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…

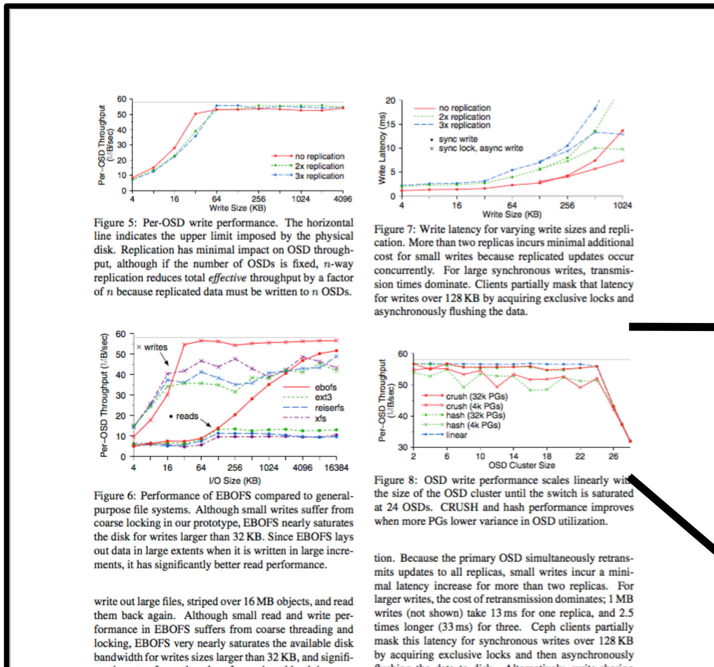


Figure 5: Per-OSD write performance. The horizontal line indicates the upper limit imposed by the physical disk. Replication has minimal impact on OSD throughput, although if the number of OSDs is fixed, n-way replication reduces total *effective* throughput by a factor of n because replicated data must be written to n OSDs.

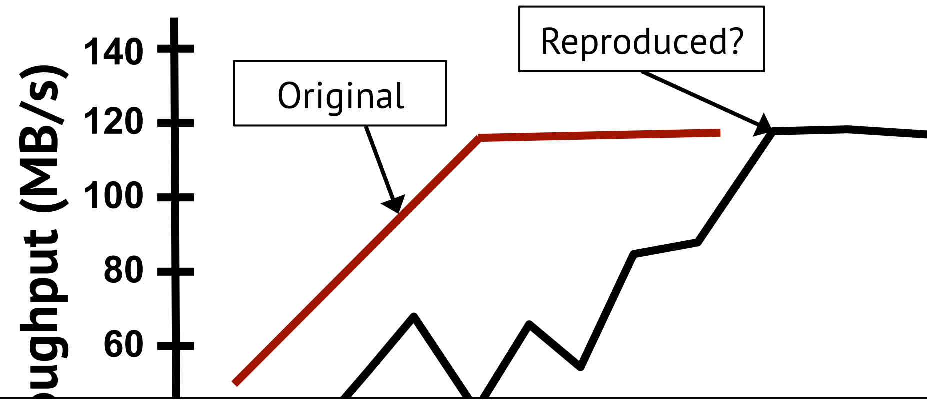
Figure 6: Performance of EBOFS compared to general-purpose file systems. Although small writes suffer from coarse locking in our prototype, EBOFS nearly saturates the disk for writes larger than 32 KB. Since EBOFS lays out data in large extents when it is written in large increments, it has significantly better read performance.

write out large files, striped over 16 MB objects, and read them back again. Although small read and write performance in EBOFS suffers from coarse threading and locking, EBOFS very nearly saturates the available disk bandwidth for writes sizes larger than 32 KB, and significantly outperforms the others for read workloads because

Figure 7: Write latency for varying write sizes and replication. More than two replicas incurs minimal additional cost for small writes because replicated updates occur concurrently. For large synchronous writes, transmission times dominate. Clients partially mask that latency for writes over 128 KB by acquiring exclusive locks and asynchronously flushing the data.

Figure 8: OSD write performance scales linearly with the size of the OSD cluster until the switch is saturated at 24 OSDs. CRUSH and hash performance improves when more PGs lower variance in OSD utilization.

tion. Because the primary OSD simultaneously retransmits updates to all replicas, small writes incur a minimal latency increase for more than two replicas. For larger writes, the cost of retransmission dominates; 1 MB writes (not shown) take 13 ms for one replica, and 2.5 times longer (33 ms) for three. Ceph clients partially mask this latency for synchronous writes over 128 KB by acquiring exclusive locks and then asynchronously flushing the data to disk. Alternatively, write-sharing



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?

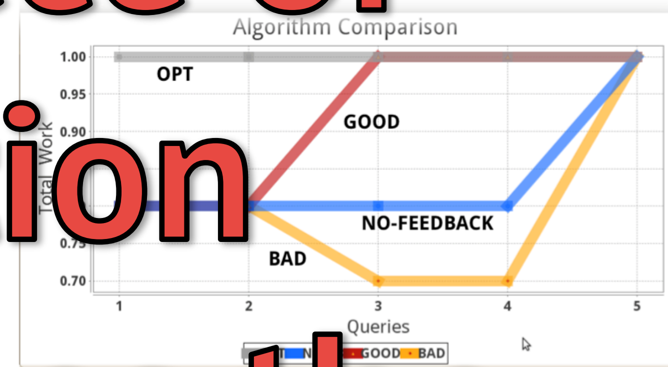
Need a piece of information that describes the relationship between these

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 illus... the time required to complete MySQL's test- ... 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.

Outline

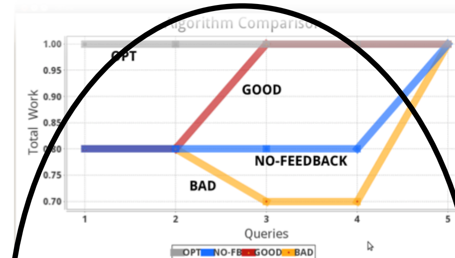
- Re-execution vs. validation
- **Declarative Experiment Specification (ESF)**
- Case Study
- Benefits & Challenges

```yaml
goal_location:
  sec: '6.1'
  par: '5'
goal_text: >
    demonstrate that Ceph scales linearly with
    the size of the cluster
goal_category: 'proof-of-concept'
experiments:
- reference: 'figure-8'
  name: 'scalability experiment'
  tags: [ 'throughput' ]
  hardware_dependencies:
  - type: 'hdd'
    bw: '58 MB/s'
  - type: 'network'
    bw: '1GbE'
  software_dependencies:
  - type: 'os'
    kernel: 'linux 2.6.32'
    distro: 'debian 6.0'
  - type: 'storage'
    name: 'ceph'
    version: '0.1.67'
  workload:
  - type: 'rados-benchmark'
    configuration:
      object-size: '4mb'
      time: '120s'
      threads: 16
      mode: 'write'
  independent_variables:
  - type: method
    values: [ 'raw', 'ceph' ]
    desc: >
      raw corresponds to hdd sequential write
      performance, expressed in MB/s
  - type: 'size'
    values: [ '2-24', '2' ]
  dependent_variable:
  - type: 'throughput'
    scale: 'mb/s'
  statistical_functions:
    functions: ['avg', 'stddev']
    repetitions: 10
  validations:
  - >
    for
      size=*
    expect
      ceph >= (raw * 0.9)
```


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

Means of Experiment


```
Src,Eqid,Version,Datetime,Lat,Lon,Magnitude,Depth,NST,Region
ci,14692356,1,"Tuesday, May  4, 2010 03:21:38 UTC",32.6443,-1
ci,14692348,1,"Tuesday, May  4, 2010 03:19:38 UTC",32.1998,-1
ci,14692332,1,"Tuesday, May  4, 2010 03:16:56 UTC",32.6756,-1
ci,14692324,1,"Tuesday, May  4, 2010 03:08:47 UTC",32.6763,-1
ci,14692316,1,"Tuesday, May  4, 2010 03:08:08 UTC",32.6778,-1
ci,14692308,1,"Tuesday, May  4, 2010 03:06:20 UTC",32.7071,-1
ci,14692300,1,"Tuesday, May  4, 2010 03:01:52 UTC",32.1948,-1
ak,10047267,1,"Tuesday, May  4, 2010 03:01:04 UTC",61.2695,-1
ci,14692284,1,"Tuesday, May  4, 2010 02:58:51 UTC",32.7016,-1
ci,14692276,1,"Tuesday, May  4, 2010 02:57:46 UTC",32.6998,-1
ak,10047263,1,"Tuesday, May  4, 2010 02:56:28 UTC",63.5779,-1
ak,10047261,1,"Tuesday, May  4, 2010 02:52:00 UTC",60.4986,-1
ci,14692268,1,"Tuesday, May  4, 2010 02:48:40 UTC",32.6813,-1
ci,14692260,1,"Tuesday, May  4, 2010 02:35:27 UTC",32.2006,-1
nc,71392116,0,"Tuesday, May  4, 2010 02:15:24 UTC",38.8415,-1
ci,14692244,1,"Tuesday, May  4, 2010 02:05:07 UTC",33.5248,-1
ci,14692228,1,"Tuesday, May  4, 2010 01:57:08 UTC",32.6823,-1
ci,14692220,1,"Tuesday, May  4, 2010 01:53:28 UTC",32.6881,-1
ci,14692212,1,"Tuesday, May  4, 2010 01:48:53 UTC",32.6398,-1
ci,14692188,1,"Tuesday, May  4, 2010 01:26:58 UTC",32.5003,-1
ci,14692180,1,"Tuesday, May  4, 2010 01:19:44 UTC",32.6836,-1
ci,14692172,1,"Tuesday, May  4, 2010 01:12:01 UTC",32.5321,-1
ci,14692164,1,"Tuesday, May  4, 2010 01:08:24 UTC",32.6833,-1
```

Multiple instances of WFIT running in para

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 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.

