Only Relative Speed Matters: Virtual Causal Profiling

Behnam Pourghassemi , Ardalan Amiri Sani , Aparna Chandramowlishwaran

University of California, Irvine

38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (Performance ’20)

Performance {Really} Matters!

- Business Revenue
- User satisfaction
- Research cost
- etc.

1 second lag in loading Amazon would **cost $1.6 billion** in sale

- Amazon

53% of mobile site visitors **leave** a page that takes **longer than 3 seconds** to load

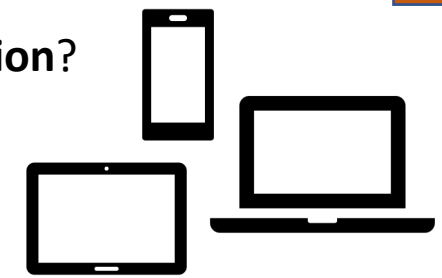
- Google

~1000$/hr for code runs on supercomputer/cluster

- ExtremeTech

Performance Analysis

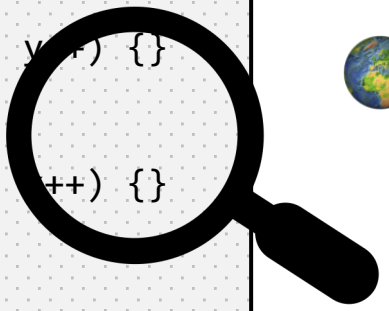
1. What are the **bottlenecks** or **critical spots** in the program?
2. How much performance improvement can we realistically achieve by optimizing these critical spots? A or B
3. How much is the measured performance gain related to the **system** and **configuration**?



What-if Analysis

What-if Analysis with Conventional Profilers

```
01  void A() { // ~5.9 seconds
02    for(volatile size_t x=0; x<2500000000; x++) {}
03  }
04  void B() { // ~5.4 seconds
05    for(volatile size_t y=0; y<2200000000; y++) {}
06  }
07  void C() { // ~5.1 seconds
08    for(volatile size_t x=0; x<2100000000; x++) {}
09  }
10 int main() {
11   thread A_thread(A), B_thread(B);
12   A_thread.join(); B_thread.join();
13   C();
14 }
```



