Java performance evaluation methodology

DISTRIBUTED SYSTEMS RESEARCH GROUP
http://dsrg.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE
FACULTY OF MATHEMATICS AND PHYSICS
Introduction

• Georges A., Buytaert D., Eeckhout L.: Statistically Rigorous Java Performance Evaluation

Department of Electronics and Information Systems, Ghent University, Belgium
Problems

• Non-determinism
  ▪ JIT compilation
    • runtime profile
    • different optimization levels
  ▪ Thread scheduling
Methodologies in papers

- 2001-2006
- Survey of 50 papers
  - 16 – not specified
  - 10 – best run
  - 8 – average
  - 4 – median
  - 4 – second best
  - 3 – worst
  - Replay compilation – 7
  - Confidence intervals - 4
Experimental design options

- One/multiple VM invocations
  - Startup/steady-state performance
- Including/excluding compilation
- Forced GC before measurement
Intermezzo – Replay compilation

- Ogata et al.: Replay Compilation: Improving Debuggability of a Just-in-Time Compiler
- state-saving compiler
- replaying compiler
- parameters stored in a log in the memory
- available in crash dumps
Statistics...

- Confidence interval
- Central limit theory
  - means are approximately normally distributed, $n \geq 30$
  - Formulas for confidence interval
- $n < 30$ – Student's t-distribution
Comparing two alternatives

- Are the results from benchmarks different?
- Confidence intervals overlap => difference can be result of random effects
- Do not overlap – only with probability $\alpha$ the difference is due to random effects
- Another formulas for confidence interval of difference of measurements – 0 must not be in the interval
More than 2 alternatives

- Analysis of Variance (ANOVA)
- $n$ measurements for each of $k$ alternatives
- Calculation of variation in alternatives – column means vs. overall mean (SSA)
- Calculation of variation of the differences – single values vs. column mean (SSE)
- SST – all single values vs. overall mean
- SST = SSA + SSE
- F-test: determine if SSA is 'larger' than SSE
Recommended methodology I

- Startup performance
  - Measure execution time of multiple VM invocations, each running a single benchmark iteration
  - Compute confidence intervals (for given confidence level $1-\alpha$)
Recommended methodology II

• Steady state performance
  ▪ Execute more VM invocations, each running more benchmark iterations
  ▪ Determine in which iteration the steady state was reached and add defined number of iterations to results
  ▪ For each VM invocation, compute mean
  ▪ Compute confidence interval for means from above

• Independence required
Comparison

<table>
<thead>
<tr>
<th>statistically rigorous methodology</th>
<th>overlapping intervals</th>
<th>prevalent methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>non-overlapping intervals, same order</td>
<td>performance difference &lt; θ</td>
</tr>
<tr>
<td></td>
<td>non-overlapping intervals, not same order</td>
<td>misleading but correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>misleading and incorrect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>incorrect</td>
</tr>
</tbody>
</table>

Table 4. Classifying conclusions by a prevalent methodology in comparison to a statistically rigorous methodology.

- Confidence in methodologies
- SPECjvm98 and DaCapo benchmarks
javac benchmark

- correct – complement
- larger threshold → less incorrect conclusions

Figure 5. The classification for javac as a function of the threshold $\theta \in [0; 3]$ for the ‘best’ prevalent method, on the AMD Athlon.
Results

• Up to 16% conclusions misleading
• Incorrect conclusions – for some methodologies more than 3%
• Mean and median are better than best, second best and worst
The End

Questions?