Validation Language Syntax

```
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

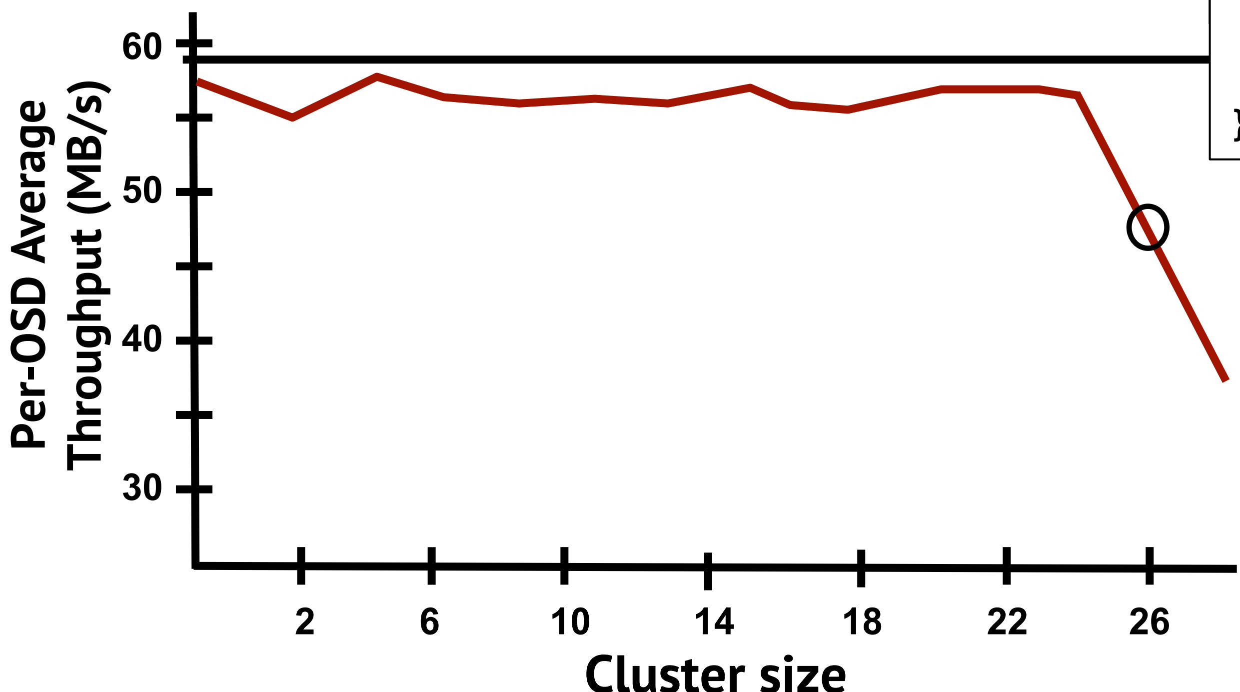
Ceph OSDI '06

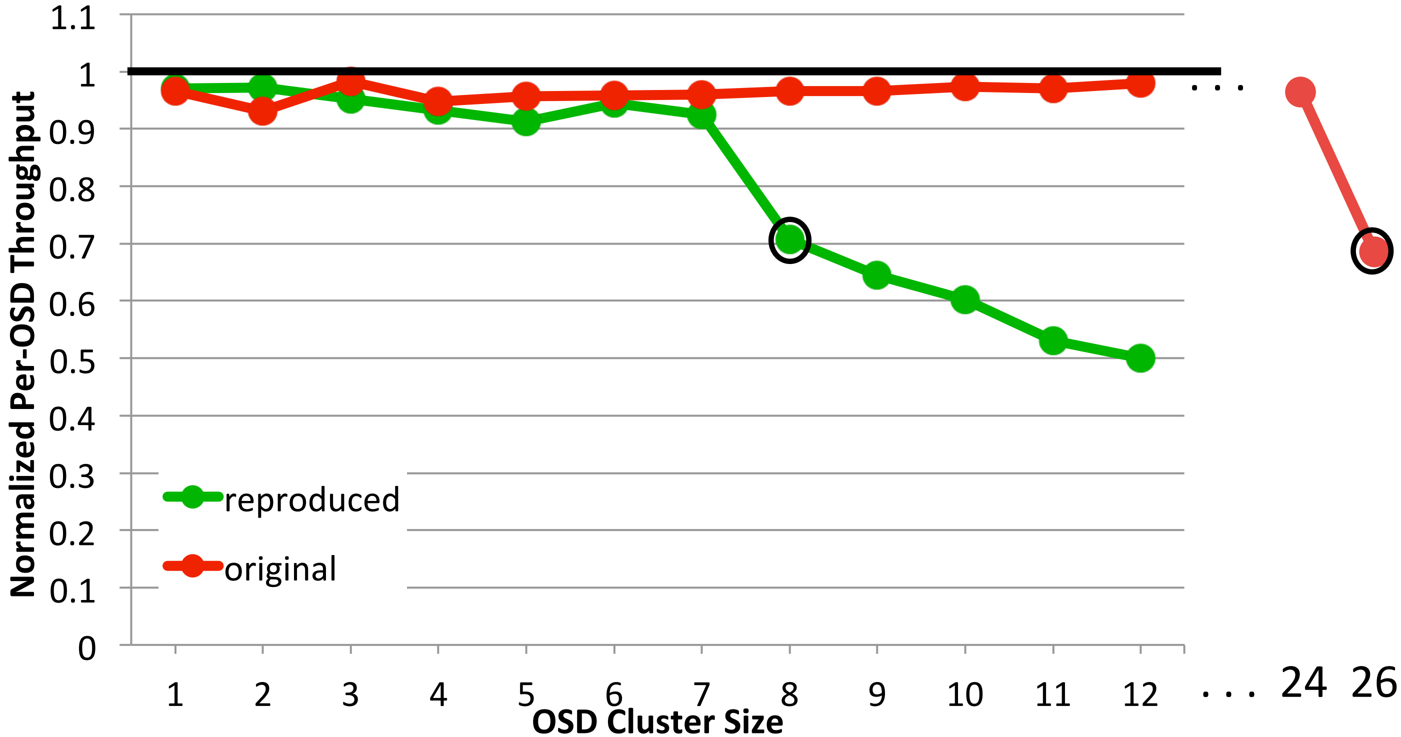
- Select scalability experiment.
  - Distributed; makes use of all resources
  - Main bottlenecks: I/O and network
- Why this experiment?
  - Top conference
  - 10 year old experiment
  - Ideal reproducibility conditions
    - Access to authors, topic familiarity, same hardware,
  - Even in an ideal 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"
},
```



Component	Original	Reproduced
CPU	AMD 2212 @2.0GHz	Intel E5-2630 @2.3GHz
Disk drive	Seagate ST3250620NS	HP 6G 658071-B21
Disk BW	58 MB/s	120 MB/s (15 MB/s limit)
Linux	2.6.9	3.13.0
Ceph	commit from 2005	0.87.1
Storage	26 nodes	12 nodes
Clients	20 nodes	1 node
Network	Netgear GS748T	Same as original
Network BW	1400 MB/s	110 MB/s

