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.

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?

3. How much is the measured performance gain related to the **system** and **configuration**?

What-if Analysis

- **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

2 seconds lag in loading Amazon would cost **$6 billion** in sale
- Google

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

~$1000/hr for code runs on supercomputer/cluster
- ExtremeTech
What-if Analysis with Conventional Profilers

```c
void A() { // ~5.9 seconds
    for (volatile size_t x=0; x<2500000000; x++) {} } // ~5.9 seconds
void B() { // ~5.4 seconds
    for (volatile size_t y=0; y<2200000000; y++) {} } // ~5.4 seconds
void C() { // ~5.1 seconds
    for (volatile size_t x=0; x<2100000000; x++) {} } // ~5.1 seconds
int main() {
    thread A_thread(A), B_thread(B);
    A_thread.join(); B_thread.join();
    C();
}
```

```plaintext
% cumulative
total
name
time (seconds) calls
A() 35.90 1
B() 32.90 1
C() 31.10 1
% time
self children called
A() 35.9 7.2 0.00
B() 32.9 5.9 0.00
C() 31.2 5.9 0.00

What-if Analysis with Conventional Profiler

- gprof
- Perf
- Valgrind

program speedup (%)

function speedup (%)

3%
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**.

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

```c
void A() { // ~5.9 seconds
    for(volatile size_t x=0; x<2500000000; x++) {}
}
void B() { // ~5.4 seconds
    for(volatile size_t y=0; y<2200000000; y++) {}
}
void C() { // ~5.1 seconds
    for(volatile size_t x=0; x<2100000000; x++) {}
}
int main() {
    thread A_thread(A), B_thread(B);
    A_thread.join(); B_thread.join();
    C();
}
```

What-if graph
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 time-consuming

- 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.

Comparison:

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

Results:

- A: 78 ms, B: 56 ms
- A: 84 ms, B: 71 ms

Ratio:

A: 78/56 = 1.39
B: 84/71 = 1.18

The ratio of 1.39 is not equal to 1.18, indicating that the code segments are not preserved in ratio when controlling the speed of CPU, I/O, and Memory.
Theorem. If the execution time of all the segments scale by a constant factor $\alpha$ (e.g. $E(A_i) = \alpha E(A_i)$) as well as the speedup in a selected segment (e.g. $\varepsilon_{new} = \alpha \varepsilon$), then the new program speedup, $S_{new}$, is the same as $S_{old}$ and is given by $S_{new} = \frac{(T + \varepsilon) - T_{virtual}}{T}$.

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

VCoz

- Match host & target hardware
- Performance tests
- Scaling factors
- Systems spec
- Database
- Normalization
- Normalized factors
- Component tuning
  - CPU
  - Memory
  - I/O
  - Run application with Coz profiler
  - What-if graph

Target device (spec)
Results and Validation

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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>mobile</th>
<th>desktop</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Mult.</td>
<td>3.1 s</td>
<td>1.5 s</td>
<td>2.1</td>
</tr>
<tr>
<td>FFT</td>
<td>56 ms</td>
<td>23 ms</td>
<td>2.4</td>
</tr>
<tr>
<td>LU</td>
<td>2.3 s</td>
<td>1.0 s</td>
<td>2.3</td>
</tr>
<tr>
<td>Word Count</td>
<td>38 s</td>
<td>16 s</td>
<td>2.3</td>
</tr>
<tr>
<td>Cholesky</td>
<td>1.1 s</td>
<td>450 ms</td>
<td>2.4</td>
</tr>
<tr>
<td>PCA</td>
<td>690 ms</td>
<td>300 ms</td>
<td>2.3</td>
</tr>
</tbody>
</table>

\[ \alpha_{cpu} = 2.3 \]
Results and Validation

Coz on Nexus 6P
VCoz
Coz on the host
VCoz 1.5 GHz
Results and Validation

- CPU-CPU
- CPU-Memory
- CPU-I/O (without normalization)
- CPU-I/O (with normalization)

Less than 16% variance

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

Behavior 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/