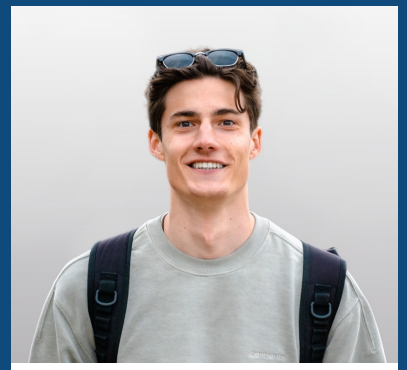
@Large Research
Massivizing Computer Systems

Design and Implementation of a Framework for Performance Characterization of Resource Sharing Mechanisms in Distributed Computing



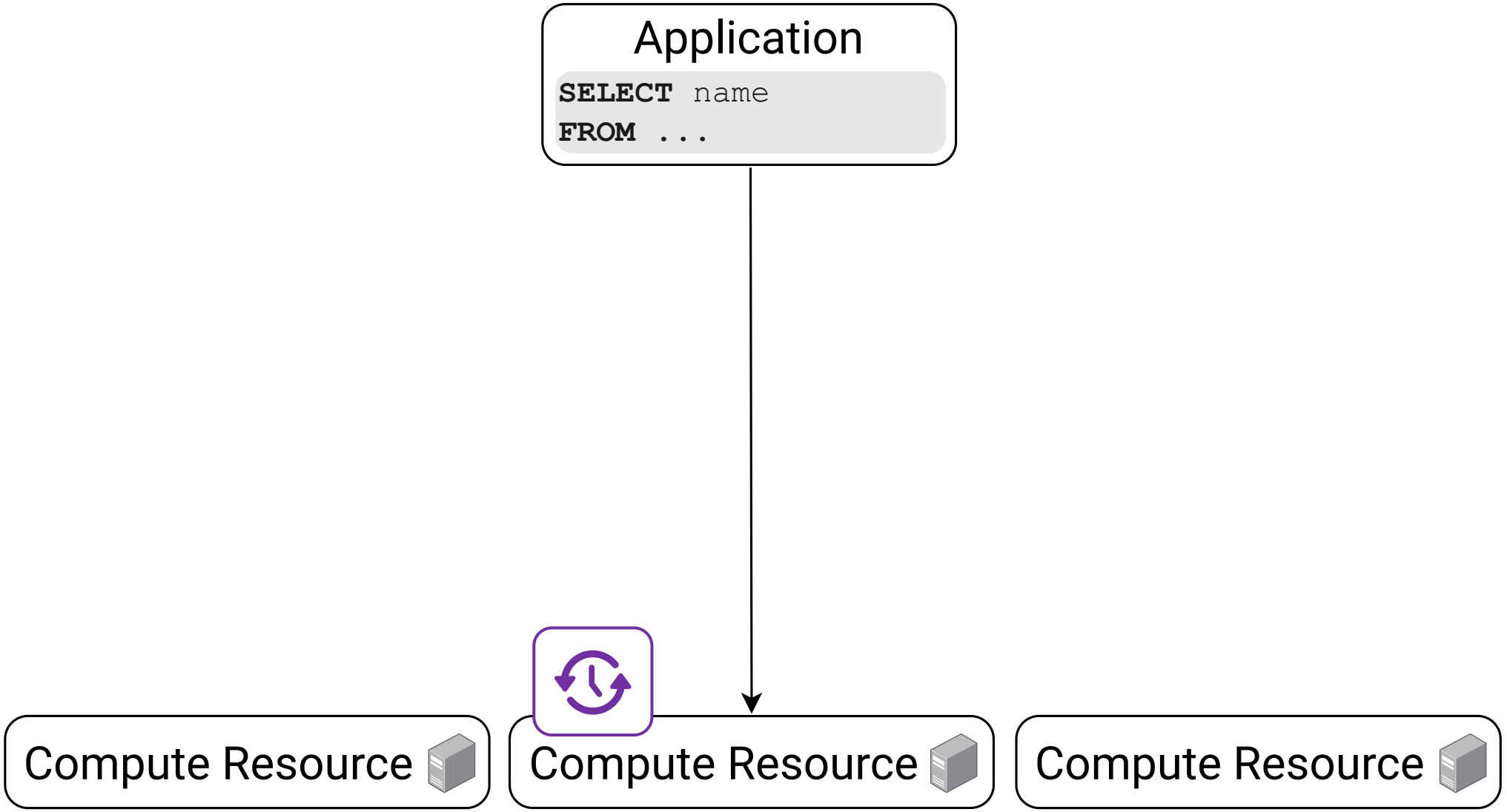
VU VRIJE UNIVERSITEIT AMSTERDAM

BSc Thesis by
Lennart K. M. Schulz

under supervision from
Prof. Dr. Ir. Alexandru Iosup and **Sacheendra Talluri**

at
Vrije Universiteit Amsterdam, NL

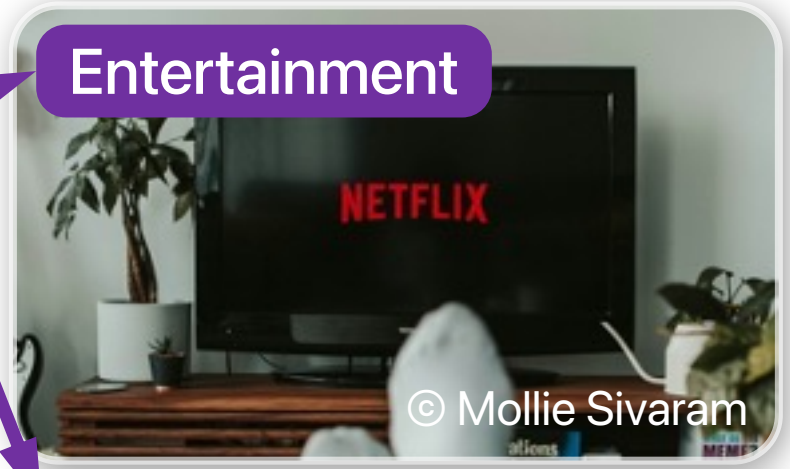
Context – *Where Are We?*

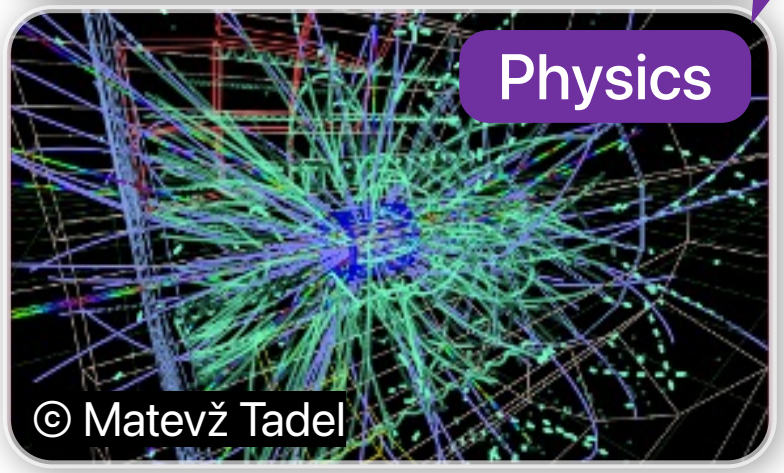


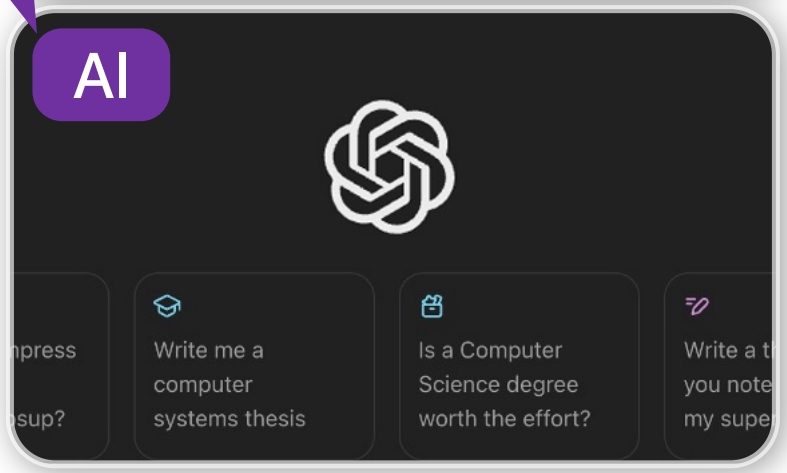
Context – *Where Are We?*

Application

```
SELECT name
FROM ...
```







Compute Resource

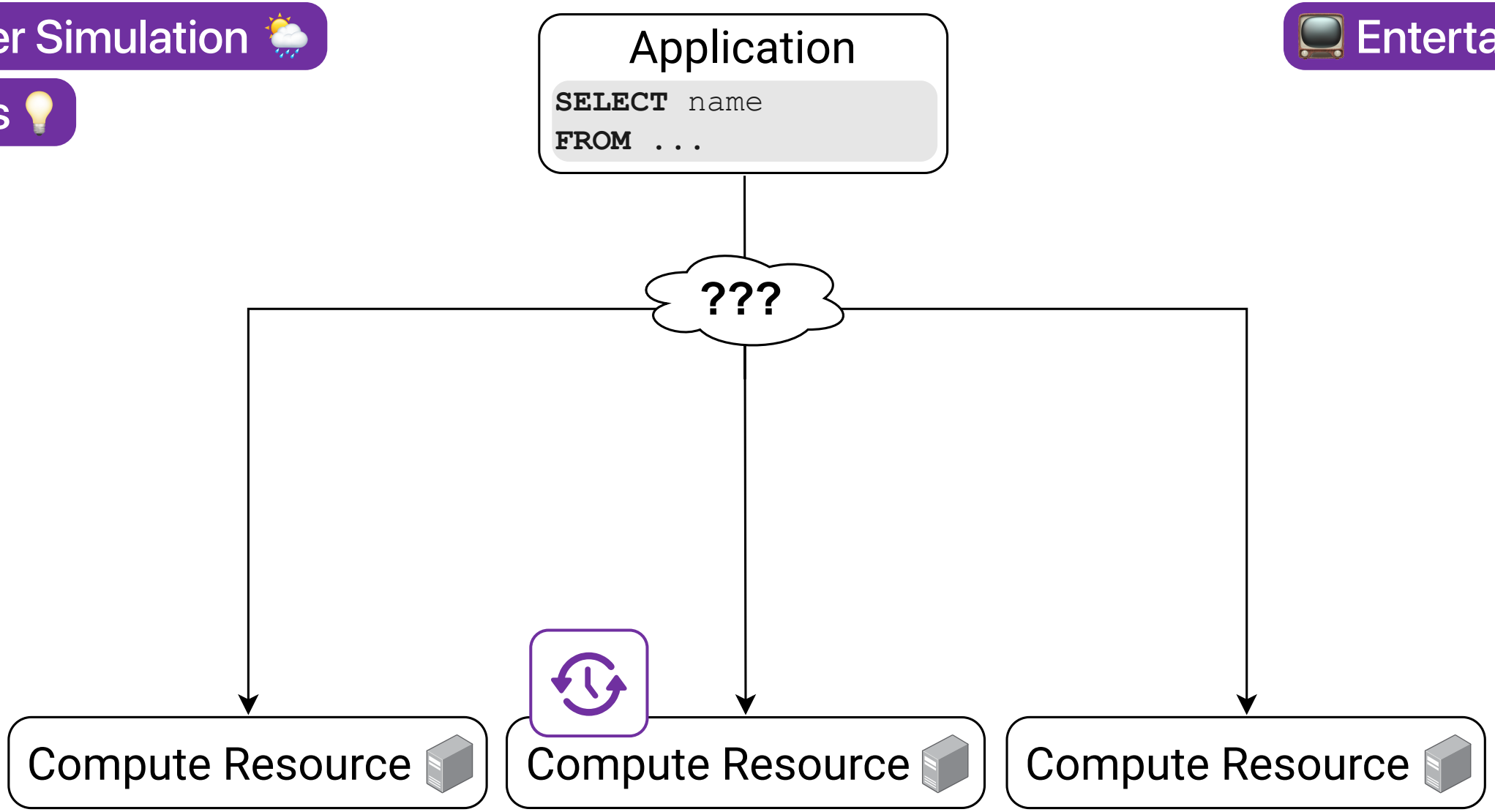
Compute Resource

Compute Resource

Context – *Where Are We?*

Weather Simulation 🌦️

Physics 💡



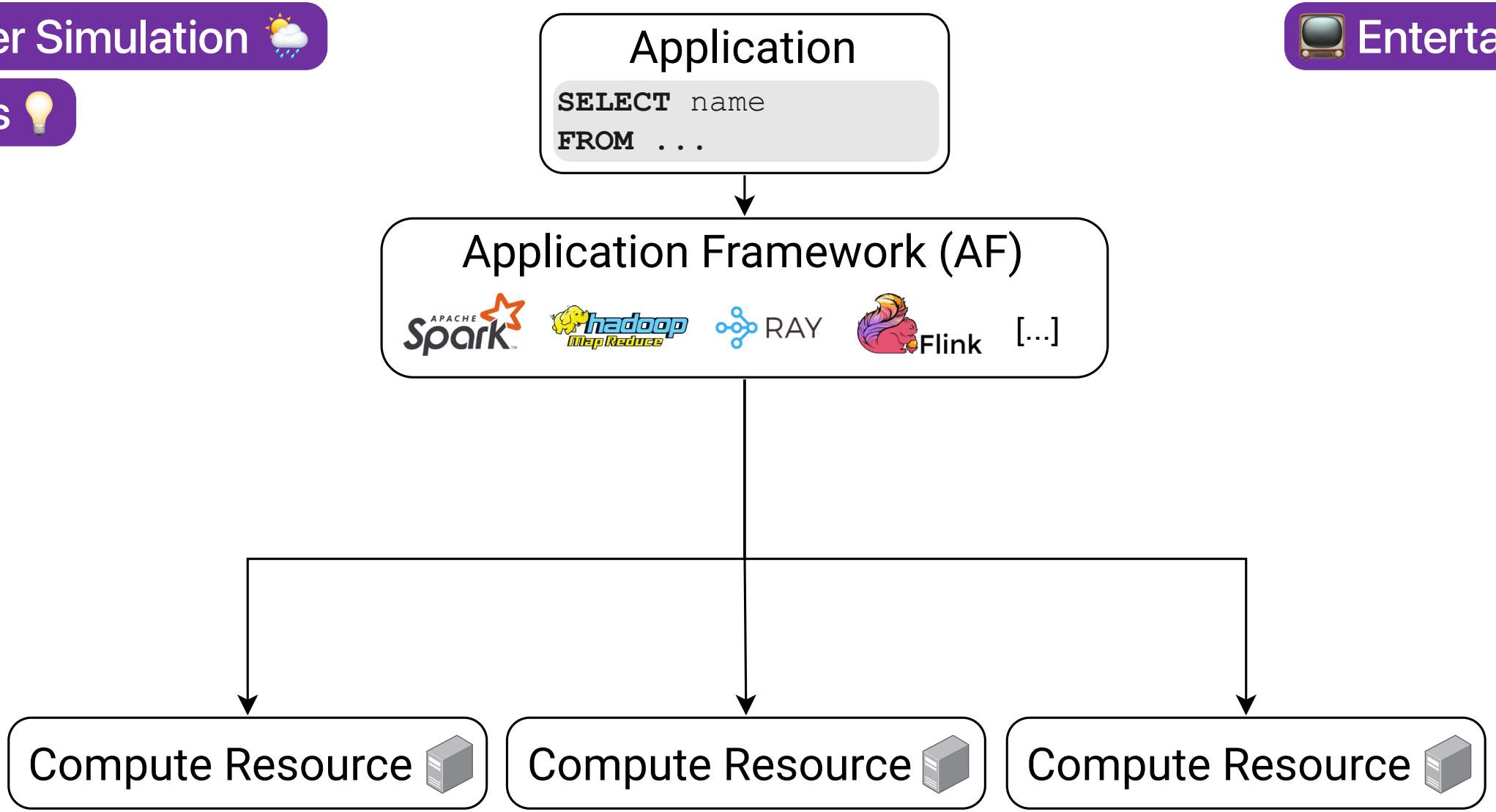
🖥️ Entertainment

🤖 AI

Context – *Where Are We?*

Weather Simulation 🌦️

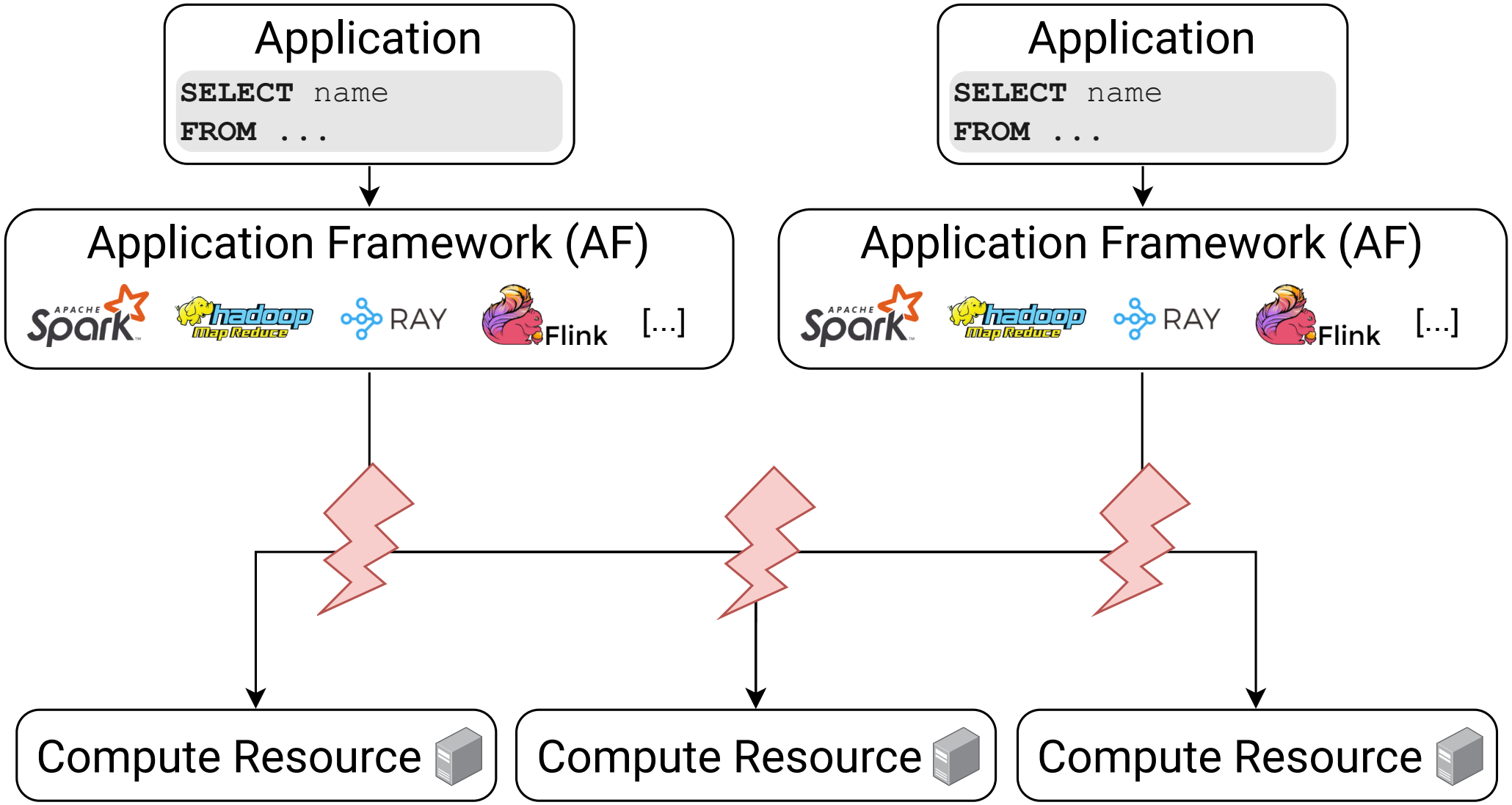
Physics 💡



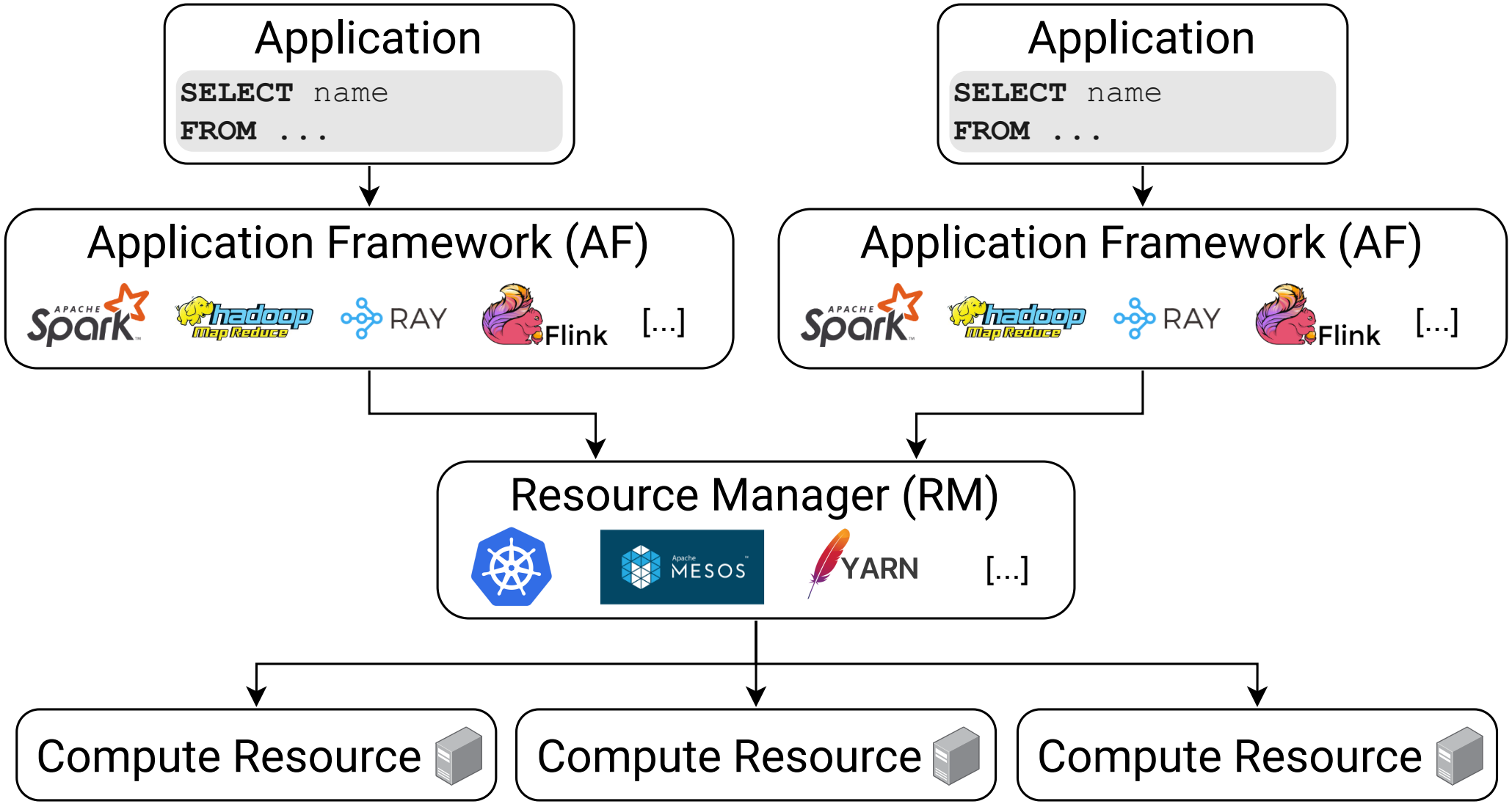
📺 Entertainment

🤖 AI

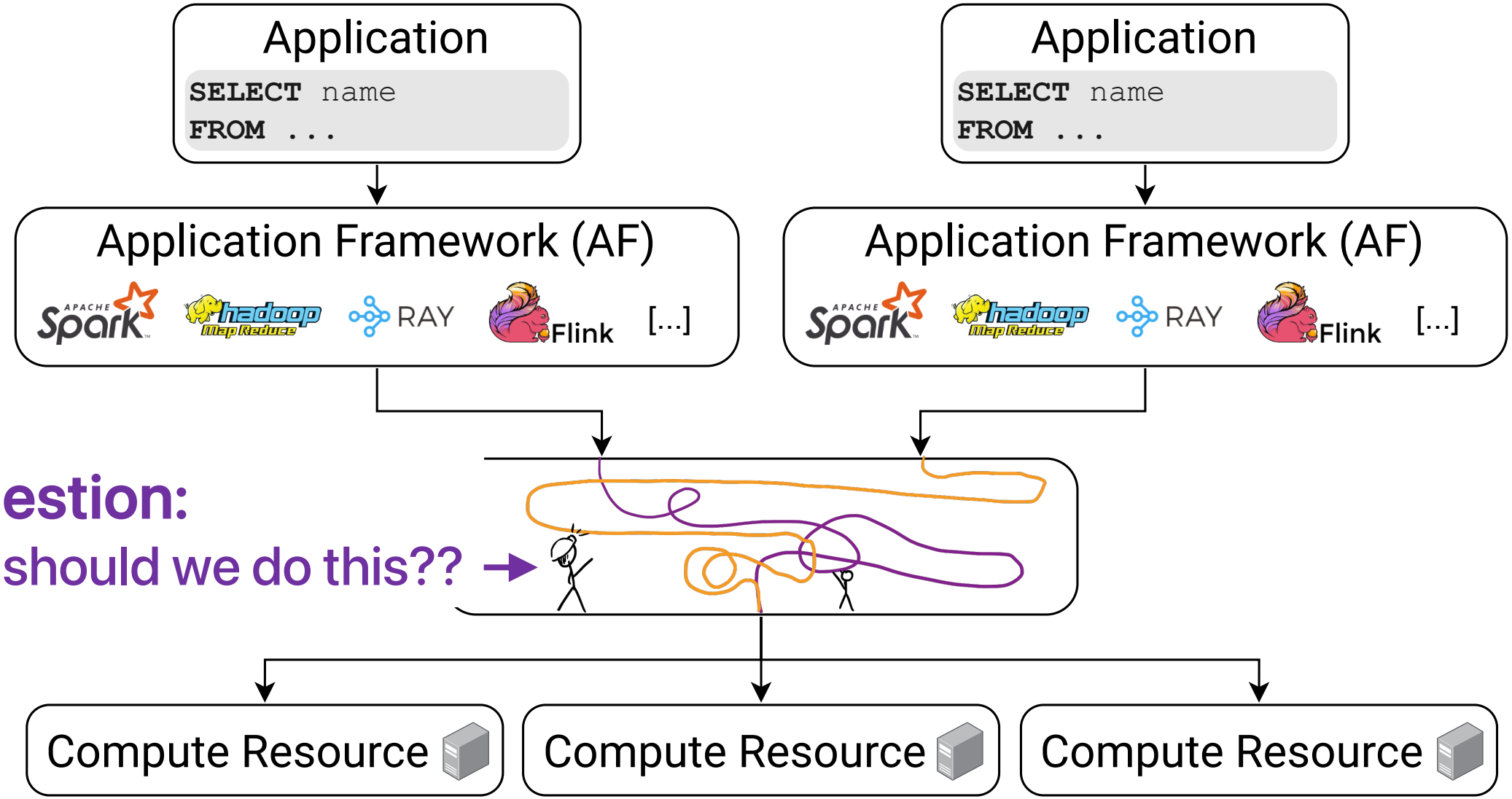
Context – *Where Are We?*



Context – *Where Are We?*



Context – *Where Are We?*



Question:

How should we do this??

Research Questions

Main RQ: How to evaluate the performance characteristics of resource sharing mechanisms in distributed computing?

using SparkSQL on Kubernetes

