Evaluating Performance using Ratio of Execution Times

Tomas Kalibera
My Background

- **PL/Systems**
  - R language: GNU R, (Purdue) FastR
  - Java: Ovm, OpenJDK
  - Garbage collection, interpretation, analysis

- **Performance/Benchmarking**
  - Methodology: modeling non-determinism
  - DaCapo benchmarks: observational study
  - Practice: DaCapo, SPEC CPU/JBB/JVM, Shootout, CD, CSIBE, FFT&kernels – Mono, Java, R
  - Teaching; Evaluate, Dagstuhl workshops
Talking about Performance
(fictional conversations in PL/systems)

Lunch at SW company

Joe: Any numbers yet for your compiler patch?
Ann: 9% on average, no big slowdowns.
Joe: That's really good!
Ann: Yes:) Or too good to be true, have to run more tests.

Coffee at CS dept of a uni

Cristine: How much slower is our VM than production VM X?
John: Now within 2x.
Cristine: Perfect, that allows us to claim our speedups are relevant.

Dissertation (MSc) committee meeting, the student got 18% speedup on FFT with kernel patch and claimed he could speed up applications by 18%

Erik: 18% speedup is far too small. We should reject.
Tim: 18% is great even for just FFT, great work. The generalizing claim is naïve.
# Evaluating Time Ratio In Papers

<table>
<thead>
<tr>
<th></th>
<th>Papers</th>
<th>Reported Time Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2011</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASPLOS</td>
<td>32</td>
<td>22</td>
</tr>
<tr>
<td>ISMM</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>PLDI</td>
<td>55</td>
<td>27</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>218</td>
<td>127 (58%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Papers</th>
<th>Reported Time Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2015</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASPLOS</td>
<td>48</td>
<td>37</td>
</tr>
<tr>
<td>ISMM</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>PLDI</td>
<td>58</td>
<td>22</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>218</td>
<td>127 (58%)</td>
</tr>
</tbody>
</table>
Important Decisions in Evaluations involving Time Ratio

- Which ratio?
  - Opinions, ratio games and confusion

- Averaging
  - Which mean, averaging over benchmarks

- Error estimate
  - Hardly ever any at all

Warning: some options given in the following are questionable and some are outright wrong!
Time Ratio: But Which One?

GNU-R, byte-code interpreter (B): 58s
Purdue FastR (F): 16s
(spectralnorm-alt4 [sn5] benchmark)

\[
\frac{T_{\text{new}}}{T_{\text{old}}} = 0.28 \text{ (28\%)}
\]

\[
1 - \frac{T_{\text{new}}}{T_{\text{old}}} = 0.72 \text{ (72\%)}
\]

\[
\frac{T_{\text{old}}}{T_{\text{new}}} = 3.63 \text{ (363\%, 3.63x)}
\]

\[
\frac{T_{\text{old}}}{T_{\text{new}}} - 1 = 2.63 \text{ (263\%)}
\]

\[
\frac{T_{\text{old}}}{T_{\text{old}} - T_{\text{new}}} = 1.38 \text{ (138\%)}
\]

\[
\frac{T_{\text{old}}}{T_{\text{old}} - T_{\text{new}}} \times 100\% = 250\%
\]
Time Ratio: The Right Baseline?

GNU-R, byte-code compiler (B): 58s $T_B$
Purdue FastR (F): 16s $T_F$
GNU-R, AST interpreter (A): 154s $T_A$

\[
\frac{T_F}{T_B} = 0.28
\]
We reduced execution time to 28% of best performing alternative. We are 3.63x faster.

\[
\frac{T_B}{T_F} = 3.63
\]

\[
\frac{T_F}{T_A} = 0.10 \quad \frac{T_B}{T_A} = 0.38
\]
We reduced execution time of an existing system to 10%. The best performing alternative reduced it to 38%. We are 9.63x faster but the alternative only 2.66x faster.
Summarizing over Benchmarks

Language Shootout Benchmark Suite for R: \( n = 37 \) benchmarks.

