Avoiding the Ordering Trap in Systems Performance Measurement

Dmitry Duplyakin[*] Nikhil Ramesh[*] Carina Imburgia[^]
Hamza Fathallah Al Sheikh[*] Semil Jain[*] **Prikshit Tekta**[*]
Aleksander Maricq[*] Gary Wong[*] Robert Ricci[*]

[*] University of Utah [^] University of Washington

USENIX ATC 2023

Benchmarking story

While working on OSDI'18 "Taming Performance Variability" paper, we measured memory bandwidth (using STREAM) on CloudLab's `c220g2` servers, which had an unbalanced DIMM configuration.

Observation 1:

Results from older (balanced) servers were better by **3x**.

Observation 2:

Running a large CPU benchmark **before** STREAM "recovered" the memory bandwidth and increased STREAM's results by **3x**.

The order of benchmark execution may affect the benchmarking results.

More broadly

Performance tests may suggest that **system A is better than system B.**

Such a conclusion **may or may not hold true**

if A was always tested before B and A tests systematically impact B tests

The ordering trap

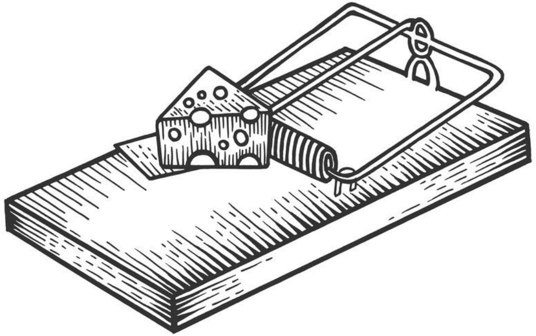
It is assumed that the results obtained from individual performance experiments are independent

No attention is paid to the order of execution of experiments

Incorrect or unreliable conclusions

The ordering trap

“Root causes”: performance-affecting system states that carry over or change between performance tests



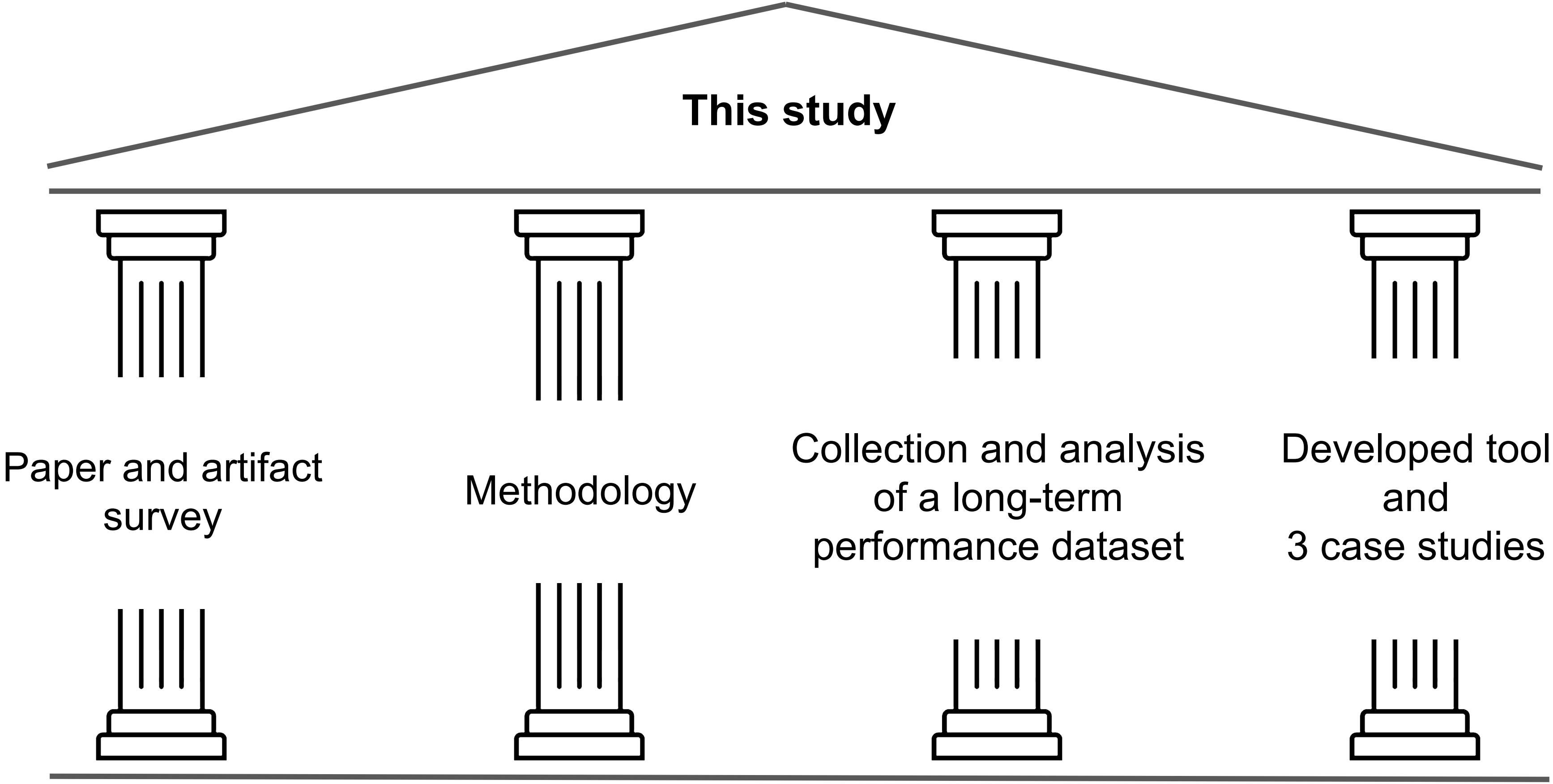
- caches
- data layout in RAM
- data layout on disk
- application and operating system dynamic parameters
- CPU temperatures and thermal throttling
- environment variables
- …
- many more complex “**behind the scenes” factors**