Benefits & Challenges

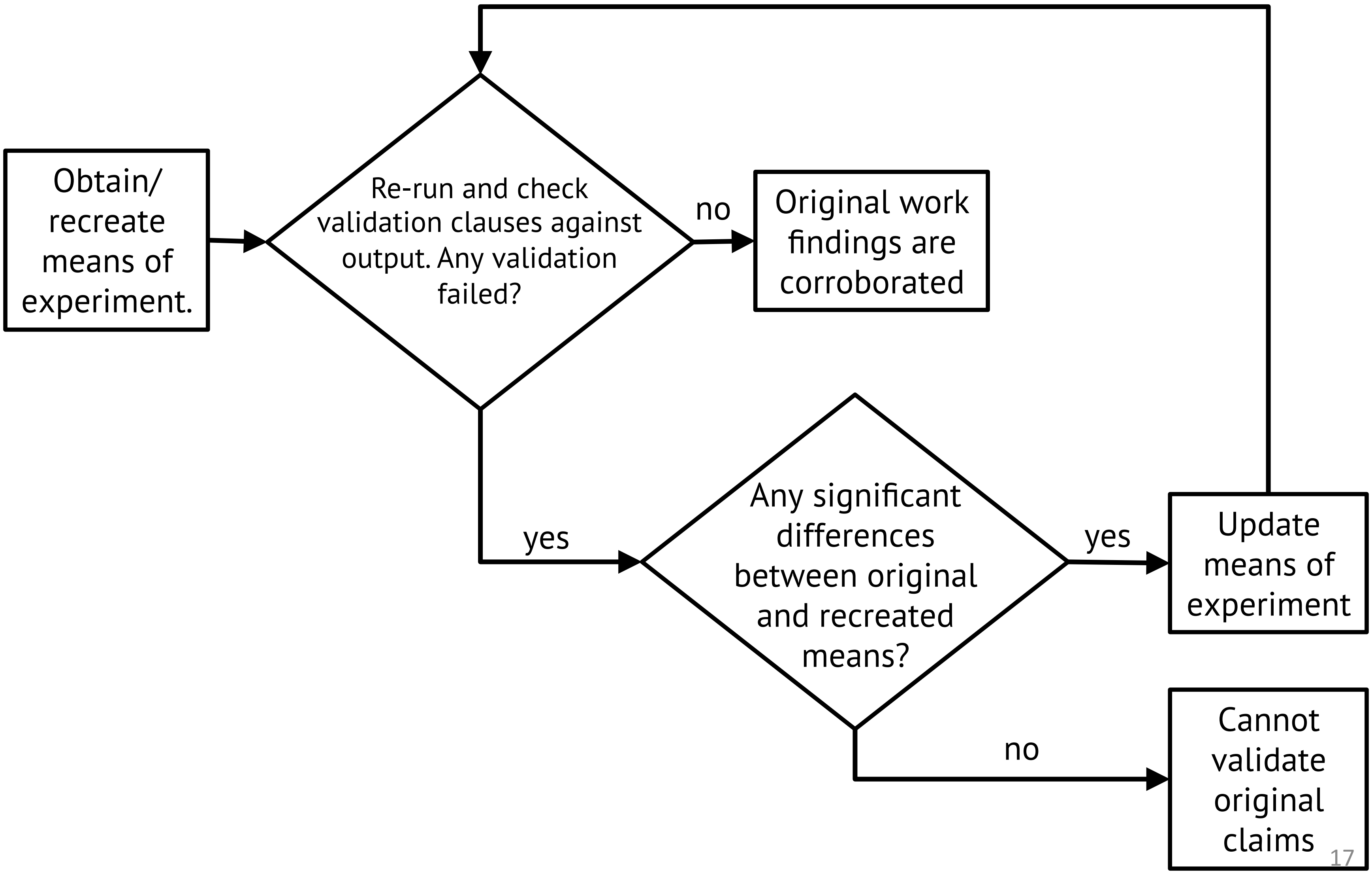
Why care about Reproducibility?

- 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 falsibiability to our field
  - Statements can be proven false
- Automate validation
  - Validation becomes an objective task

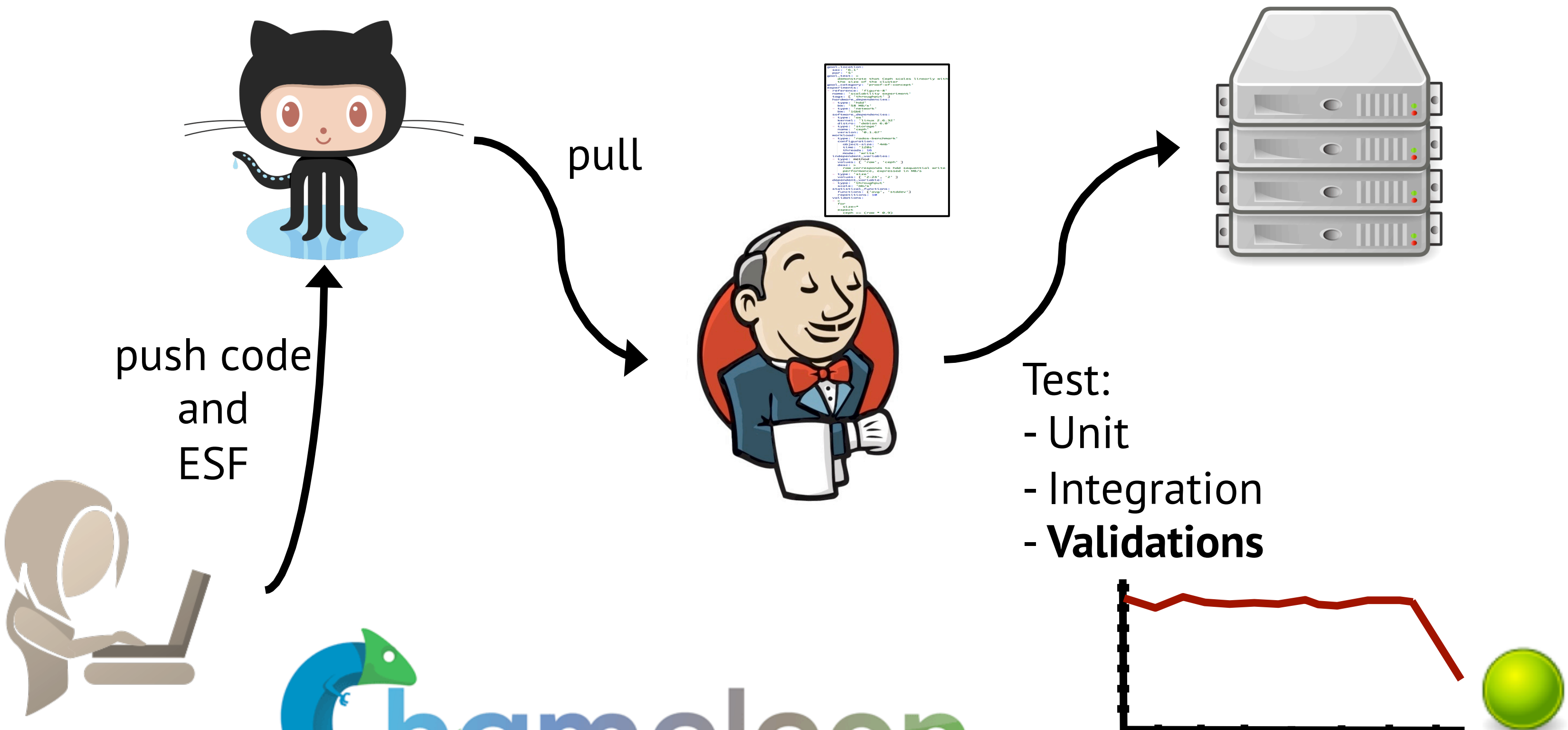
Validation Workflow



Benefits of ESF-based methodology

- Brings falsibiability 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



Chameleon

CloudLab

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

```
for
 workload=*
expect
 cuckoo > raw and trie > raw
for
 lookup
expect
 cuckoo > trie
and
for
 individual and bulk
expect
  cuckoo > trie
```

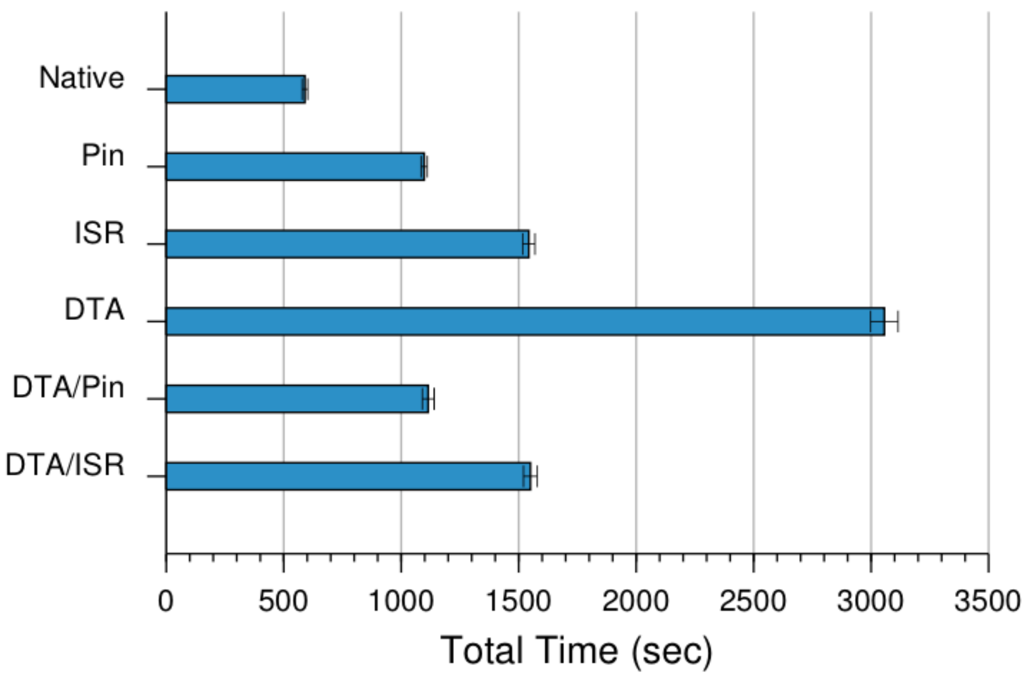
The h
flash
budg
opera
This
demo
meet
comp


```
{
  "type": ”method”,
  "values": [“raw”, "cuckoo", "trie"]
},
{
  "type": "workload",
  "values": [
    "individual", "bulk", "lookup”
  ]
},
"dependent_variable": {
  "type":  "throughput",
  "scale": ”bytes/second"
}
```


