Tackling the Reproducibility Problem in Systems Research with Declarative Experiment Specifications

Ivo Jimenez, Carlos Maltzahn (UCSC)
Adam Moody, Kathryn Mohror (LLNL)
Jay Lofstead (Sandia)
Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau (UWM)
The Reproducibility Problem

- Network
- Disks
- BIOS
- OS conf.

- Magic numbers
- Workload
- Jitter
- etc...

**Goal:** define methodology so that we don’t end up in this situation
Outline

• Re-execution vs. validation
• Declarative Experiment Specification (ESF)
• Case Study
• Benefits & Challenges
Outline

• Re-execution vs. validation
• Declarative Experiment Specification (ESF)
• Case Study
• Benefits & Challenges
Reproducibility Workflow

1. Re-execute experiment
   – Recreate original setup, re-execute experiments
   – Technical task

2. Validate results
   – Compare against original
   – A subjective task
     • How do we express objective validation criteria?
     • What contextual information to include with results?
Experiment Goal:
Show that my algorithm/system/etc. is better than the state-of-the-art.

Means of Experiment

Observations

The relationship that describes the information needed is part of the experiment. The data shows that my approach outperforms the state-of-the-art.

Figure 5 illustrates the time required to complete MySQL's test. The partitioned execution runs only for a short period of time. In general, the overhead of applying DTA diminishes as the unpartitioned execution performs similarly to the mechanism applied on the authenticated partition.
Outline

• Re-execution vs. validation
• **Declarative Experiment Specification (ESF)**
• Case Study
• Benefits & Challenges
**Experiment Goal:** Show that my algorithm/system/etc. is better than the state-of-the-art.
Validation Language Syntax

```plaintext
validation
  : 'for' condition ('and' condition)* 'expect' result ('and' result)*
    ;

condition
  : vars ('in' range | ('=' | '<' | '>' | '!=') value)
    ;

result
  : condition
    ;

vars
  : var (',' var)*
    ;

range
  : '[ ' range_num (',' range_num)* ' ]'
    ;

range_num
  : NUMBER '-' NUMBER | '*'
    ;

value
  : '*' | 'NUMBER (',' NUMBER)*
    ;
```
Outline

• Re-execution vs. validation
• Declarative Experiment Specification (ESF)
• Case Study
• Benefits & Challenges
Ceph OSDI ‘06

• Select scalability experiment.
  – Distributed; makes use of all resources
  – Main bottlenecks: I/O and network

• Why this experiment?
  – Top conference
  – 10 year old experiment
  – Ideal reproducibility conditions
    • Access to authors, topic familiarity, same hardware,
  – Even in an ideal scenario, we still struggle
    • Demonstrates which missing info is captured by an ESF!
Validation Statement

```
for cluster_size = * and not net_saturated
expect ceph >= (raw * .90)
```

```
"independent_variables": [{
  "type": "cluster_size",
  "values": "2-28"
}, {
  "type": "method",
  "values": ["raw", "ceph"]
}, {
  "type": "net_saturated",
  "values": ["true", "false"]
}],
"dependent_variable": {
  "type": "throughput",
  "scale": "mb/s"
},
```

---

**Cluster size**

**Per-OSD Average Throughput (MB/s)**

- 2 MB/s
- 6 MB/s
- 10 MB/s
- 14 MB/s
- 18 MB/s
- 22 MB/s
- 26 MB/s
- 30 MB/s
- 40 MB/s
- 50 MB/s
- 60 MB/s
<table>
<thead>
<tr>
<th>Component</th>
<th>Original</th>
<th>Reproduced</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>AMD 2212 @2.0GHz</td>
<td>Intel E5-2630 @2.3GHz</td>
</tr>
<tr>
<td>Disk drive</td>
<td>Seagate ST3250620NS</td>
<td>HP 6G 658071-B21</td>
</tr>
<tr>
<td>Disk BW</td>
<td>58 MB/s</td>
<td>20 MB/s (15 MB/s limit)</td>
</tr>
<tr>
<td>Linux</td>
<td>2.6.9</td>
<td>3.13.0</td>
</tr>
<tr>
<td>Ceph</td>
<td>commit from 2005</td>
<td>0.87.1</td>
</tr>
<tr>
<td>Storage</td>
<td>26 nodes</td>
<td>12 nodes</td>
</tr>
<tr>
<td>Clients</td>
<td>20 nodes</td>
<td>1 node</td>
</tr>
<tr>
<td>Network</td>
<td>Netgear GS748T</td>
<td>Same as original</td>
</tr>
<tr>
<td>Network BW</td>
<td>1400 MB/s</td>
<td>110 MB/s</td>
</tr>
</tbody>
</table>
Benefits & Challenges
Why care about Reproducibility?

• Good enough is not an excuse
  – We can always improve the state of our practice
  – How do we compare hardware/software in a scientific way?

