leaks, overruns, hardware failures
- real applications are orders of magnitude larger than SPEC et. al.
- complexity of production behavior is hard to model
- contention is difficult to predict and plan for [2]
- testing matrix is combinatoric explosion

hard problems cross boundaries

datacenter applications employ many types of parallelism *and they are distributed*

- client, balancer, frontend, backend, storage
- at each hop
  - queues in kernel and userspace
  - contention with other jobs on machine
- a single query may touch hundreds of machines
- long tail latency, RPC fanout [1]

what would help?

really low-overhead tools, sub 3%

- especially for error detection, like [1]
- hardware support welcome