Type	Cuckoo hashing (K keys/s)	Trie (K keys/s)
Individual insertion	10182	–
Bulk insertion	–	7603
Lookup	1840	208

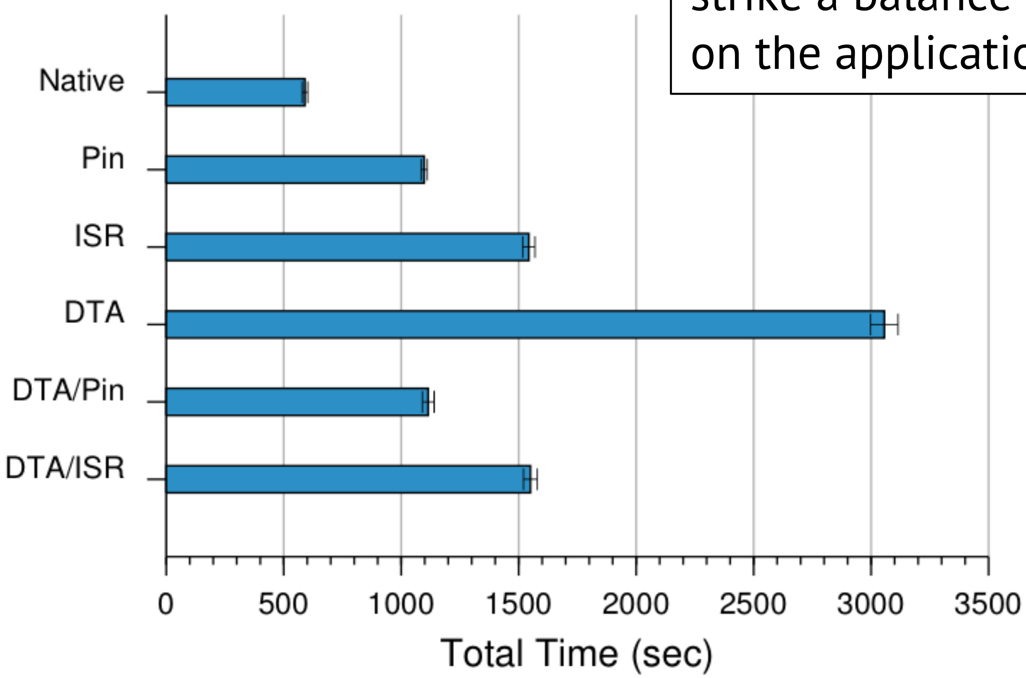
Table 5: In-memory performance of index data structures in SILT on a single CPU core.

Geneiatakis et. al. CCS ‘12



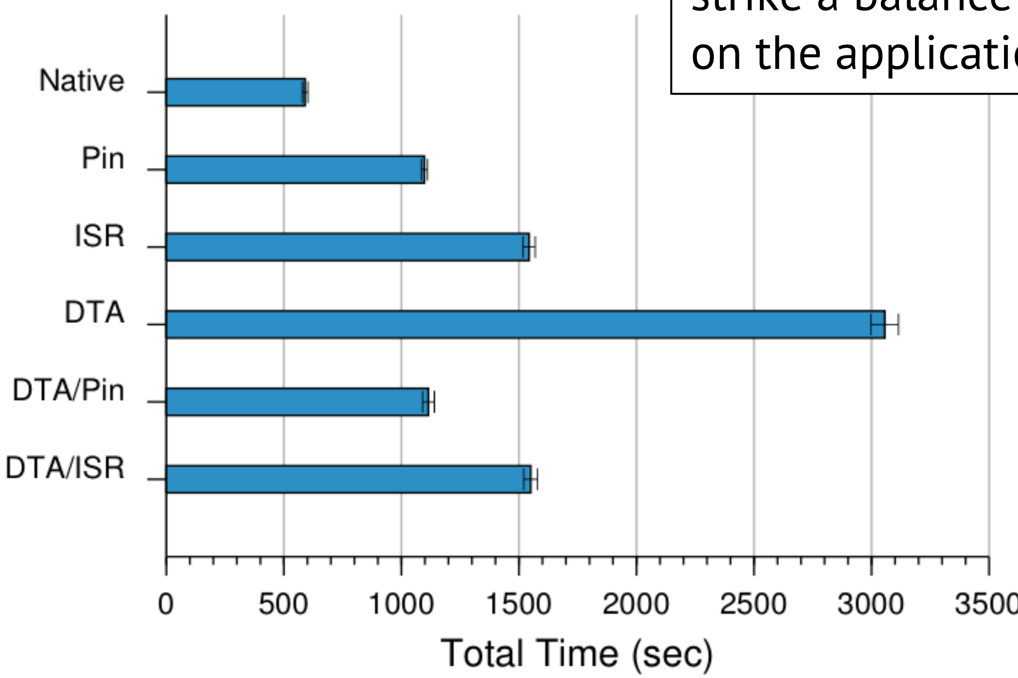
Experiment Goal

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.

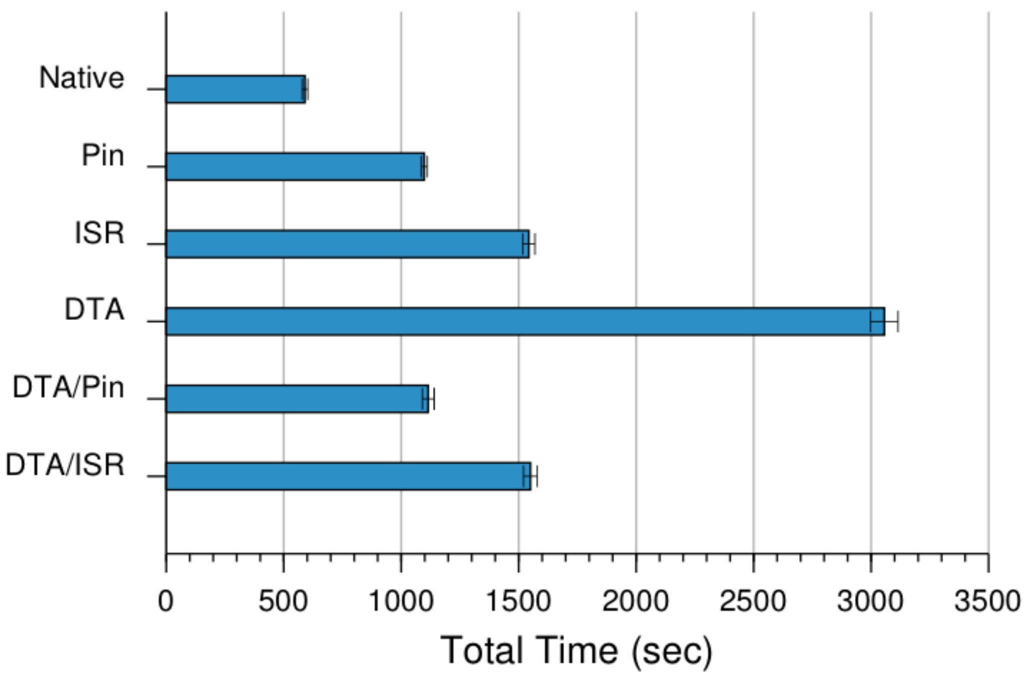


Experiment Goal

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.