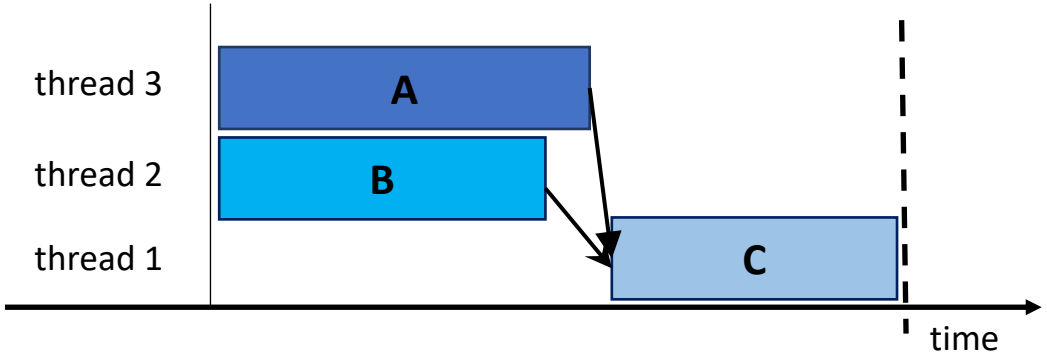
PROFILE

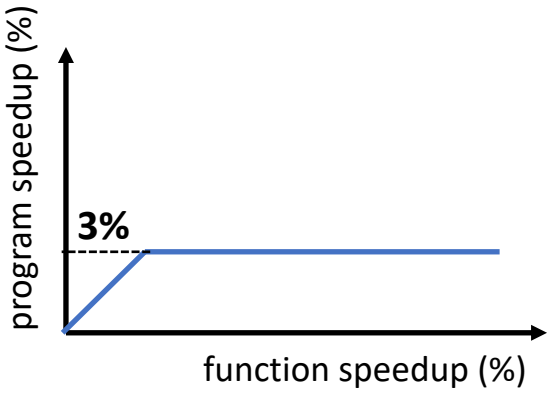
Perf

Valgrind

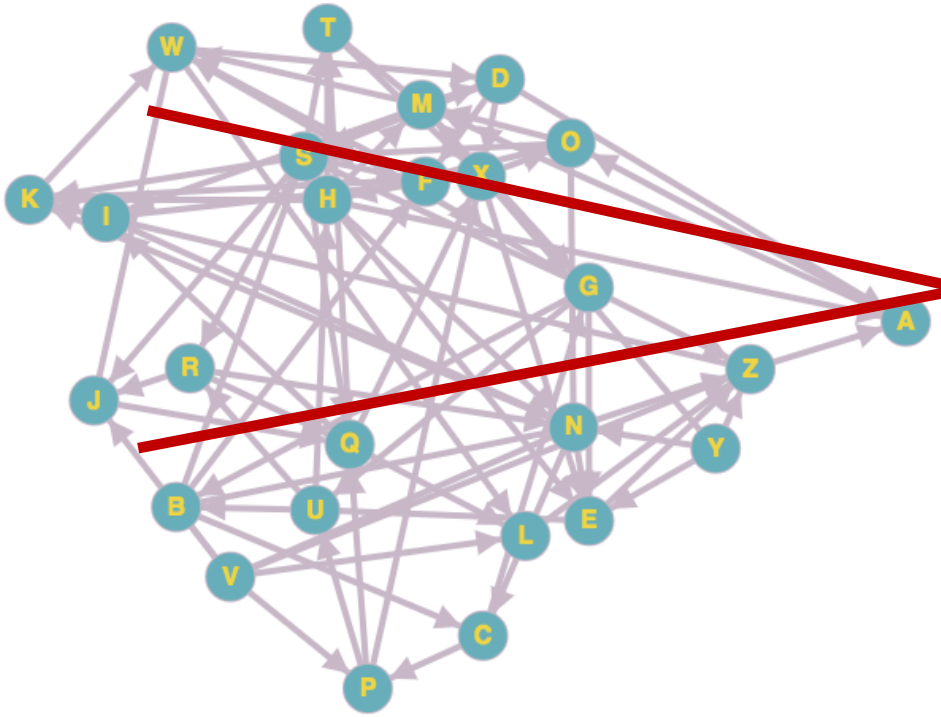
```
[~] gprof
                                  time (seconds)  calls  name
                                                  1      A()
32.91                             5.39          1      B()
31.19                             5.11          1      C()

% time   s                                      led    name
35.9     7                                             A()
-----------------------------------------------------------
32.9     5.89     0.00                         B()
-----------------------------------------------------------
31.2     5.89     0.00                         C()
```