Execution times with FastR: \( T_{Fi} \)

Execution times with GNU-R AST: \( T_{Ai} \)

Summarying ratio \( \frac{T_A}{T_F} \)

\[
\frac{1}{n} \sum_{i=1}^{n} \frac{T_{Ai}}{T_{Fi}} = 12.91 \quad \text{Arithmetic mean of ratios}
\]

\[
\frac{\sum_{i=1}^{n} T_{Ai}}{\sum_{i=1}^{n} T_{Fi}} = 7.00 \quad \text{Ratio of sums}
\]

\[
\sqrt[n]{\prod_{i=1}^{n} \frac{T_{Ai}}{T_{Fi}}} = 8.53 \quad \text{Geometric mean of ratios}
\]

\[
\frac{n}{\sum_{i=1}^{n} \frac{T_{Fi}}{T_{Ai}}} = 5.02 \quad \text{Harmonic mean of ratios}
\]
What is Hiding Behind the Mean?

\[ \sqrt[n]{\prod_{i=1}^{n} \frac{T_{Ai}}{T_{Fi}}} = 8.53 \]

**Geometric mean of ratios**

66x speedup!
Repetition and Error Estimate

Iteration times for sn5 (FastR)

Percentile bootstrap 95% confidence interval for the mean

```r
cfsingle <- function(x) {
  means <- sapply(1:10000,
                  function(i) mean(sample(x, replace = TRUE))
    )
  sort(means)[c(250, 9750)]
}
```

Sn5 with FastR takes 16.6 ± 2.0s with 95% confidence.
Repetition and Error Estimate

Percentile bootstrap 95% confidence interval for the ratio of means.
Input:
   x – vector of iteration times for nominator
   Y – vector of iteration times for denominator

```r
cfratio <- function(x, y) {
  means <- sapply(1:10000, function(i) {
    xs <- sample(x, replace = TRUE)
    ys <- sample(y, replace = TRUE)
    mean(xs) / mean(ys)
  })
  sort(means)[c(250, 9750)]
}
```

The speedup of FastR over GNU-R AST on sn5 is 9.4 ± 1.1x.

FastR reduces execution time of sn5 over GNU-R AST to 10.8 ± 1.3%.
Relative Time of FastR and GNUR-BC over GNUR-AST

- FastR
- GNUR-BC
Repetition and Error Estimate

Percentile bootstrap 95% confidence interval for the geometric mean.

Input:
  \(xr\) – vector of ratios (one for each benchmark, calculated as ratio of iteration means)

```r
cfgmean <- function(xr) {
  gmean <- function(x) exp(mean(log(x)))
  gmeans <- sapply(1:10000, function(i)
    gmean(sample(xr, replace = TRUE)))
  sort(gmeans)[c(250, 9750)]
}
```

The geomean speedup of FastR over GNU-R AST is 8.9 ± 2.7x.

On geomean, FastR reduces execution time over GNU-R AST to 12.4 ± 3.8%.
Summary

• Decisions for R study
  - Ratio for graphs \( \frac{T_{\text{new}}}{T_{\text{old}}} \)
  - Ratio in text given as inverse \( \frac{T_{\text{old}}}{T_{\text{new}}} \)
  - 95% bootstrap confidence intervals for ratios of individual benchmarks
  - Geometric mean over suite in text with huge disclaimer

• References
  - ISMM'13, Rigorous benchmarking in reasonable time
  - OOPSLA'12, A black-box approach to understanding concurrency in DaCapo
  - VEE'15, A Fast Abstract Syntax Tree Interpreter for R
  - Uni of Kent technical report, https://kar.kent.ac.uk/30809, Quantifying Performance Changes with Effect Size Confidence Intervals
Additional Resources

Jain: The Art of Computer Systems Performance Analysis


Kirkup: Experimental Methods: An Introduction to the Analysis and Presentation of Data


Wasserman: All of Statistics: A Concise Course in Statistical Inference