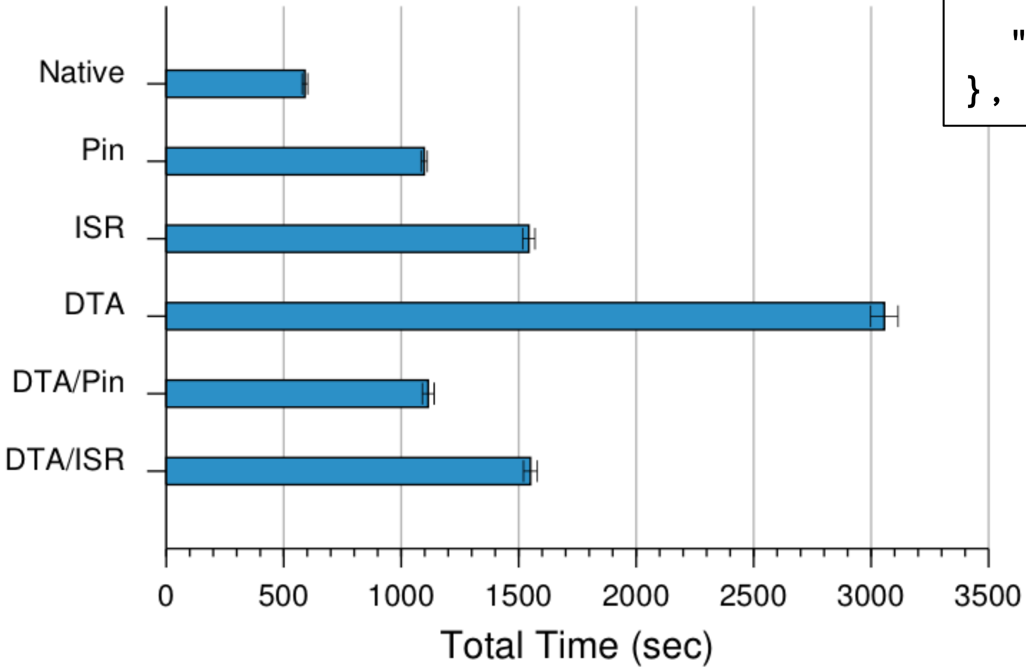
Schema



Native
Pin
ISR
DTA
DTA/Pin
DTA/ISR
0
500
1000
1500
2000
2500
3000
3500
Total Time (sec)

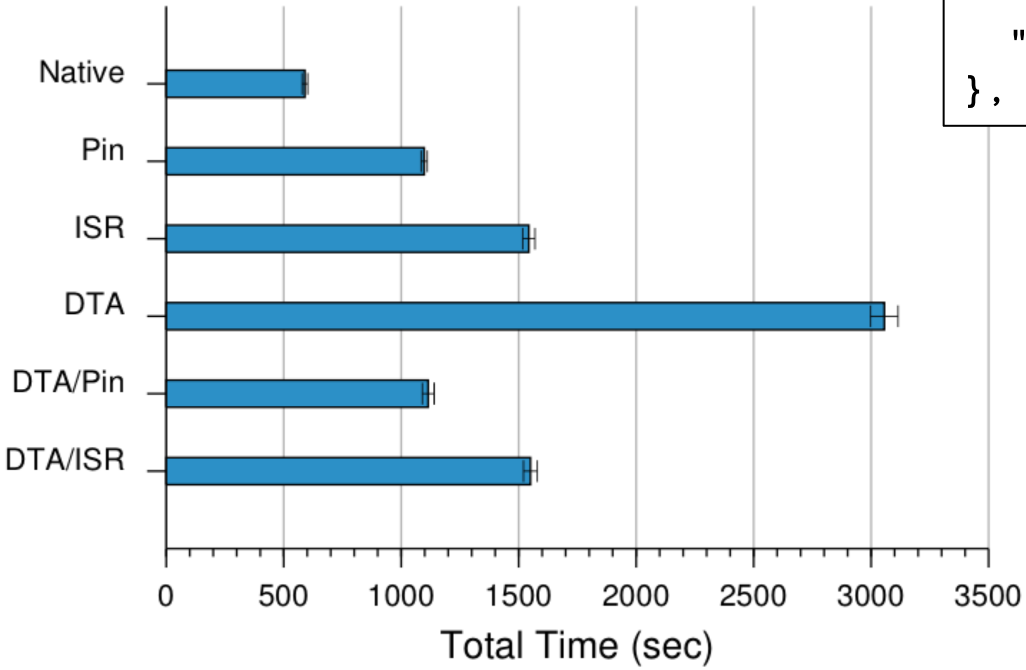
Schema

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



Validations

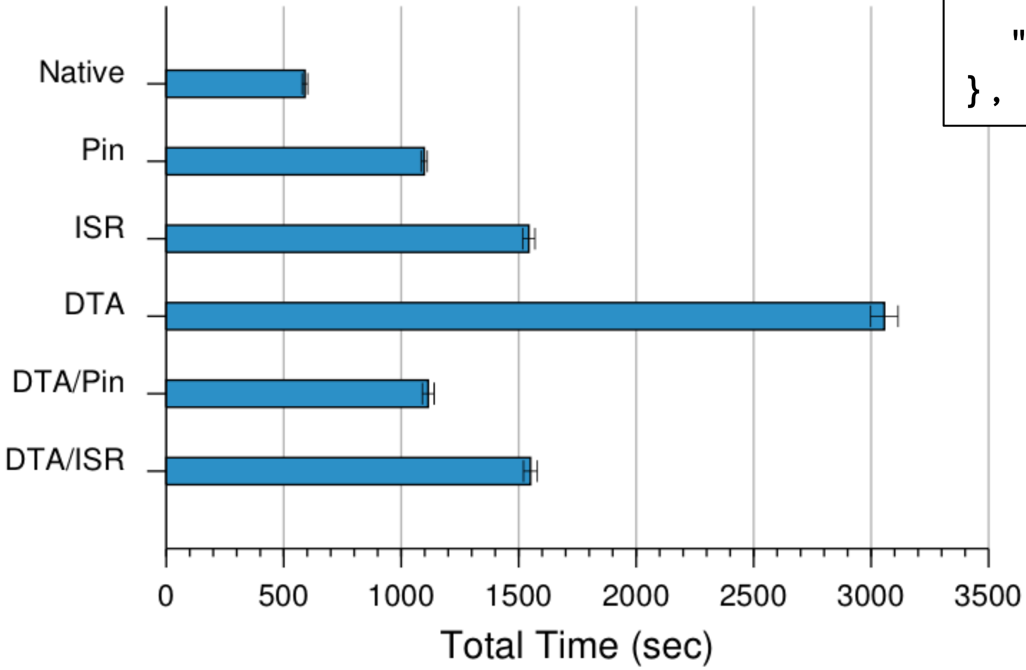

```
"independent_variables": [
  {
    "type":   "method",
    "alias":  ["technique"],
    "values": [
      "native", "pin", "isr", "dta",
      "dta_pin", "dta_isr"
    ]
  }
],
"dependent_variable": {
  "type":  "runtime",
  "scale": "s"
},
```

Validations

```
expect
 native < any
```

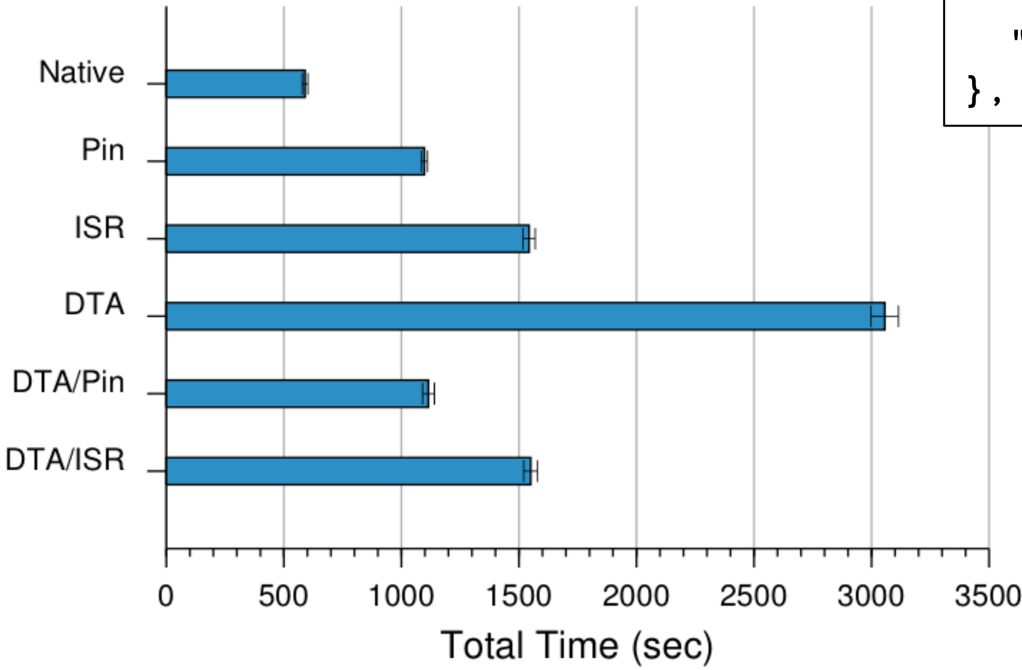

```
"independent_variables": [
  {
    "type":   ”method",
    "alias":  [”technique”],
    "values": [
      “native", “pin", “isr”, “dta”,
      “dta_pin”, “dta_isr”
    ]
  }
],
"dependent_variable": {
  "type":  ”runtime",
  "scale": “s"
},
```

