Performance Tuning: How to Prepare to be Surprised, and then… How to Hunt for the Surprises

Presented to ATPESC 2017 Participants

James Reinders
HPC Consultant, Retired from Intel in 2016 (after 10,001 days)
Q Center, St. Charles, IL (USA)
Date 08/08/2017
Performance Tuning: How to Prepare to be Surprised, and then... How to Hunt for the Surprises

APPEARING SOON

SUMMARY OF MY ENTIRE TALK

Argonne National Laboratory
Performance Tuning: How to Prepare to be Surprised, and then... How to Hunt for the Surprises

• Have Expectations
  – make sure you can be surprised

• Validate your expectations
  – follow-up on discrepancies

• Follow a methodology
  – Use tools but don’t let them wag you
Performance Tuning: How to Prepare to be Surprised, and then… How to Hunt for the Surprises

• Have Expectations
  – make sure you can be surprised

• Validate your expectations
  – follow-up on discrepancies

• Follow a methodology
  – Use tools but don’t let them wag you

Доверяй, но проверяй
Performance Tuning: How to Prepare to be Surprised, and then... How to Hunt for the Surprises

• Have Expectations
  – make sure you can be surprised

• Validate your expectations
  – follow-up on discrepancies

• Follow a methodology
  – Use tools but don’t let them wag you
Performance Tuning: How to Prepare to be Surprised, and then… How to Hunt for the Surprises

• Have Expectations
  – make sure you can be surprised

• Validate your expectations
  – follow-up on discrepancies

• Follow a methodology
  – Use tools but don’t let them wag you
What is a **cache**?

**KEEP CALM**

**IT'S**

**QUIZ TIME**
how hard can it be? 10% efficient
I have a 10,000 core machine – but I insist on 50% efficiency.

How many cores should I allow you to use?
I have a 10,000 core machine – but I insist on 50% efficiency.

How many cores should I allow you to use?

<table>
<thead>
<tr>
<th>max that is 50% efficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.6</td>
</tr>
<tr>
<td>0.75</td>
</tr>
<tr>
<td>0.9</td>
</tr>
<tr>
<td>0.91</td>
</tr>
<tr>
<td>0.92</td>
</tr>
<tr>
<td>0.93</td>
</tr>
<tr>
<td>0.94</td>
</tr>
<tr>
<td><strong>0.95</strong></td>
</tr>
<tr>
<td>0.96</td>
</tr>
<tr>
<td>0.97</td>
</tr>
<tr>
<td>0.98</td>
</tr>
<tr>
<td>0.99</td>
</tr>
<tr>
<td>0.999</td>
</tr>
<tr>
<td>0.9999</td>
</tr>
</tbody>
</table>

_at 95% parallel we can only keep 21 cores busy and be 50% efficient_

(at 22 cores, we would slip to 48.8%)
I have a 10,000 core machine – but I insist on 50% efficiency.

How many cores should I allow you to use?

<table>
<thead>
<tr>
<th></th>
<th>max that is 10% efficient</th>
<th>max that is 33% efficient</th>
<th>max that is 50% efficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>19</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>0.6</td>
<td>23</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>0.75</td>
<td>37</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>0.9</td>
<td>91</td>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>0.91</td>
<td>101</td>
<td>23</td>
<td>12</td>
</tr>
<tr>
<td>0.92</td>
<td>113</td>
<td>26</td>
<td>13</td>
</tr>
<tr>
<td>0.93</td>
<td>129</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>0.94</td>
<td>151</td>
<td>34</td>
<td>17</td>
</tr>
<tr>
<td>0.95</td>
<td>181</td>
<td>41</td>
<td>21</td>
</tr>
<tr>
<td>0.96</td>
<td>226</td>
<td>51</td>
<td>26</td>
</tr>
<tr>
<td>0.97</td>
<td>301</td>
<td>68</td>
<td>34</td>
</tr>
<tr>
<td>0.98</td>
<td>451</td>
<td>102</td>
<td>51</td>
</tr>
<tr>
<td>0.99</td>
<td>900</td>
<td>204</td>
<td>101</td>
</tr>
<tr>
<td>0.999</td>
<td>9,001</td>
<td>2,031</td>
<td>1,001</td>
</tr>
<tr>
<td>0.9999</td>
<td>90,000</td>
<td>20,304</td>
<td>10,000</td>
</tr>
</tbody>
</table>
REAL WORLD EXAMPLE

Glacier National Park, photo by James Reinders, August 4, 2017
Single Frame

Underutilized?

Adapted with permission, courtesy of Florian Zitzelsberger, Pixar Animation Studios, from “Building a Scalable Evaluation Engine for Presto” segment of “Multithreading for Visual Effects” talk at SIGGRAPH2017.
Results

![Graph showing the relative speedup of Out of the Gate and Best Case with the number of cores increasing from 1 to 16. The graph indicates a significant improvement in performance, with a speedup of 3.3x at 16 cores.]