Avoiding the ordering trap

Define relevant “Reset to Clean State” procedure

Run experiments in both **baseline** and **multiple random orders,** with repetition of individual tests and with calls to the reset procedure

Compare results using appropriate statistical tests



This study

Paper and artifact survey

Methodology

Collection and analysis of a long-term performance dataset

Developed tool and 3 case studies

This study

Paper and artifact survey

Methodology

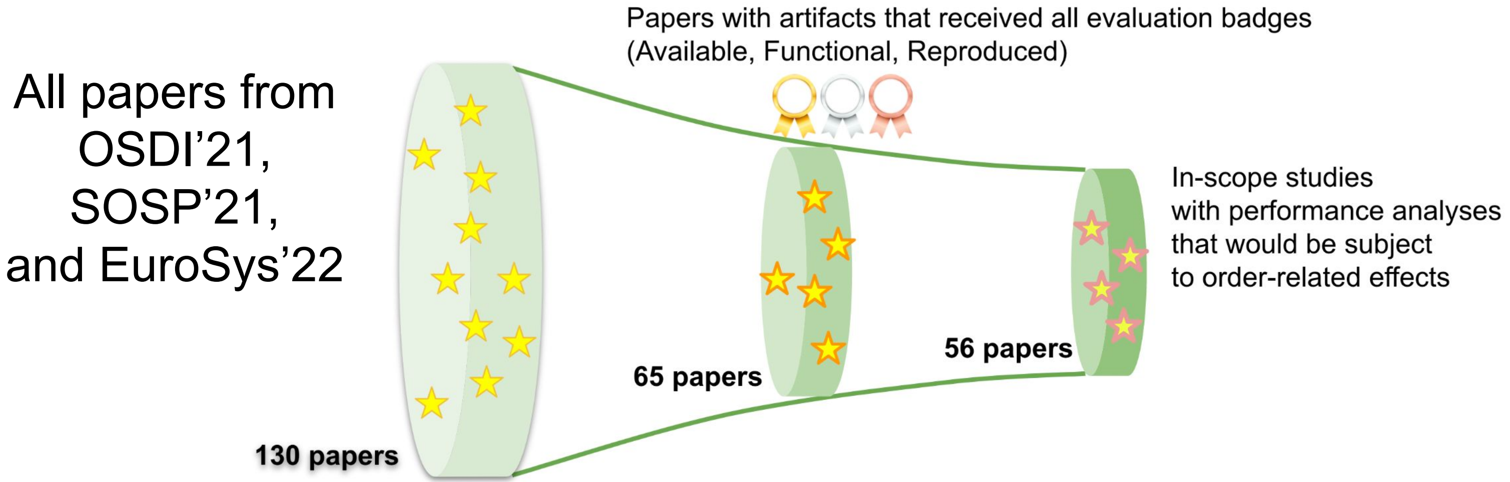
Collection and analysis of a long-term performance dataset

Developed tool and 3 case studies

Paper and artifact survey

Our Impression: ordering effects are **rarely** considered in computer systems research

Proving / disproving it:



All papers from OSDI'21, SOSP'21, and EuroSys'22

130 papers

Papers with artifacts that received all evaluation badges (Available, Functional, Reproduced)

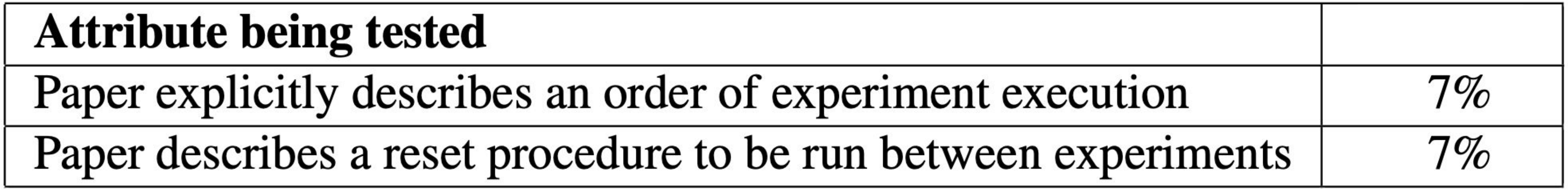
65 papers

56 papers

In-scope studies with performance analyses that would be subject to order-related effects

