The case for the Three R’s of Systems Research:

Repeatability
Reproducibility
& Rigor

Jan Vitek

Kalibera, Vitek. Repeatability, Reproducibility, and Rigor in Systems Research, EMSOFT'11

Out of 122 papers in ASPLOS, ISMM, PLDI, TACO, TOPLAS

90 evaluated execution time based on experiments

71 of these 90 papers ignored uncertainty

Science Done Bad

In 2006, Potti&Nevins claim they can predict lung cancer

In 2010, papers retracted, bankruptcy, resignations & investigation

Bad science ranging from fraud, unsound methods, to off-by-one errors in Excel

Uncovered by a repetition study conducted by Baggerly&Coombes with access to raw data and 2,000 hours of effort

Producing Wrong Data

Without Doing Anything Obviously Wrong!

ASPLOS’09

A parté

Mytkowicz, Diwan, Hauswirth, Sweeney. Producing Wrong Data
Out of 122 papers in ASPLOS, ISMM, PLDI, TACO, TOPLAS

90 evaluated execution time based on experiments

71 of these 90 papers ignored uncertainty

This lack of rigor undermines the results
Yet, no equivalent to the Duke Scandal.

Are we better?
Is our research not worth reproducing?
Is our research too hard to reproduce?

Reproduction

...independent researcher implements/realizes the published solution from scratch, under new conditions

Repetition

...re-doing the same experiments on the same system and using the same evaluation method

Is our research hard to repeat?
Is our research hard to reproduce?

Goal

Break new ground in

**hard real-time concurrent garbage collection**

Obstacles

No real-time benchmarks for GCed languages

No clear competition, two GC algorithms claim to be best

No accepted measurement methodology

No open source experimental platform for comparison
Step 1

Develop an open source experimental platform

*Picked the Real-time Specification for Java*

First generation system, about 15 man/years

*Flew on a Boeing ScanEagle*

Second generation system, about 6 man/years

*Competitive with commercial JVMs*

A Real-time Java Virtual Machine for Avionics. TECS, 2006

--

Step 2

Develop an open source benchmark

(Collision Detector Benchmark

*In Java, Real-time Java, and C (Linux/RTEMS)*

Measure response time, release time jitter

*Simulate air traffic control*

*Hard RT collision detector thread*

*Scalable stress on garbage collector*

About 1.5 man/years

A family of Real-time Java benchmarks. CC:PE 2011

--

Observations

Results on noncompetitive systems not relevant

Much of work went into a credible research platform

A family of Real-time Java benchmarks. CC:PE 2011

--

Observations

Understanding what you measure is critical

Running on a real embedded platform and real-time OS, difference between Java & C small…

Good news?

No. The LEON3 lacks a FP unit, & the benchmark is FP intensive...
**Step 3**

Gain experience with the state of the art

Experiment with different GC techniques

- GC in uncooperative environment
- Brooks forwarding
- Object replication
- Object handles

About 2 man/years

---

**Observation**

Trust but verify, twice.

From workshop to journal, speed 30% better

Good news?

Later realized switching to GCC 4.4 slowed baseline (GCC didn’t inline a critical function)

Once accounted for this our speed up was 4%…

A correction was issued...

---

**Step 4**

Reproduce state of the art algorithms from IBM and Oracle

- Metronome, Sun Java RTS

Choose measurement methodology

- Existing metric (MMU) inadequate

About 3 man/years

---

**Observation**

Reproduction was difficult because of closed-source implementations & partial description of algorithms

Repetition was impossible because no common platform
Step 5

Develop a novel algorithm
*Fragmentation tolerant*
*Constant-time heap access*

About 0.6 man/years

Schism: Fragmentation-Tolerant Real-Time Garbage Collection. *PLDI 2011*

In summary, 28 m/y reproduction .6 m/y novel work

<table>
<thead>
<tr>
<th>Activity</th>
<th>Man/years</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental platform</td>
<td>21</td>
</tr>
<tr>
<td>Benchmark</td>
<td>2</td>
</tr>
<tr>
<td>Implementing basic techniques</td>
<td>2</td>
</tr>
<tr>
<td>Reproduction of state-of-the art +methodology</td>
<td>3</td>
</tr>
<tr>
<td>Implementing novel algorithm</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Rigor

Cater for random effects, non-determinism
*Repeat experiment runs, summarize results*
*Threat to validity detectable by failure to repeat*

Guard against bias
*Use multiple configurations, hardware platforms*
*Threat to validity detectable by failure to reproduce*

Jain: The Art of Computer Systems Performance Analysis
Lilja: Measuring Computer Performance, A Practitioner’s Guide

Reproducibility

Community support for focused reproductions
*Open benchmarks and platforms*

Reward system for reproductions
*Publish reproduction studies*
*Regard them as 1st class publications*

[http://evaluate.inf.usi.ch](http://evaluate.inf.usi.ch)
Repeatability

Enable repetition studies
Archival
    Automate and archive
Disclosure
    Share experimental details

Action

In 2011, FSE led by Shriram Krishnamurthi attempted to institutionalize repetition

*The SE community pushed back*

In 2012, I was elected chair of SIGPLAN by accident

*Here is what we did together*

A Reviewing Pattern

PLDI, ECOOP, OOPSLA and POPL have an artifact evaluation committee

Close to 50% of the papers have submitted artifacts

Over 90% of artifacts have been validated

Conclusions

Develop open source benchmarks

Codify documentation, methodologies & reporting standards

Require executable artifacts

Publish reproduction studies

Separate “idea” papers from “evaluation” papers