• Experimental Cloud Infrastructure
  – PRObE / CloudLab / Chameleon
  – Having reproducible / validated experiments would represent a significant step toward embodying the scientific method as a core component of these infrastructures
Benefits of ESF-based methodology

• Brings falsiibility to our field
  – Statements can be proven false

• Automate validation
  – Validation becomes an objective task
Validation Workflow

- Obtain/recreate means of experiment.
- Re-run and check validation clauses against output. Any validation failed?
  - no: Original work findings are corroborated
  - yes: Any significant differences between original and recreated means?
    - yes: Update means of experiment
    - no: Cannot validate original claims
Benefits of ESF-based methodology

• Brings falsibility to our field
  – Statements can be proven false
• Automate validation
  – Validation becomes an objective task
• Usability
  – We all do this anyway, albeit in an ad-hoc way
• Integrate into existing infrastructure
Integration with Existing Infrastructure

- Push code
- Test:
  - Unit
  - Integration
  - Validations

pull

push code
and
ESF

CloudLab

Chameleon

ProbE

Parallel Reconfigurable Observational Environment
Challenges

• Reproduce every time
  – Include sanity checks as part of experiment
  – Alternative: corroborate that network/disk observes expected behavior at runtime

• Reproduce everywhere
  – Example: GCC’s flags, $10^{806}$ combinations
  – Alternative: provide image of complete software stack (e.g. Linux containers)
Conclusion

ESFs:

• Embody all components of an experiment
• Enable automation of result validation
• Brings us closer to the scientific method
• Our ideal future:
  – Researchers use ESFs to express an hypothesis
  – Toolkits for ESFs produce metadata-rich figures
  – Machine-readable evaluation section

https://github.com/systemslab/esf
Thanks!
Validations

The high random read speed of flash drives means that the CPU budget available for each index operation is relatively limited. This microbenchmark demonstrates that SILT’s indexes meet their design goal of computation-efficient indexing.

<table>
<thead>
<tr>
<th>Type</th>
<th>Cuckoo hashing (K keys/s)</th>
<th>Trie (K keys/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual insertion</td>
<td>10182</td>
<td>–</td>
</tr>
<tr>
<td>Bulk insertion</td>
<td>–</td>
<td>7603</td>
</tr>
<tr>
<td>Lookup</td>
<td>1840</td>
<td>208</td>
</tr>
</tbody>
</table>

Table 5: In-memory performance of index data structures in SILT on a single CPU core.
In this section, our goal is to evaluate the performance benefits that can be reaped, by utilizing virtual partitioning to apply otherwise expensive protection mechanisms on the most exposed part of applications. This allows us to strike a balance between the overhead imposed on the application and its exposure to attacks.
In this section, our goal is to evaluate the performance benefits that can be reaped, by utilizing virtual partitioning to apply otherwise expensive protection mechanisms on the most exposed part of applications. This allows us to strike a balance between the overhead imposed on the application and its exposure to attacks.

Experiment Goal
Schema

[Bar chart showing total time in seconds for different methods: Native, Pin, ISR, DTA, DTA/Pin, DTA/ISR. The chart indicates that DTA/ISR has the longest total time, followed by DTA, and Native has the shortest.]
Schema

"independent_variables": [ 
  {
    "type": "method",
    "alias": ["technique"],
    "values": [
      "native", "pin", "isr", "dta",
      "dta_pin", "dta_isr"
    ]
  }
],
"dependent_variable": {
  "type": "runtime",
  "scale": "s"
},

![Bar chart showing total time for different conditions]
Validations

```
"independent_variables": [{
  "type":   "method",
  "alias":  ["technique"],
  "values": [
    "native", "pin", "isr", "dta",
    "dta_pin", "dta_isr"
  ]
}
],
"dependent_variable": {
  "type":  "runtime",
  "scale": "s"
}
```
Validations

```
expect
native < any
```

```
"independent_variables": [
{
   "type": "method",
   "alias": ["technique"],
   "values": [
      "native", "pin", "isr", "dta",
      "dta_pin", "dta_isr"
   ]
}
],
"dependent_variable": {
   "type": "runtime",
   "scale": "s"
},
```

![Bar chart showing total time in seconds for different methods: Native, Pin, ISR, DTA, DTA/Pin, DTA/ISR. The chart indicates that DTA/ISR has the highest total time, followed by DTA, with native and pin having the least.](chart.png)
Validations

```
expect
native < any and
```

```
"independent_variables": [ 
  {
    "type":  "method",
    "alias":  ["technique"],
    "values": [ 
      "native", "pin", "isr", "dta",
      "dta_pin", "dta_isr"
    ]
  }
],
"dependent_variable": { 
  "type":  "runtime",
  "scale": "s"
},
```
Validations

```
expect
native < any and
dta_pin between pin and isr
```

```
"independent_variables": [
{
    "type": "method",
    "alias": ["technique"],
    "values": [
        "native", "pin", "isr", "dta",
        "dta_pin", "dta_isr"
    ]
}
],
"dependent_variable": {
    "type": "runtime",
    "scale": "s"
},
```
Validations

\[
\text{expect} \quad \text{native} < \text{any and}
\quad \text{dta\_pin} \quad \text{between pin and isr and}
\]
Validations

expect
native < any and
dta_pin between pin and isr and
dta_isr between isr and dta