Summary of 56 studied papers

*

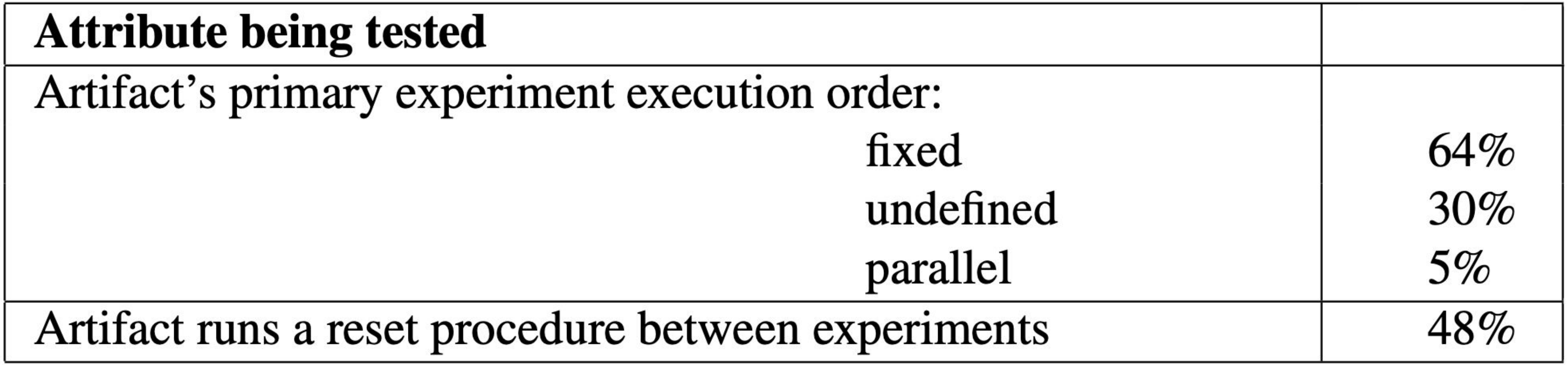


Attribute being tested	
Paper explicitly describes an order of experiment execution	7%
Paper describes a reset procedure to be run between experiments	7%

Very few research papers describe
order of execution and inter-experiment reset procedures.

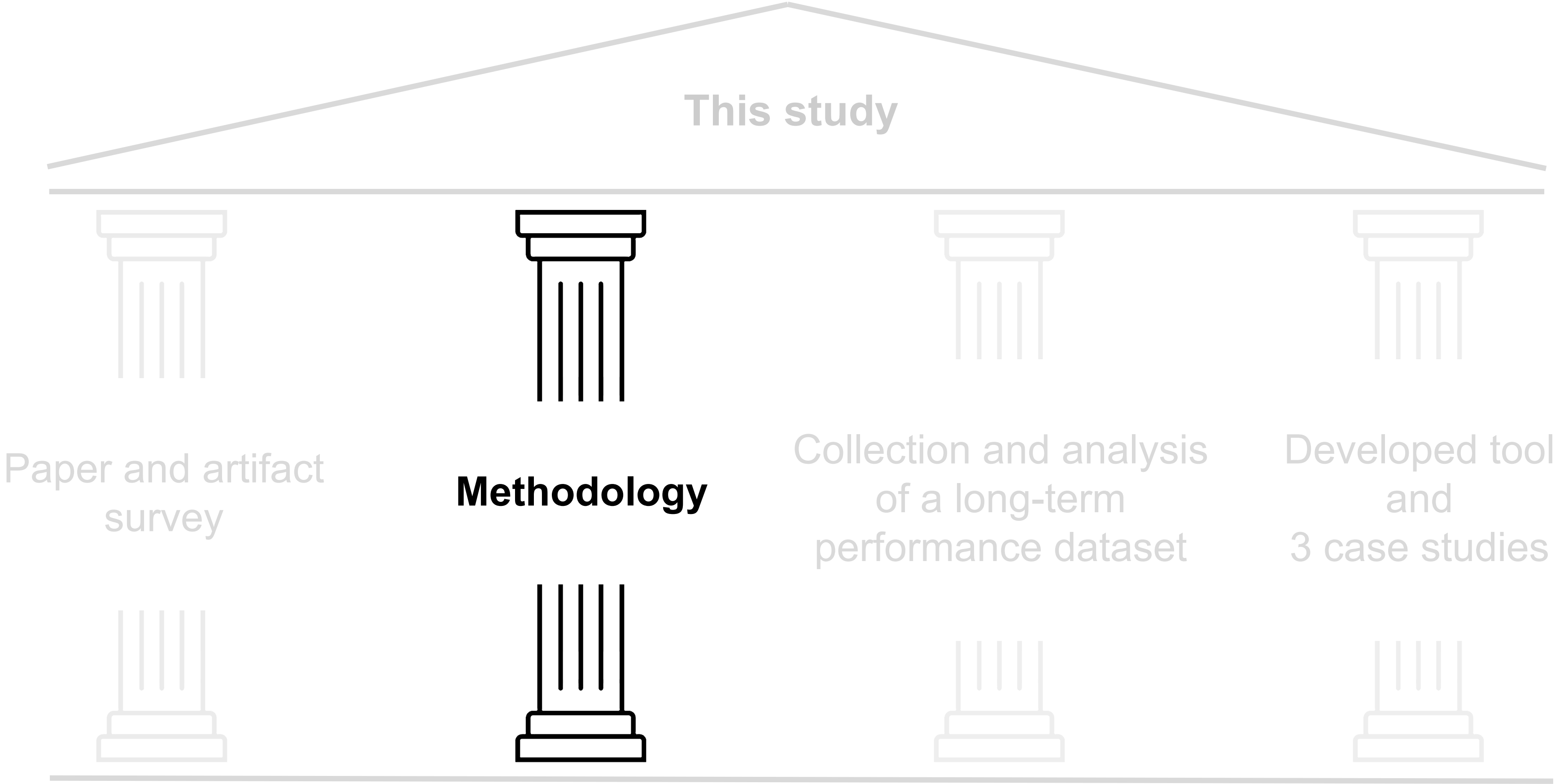
Summary of 56 studied artifacts

*



Attribute being tested	
Artifact’s primary experiment execution order:	
fixed	64%
undefined	30%
parallel	5%
Artifact runs a reset procedure between experiments	48%

A **randomized experiment design was not found** in the studied artifacts.