Causal Profiling



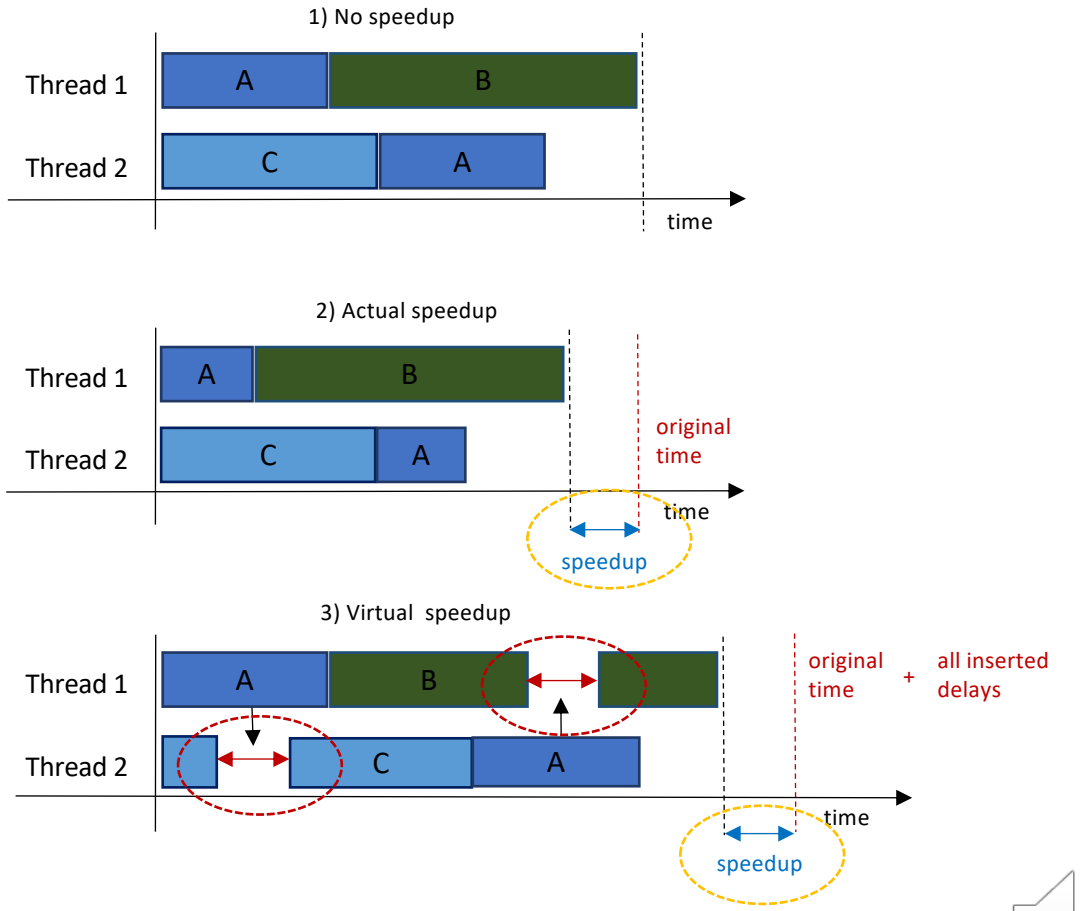
- Causal profiling determines *impact of optimization* in a line of code on the total execution time.
- Does not require dependency graph generation and subsequent graph processing.
- Dependencies and impact of optimization are captured at runtime.

Curtsinger et al., "Coz: Finding code that counts with causal profiling." *SOSP* 2015.

Causal Profiling (Methodology)

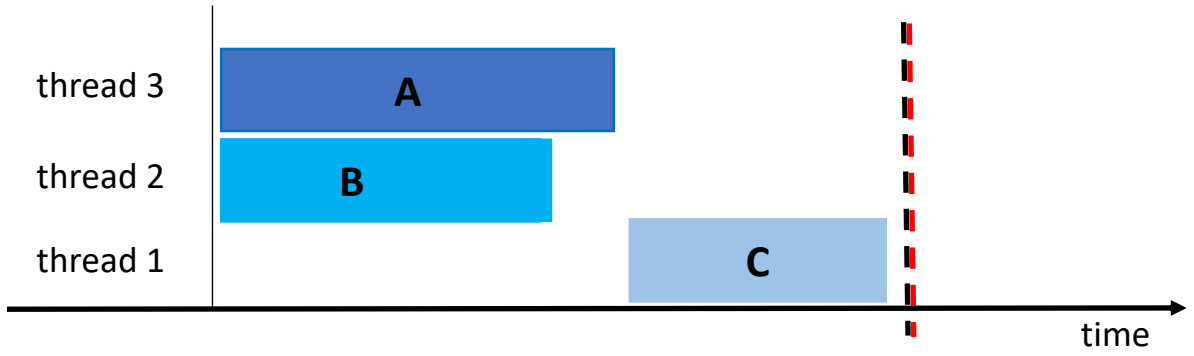
- Key idea: virtually speedup a selected code segment during runtime.