Single Frame

\[ VdfParallelExecutorEngineBase::RunSchedule \]

~ 20% Underutilized

Underutilization

Optimized Rig

Adapted with permission, courtesy of Florian Zitzelsberger, Pixar Animation Studios, from “Building a Scalable Evaluation Engine for Presto” segment of “Multithreading for Visual Effects” talk at SIGGRAPH2017.
Optimized Rig

≈ 15% Underutilized

Optimized Rig

KEEP CALM AND LOOK FOR WHAT YOU DO NOT SEE
It is hard to “see” if you do not look.
It is hard to “see” if you do not look.

We could guess,
after all – we are smart enough to believe we know what is happening.
Look for:

?
Look for:

- Confirmation
Look for:

- Confirmation
- Surprises
Look for:

- Confirmation
- Surprises

Your EXPERTISE will grow as you investigate.
Optimization: A Top-down Approach

Optimization: A Top-down Approach

H/W tuning:
- BIOS (TB, HT)
- Memory
- Network I/O

OS tuning:
- Page size
- Swap file
- RAM Disk
- Power settings

Better application design:
- Parallelization
- Fast algorithms / data bases
- Programming language and RT libs
- Performance libraries
- Driver tuning

Tuning for Microarchitecture:
- Compiler settings/Vectorization
- Memory/Cache usage
- CPU pitfalls

Optimization: A Top-down Approach

H/W tuning:
- BIOS (TB, HT)
- Memory
- Network I/O

OS tuning:
- Page size
- Swap file
- RAM Disk
- Power settings

Better application design:
- Parallelization
- Fast algorithms / data bases
- Programming language and RT libs
- Performance libraries
- Driver tuning

Tuning for Microarchitecture:
- Compiler settings/Vectorization
- Memory/Cache usage
- CPU pitfalls

Application Tuning

Who: Software Developers, Performance Engineers, Domain Experts

How:
- Workload selection
  - Repeatable results
  - Steady stat
- Define Metrics and Collect Baseline
  - Wall-clock time, FLOPS, FPS
  - <insert your metric here>
- Identify Hotspots
  - Focus effort where it counts
  - Use Tools
- Determine inefficiencies
  - Is there parallelism?
  - Are you memory bound?
  - Will better algorithms or programming languages help?

This step often requires some knowledge of the application and its algorithms.
Application Tuning

Find Hotspots

- This could be at the module, function, or source code level
- Determine your own granularity

```bash
$ oproreport --exclude-dependent --demangle=smart --symbols `which lyx`
```

CPU: PIII, speed 863.195 MHz (estimated)
Counted CPU_CLK_UNHALTED events (clocks processor is not halted) with a unit mask of 0x00 (No unit mask)
sym  samples  symbol name
081ec974 5016  8.5096 _Rb_tree<unsigned short, pair<unsigned short const, int>>, unsigned short
0810c4ec 3323  5.6375 Paragraph::getFontSettings(BufferParams const&, int) const
081319d8 3220  5.4627 LyXText::getFont(Buffer const*, Paragraph*, int) const
080e45d8 3011  5.1082 LyXFont::realize(LyXFont const&)
080e3d78 2623  4.4499 LyXFont::LyXFont()
081255a4 1823  3.0927 LyXText::singleWidth(BufferView*, Paragraph*, int, char) const
080e3cf0 1804  3.0605 operator==(LyXFont::FontBits const&, LyXFont::FontBits const&)
081128e0 1729  2.9332 Paragraph::Pimpl::getChar(int) const
081ed020 1380  2.3412 font_metrics::width(char const*, unsigned, LyXFont const&)
08110d60 1310  2.2224 Paragraph::getChar(int) const
081abc94 1227  2.0816 qfont_loader::getFontInfo(LyXFont const&)
```

Application Tuning

Find Hotspots

- This could be at the module, function, or source code level
- Determine your own granularity

sysprof: http://sysprof.com
Application Tuning

Find Hotspots

- This could be at the module, function, or source code level
- Determine your own granularity

Application Tuning

Find Hotspots

- This could be at the module, function, or source code level
- Determine your own granularity

Application Tuning
Find Hotspots

- This could be at the module, function, or source code level
- Determine your own granularity

This may reinforce your understanding of the application but often reveals surprises
## Application Tuning
### Resource Utilization

- Is the application parallel?
- Multi-thread vs. Multi-process
- Memory Bound?

