Motivating Proxy Research via Industrial Use Cases

IISWC 2020

Emily Shriver
Principal Engineer / Research Scientist
Intel Labs

Acknowledgments: Karthik Sankaranarayanan, Intel Labs

Outline

- Why?
- What is a good proxy?
- Future

Outline

- Why?
  - Microarchitecture and Design
  - Performance and Power Validation
  - Post-silicon Tuning and Validation
  - Proprietary Workloads
- What is a good proxy?
- Future

Microarchitecture & Design

- Pain Points:
  - Unable to capture the complexities of real workload applications
  - Why? Short traces to run on (slow) near cycle-accurate simulators, capture user-space only
- Desire
  - Representative proxies with runtime in hours (instead of days)
  - Incorporating OS effects
- Opportunities
  - Representative proxies that are orders of magnitude shorter
  - Capture OS effects run overnight on virtual prototyping and emulation pre-si systems

Comparing Runtime on Different Platforms	
Silicon	1 Minute
Emulation	11.8 days
Virtual Prototyping w/ Power/Perf	2.4 days

Desire hours

Assumptions:
- Silicon (3.4 GHz),
- Emulation (200 KHz),
- Virtual Prototyping (1 MHz)

Note: There is a wide variation in emulation speed; assume a lower bound here based on a large, complex system in emulation.

Performance and Power Validation

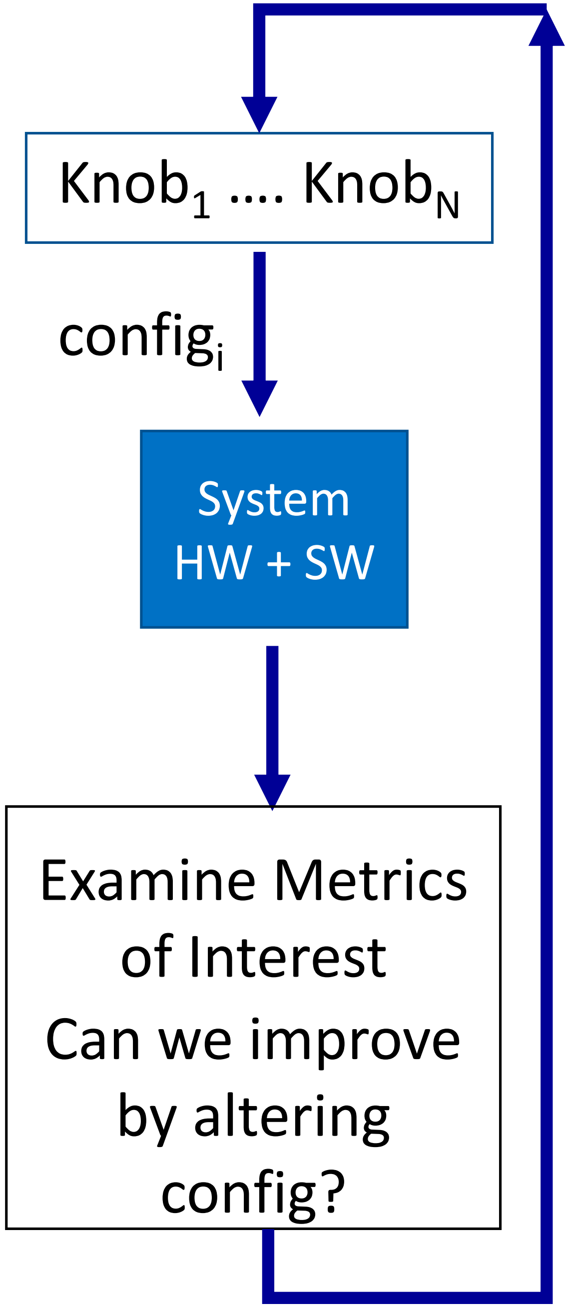
- Pain Points:
  - Pre-Si performance and power predictions do not match post-Si measurements
  - Hard to root-cause: very little content runs on both pre-Si and post-Si systems
  - Limited visibility on post-Si

- Desire:
  - Ability to run same workload on n-1 Silicon, pre-Si, and post-Si systems

- Opportunity:
  - Increase content: portable proxies that run on both pre-Si and post- Si
  - Increased visibility: on pre-Si platforms to facilitate post-Si validation

Post-Silicon: Tuning & Validation

- What:
  - Many HW & SW knobs to tune to optimize system (e.g. HW prefetchers)
- Pain Points:
  - Long running benchmarks makes each set of runs expensive (days)
- Desire:
  - Short runs with turnaround time in minutes instead of hours
  - Reduce tuning to hours instead of days
- Opportunity:
  - Representative proxies that are orders of magnitude shorter than original benchmark