Validations

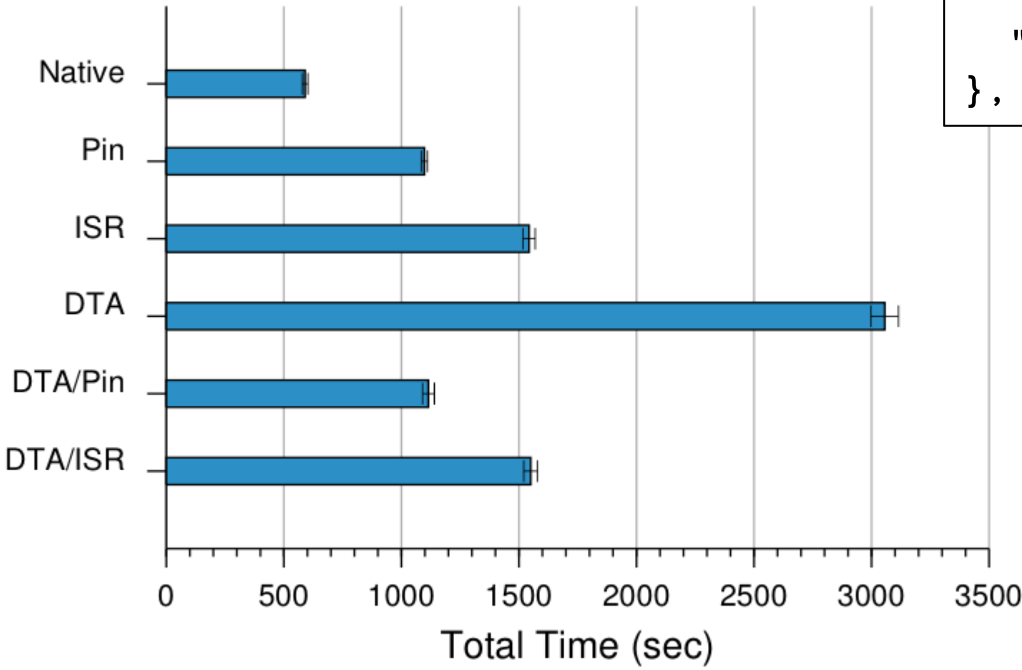
```
expect
 native < any and
```


```
"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
```

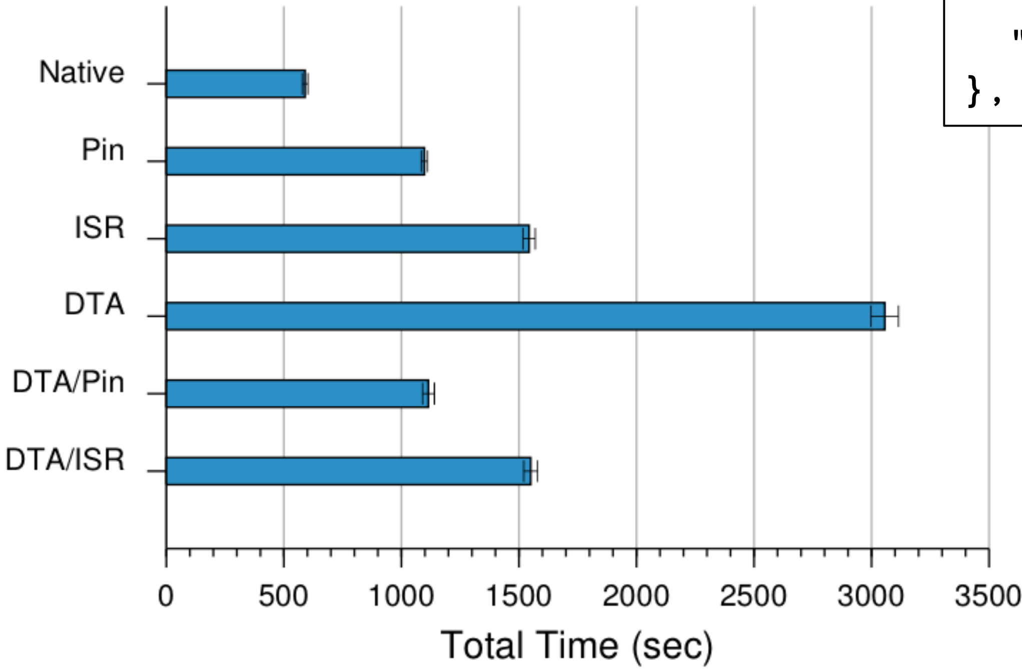



```
"independent_variables": [
  {
    "type":   ”method",
    "alias":  [”technique”],
    "values": [
      “native", “pin", “isr”, “dta”,
      “dta_pin”, “dta_isr”
    ]
  }
],
"dependent_variable": {
  "type":  ”runtime",
  "scale": “s"
},
```

Validations

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

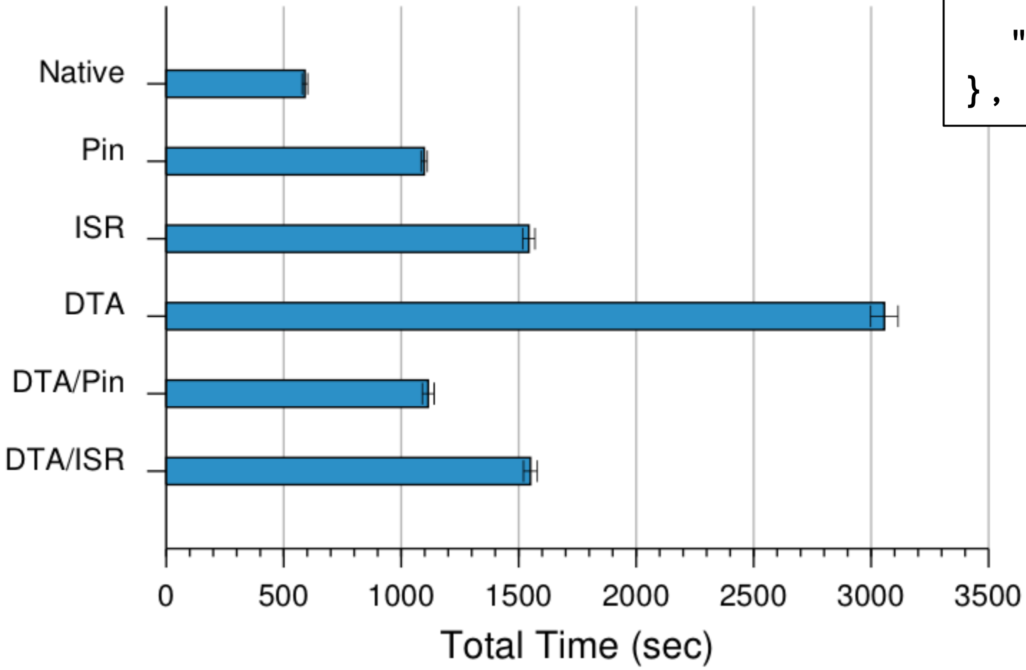
```
"independent_variables": [
  {
    "type":   ”method",
    "alias":  [”technique”],
    "values": [
      “native", “pin", “isr”, “dta”,
      “dta_pin”, “dta_isr”
    ]
  }
],
"dependent_variable": {
  "type":  ”runtime",
  "scale": “s"
},
```



Validations

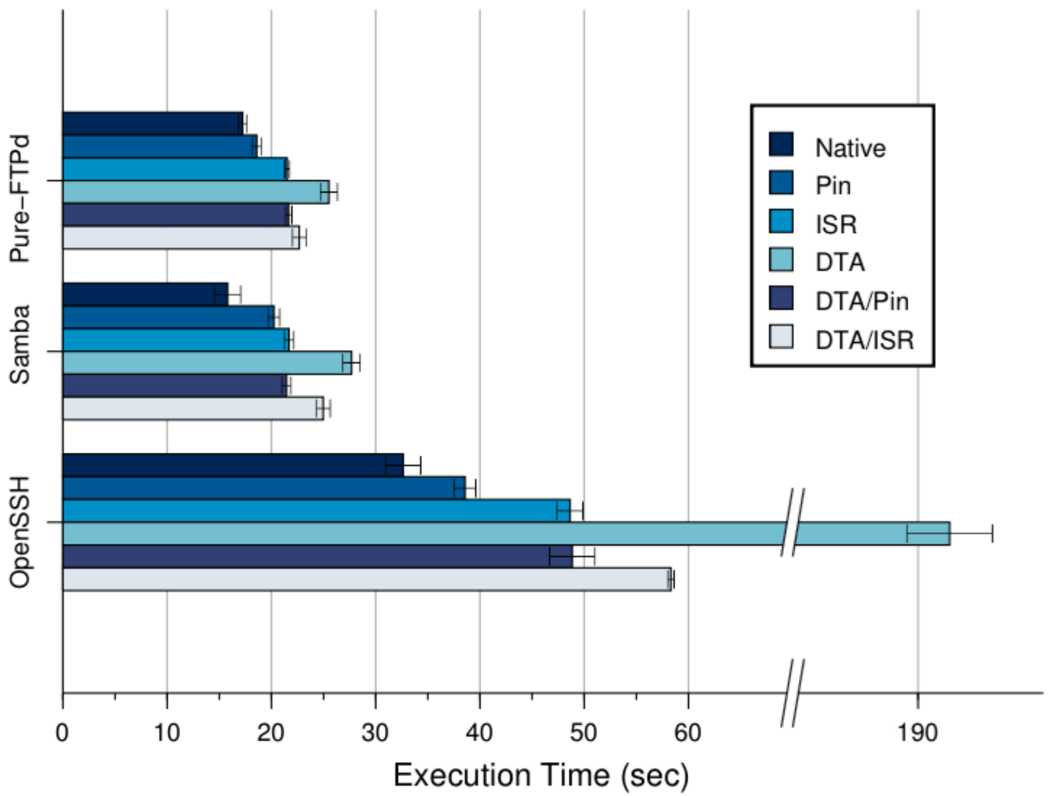
```
expect
 native < any and
 dta_pin between pin and isr and
 dta_isr between isr and dta
```