RQ1: How to generate workloads to evaluate the performance characteristics of resource sharing mechanisms?

RQ2: How to implement an infrastructure framework for automated comparison of resource sharing mechanisms?

RQ3: What are the performance characteristics of three selected resource sharing mechanisms?

Research Questions

Main RQ: How to evaluate the performance characteristics of resource sharing mechanisms in distributed computing?

using SparkSQL on Kubernetes

⚠ No system exists!

Answering these research questions helps to address:

- Responsibility,
- Sustainability, and
- Usability

of computer systems, as described in the CompSys NL Manifesto.

Workload Generator Design

RQ1: How to generate workloads to evaluate the performance of resource sharing mechanisms?

Workload:

A (structured) **Collection** of **Queries** on **Data.**

Workload Generator Design

Workload: A (structured) **Collection** of **Queries** on **Data.** ✓

1. The TPC-DS Dataset

Synthetic dataset, modelling the sales and returns process for a multichannel (stores, catalogs, internet) commercial organization

2. Adapting the TPC-DS Queries

TPC-DS queries are good - but limited in number

Solution:
Modify queries into numerous new ones of different scales

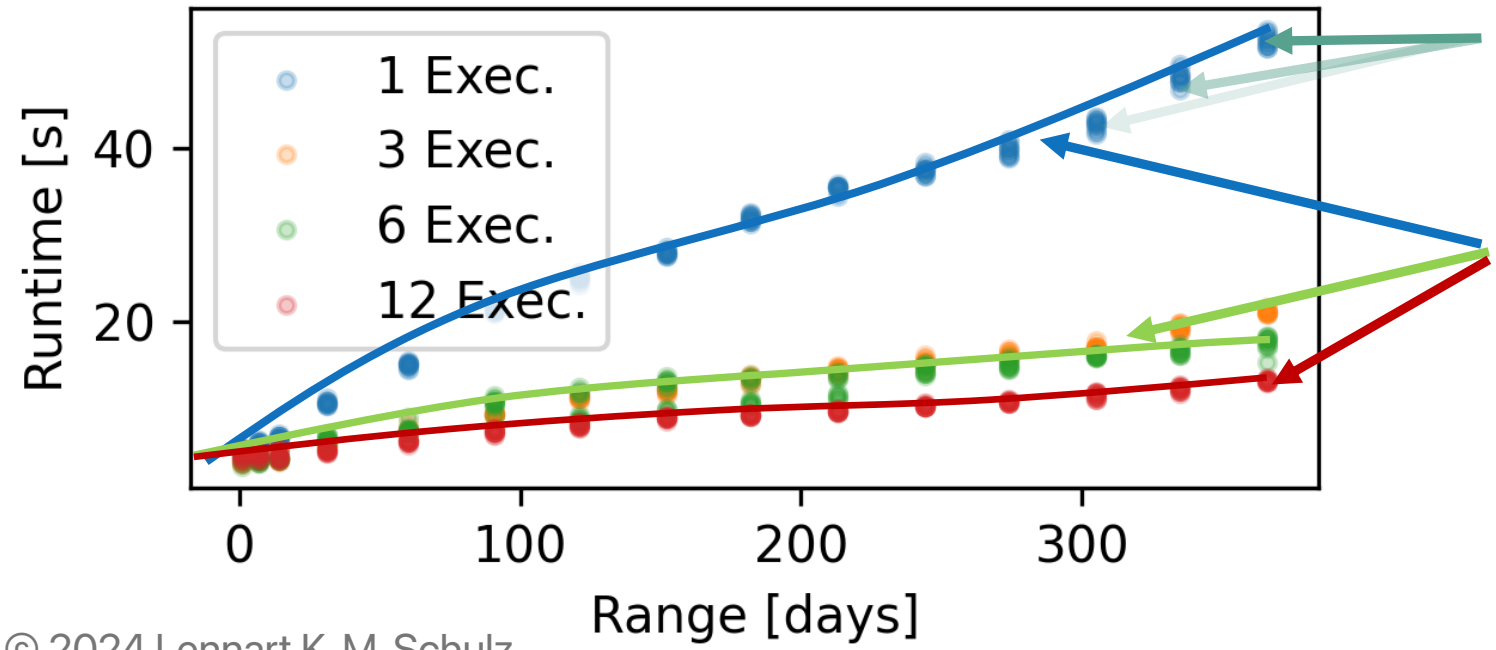
Workload Generator Design

Workload: A (structured) **Collection** of **Queries** on **Data**.✓