Proprietary Workloads

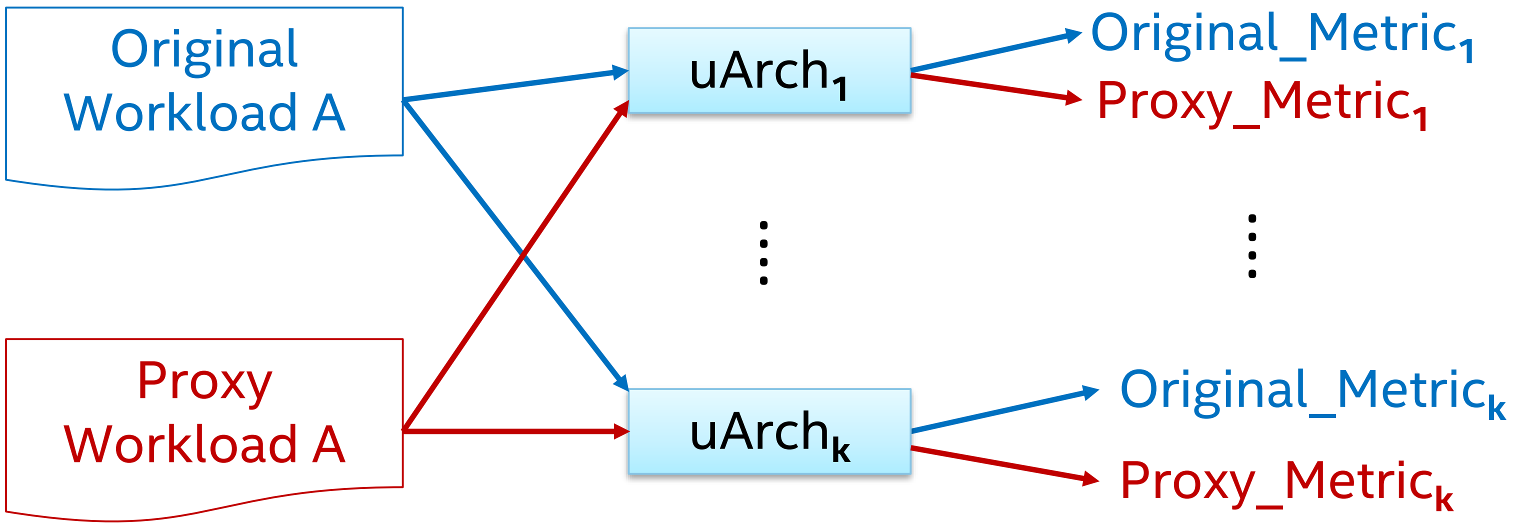
- What
  - Customers often do not share their workloads (their secret sauce)
- Pain Point
  - Design optimal HW & SW without workloads
- Desire
  - Ability to generate a workload with similar characteristics to customer workload
- Opportunity
  - Proxies enable customers to either 1) run the proxy generation code or 2) run the profiling code used as input to proxy generation code

Outline

- Why?
- What is a good proxy?
  - Micro-Architectural Independence
  - Capture over time phase behavior
  - Proxy ≈ Real ?
- Future

Micro-Architecture Independence

Portable across micro-architectures and uArch configurations



Metrics Include:

- Performance
- Power
- Cache behavior
- Branch predictor behavior, etc…

Use Cases:

- Generational comparisons
- Exploration of micro-architectural features, accelerators, software stack, etc…

Results: Micro-Architectural Independence

[1] Reena Panda, et.al. " Accurate Address Streams for LLC and Byeond (SLAB):..." ISPASS 2017

SPEC CPU 2017 Integer benchmarks (10 total)

- Proxies created using techniques in [1]
- Proxies created and manually tuned on Broadwell
- Proxies: 1000x – 7500x reduction in retired instructions

Compare performance (IPC) of proxy vs original benchmark on 2 different micro-architectures

- On Broadwell system 10/10 proxies <13% IPC error
- On Kabylake system 5/10 proxies <13% IPC error
  but 5/10 proxies 37-75% IPC error
  50% have high IPC error

Significant manual tuning to achieve the good Broadwell IPC results

More research needed to
1) achieve micro-architectural independence
2) improve automation