```plaintext
last pid: 86494; load averages: 0.83, 0.65, 0.69 up 67:22:48:43 14:44:15
227 processes: 1 running, 224 sleeping, 2 zombie
CPU: 20.2% user, 0.0% nice, 6.5% system, 0.2% interrupt, 73.1% idle
Mem: 1657M Active, 1868M Inact, 273M Wired, 190M Cache, 112M Buf, 11M Free
Swap: 4508M Total, 249M Used, 4251M Free, 5% Inuse

<table>
<thead>
<tr>
<th>PID</th>
<th>USERNAME</th>
<th>THR</th>
<th>PRI</th>
<th>NICE</th>
<th>SIZE</th>
<th>RES</th>
<th>STATE</th>
<th>C</th>
<th>TIME</th>
<th>WCPU</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>86460</td>
<td>www</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>150M</td>
<td>30204K</td>
<td>accept</td>
<td>1</td>
<td>0:02</td>
<td>11.18%</td>
<td>php-cgi</td>
</tr>
<tr>
<td>86458</td>
<td>www</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>150M</td>
<td>29912K</td>
<td>accept</td>
<td>0</td>
<td>0:02</td>
<td>8.98%</td>
<td>php-cgi</td>
</tr>
<tr>
<td>86463</td>
<td>psql</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>949M</td>
<td>99M</td>
<td>sbwait</td>
<td>1</td>
<td>0:01</td>
<td>7.96%</td>
<td>postgres</td>
</tr>
<tr>
<td>85005</td>
<td>www</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>150M</td>
<td>35204K</td>
<td>accept</td>
<td>1</td>
<td>0:01</td>
<td>7.57%</td>
<td>php-cgi</td>
</tr>
<tr>
<td>85274</td>
<td>www</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>149M</td>
<td>40866K</td>
<td>sbwait</td>
<td>3</td>
<td>0:27</td>
<td>5.13%</td>
<td>php-cgi</td>
</tr>
<tr>
<td>85257</td>
<td>www</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>151M</td>
<td>40644K</td>
<td>sbwait</td>
<td>2</td>
<td>0:33</td>
<td>4.59%</td>
<td>php-cgi</td>
</tr>
<tr>
<td>85284</td>
<td>www</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>150M</td>
<td>41504K</td>
<td>accept</td>
<td>2</td>
<td>0:14</td>
<td>4.59%</td>
<td>php-cgi</td>
</tr>
<tr>
<td>85687</td>
<td>psql</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>951M</td>
<td>128M</td>
<td>sbwait</td>
<td>1</td>
<td>0:04</td>
<td>4.20%</td>
<td>postgres</td>
</tr>
<tr>
<td>85886</td>
<td>psql</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>949M</td>
<td>161M</td>
<td>sbwait</td>
<td>0</td>
<td>0:08</td>
<td>3.37%</td>
<td>postgres</td>
</tr>
<tr>
<td>86459</td>
<td>psql</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>949M</td>
<td>75968K</td>
<td>sbwait</td>
<td>2</td>
<td>0:01</td>
<td>3.37%</td>
<td>postgres</td>
</tr>
<tr>
<td>85279</td>
<td>psql</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>950M</td>
<td>192M</td>
<td>sbwait</td>
<td>2</td>
<td>0:14</td>
<td>2.39%</td>
<td>postgres</td>
</tr>
<tr>
<td>85269</td>
<td>psql</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>950M</td>
<td>199M</td>
<td>sbwait</td>
<td>1</td>
<td>2:20%</td>
<td>2.20%</td>
<td>postgres</td>
</tr>
<tr>
<td>85268</td>
<td>www</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>152M</td>
<td>44356K</td>
<td>sbwait</td>
<td>2</td>
<td>0:32</td>
<td>1.17%</td>
<td>php-cgi</td>
</tr>
<tr>
<td>85273</td>
<td>psql</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>950M</td>
<td>215M</td>
<td>sbwait</td>
<td>1</td>
<td>0:19</td>
<td>1.17%</td>
<td>postgres</td>
</tr>
<tr>
<td>97982</td>
<td>psql</td>
<td>1</td>
<td>44</td>
<td>0</td>
<td>26020K</td>
<td>6832K</td>
<td>select</td>
<td>0</td>
<td>46:55</td>
<td>0.00%</td>
<td>postgres</td>
</tr>
<tr>
<td>892</td>
<td>root</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3160K</td>
<td>8K</td>
<td>-</td>
<td>2</td>
<td>13:33</td>
<td>0.00%</td>
<td>ntfs</td>
</tr>
<tr>
<td>1796</td>
<td>root</td>
<td>1</td>
<td>44</td>
<td>0</td>
<td>19780K</td>
<td>13650K</td>
<td>select</td>
<td>3</td>
<td>12:43</td>
<td>0.00%</td>
<td>Xvfb</td>
</tr>
</tbody>
</table>
```
Application Tuning
Resource Utilization

- Is the application parallel?
Application Tuning
Resource Utilization

- Memory Bound?

- Know your max theoretical memory bandwidth
Application Tuning
Resource Utilization

MPI applications have added communication complexity

Application Tuning
What’s Next?

- If your Hotspots are common algorithms:
  - Look for optimized libraries
- If your Hotspots are uncommon:
  - Compiler optimizations
  - Expert analysis and refactoring of an algorithm
    - The opposite of “low-hanging fruit”