1. The TPC-DS Dataset 2. Adapting the TPC-DS Queries

3. Getting Query Times

Run each query with various scales to get insights into its runtimes



Currently:
Only using measured queries.

Future Work:
Can these relations be modelled?
(→ interpolation & better scaling)

Workload Generator Design

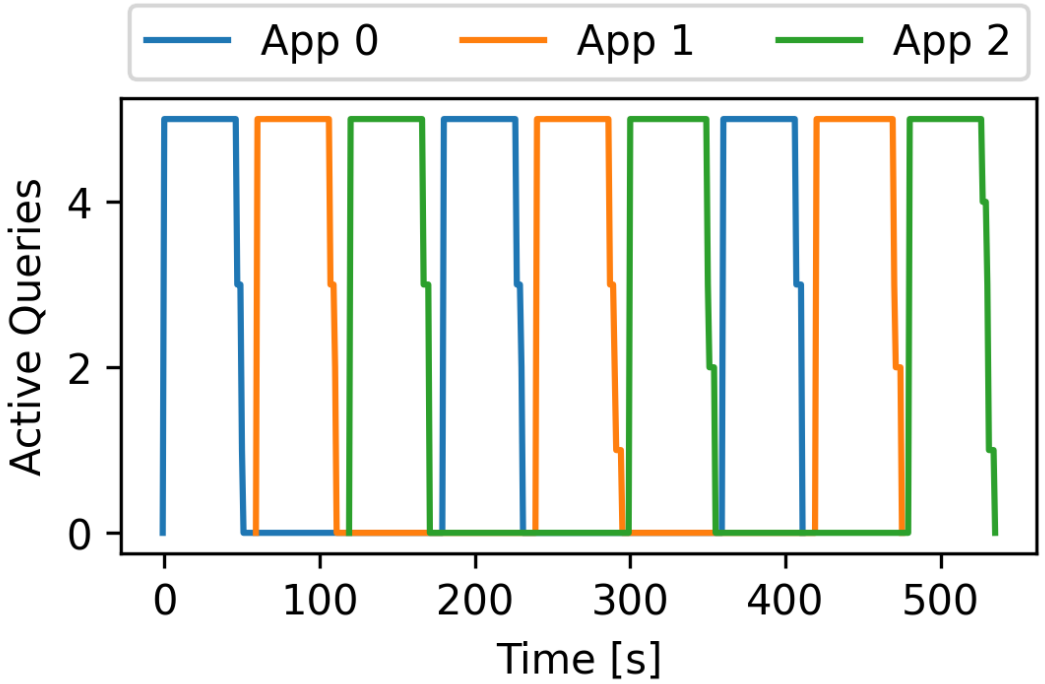
Workload: A (structured) **Collection** of **Queries** on **Data**.

1. The TPC-DS Dataset
2. Adapting the TPC-DS Queries
3. Getting Query Times

4. Building Workloads

Combine the queries into workloads, using the recorded time information:

```
1  workload = create_bursty(
2      queries,
3      num_apps=3,
4      app_offset=60,
5      burst_count=3,
6      burst_interval=60 * 3,
7      ...
8  )
```



Workload Generator Design

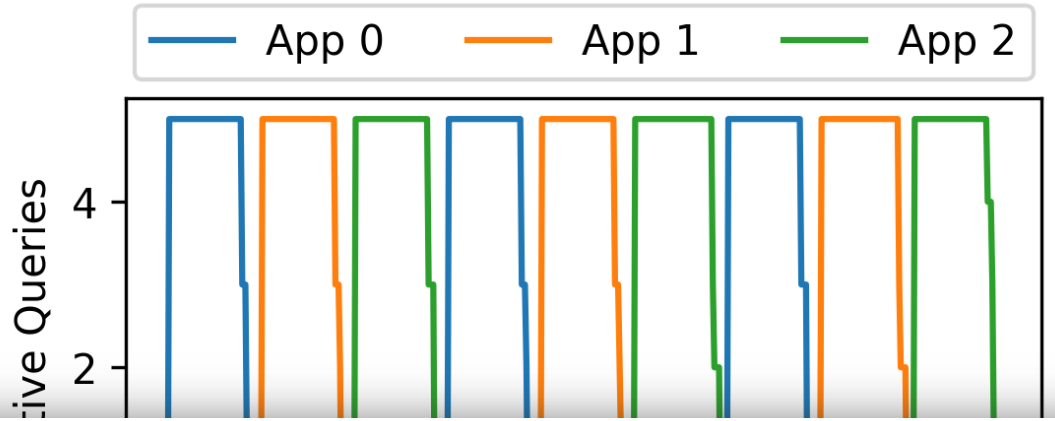
Workload: A (structured) **Collection** ✓ of **Queries** ✓ on **Data.** ✓

1. The TPC-DS Dataset 2. Adapting the TPC-DS Queries 3. Getting Query Times

4. Building Workloads

Combine the queries into workloads, using the recorded time information:

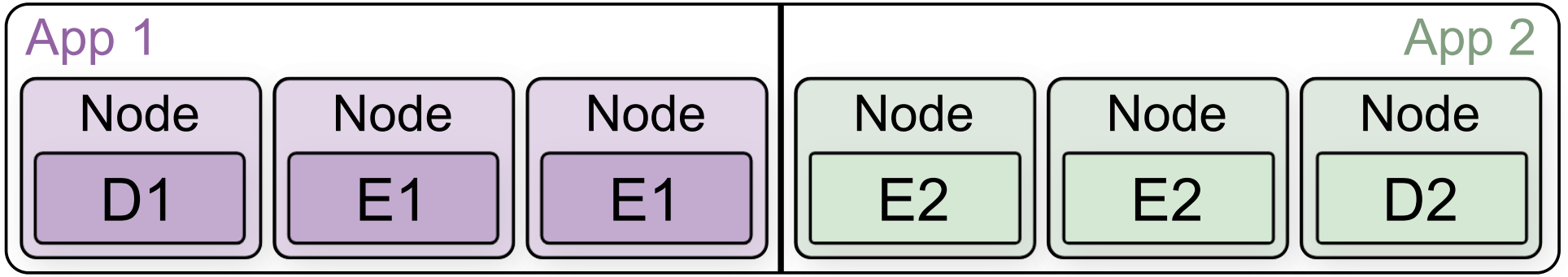
```
1  workload = create_bursty(
2      queries,
3      num_apps=3,
4      app_offset=60,
5      burst_count=3,
6      burst_interval=60 * 3
```



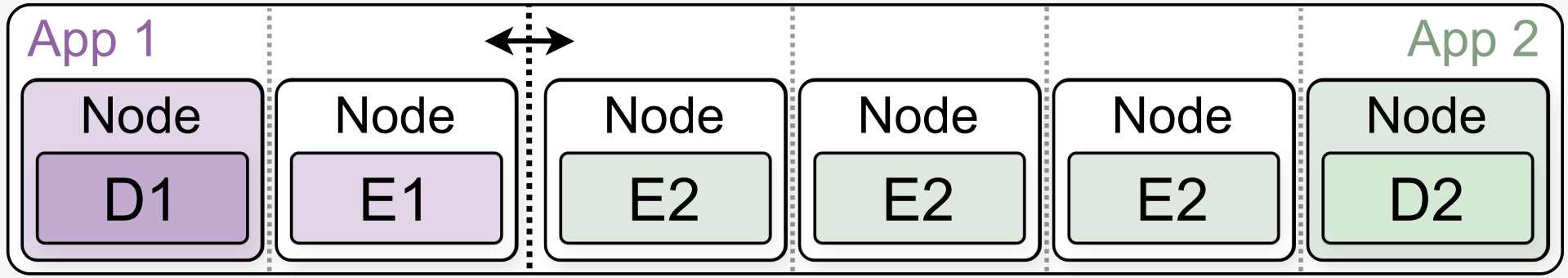
✓ **RQ1:** How to generate workloads to evaluate the performance characteristics of resource sharing mechanisms?

