Profiling a warehouse-scale computer

Svilen Kanev
Juan Pablo Darago
Kim Hazelwood
Parthasarathy Ranganathan, Tipp Moseley
Gu-Yeon Wei, David Brooks

Harvard University
Universidad de Buenos Aires
Yahoo Labs
Google Inc.
Harvard University
The cloud is here to stay

[http://google.com/trends, 2015]
Warehouse-scale computers (of yore)

- Datacenters built around a few “killer workloads”
- Problem sizes >> 1 machine
- Distributed, but tightly interconnected services
- Communication through remote-procedure calls (RPCs)
Now “the datacenter is the computer” (the WSC model has caught on)

“microservice architecture”

thousands of services are “one RPC away”

“... about a hundred of services that comprise Siri’s backend...” [Apple, Mesos meetup 2015]

Did you mean: #pldi15

frequency[“#isca15”]++
How do modern WSC applications interact with hardware?

And what does that imply for future server processors?
Traditional profiling: load testing

Isolate a service

Find representative inputs

Find representative operating point

Profile / optimize

Repeat
Live datacenter-scale profiling

*(Google-wide profiling)*

Select random production machines  
~20,000 / day

Profile each one (for a while)  
without isolation  
while running live traffic  
for billions of users

Aggregate days, weeks, years worth of execution

[Ren et al. *Google-wide profiling*, 2010]
Live WSC profiling insights

Where are cycles spent in a datacenter?
Are there really no killer applications?
How do WSC applications interact with instruction caches?
How much ILP is there? Big / small cores?
DRAM latency vs. bandwidth?
Hyperthreading?
Where are WSC cycles spent?
No “killer” application to optimize for

Instead: a long tail of various different services

[1 week of sampled WSC cycles]
Ongoing application diversification

Optimizing hardware one-application-at-a-time has diminishing returns
Within applications: no hotspots

Corollary: hunting for per-application hotspots is not justified
Hotspots across applications: “datacenter tax”

Shared low-level routines; typical for larger-than-1-server problems
Hotspots across applications: “datacenter tax”

Only 6 self-contained routines account for ~30% of WSC cycles

Prime candidates for accelerators in server SoCs
Live WSC profiling insights

Where are cycles spent in a datacenter? Everywhere.

Are there really no killer applications? Datacenter tax.

How do WSC applications interact with instruction caches?

How much ILP is there? Big / small cores?

DRAM latency vs. bandwidth?

Hyperthreading?
Microarchitecture:
WSC i-cache pressure
Severe instruction cache bottlenecks

15-30% of core cycles wasted on instruction-supply stalls

20,000 Intel IvyBridge servers
2 days
Top-Down analysis [Yasin 2014]
Severe instruction cache bottlenecks

15-30% of core cycles wasted on instruction-supply stalls

Fetching instructions from L3 caches
Very high i-cache miss rates
10x the highest in SPEC
50% higher than CloudSuite

Lots of lukewarm code
100s MBs of instructions per binary; no hotspots
A problem in the making

I-cache working sets 4-5x larger than largest in SPEC

Growing almost 30% / year

significantly faster than i-caches

One solution: L2 i/d partitioning
Where are cycles spent in a datacenter? Everywhere.

Are there really no killer applications? Datacenter tax.

How do WSC applications interact with instruction caches? Poorly.

How much ILP is there? Big / small cores? Bimodal.

DRAM latency vs. bandwidth? Latency.

Hyperthreading? Yes.
To sum up

A growing number of programs cover “the world’s WSC cycles”. There is no “killer application”, and hand-optimizing each program is suboptimal.

Low-level routines (datacenter tax) are a surprisingly high fraction of cycles. Good candidates for accelerators in future server processors.

Common microarchitectural footprint: working sets too large for i-caches; many d-cache stalls; generally low IPC; bimodal ILP; low memory bandwidth utilization.