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

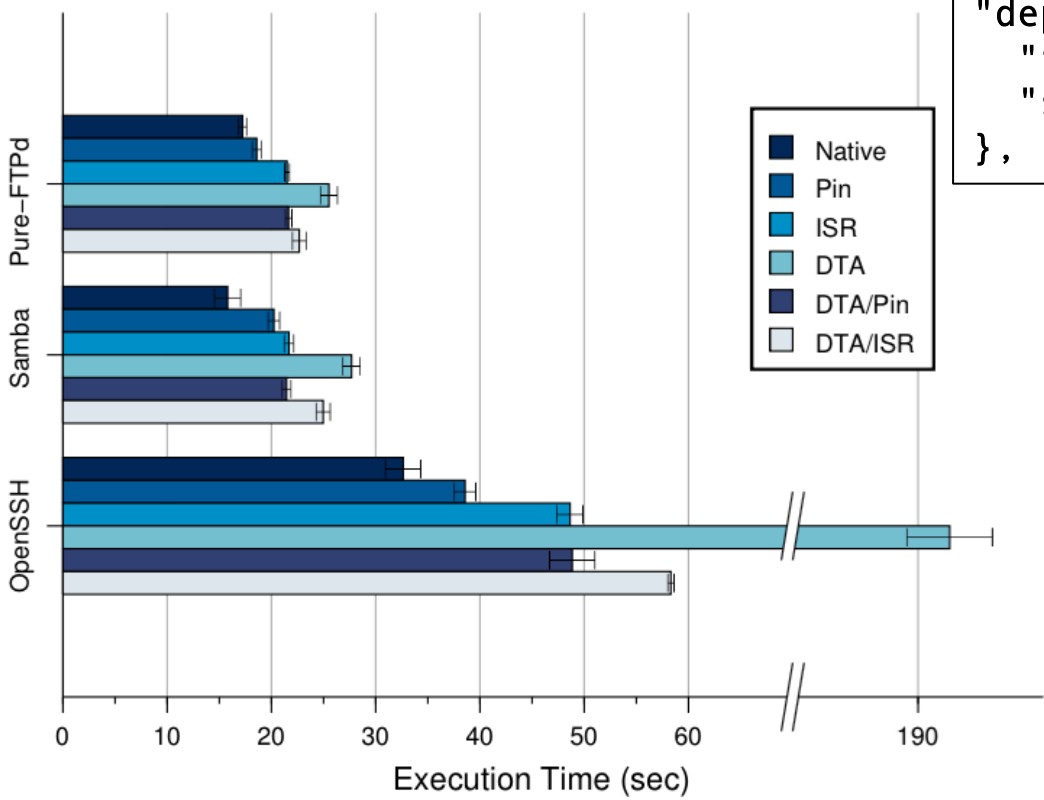
Example 2

Example 2

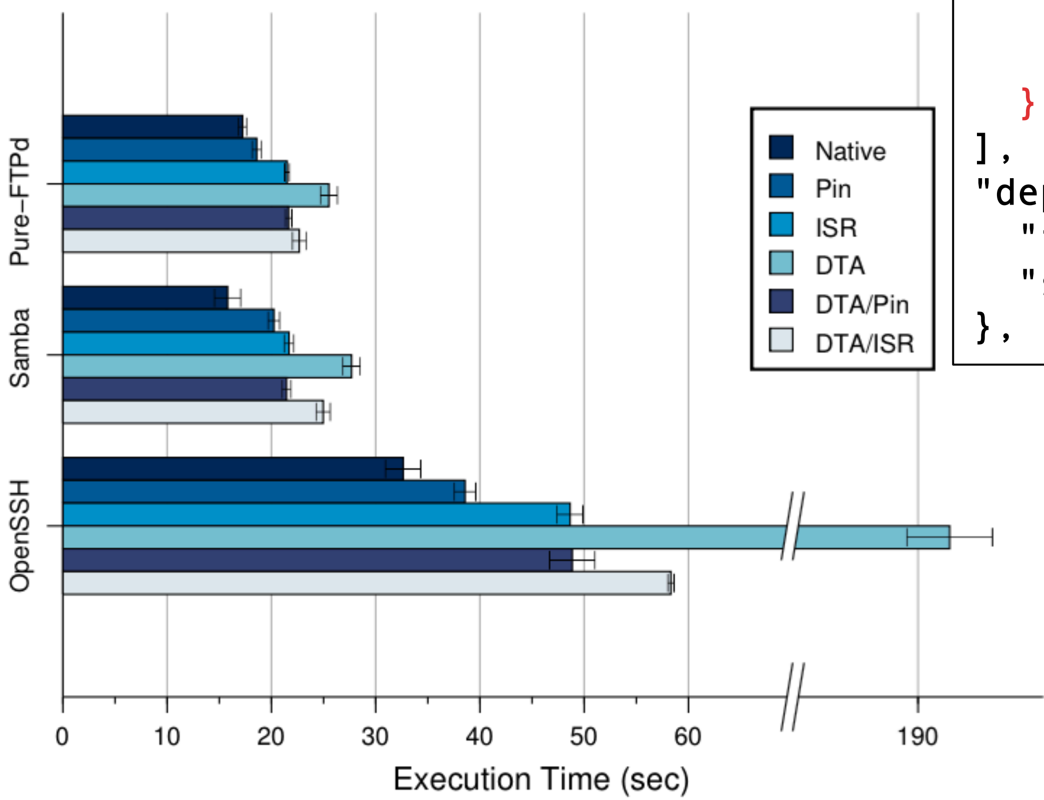

Native
Pin
ISR
DTA
DTA/Pin
DTA/ISR
Pure-FTPd
Samba
OpenSSH
0
10
20
30
40
50
60
190
Execution Time (sec)

Schema


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

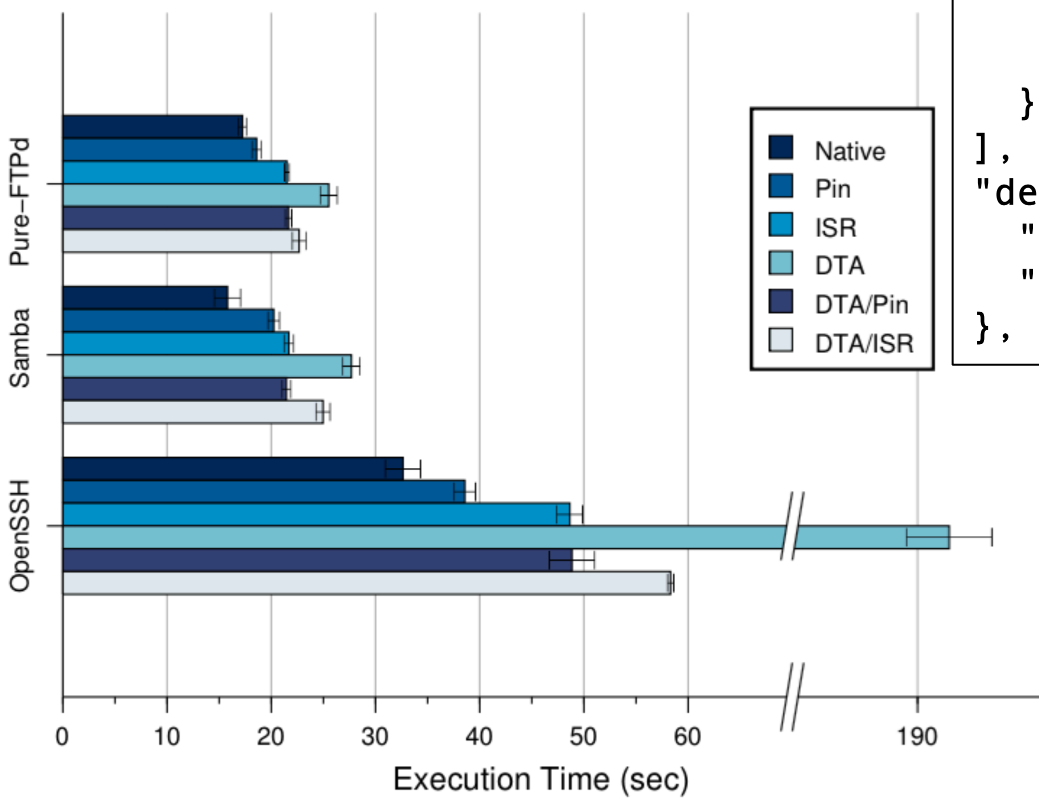
Schema




```
"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
```