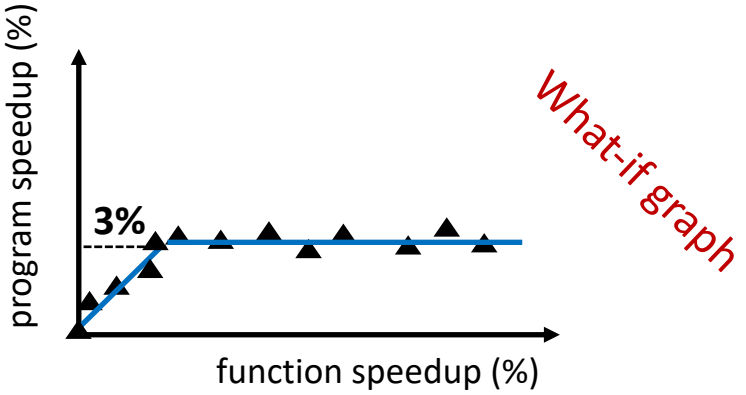
- Virtual speedup
  - Run concurrent execution paths **slower** whenever the selected function is running.
- Amount of delay $\propto$ selected speedup
- Coz profiler is implemented based on this idea



What-if Analysis with Coz profiler

```
01  void A() { // ~5.9 seconds
02    for(volatile size_t x=0; x<2500000000; x++) {}
03  }
04  void B() { // ~5.4 seconds
05    for(volatile size_t y=0; y<2200000000; y++) {}
06  }
07  void C() { // ~5.1 seconds
08    for(volatile size_t x=0; x<2100000000; x++) {}
09  }
10 int main() {
11   thread A_thread(A), B_thread(B);
12   A_thread.join(); B_thread.join();
13   C();
14 }
```





Problems with Causal Profiling

- Large number of experiments
  - Various systems and configuration
  - A study on page load time using causal profiling had over **12000** runs*
- Cross-platform what-if analysis
  - Limited access to resources
  - Develop and maintain Coz profiler for different architecture and OS
    - Using cycle-accurate emulators is tim

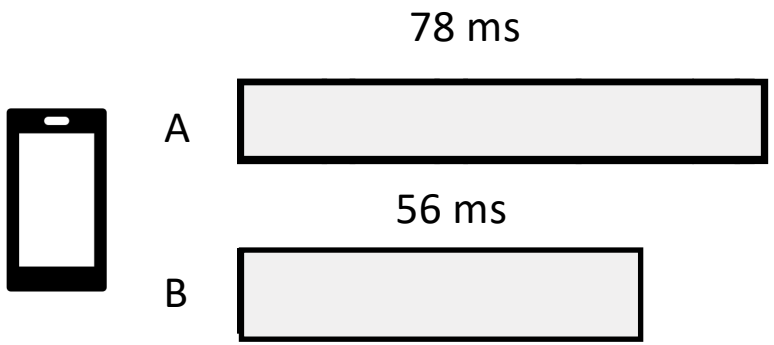
virtual causal profiling

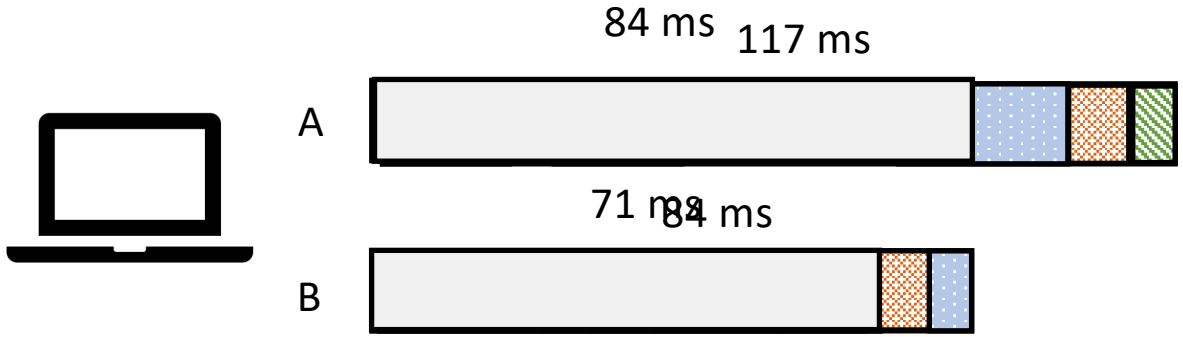
- Do we need **precise** timing? Can we use the idea of virtualization to scale the analysis?