Observation of benchmarks with high error

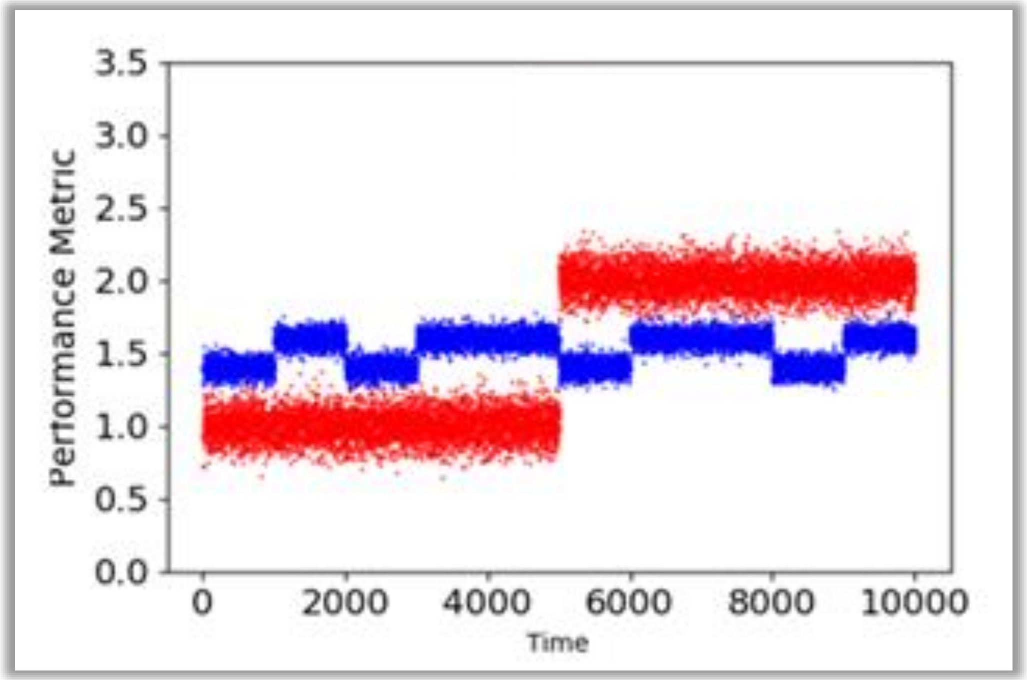
- memory bound
- branch

Overtime Phase Behavior

- Stress shared resources in the same manner as the original
  - memory accesses
  - shared data

- Modeling of stressing shared resources in presence of concurrency:
  - multi-process, multi-threaded, multi-accelerators, etc

- Energy Management



Two programs
same average behavior
different phase behavior

Is Proxy ≈ Real?

- Represent workload with a single metric comparison
  - average: APE, MAPE (mean absolute percent error)
  - peak: max
  - sum: area under the curve
  - Use Cases: compare different uArch, power delivery di/dt, energy

Well established techniques

- Behavior over time comparison
  - Represent workload with multiple numbers & calculate similarity[1]
  - Many use cases lend themselves to accurate phase behavior over time: Concurrency, power management

Less established techniques in uArch community

Similarity techniques for system analysis is an open area of research

[1] DTW – Dynamic Time Warping - Kalaba, "On adaptive control processes,"Automatic Control, IRE Transactions on, vol 4, 1-9, 1959

Outline

- Why?
- What is a good proxy?
- Future

Future – Scale to Systems

Current State of Proxies

- Proxies primarily single threaded, CPU-centric

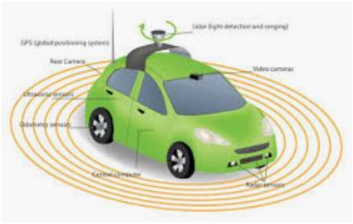
Emerging HW/SW

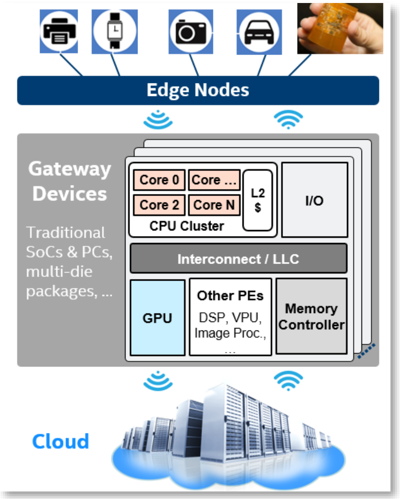
- High integration of accelerators and high concurrency
- Complex and rapidly changing software stacks

Ability to handle large benchmarks

- Emerging applications require longer runs for meaningful analysis (secs/mins instead of msecs)
- Example: cloud applications (e.g. search) with large code footprint take seconds to minutes to encounter meaningful performance bottlenecks [1][2]

Emerging applications: micro-services, autonomous driving, cloud computing, IoT, …





[1] Svilen Kanev, et.al. "Profiling a warehouse-scale computer." *ISCA 2015*
[2] Grant Ayers, et.al., "AsmDB: understanding and mitigating front-end stalls in warehouse-scale computers." *ISCA 2019*

Questions
/
Comments