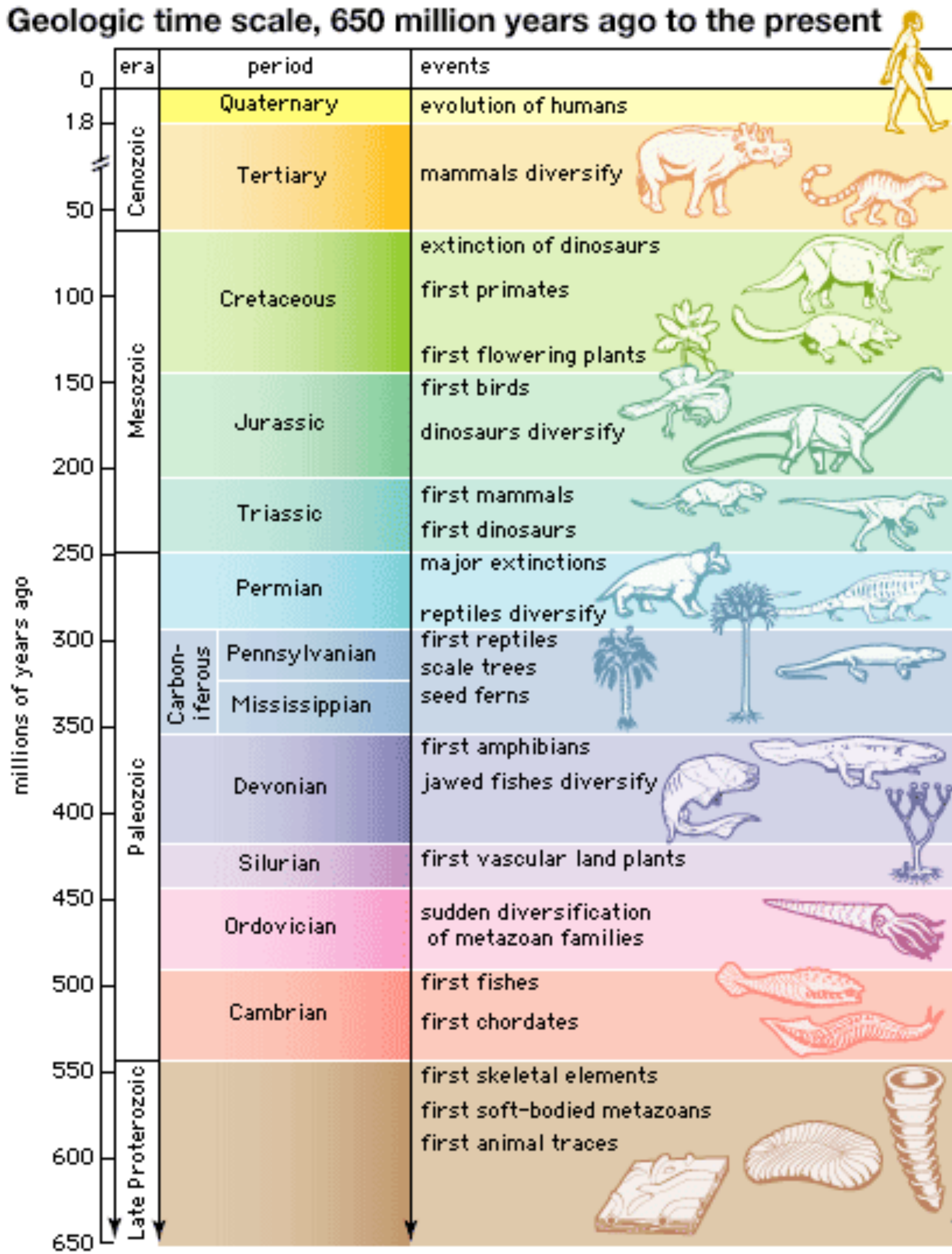

```
"independent_variables": [
  {
    "type":   ”method”,
    "values": [
      “native", “pin", “isr”, “dta”,
      “dta_pin”, “dta_isr”
    ]
  },
  {
    "type":   ”workload”,
    "values": [“ftp", “samba", “ssh”]
  }
],
"dependent_variable": {
  "type":  ”runtime",
  "scale": “s"
},
```

Falsifiability in Science

Falsibiability 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
era
period
events
0
1.8
50
100
150
200
250
300
350
400
450
500
550
600
650
millions of years ago
Cenozoic
Mesozoic
Paleozoic
Late Proterozoic
Quaternary
Tertiary
Cretaceous
Jurassic
Triassic
Permian
Carbon-iferous
Pennsylvanian
Mississippian
Devonian
Silurian
Ordovician
Cambrian
evolution of humans
mammals diversify
extinction of dinosaurs
first primates
first flowering plants
first birds
dinosaurs diversify
first mammals
first dinosaurs
major extinctions
reptiles diversify
first reptiles
scale trees
seed ferns
first amphibians
jawed fishes diversify
first vascular land plants
sudden diversification of metazoan families
first fishes
first chordates
first skeletal elements
first soft-bodied metazoans
first animal traces

Falsifiability in Systems

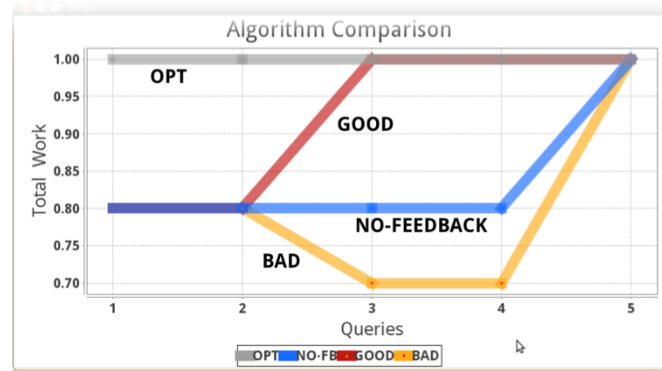
Experiment Goal: Show that my algorithm/system/etc. is better than the state-of-the-art.

Means of Experiment

Raw data

Src,Eqid,Version,Datetime,Lat,Lon,Magnitude,Depth,NST,Region
ci,14692356,1,"Tuesday, May 4, 2010 03:21:38 UTC",32.6443,-1
ci,14692348,1,"Tuesday, May 4, 2010 03:19:38 UTC",32.1998,-1
ci,14692332,1,"Tuesday, May 4, 2010 03:16:56 UTC",32.6756,-1
ci,14692324,1,"Tuesday, May 4, 2010 03:08:47 UTC",32.6763,-1
ci,14692316,1,"Tuesday, May 4, 2010 03:08:08 UTC",32.6778,-1
ci,14692308,1,"Tuesday, May 4, 2010 03:06:20 UTC",32.7071,-1
ci,14692300,1,"Tuesday, May 4, 2010 03:01:52 UTC",32.1948,-1
ak,10047267,1,"Tuesday, May 4, 2010 03:01:04 UTC",61.2695,-1
ci,14692284,1,"Tuesday, May 4, 2010 02:58:51 UTC",32.7016,-1
ci,14692276,1,"Tuesday, May 4, 2010 02:57:46 UTC",32.6998,-1
ak,10047263,1,"Tuesday, May 4, 2010 02:56:28 UTC",63.5779,-1
ak,10047261,1,"Tuesday, May 4, 2010 02:52:00 UTC",60.4986,-1
ci,14692268,1,"Tuesday, May 4, 2010 02:48:40 UTC",32.6813,-1
ci,14692260,1,"Tuesday, May 4, 2010 02:35:27 UTC",32.2006,-1
nc,71392116,0,"Tuesday, May 4, 2010 02:15:24 UTC",38.8415,-1
ci,14692244,1,"Tuesday, May 4, 2010 02:05:07 UTC",33.5248,-1
ci,14692228,1,"Tuesday, May 4, 2010 01:57:08 UTC",32.6823,-1
ci,14692220,1,"Tuesday, May 4, 2010 01:53:28 UTC",32.6881,-1
ci,14692212,1,"Tuesday, May 4, 2010 01:48:53 UTC",32.6398,-1
ci,14692188,1,"Tuesday, May 4, 2010 01:26:58 UTC",32.5003,-1
ci,14692180,1,"Tuesday, May 4, 2010 01:19:44 UTC",32.6836,-1
ci,14692172,1,"Tuesday, May 4, 2010 01:12:01 UTC",32.5321,-1
ci,14692164,1,"Tuesday, May 4, 2010 01:08:24 UTC",32.6833,-1

Figure



4: Multiple instances of WFIT running in paralle

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:

We observe that our system outperforms the alternatives

To:

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