Context – *Resource Sharing Mechanisms*

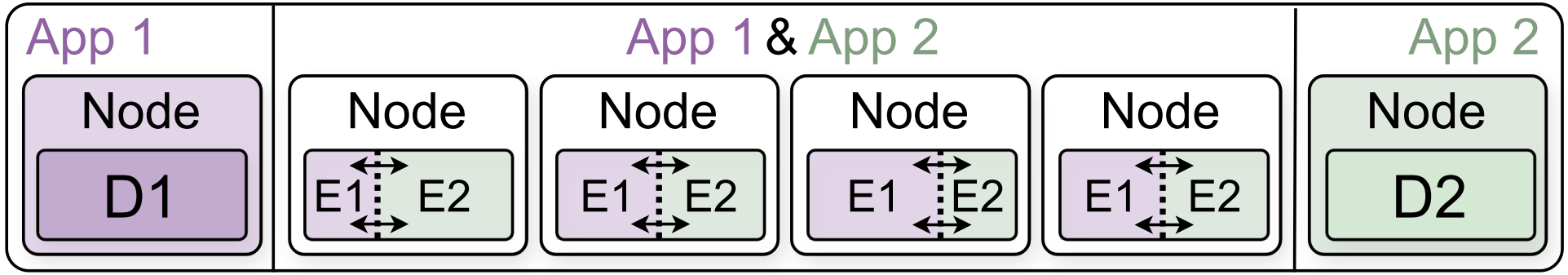
Static Partitioning



Dynamic Partitioning



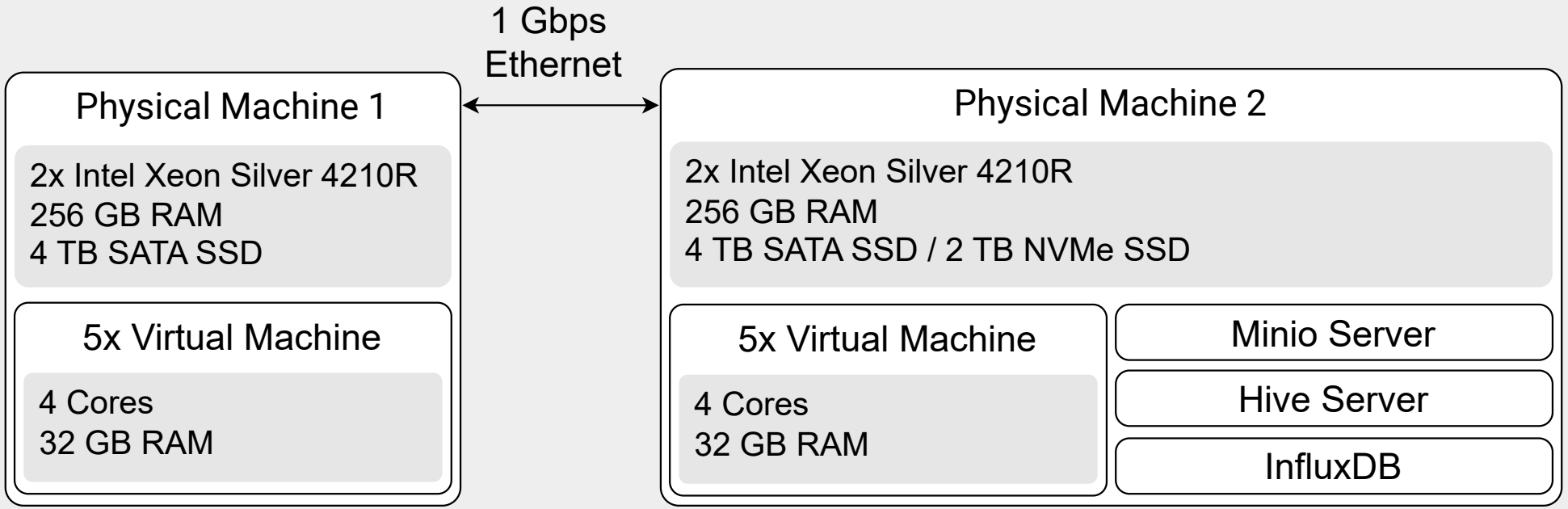
Node-Level Sharing



Kubernetes Node | D Spark Driver | E Spark Executor

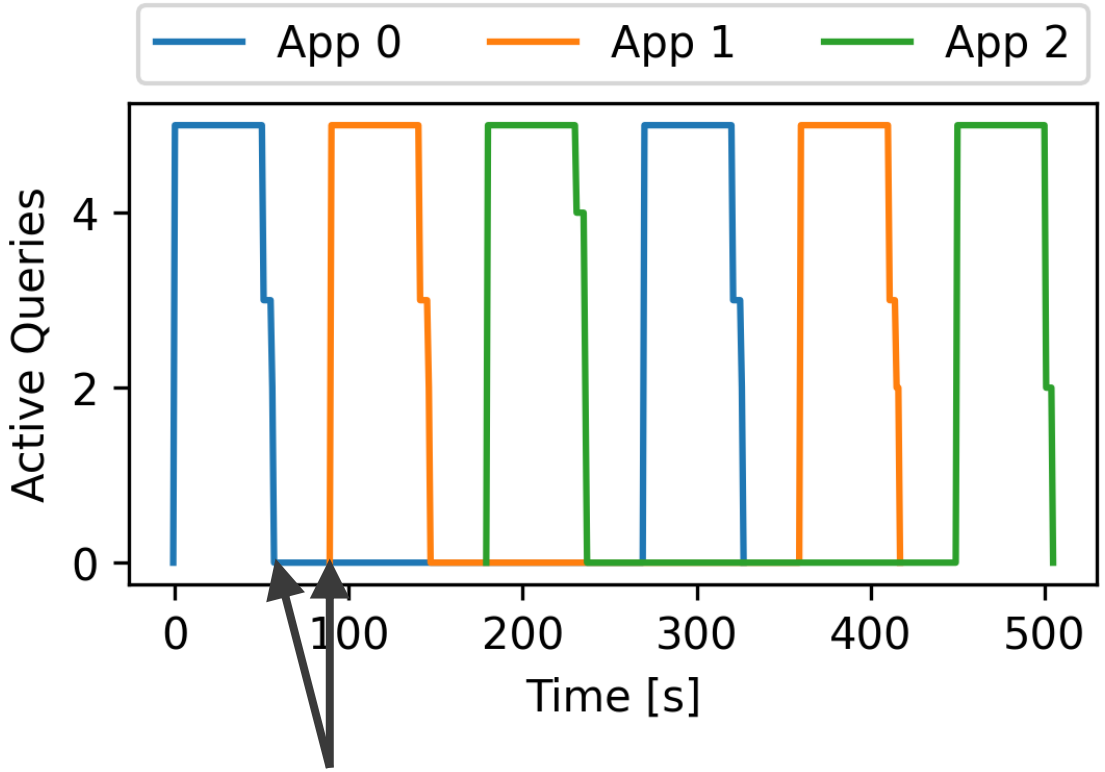
RQ3: What are the performance characteristics of the selected resource sharing mechanisms?

All experiments conducted on:



Results – *Alternating Bursts (with Spacing)*

Workload:

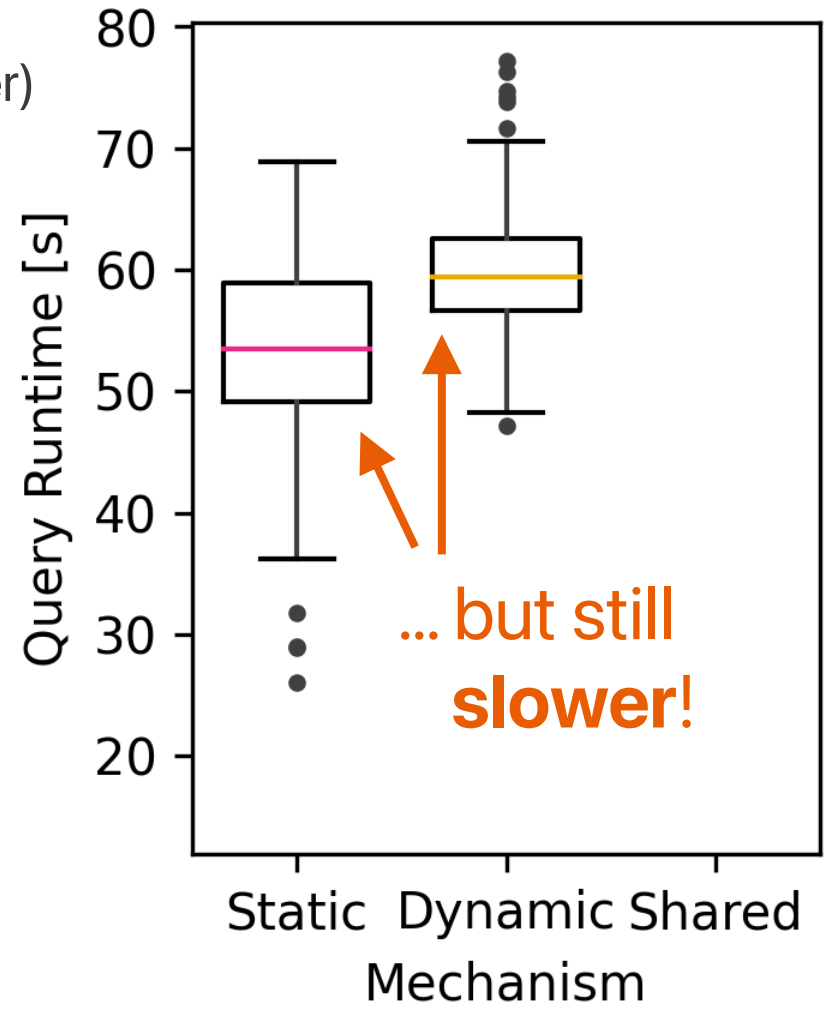


30s delay between activity!