* Pourghassemi et al.,"What-if Analysis of Page Load Time in Web Browsers Using Causal Profiling", ACM SIGMETRICS'19

Virtual Causal Profiling – Design Idea

- Preserve the ratio of the code segments by controlling the speed of CPU, I/O, and Memory



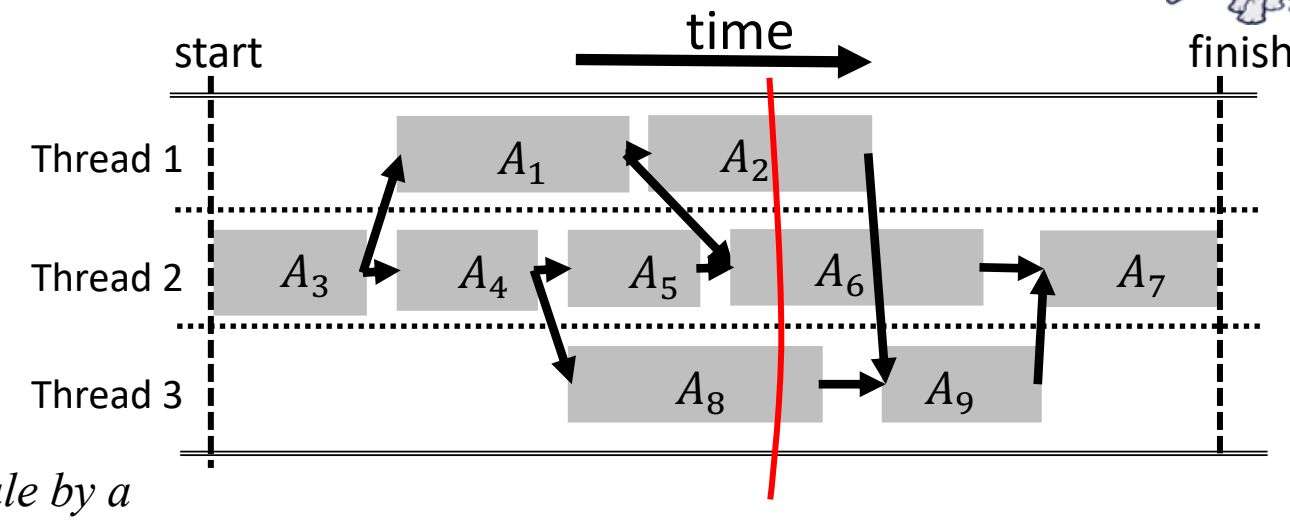


CPU	20% slower	50% slower
Memory	50% slower	50% slower
I/O	2X faster	50% slower

$$\frac{78}{56} \neq \frac{84}{84}$$

Theorem Behind Virtual Causal Profiling

- Use graph representation to model causal profiling
  - Prove soundness of causal profiling
  - Prove the necessary condition for virtual causal profiling



Theorem.. *If the execution time of all the segments scale by a constant factor α (e.g. $E(A_1) = \alpha E(A_1)$) as well as the speedup in a selected segment (e.g. $\boldsymbol{\varepsilon}_{new} = \alpha\boldsymbol{\varepsilon}$), then the new program speedup, S_{new}, is the same as S_{old} and is given by*

$$S_{new} = \frac{(T+\varepsilon) - T_{virtual}}{T}.$$

$$T = f(E(A_i), D(A_{i,j}))$$

Speedup A_2 by $\boldsymbol{\varepsilon}$:

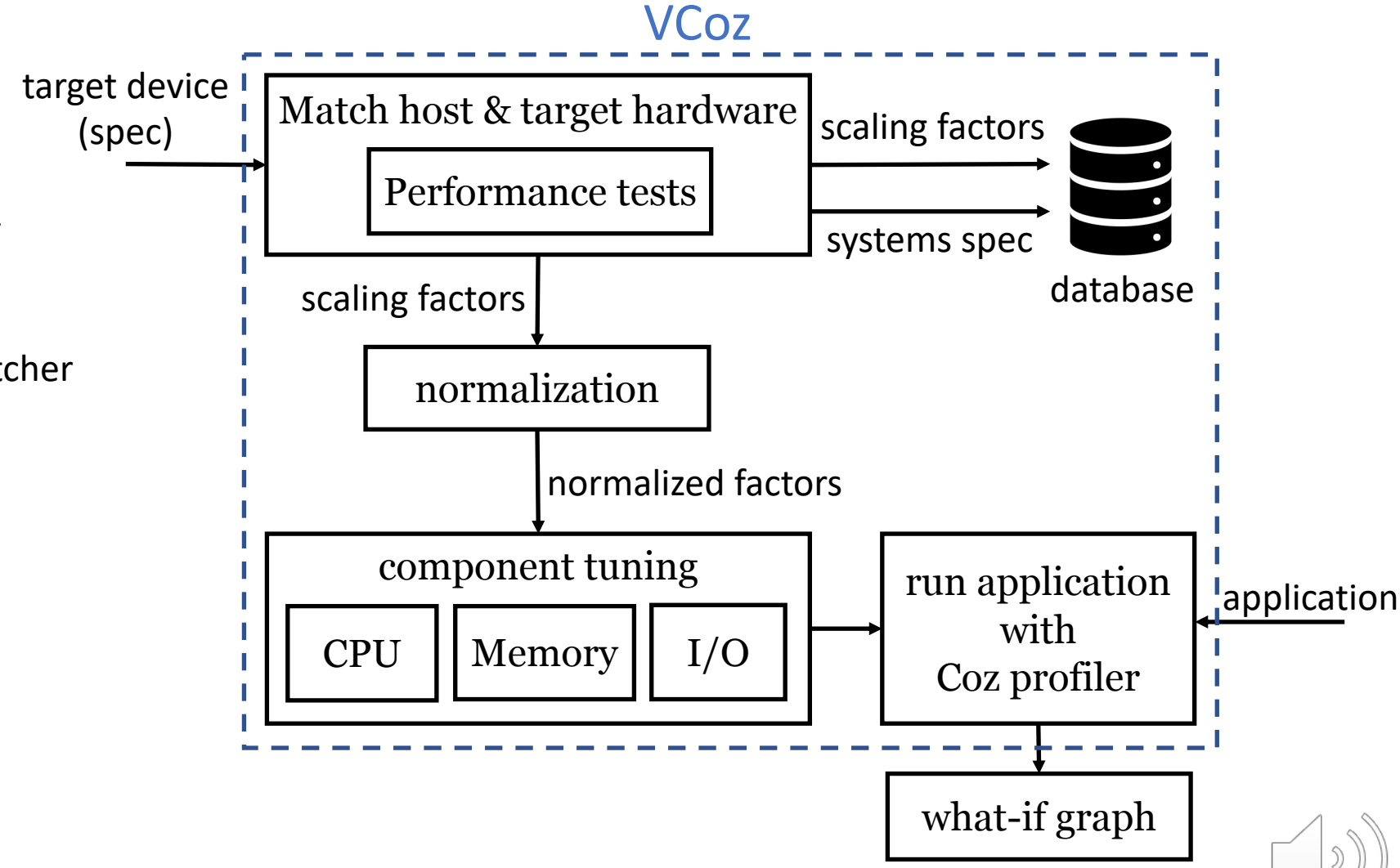
$$E'(A_6) = E(A_6) + \boldsymbol{\varepsilon}$$
$$E'(A_9) = E(A_9) + \boldsymbol{\varepsilon}$$

$$S = \frac{(T' + \varepsilon) - T}{T}$$

Only **relative** execution time of code segments is important, not the **absolute** execution time!