This study

Paper and artifact survey

Methodology

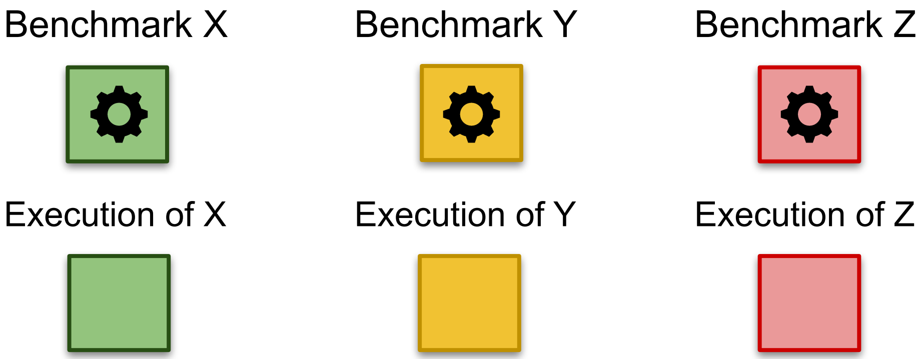
Collection and analysis of a long-term performance dataset

Developed tool and 3 case studies

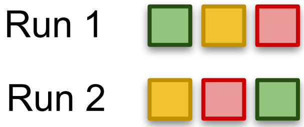
Terminology

Test: individual benchmark

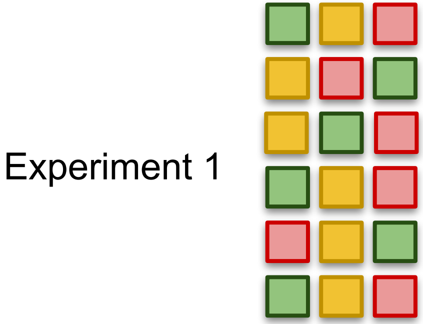
Trial: individual execution of a test



Run: set of trials, executed in a particular order, e.g., fixed-order runs, random-order runs



Experiment: collection of one or more runs executed for the purpose of reaching a conclusion



Methodology

❶ Select a “Baseline” Order

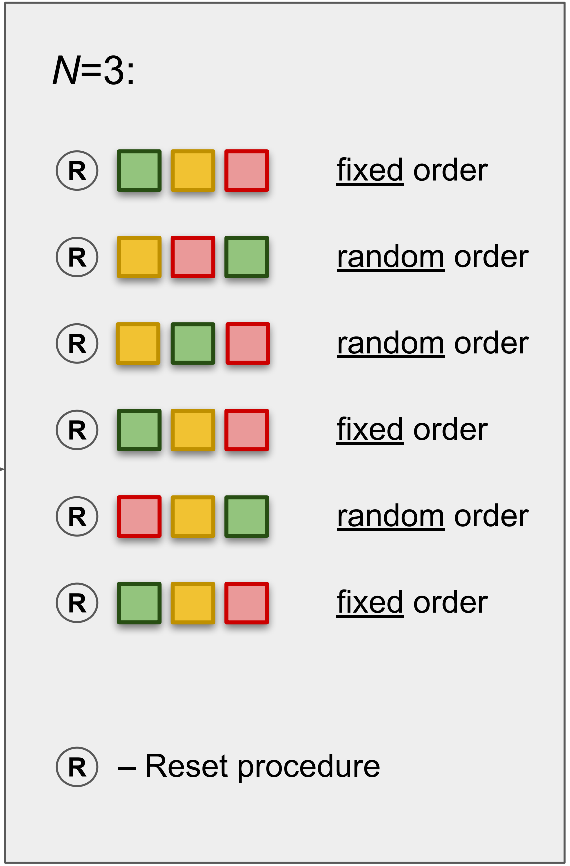
❷ Define a “Reset to Clean State” Procedure

❸ Run in Both Fixed and Random Orders

- *N* repetitions for each

❹ Compare Distributions

- Kruskal-Wallis test (instead of parametric tests: one-way ANOVA or *t*-test)
  - Hypothesis: samples come from the same distribution
  - Mann–Whitney *U* test is alternative
- Bonferroni correction for experiment-wide conclusion



N=3:

Ⓡ 🟩🟨🟥 fixed order

Ⓡ 🟨🟥🟩 random order

Ⓡ 🟨🟩🟥 random order

Ⓡ 🟩🟨🟥 fixed order

Ⓡ 🟥🟨🟩 random order

Ⓡ 🟩🟨🟥 fixed order

Ⓡ – Reset procedure

Methodology

Analysis outcomes:

- If any test's *p*-value is below the Bonferroni-corrected threshold, the **order of the tests matters**
- If all tests' *p*-values are above the Bonferroni-corrected threshold, the **order likely does not matter**

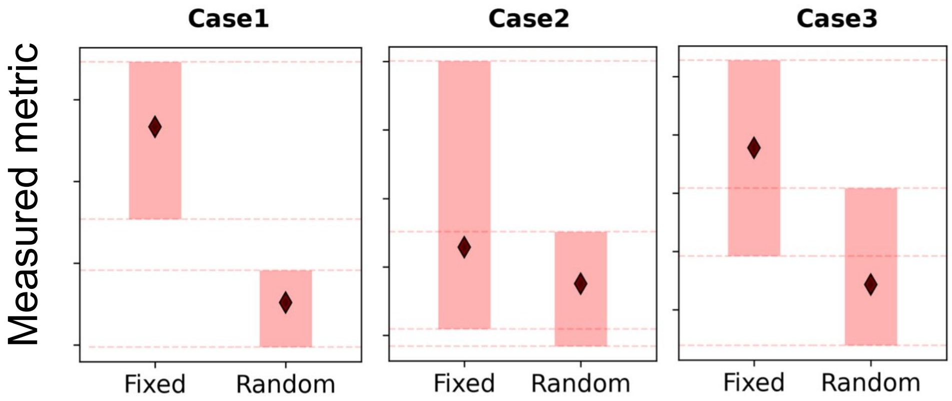
How different are the results from fixed-order and random-order runs?