(2x executor timeout → ideal conditions…)

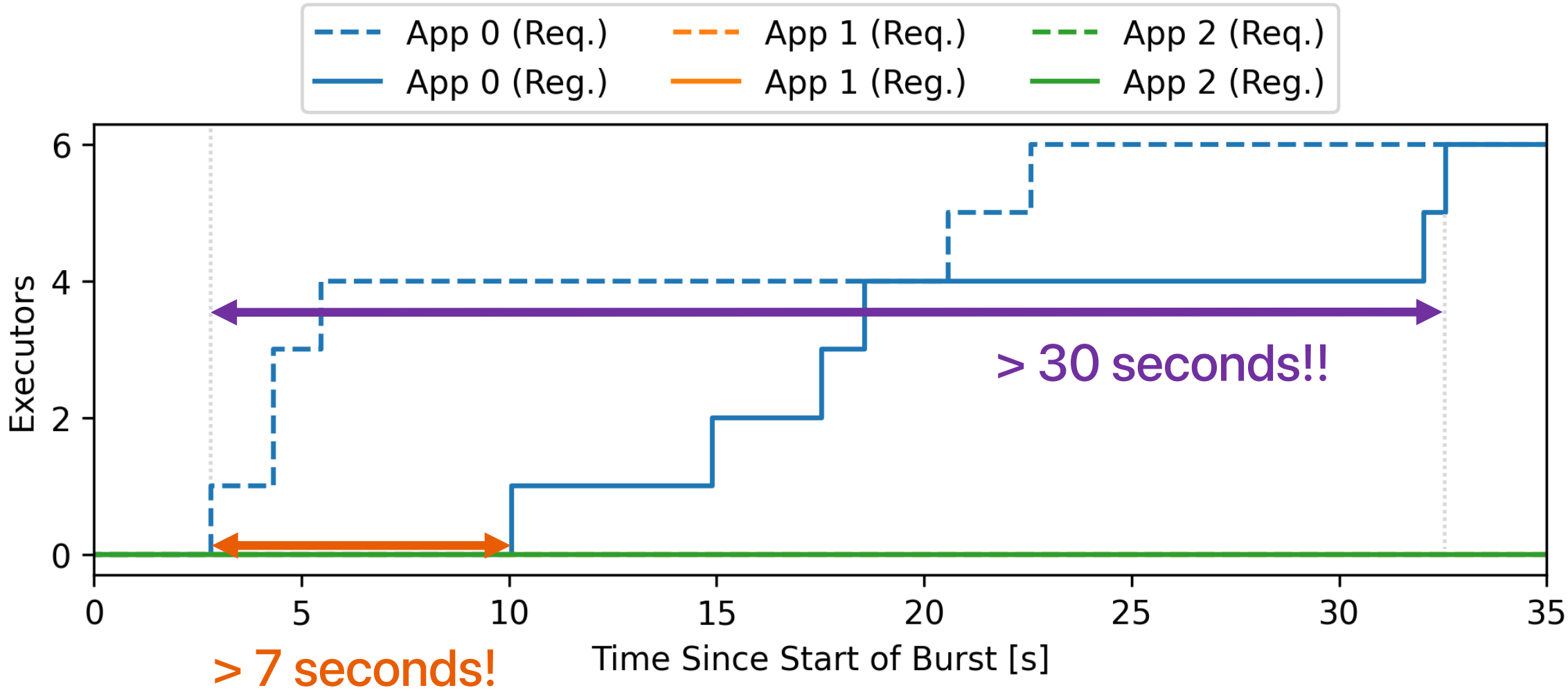
Results*:

(lower is better)

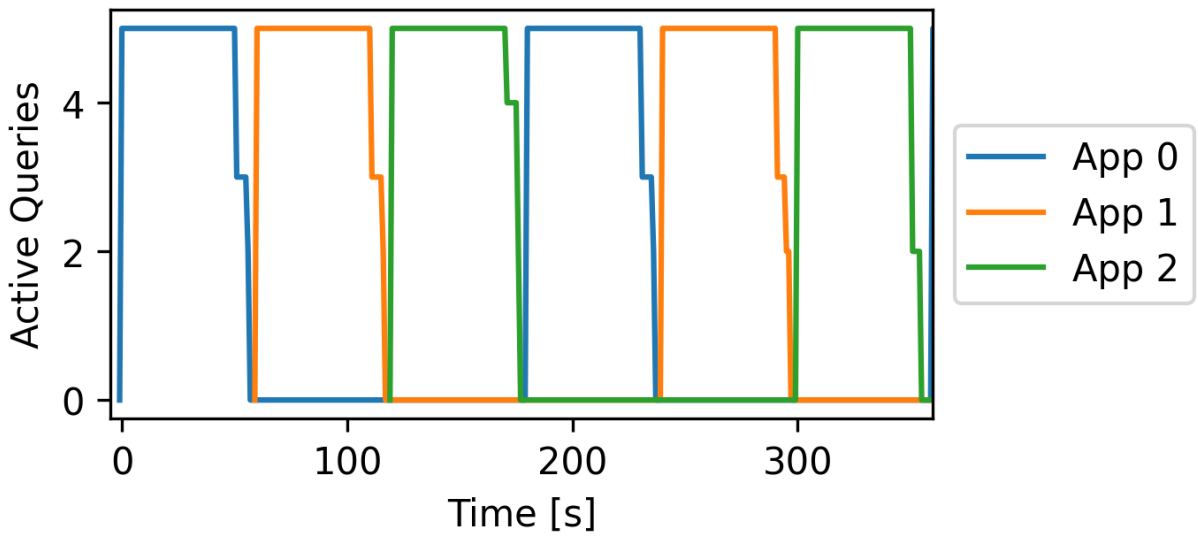


** after a 3-minute warm-up period*

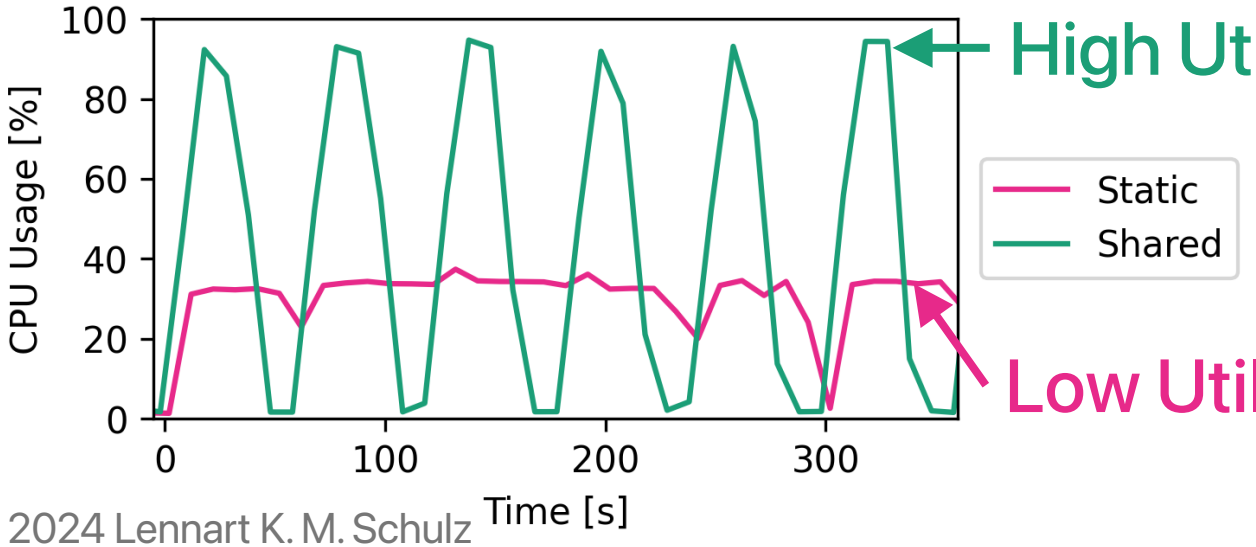
Results – *Dynamic Delays*



Results – *Alternating Bursts*



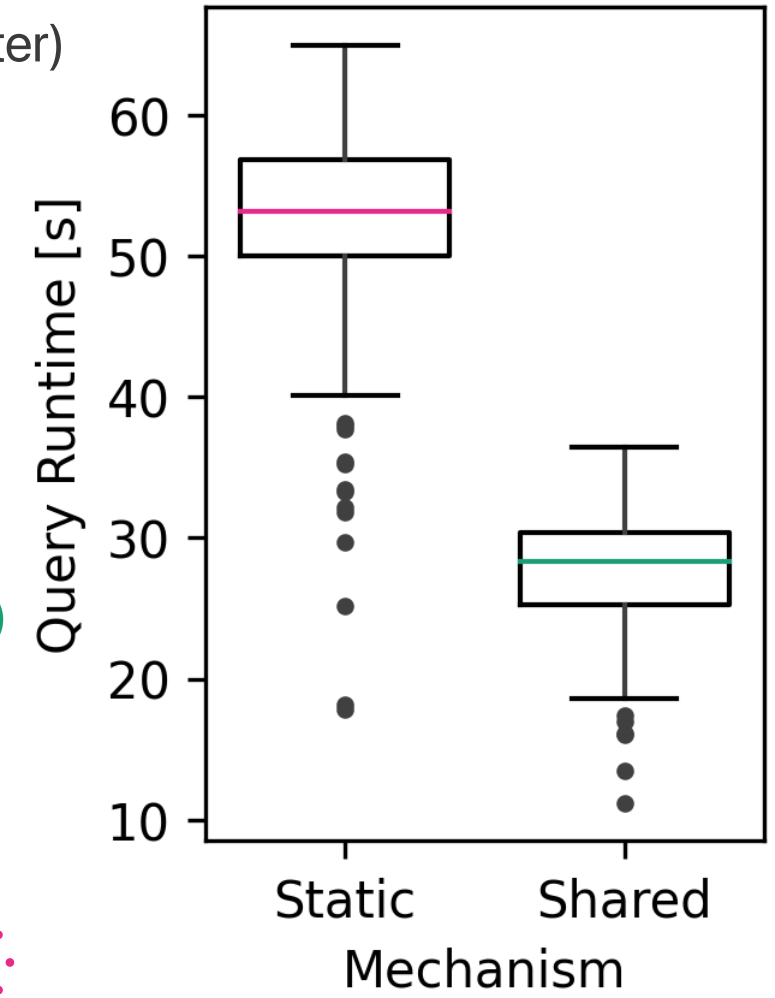
Executor CPU Usage (cluster):



High Utilization ✓

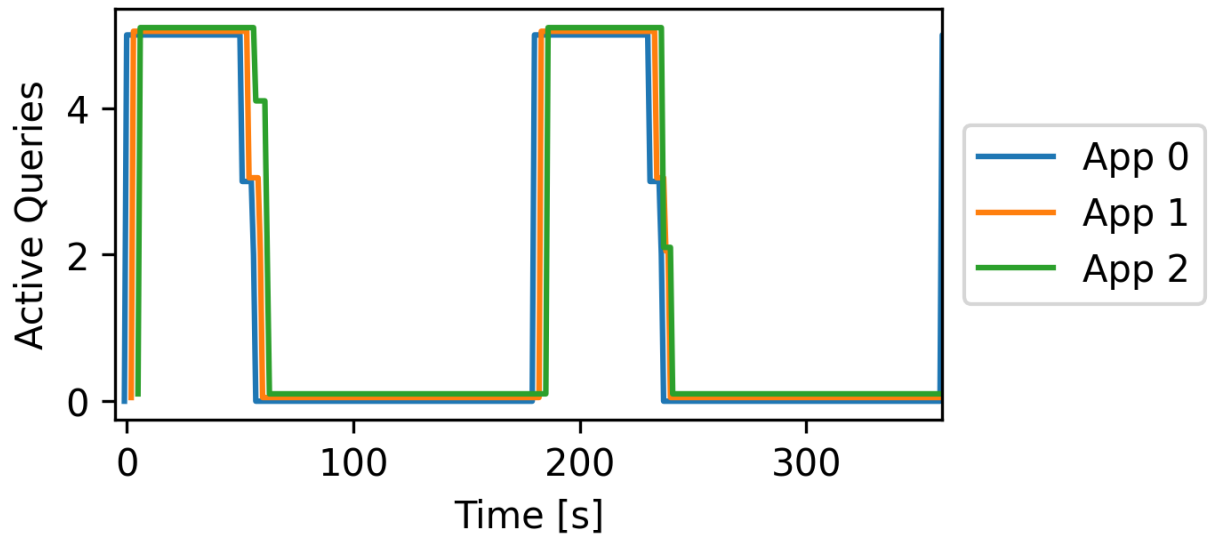
Low Utilization!