VCoz: Theory to Practice

- Implement prototype of VCoz
- VCoz modules
  - Host & target hardware matcher
  - Normalization
  - Component tuning
  - Profile application

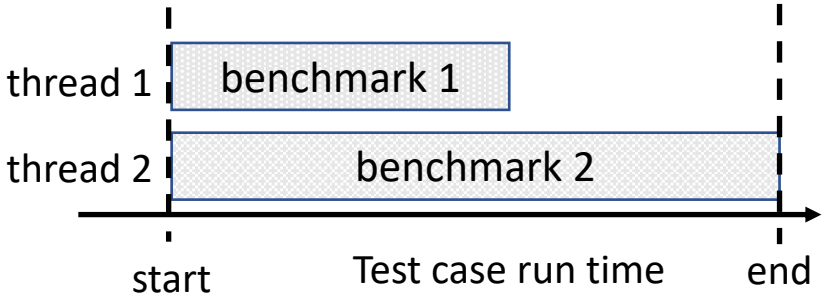


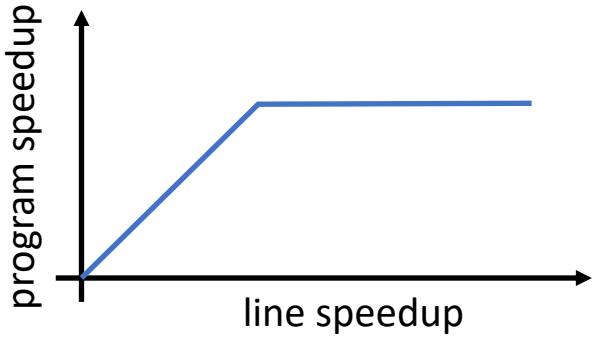
Results and Validation

- Experiment setup
  - Test case
  - Benchmarks
    - CPU-heavy: LU decomp, Cholesky
    - Memory-heavy: stream
    - I/O-heavy: Clinet-Server data stream
  - Host: **MacBook Air** Target: **Nexus 6P**

Benchmark	mobile	desktop	ratio
Matrix Mult.	3.1 s	1.5 s	2.1
FFT	56 ms	23 ms	2.4
LU	2.3 s	1.0 s	2.3
Word Count	38 s	16 s	2.3
Cholesky	1.1 s	450 ms	2.4
PCA	690 ms	300 ms	2.3

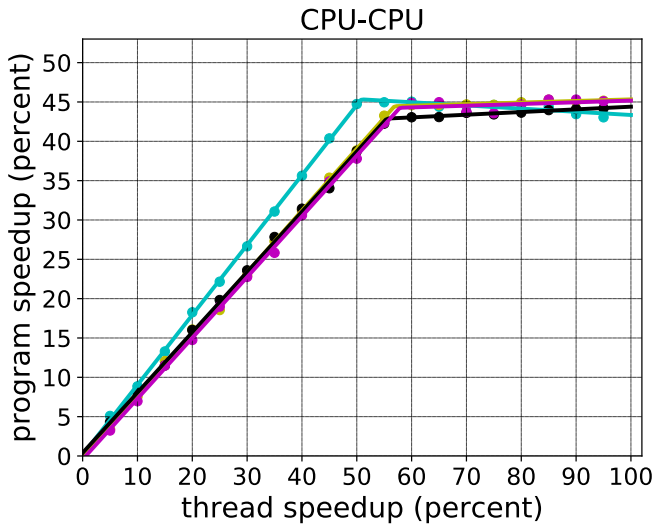
```
1  void b1(){
2     // benchmark 1
3  }
4  void b2(){
5     // benchmark 2
6  }
7  Int main(){
8    thread t1(f1), t2(b2);
9    t1._join(); t2_join();
12   }
```