Two options for answering this question:

1) Measure **relative difference** between means:

$$\Delta_{\%} = \frac{\mu_{fixed} - \mu_{random}}{\mu_{fixed}} \times 100\%$$

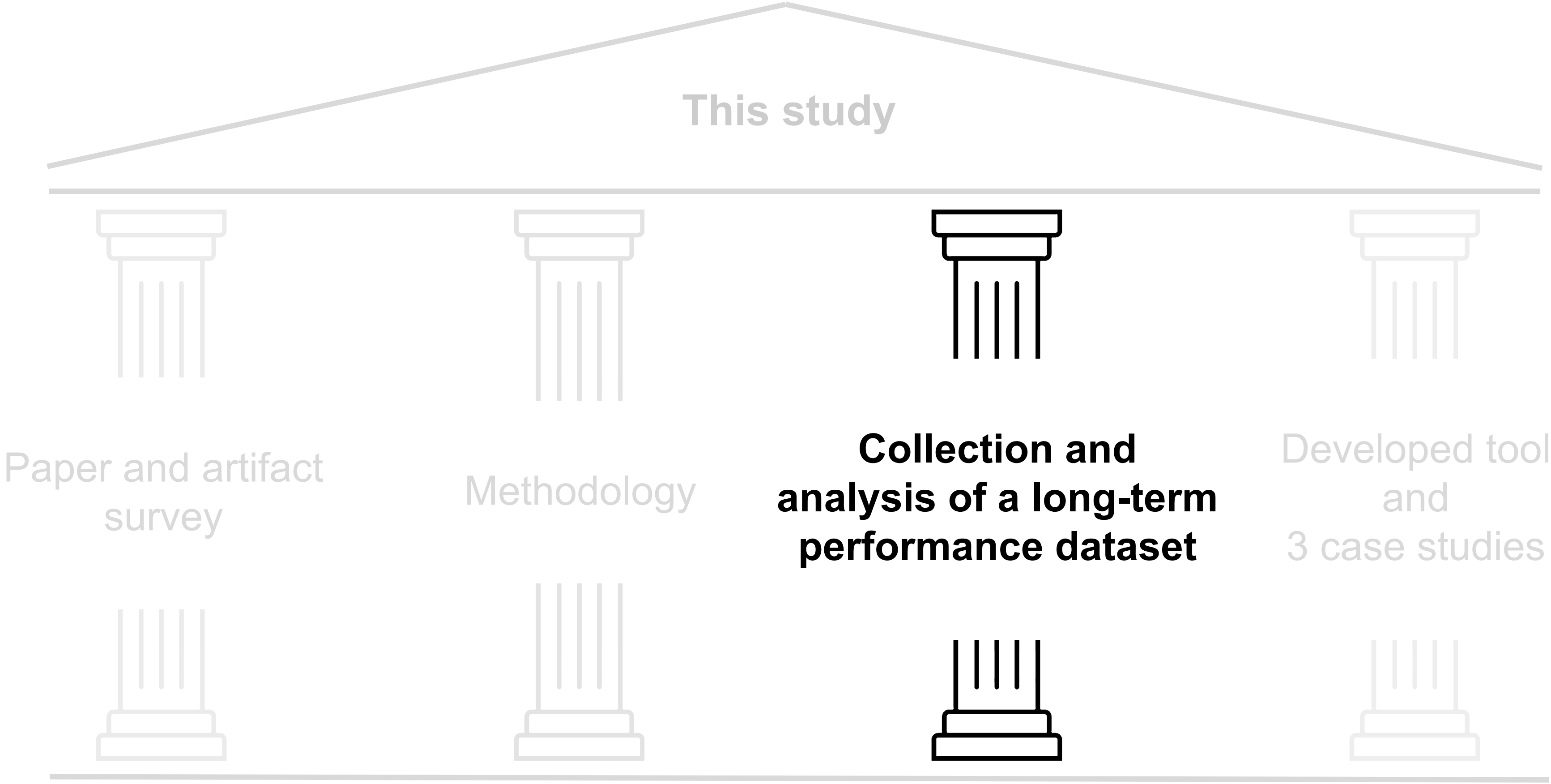
2) Visualize **medians** and **non-parametric confidence intervals** for medians:



"Red flag": different conclusions based on the order

Order is unlikely to change conclusions

More examination needed



This study

Paper and artifact survey

Methodology

Collection and analysis of a long-term performance dataset

Developed tool and 3 case studies

Long-term performance dataset

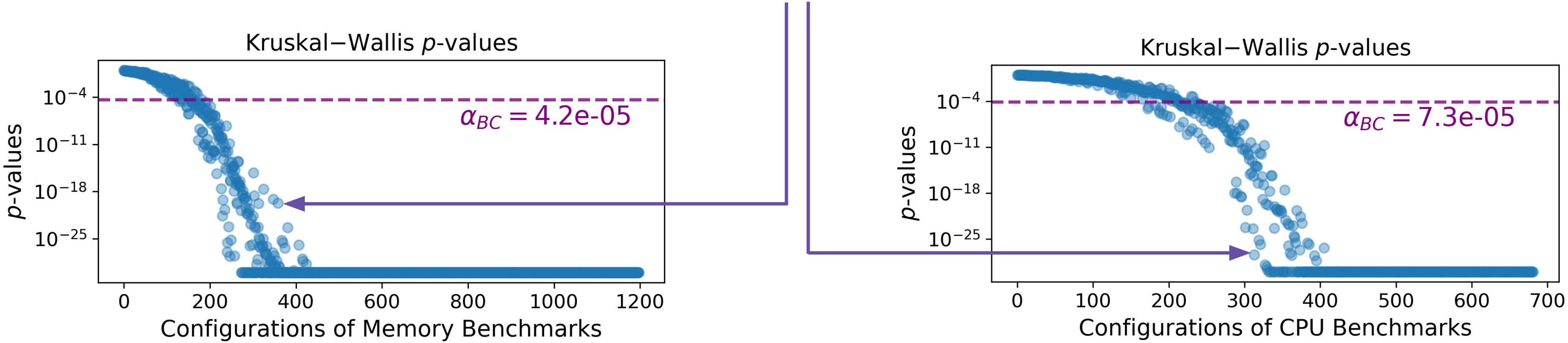
- Collected using the CloudLab testbed (www.cloudlab.us)
- CPU and memory performance evaluated using microbenchmarks
- 2.3M trials from over 9,000 runs executed on 1,700 bare-metal servers
- Truly independent runs
- Entire dataset: https://github.com/ordersage/paper-artifact