Results*:
(lower is better)

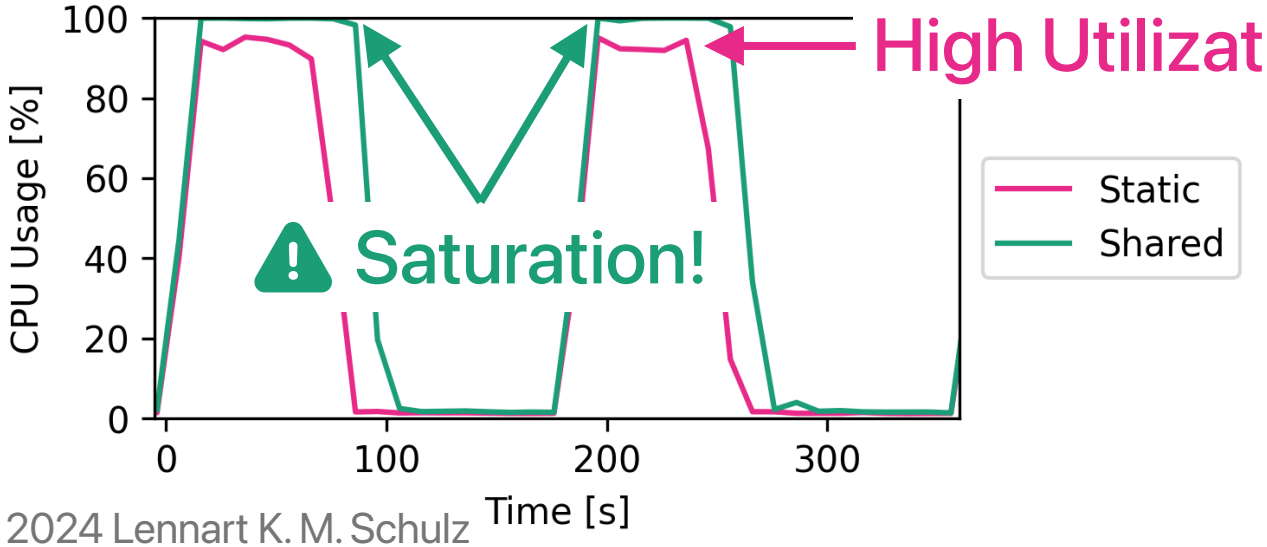


** after a 3-minute warm-up period*

Results – *Overlapping Bursts*

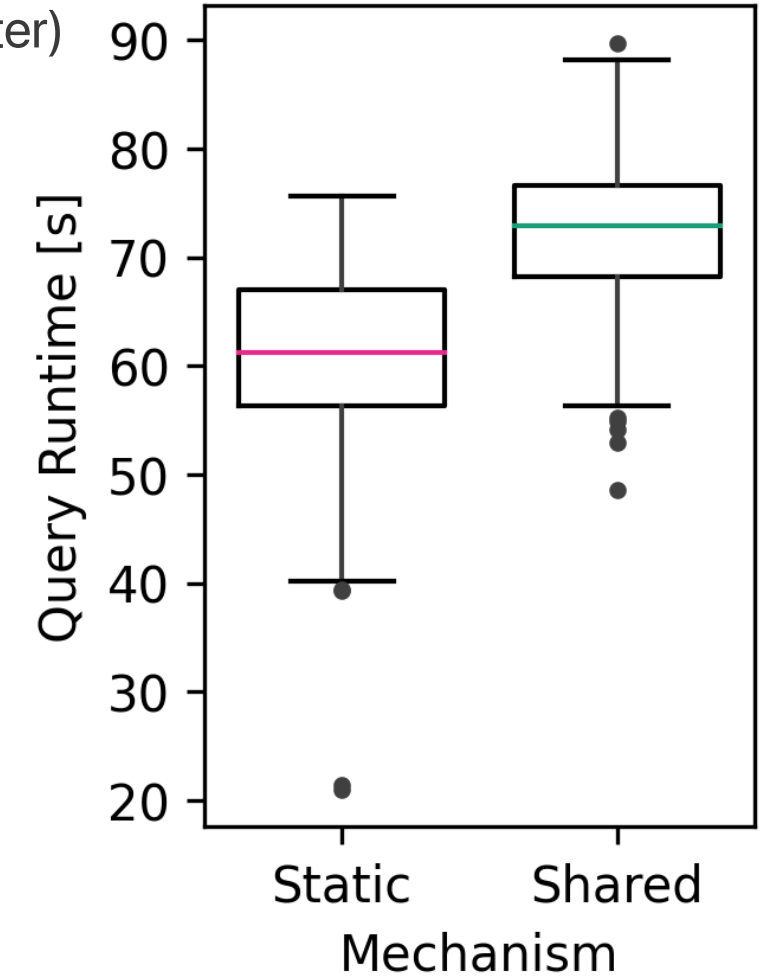


Executor CPU Usage (cluster):



High Utilization ✓

Results*:
(lower is better)



** after a 3-minute warm-up period*

Key Takeaways

→ Resource sharing has become a **ubiquitous** requirement in clusters.

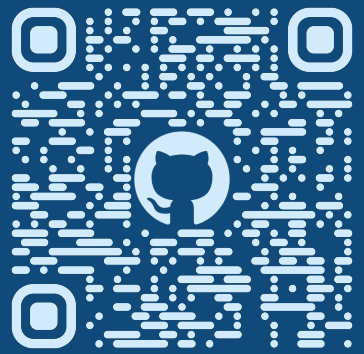
⚠ **No standard system exists** to evaluate resource sharing mechanisms!

We propose:

1. **Workload Generator Design**, and
2. **Experiment Infrastructure Framework**,

using which we evaluate the performance characteristics of three resource sharing mechanisms of Spark on Kubernetes.

available on **GitHub***



* https://github.com/lkm-schulz/FraPChaRSM

Extra Material

Workload Generator Design

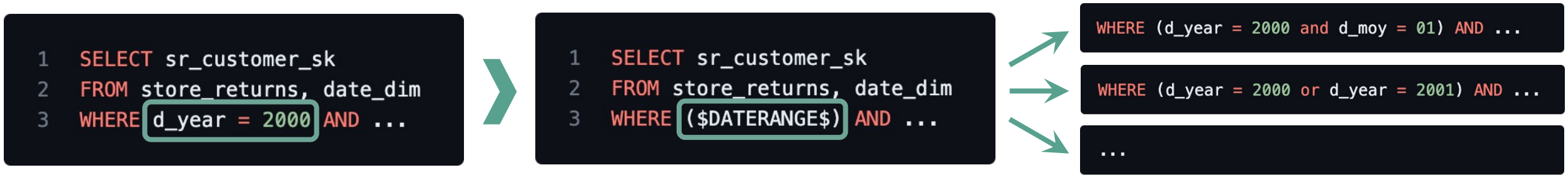
Workload: A (structured) **Collection** of **Queries** on **Data**.

1. The TPC-DS Dataset

2. Adapting the TPC-DS Queries

TPC-DS queries are good - but limited in number and inflexible.

Solution: modify given queries into numerous new ones of different scales by changing the range of included data:

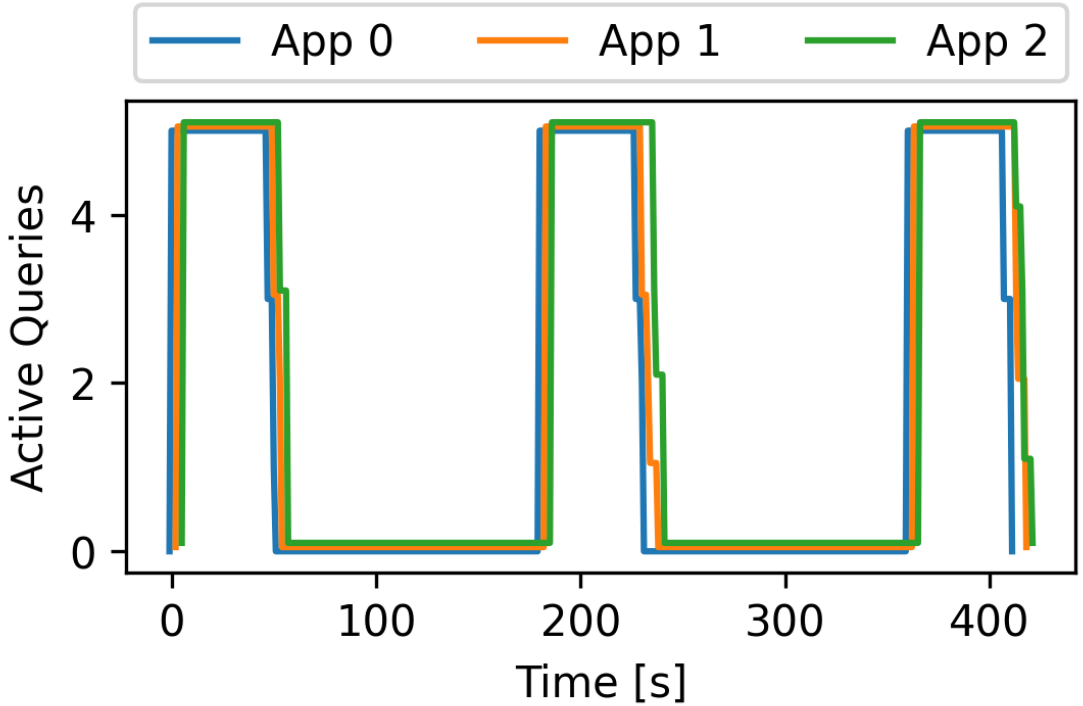


Workload Generator Design

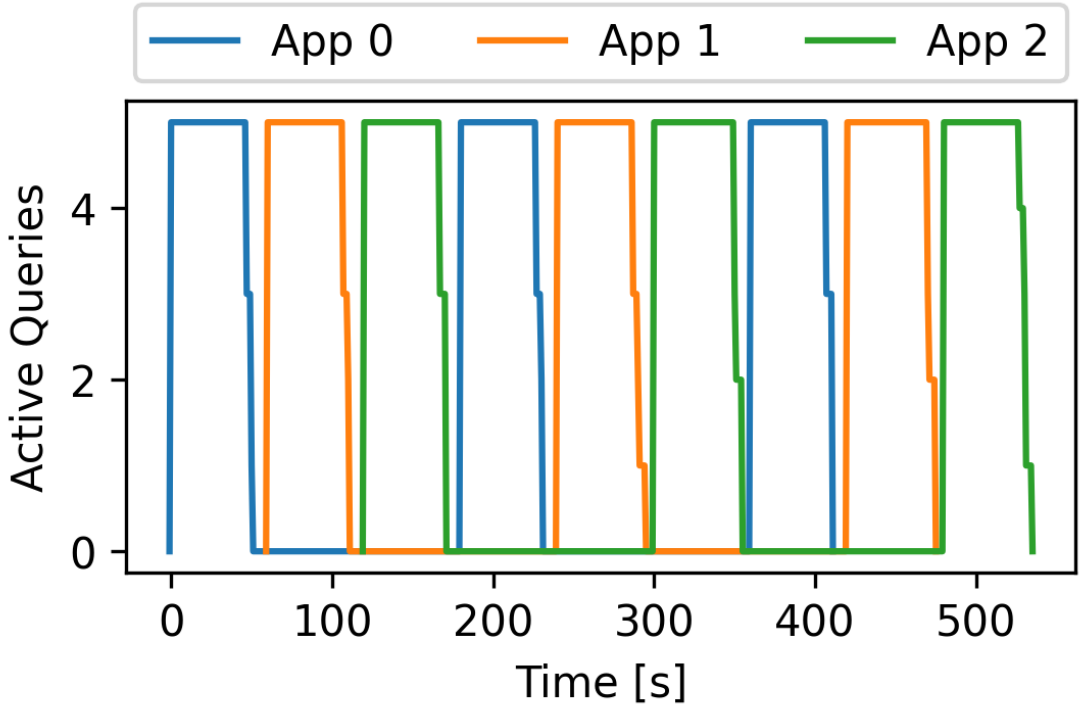
Workload: A (structured) **Collection** ✓ of **Queries** ✓ on **Data**. ✓

1. The TPC-DS Dataset
2. Adapting the TPC-DS Queries
3. Getting Query Times
4. Building Workloads

Overlapping Bursts:



Alternating Bursts:

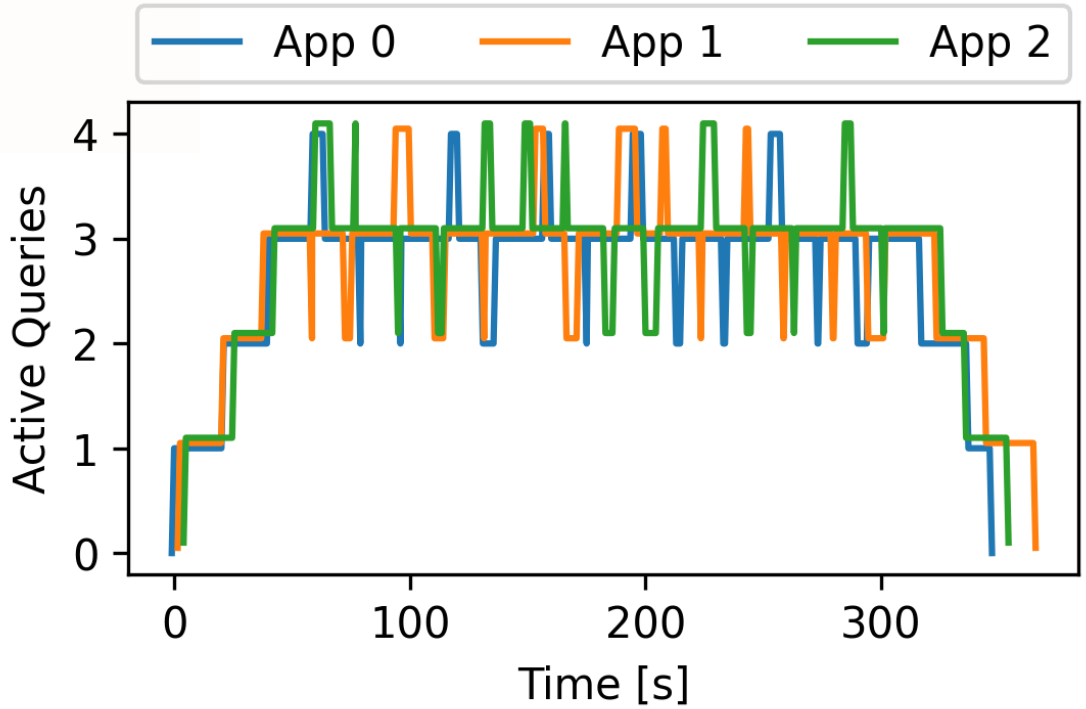


Workload Generator Design

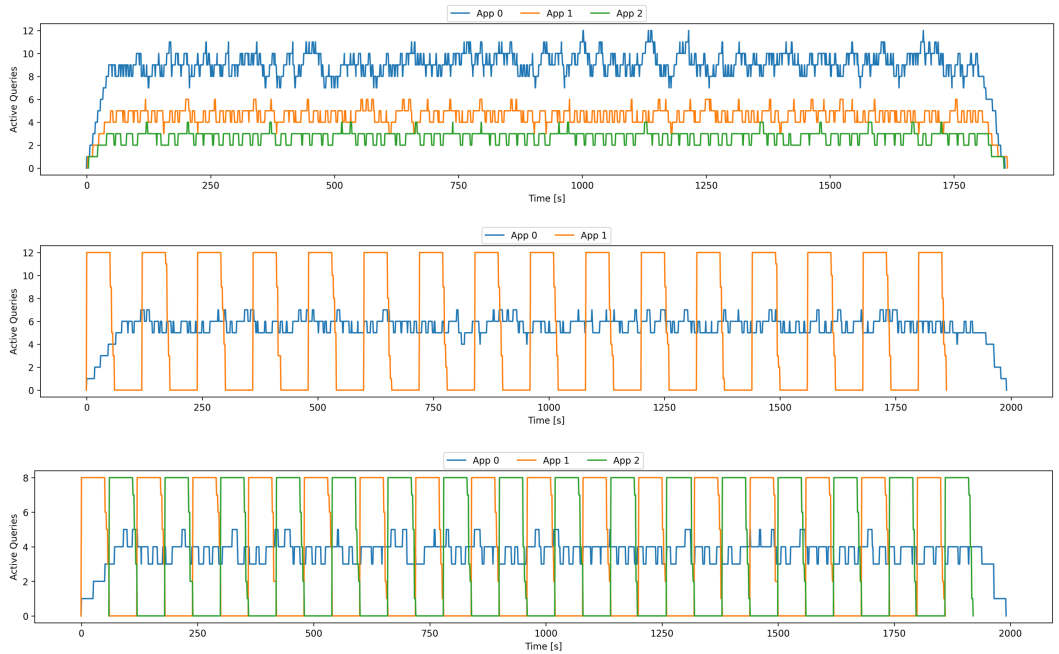
Workload: A (structured) **Collection** ✓ of **Queries** ✓ on **Data**. ✓

1. The TPC-DS Dataset
2. Adapting the TPC-DS Queries
3. Getting Query Times
4. Building Workloads

Constant Load:

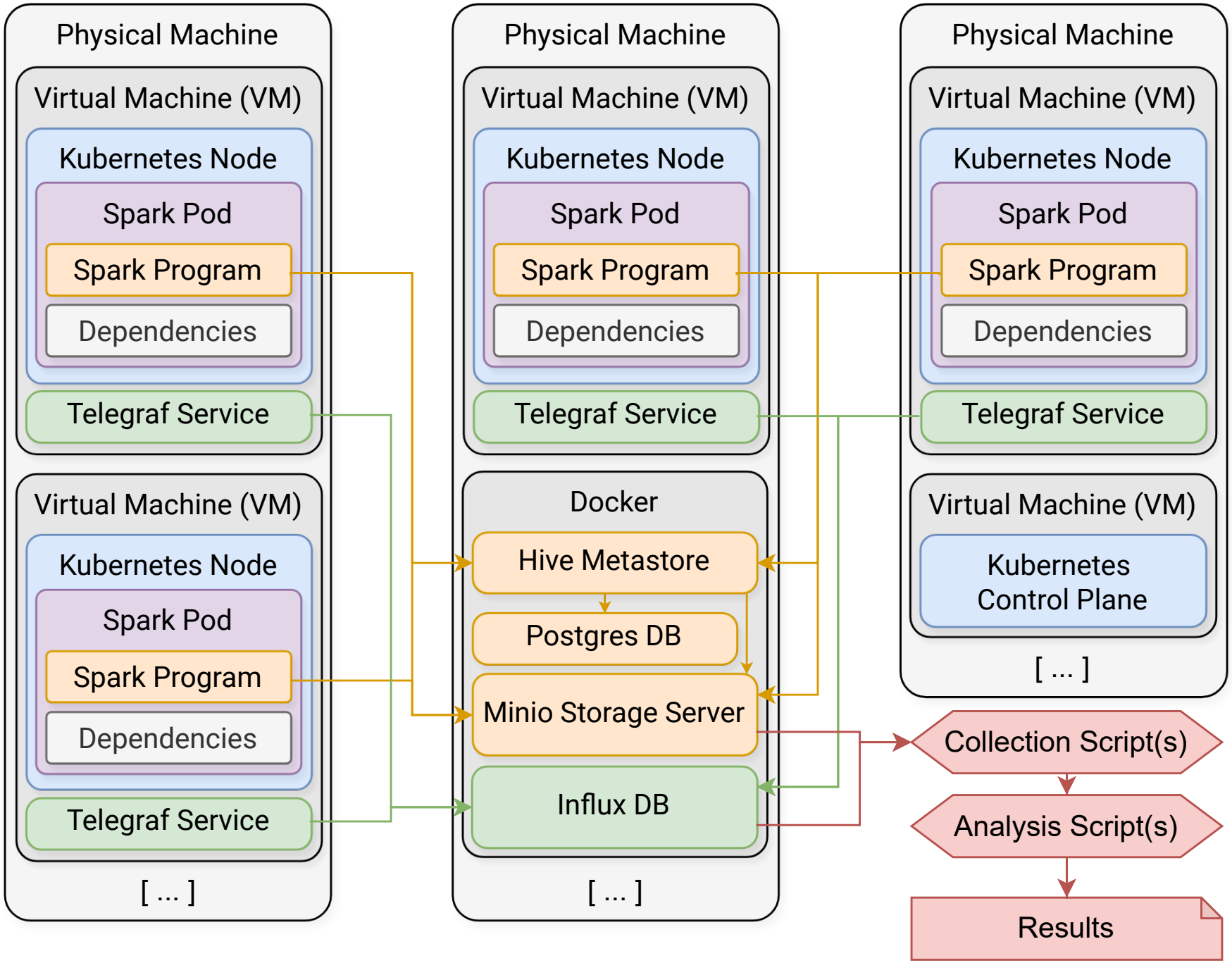


And many more...



Infra-structure

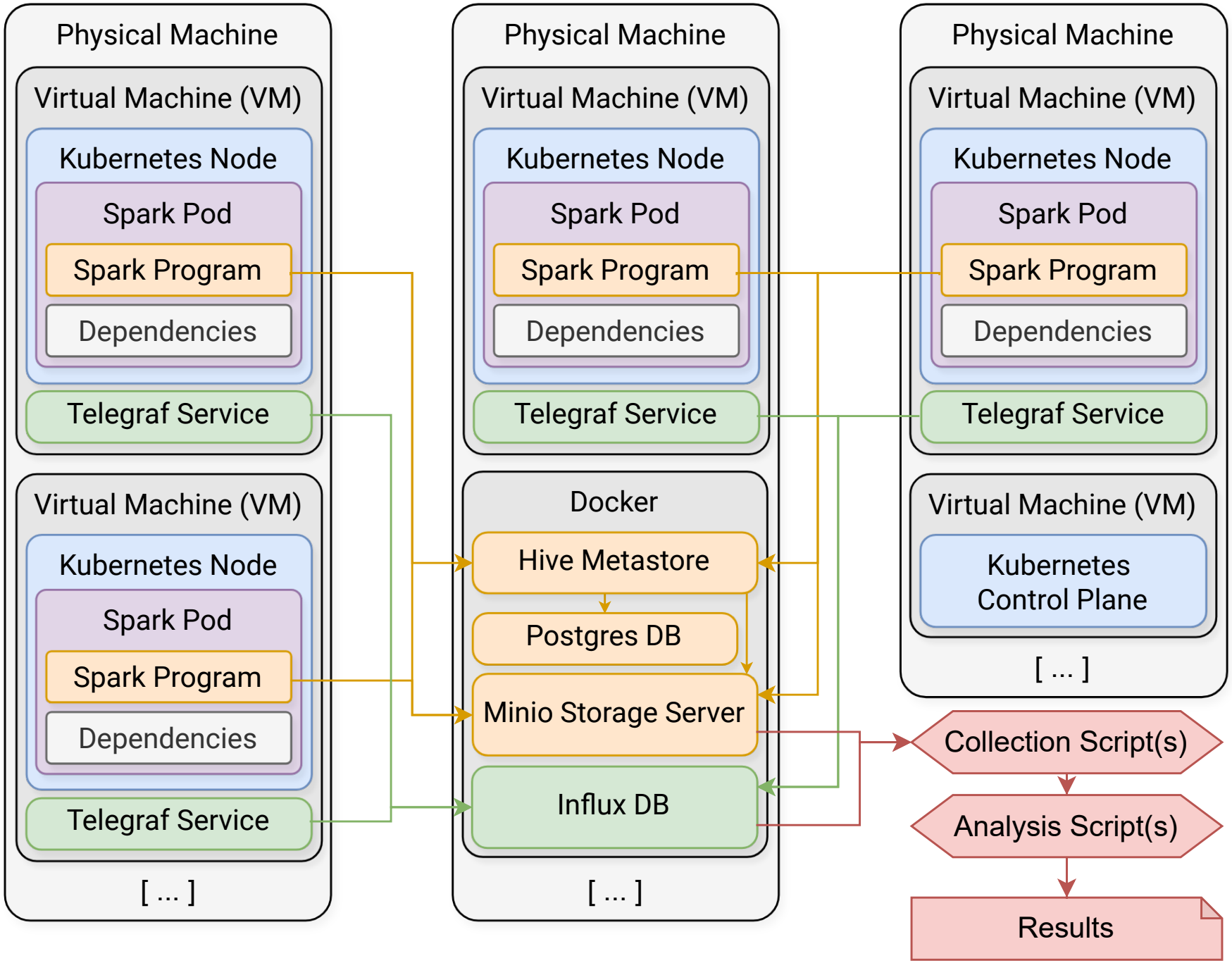
- ✓ reproducible
- ✓ automated
- ✓ extensible



@Large Research
Massivizing Computer Systems

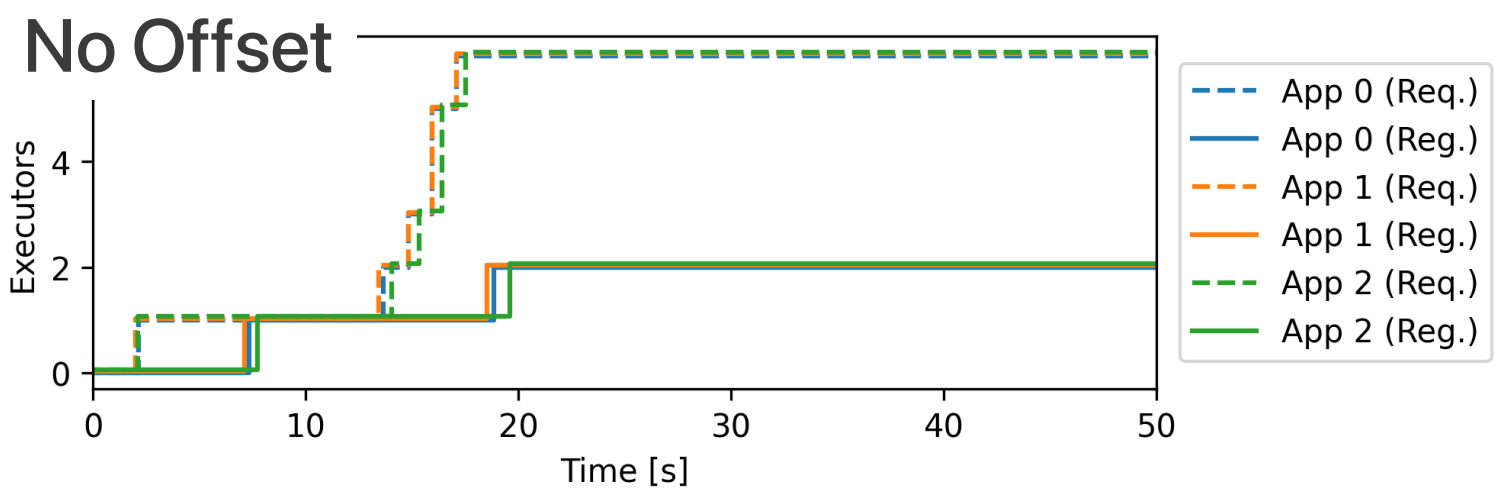
Infra-structure

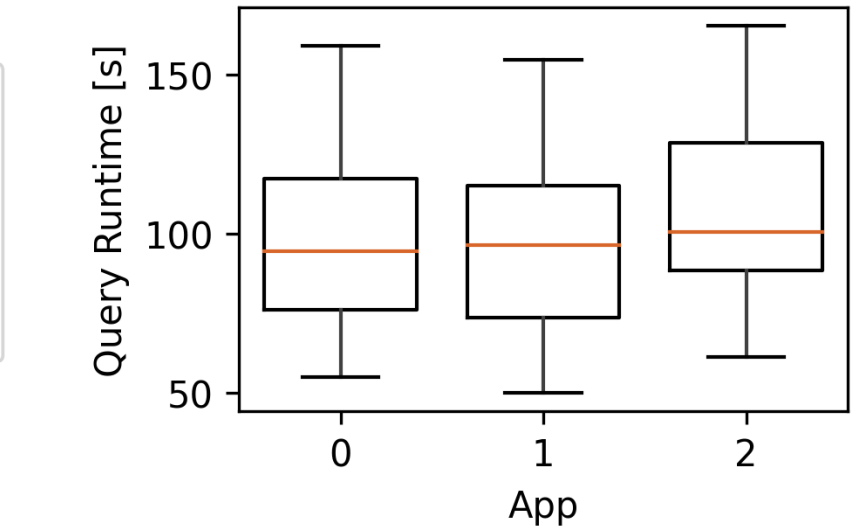
- ✓ reproducible
- ✓ automated
- ✓ extensible