  - Deeper analysis of hardware performance
    - More on this later
- If the system is underutilized:
  - Add parallelism - multi-thread or multi-process
    - OpenMP, TBB, MPI, etc…

- Tools can help you determine where to look and may identify some issues.
- Some tools may provide suggestions for fixes.
- In the end – the developer and/or expert has to make the changes and decisions – there is no silver bullet.
What is a **cache miss**?
What happens if we reduce the number of cache misses?

KEEP CALM
IT'S
QUIZ TIME
Performance Monitoring Unit (PMU)

• Registers on Intel CPUs to count architectural events
  – E.g. Instructions, Cache Misses, Branch Mispredict

• Events can be counted or sampled
  – Sampled events include Instruction Pointer

• Raw event counts are difficult to interpret
  – Use a tool like VTune or Perf with predefined metrics
Raw PMU Event Counts or Metrics – you choose as you like
Look for:

- Confirmation
- Surprises

Do not skip either
...supplements AI-based analysis with a dynamic FLOP/s profile and peak FLOPs and memory sub-system throughput levels providing enlightening “bounds and bottlenecks” analysis for complex workloads.

Learn more:
http://tinyurl.com/atpesc-roofline
Roofline = Visualize a Performance Model

Attainable Performance (Gflops/s)

Arithmetic Intensity (flops/byte)

Compute Limit

Memory Bandwidth Limit

Learn more:
http://tinyurl.com/atpesc-roofline
There Are Different Ceilings

Learn more:
http://tinyurl.com/atpesc-roofline
Ask “Why am I Here?” and “Where am I going?”

It is always useful to ask the questions:

“why am I not on a higher ceiling?” and “what should I do to reach it?”

Learn more: http://tinyurl.com/atpesc-roofline
WHAT TOOLS WILL NEVER TELL YOU

ATPESC STUDENTS BEST EVER! ACCORDING TO SOURCES CLOSE TO CHICAGO
#Moderncode: COSMOS

Book Cover Background: Photo of the COSMOS@DiRAC SGI UV2000 based Supercomputer manufactured by SGI, Inc and operated by the Stephen Hawking Centre for Theoretical Cosmology, University of Cambridge. Photo courtesy of Philip Mynott. Book Cover Foreground: 3D visualization of statistical fluctuations in the Cosmic Microwave Background, the remnant of the first measurable light after the Big Bang. CMB data is from the Planck satellite and is the topic of Chapter 10 providing insights into new physics and how the universe evolved. Visualization rendered with Intel’s OSPRay ray tracing open source software by Gregory P. Johnson and Timothy Rowley, Intel Corporation.
#ModernCode: COSMOS

Book Cover Background: Photo of the COSMOS@DiRAC SGI UV2000 based Supercomputer manufactured by SGI, Inc and operated by the Stephen Hawking Centre for Theoretical Cosmology, University of Cambridge. Photo courtesy of Philip Mynott. Book Cover Foreground: 3D visualization of statistical fluctuations in the Cosmic Microwave Background, the remnant of the first measurable light after the Big Bang. CMB data is from the Planck satellite and is the topic of Chapter 10 providing insights into new physics and how the universe evolved. Visualization rendered with Intel’s OSPRay ray tracing open source software by Gregory P. Johnson and Timothy Rowley, Intel Corporation.

ATPESC 2017, July 30 – August 11, 2017
We find that using a simple trapezium rule integrator combined with hand-selected sampling points (to improve accuracy in areas of interest) provides sufficient numerical accuracy to obtain a physically meaningful result, and the reduced space and time requirements of this simplified method give a speed-up of $O(10x)$.
KEEP CALM AND MIND YOUR ALGORITHMS

WHAT TOOLS WILL NEVER TELL YOU

ATPESC STUDENTS BEST EVER! ACCORDING TO SOURCES CLOSE TO CHICAGO
surprised verify
follow-up tools trust
methodology validate
preferred
envelope
model
expectations
back
roofline sure
let discrepancies
prepare

KEEP CALM AND THINK PARALLEL
Performance Tuning: How to Prepare to be Surprised, and then... How to Hunt for the Surprises

- Have Expectations
  - make sure you can be surprised

- Validate your expectations
  - follow-up on discrepancies

- Follow a methodology
  - Use tools but don’t let them wag you

  \[ \wedge \text{Have a Roofline Model} \]
  (back of envelope preferred)

Learn more:
http://tinyurl.com/atpesc-roofline