Different or not?

Each point represents a comparison of **fixed-order** results and **random-order** results for the same test

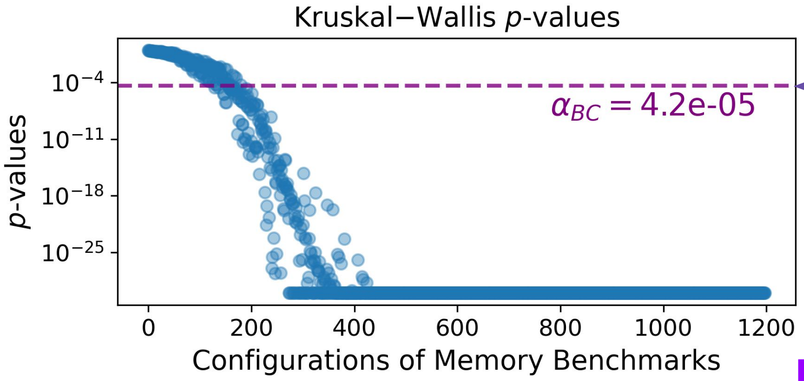
Memory benchmarks (STREAM)

CPU benchmarks (NPB)

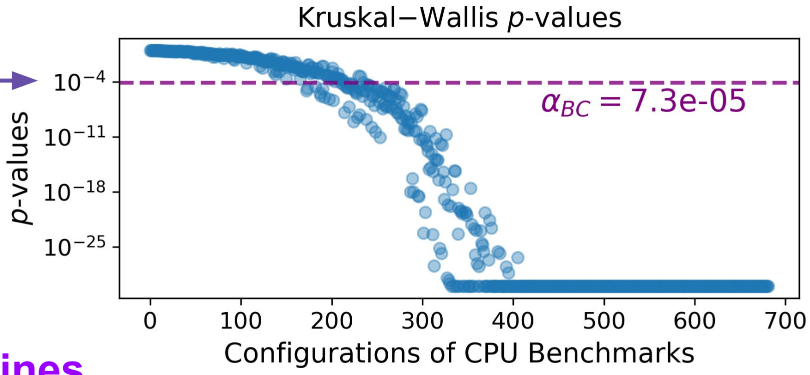


Different or not?

Memory benchmarks (STREAM)



CPU benchmarks (NPB)

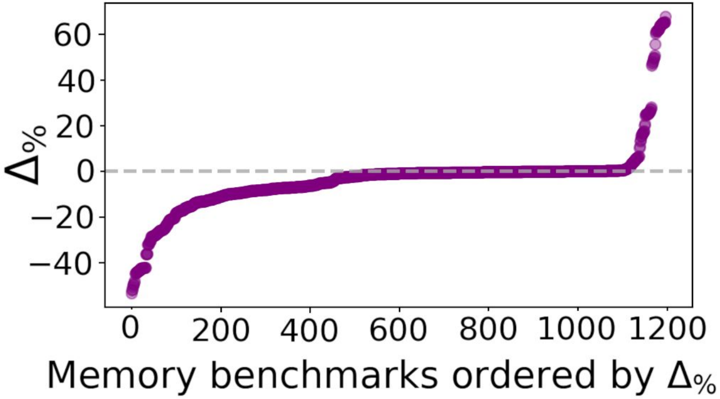


**Below these lines,
the order matters**

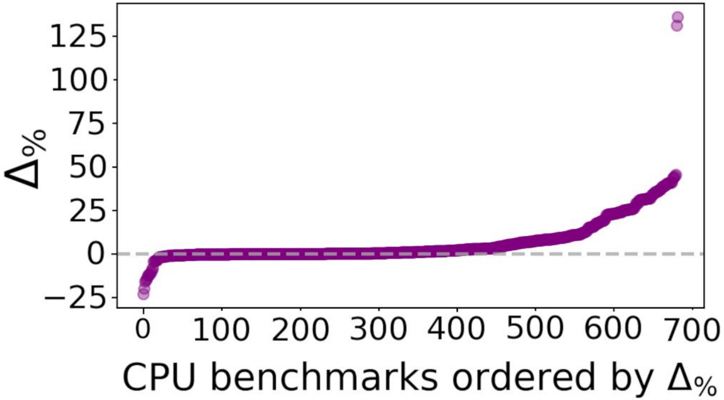
Statistically significant effects due to ordering are found
for the **majority** (over 70%) of the studied cases

How different?