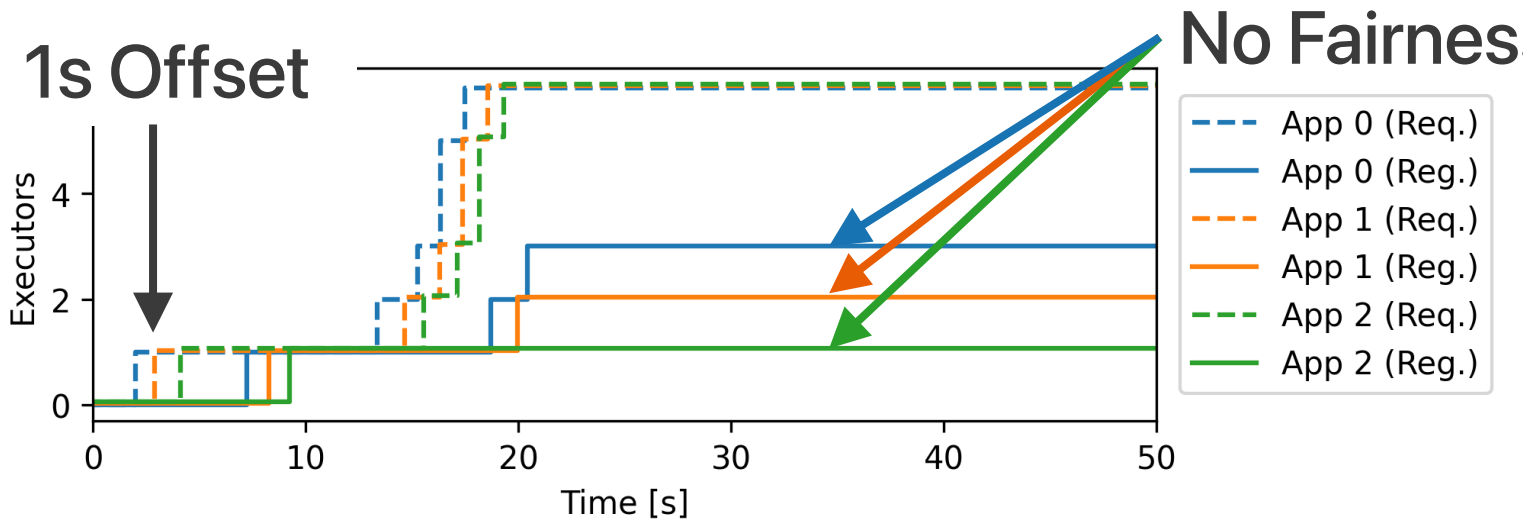


@Large Research
Massivizing Computer Systems

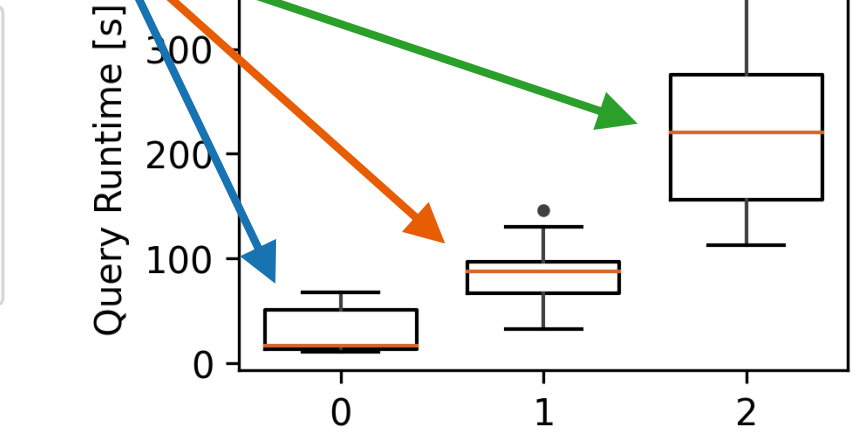
Results – *Dynamic (Un)Fairness*



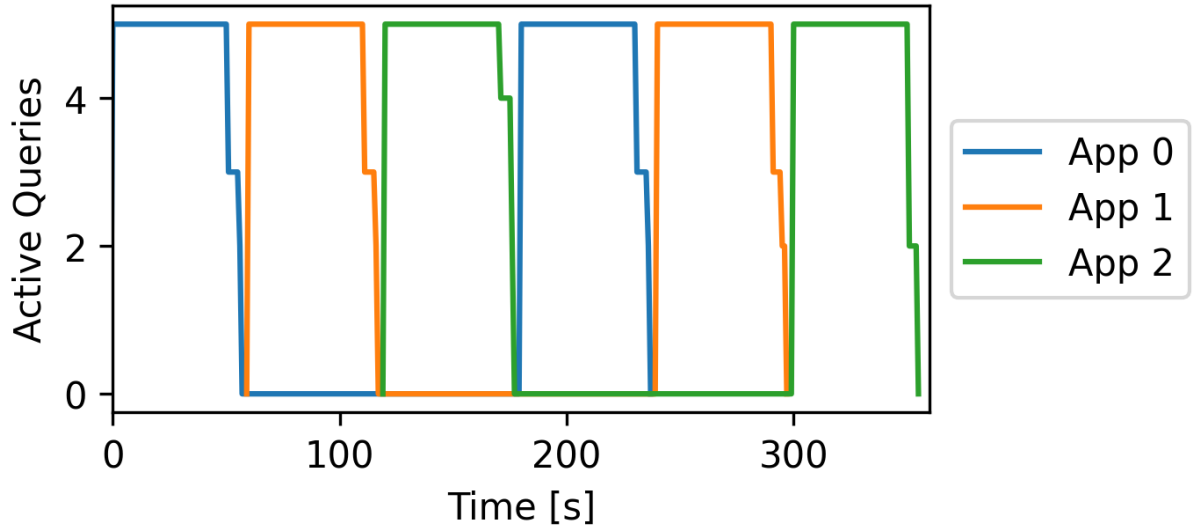


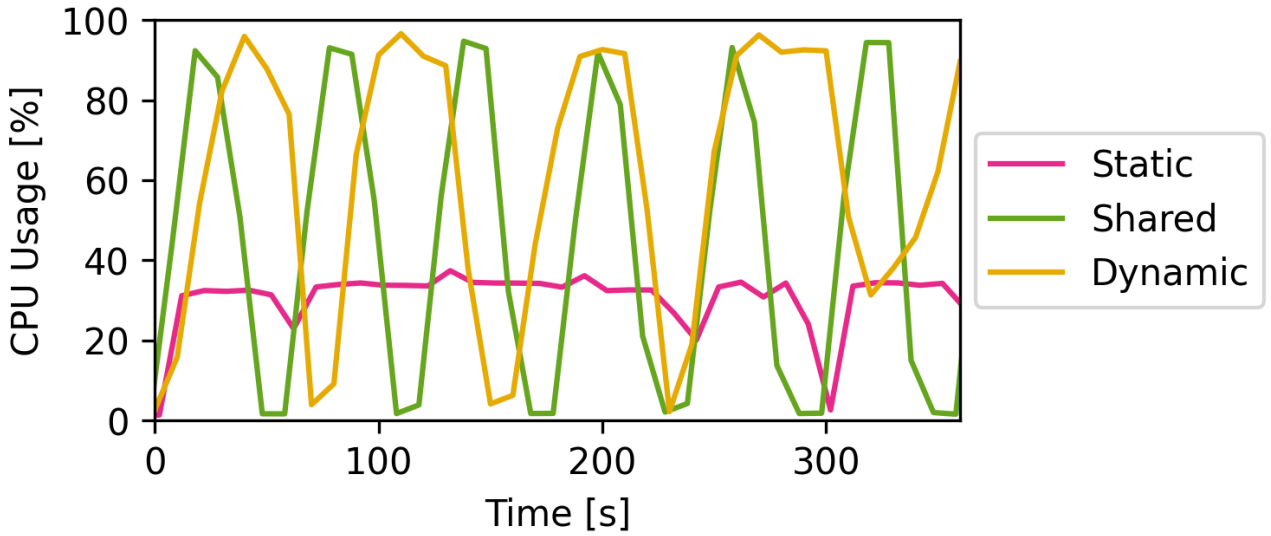


No Fairness!

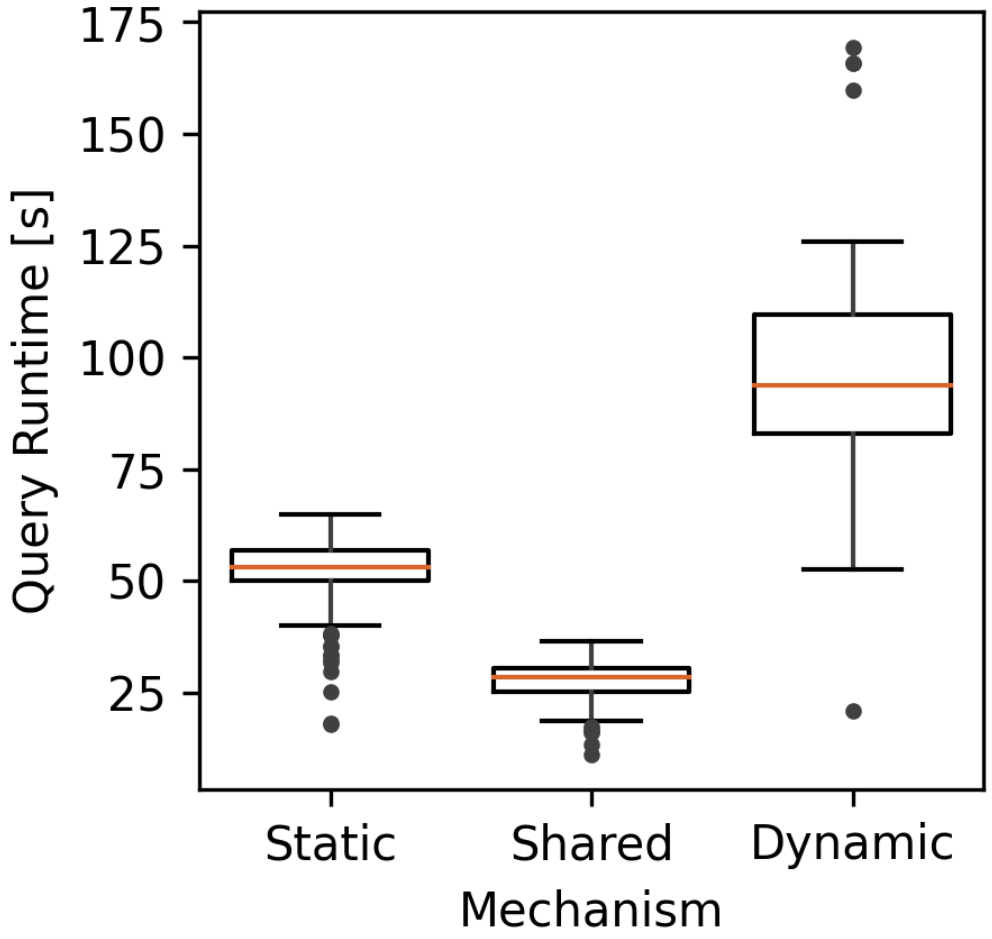


Results – Alternating Bursts





Results*:



** after a 3-minute warm-up period*