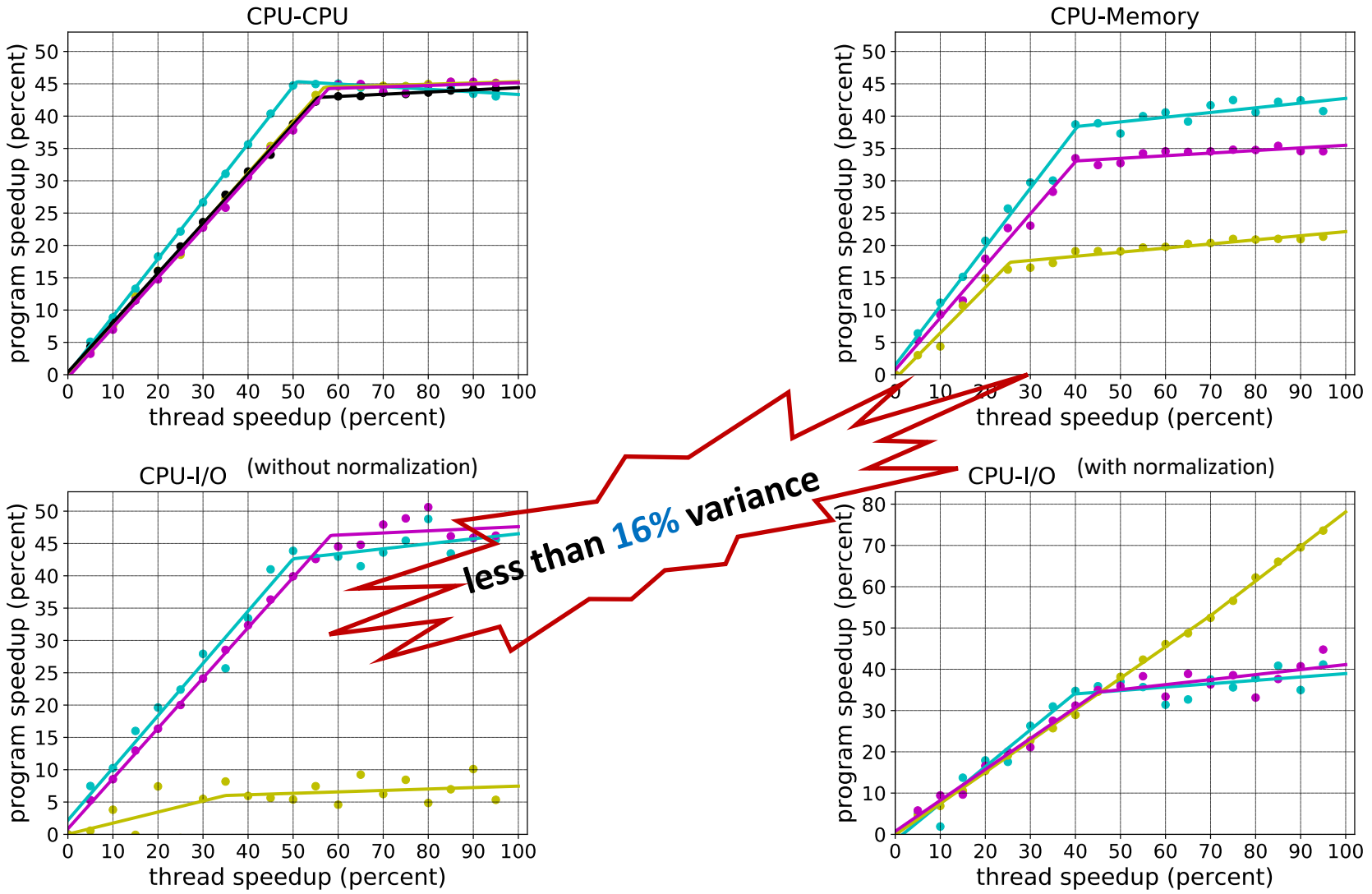
$\alpha_{cpu} = 2.3$

Results and Validation



Coz on Nexus 6P | VCoz | Coz on the host | VCoz 1.5 GHz

Results and Validation



Conclusion

- Causal profiling can be used in what-if analysis but it is not scalable
- Prove of concept and the necessary condition for perseverance of what-if graphs
- Introduce Virtual Causal Profiling and implement VCoz to **scale experiments** and **cross-platform** performance measurements
- Validation and accuracy analysis of VCoz by running experiments on different workloads

Thanks for your attention!

Behnam Pourghassemi

Prof. Aparna Chandramowlishwaran

Prof. Ardalan Amiri Sani

Personal webpage: http://newport.eecs.uci.edu/~bpourgha/

Email: bpourgha@uci.edu

HPC Forge: http://hpcforge.eng.uci.edu/