Memory benchmarks (STREAM)



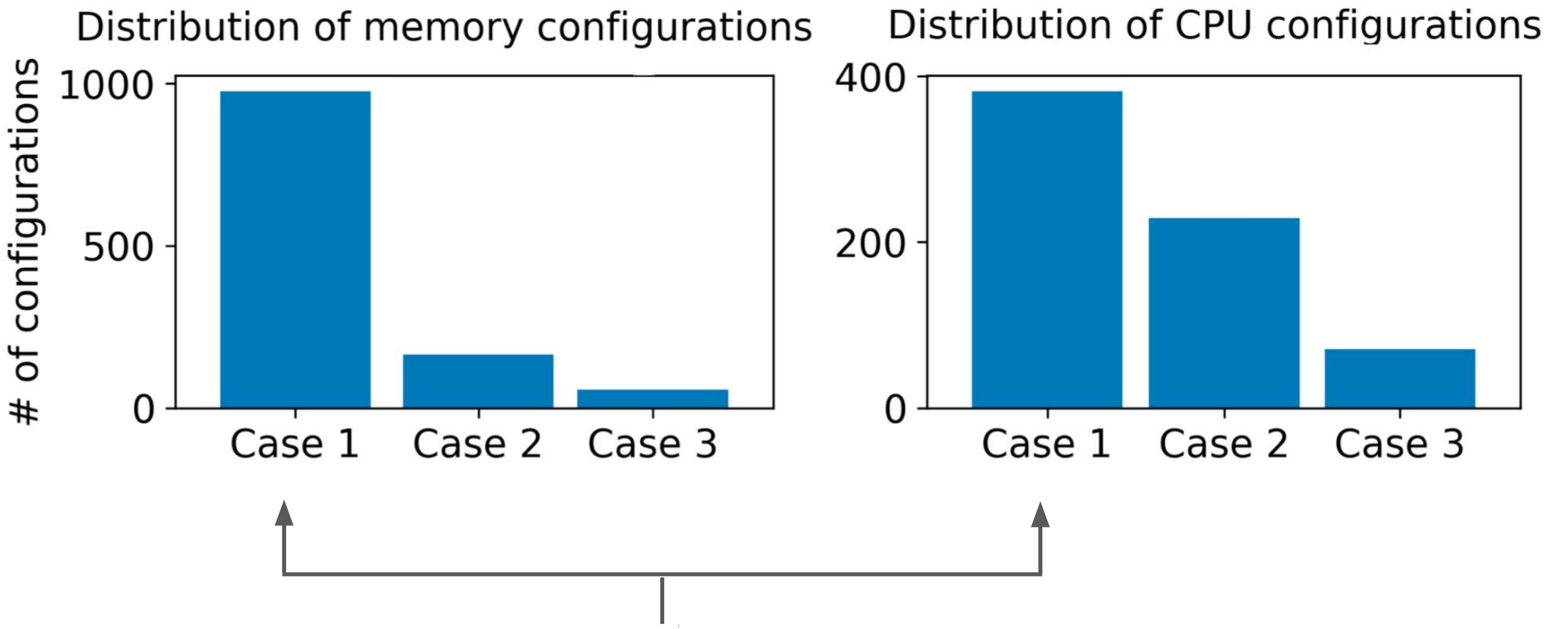
CPU benchmarks (NPB)



Mean absolute percentage differences: **8%** for memory and **7.3%** for CPU order effects.

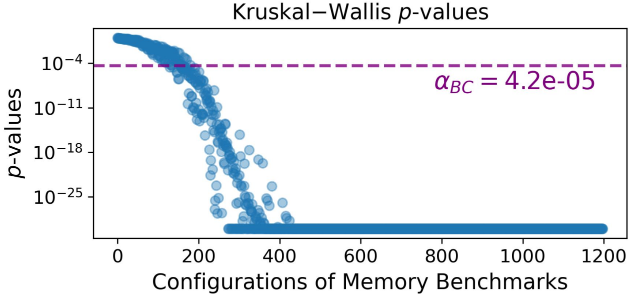
Ordering effects can be quite large, up to **tens of percents.**

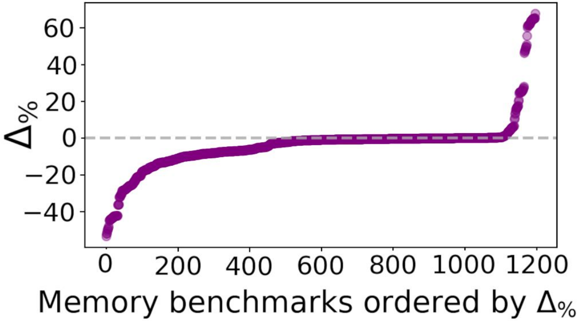
Confidence intervals

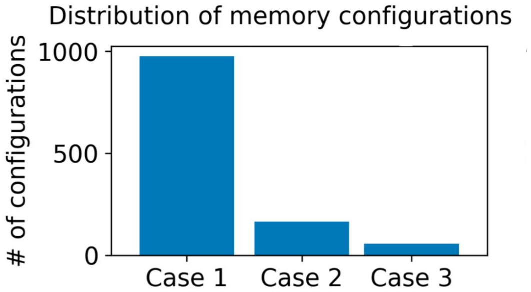


“Red flag”:
different conclusions
based on the order

Analysis summary







Rigorous performance analysis **must consider order of test execution** to ensure accurate conclusions.