"independent_variables": [
  {
    "type": "method",
    "alias": ["technique"],
    "values": [
      "native", "pin", "isr", "dta",
      "dta_pin", "dta_isr"
    ]
  }
],
"dependent_variable": {
  "type": "runtime",
  "scale": "s"
},
Example 2
Example 2
Schema

"independent_variables": [ {
    "type":   "method",
    "values": [ 
        "native", "pin", "isr", "dta",
        "dta_pin", "dta_isr"
    ]
} ],
"dependent_variable": { 
    "type":   "runtime",
    "scale": "s"
}
}
Schema

"independent_variables": [  
  {  
    "type": "method",  
    "values": [  
      "native", "pin", "isr", "dta",  
      "dta_pin", "dta_isr"  
    ]  
  },  
  {  
    "type": "workload",  
    "values": ["ftp", "samba", "ssh"]  
  }  
],
"dependent_variable": {  
  "type": "runtime",  
  "scale": "s"  
},
Validations

for workload=*
expect native < any and dta_pin between pin and isr and dta_isr between isr and dta

"independent_variables": [  
  {  
    "type": "method",  
    "values": [  
      "native", "pin", "isr", "dta",  
      "dta_pin", "dta_isr"  
    ]  
  },  
  {  
    "type": "workload",  
    "values": ["ftp", "samba", "ssh"]  
  }  
],
"dependent_variable": {  
  "type": "runtime",  
  "scale": "s"  
}
Falsifiability in Science

Falsifiability of a statement, hypothesis, or theory is an inherent possibility to prove it to be false.

• In other words, the ability to specify the conditions under which a statement is false
• Synonymous to Testability
• Example:
  – Statement: All swans are white
  – Falsifiable: Observe one black swan

source: en.wikipedia.org/wiki/Falsifiability
Geologic time scale, 650 million years ago to the present

<table>
<thead>
<tr>
<th>Era</th>
<th>Period</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cenozoic</td>
<td>Quaternary</td>
<td>evolution of humans</td>
</tr>
<tr>
<td></td>
<td>Tertiary</td>
<td>mammals diversify</td>
</tr>
<tr>
<td></td>
<td>Cretaceous</td>
<td>extinction of dinosaurs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>first primates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>first flowering plants</td>
</tr>
<tr>
<td></td>
<td>Jurassic</td>
<td>first birds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dinosaurs diversify</td>
</tr>
<tr>
<td></td>
<td>Triassic</td>
<td>first mammals</td>
</tr>
<tr>
<td></td>
<td></td>
<td>first dinosaurs</td>
</tr>
<tr>
<td></td>
<td>Permian</td>
<td>major extinctions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reptiles diversify</td>
</tr>
<tr>
<td>Paleozoic</td>
<td>Carboniferous</td>
<td>first reptiles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>scale trees</td>
</tr>
<tr>
<td></td>
<td></td>
<td>seed ferns</td>
</tr>
<tr>
<td></td>
<td>Pennsylvanian</td>
<td>first amphibians</td>
</tr>
<tr>
<td></td>
<td>Mississippian</td>
<td>jawed fishes diversify</td>
</tr>
<tr>
<td></td>
<td>Devonian</td>
<td>first vascular land plants</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sudden diversification of metazoan families</td>
</tr>
<tr>
<td></td>
<td>Ordovician</td>
<td>first fishes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>first chordates</td>
</tr>
<tr>
<td></td>
<td>Cambrian</td>
<td>first skeletal elements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>first soft-bodied metazoans</td>
</tr>
<tr>
<td></td>
<td></td>
<td>first animal traces</td>
</tr>
</tbody>
</table>
Falsifiability in Systems

**Experiment Goal:** Show that my algorithm/system/etc. is better than the state-of-the-art.

**Means of Experiment**

**Raw data**

Figure

**Observations**

Figure 5 illustrates the time required to complete MySQL’s test-insert benchmark. Applying DTA and ISR on the server for the entire duration of the test increases execution time by 4.8x and 2.6x respectively, when compared to native execution. In contrast, partitioning slows down execution by 1.8x and 2.6x, when using DTA only for the non-authenticated part of the execution, and then switching to no instrumentation and ISR respectively. We observe that the overhead of applying DTA diminishes, as the unauthenticated partition runs only for a short period of time. In general, partitioned execution performs similarly to the mechanism applied on the authenticated partition.
Falsifiability in Systems

• To falsify a claim:
  – Describe the means of the experiments
  – Provide validation statements over the output data

• Conditional statement:
  – if means are properly recreated
  – then validation statements should hold

• Go from inert observations to falsifiable statements
  From:
  \textit{We observe that our system outperforms the alternatives}
  To:
  \textit{Expect 25-30\% performance improvement on hardware platform X, on workload Y, when configured like Z}
Early Feedback

Creating an ESF helps authors to:

• Find meaningful/reproducible baselines
• Create a feedback loop in author’s mind
• Specify exactly what author means
• Make temporal context explicit