This study

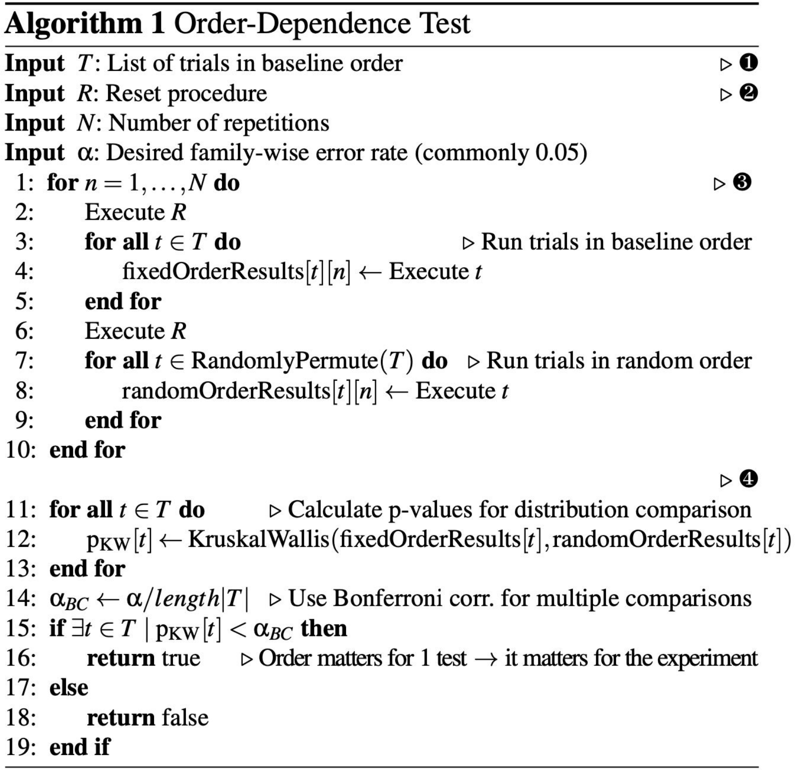
Paper and artifact survey

Methodology

Collection and analysis of a long-term performance dataset

Developed tool and 3 case studies

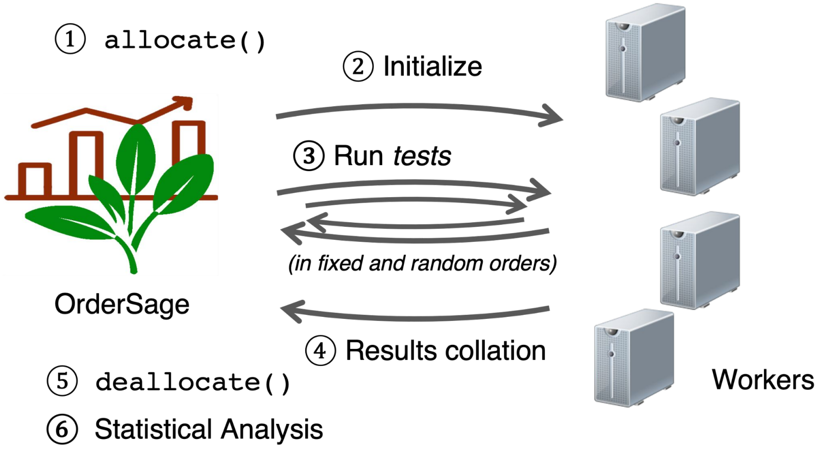
From methodology to a usable tool



Algorithm 1 Order-Dependence Test

```
Input $T$: List of trials in baseline order                                   ▷ ❶
Input $R$: Reset procedure                                                    ▷ ❷
Input $N$: Number of repetitions
Input $\alpha$: Desired family-wise error rate (commonly 0.05)
for $n = 1, \dots, N$ do                                                      ▷ ❸
    Execute $R$
    for all $t \in T$ do                                  ▷ Run trials in baseline order
        fixedOrderResults$[t][n] \leftarrow$ Execute $t$
    end for
    Execute $R$
    for all $t \in$ RandomlyPermute$(T)$ do                 ▷ Run trials in random order
        randomOrderResults$[t][n] \leftarrow$ Execute $t$
    end for
end for
                                                                              ▷ ❹
for all $t \in T$ do                    ▷ Calculate p-values for distribution comparison
    $p_{KW}[t] \leftarrow$ KruskalWallis(fixedOrderResults$[t]$, randomOrderResults$[t]$)
end for
$\alpha_{BC} \leftarrow \alpha / length|T|$          ▷ Use Bonferroni corr. for multiple comparisons
if $\exists t \in T \mid p_{KW}[t] < \alpha_{BC}$ then
    return true          ▷ Order matters for 1 test → it matters for the experiment
else
    return false
end if
```

OrderSage



Available at:

https://github.com/ordersage/ordersage

Three case studies (conducted using OrderSage)

mc-crusher benchmark suite for **memcached** key-value store

Order of the tests matters.

Largest $\Delta_{\%}$: 5.3%

NPBench (Python & NumPy) and **NPB** (NAS Parallel Benchmarks)

Order of the tests matters.

Largest $\Delta_{\%}$: -0.6%

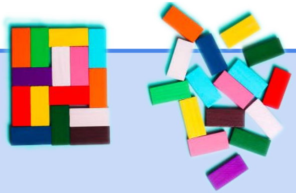
uFS Paper
artifact
(Paper: Scale and Performance in a Filesystem Semi-Microkernel)

Order of the tests matters.

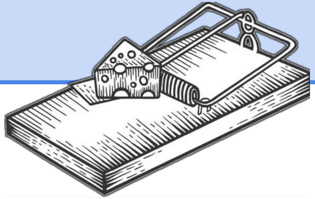
The conclusion from the uFS paper still holds.

Largest $\Delta_{\%}$: 16.8%

Takeaways



Avoid the ordering trap!



Run experiments in both **baseline** and **multiple random orders;** compare results**.**

Follow the **methodology** from our paper and use **OrderSage**.

Released artifact

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

https://github.com/ordersage/paper-artifact

Thank you!