

NSF/IUCRC CAC PROJECT

---

# INTEGRATED VISUALIZING, MONITORING, AND MANAGING HPC SYSTEMS

Jie Li

Doctoral Student, TTU

04/30/2021

Advisors:

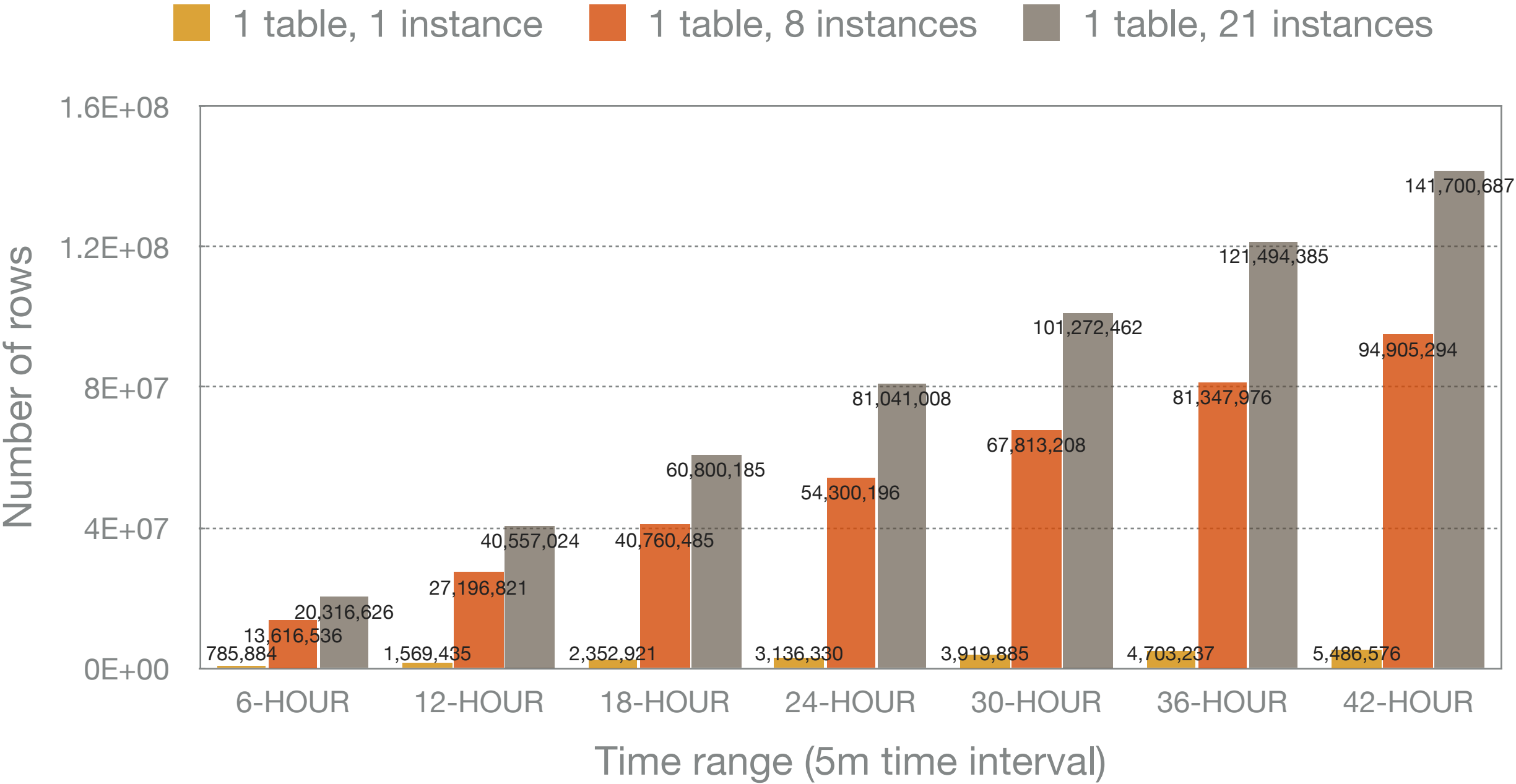
Mr. Jon Hass, SW Architect, Dell Inc.

Dr. Alan Sill, Managing Director, HPCC, TTU

Dr. Yong Chen, Associate Professor, CS Dept, TTU

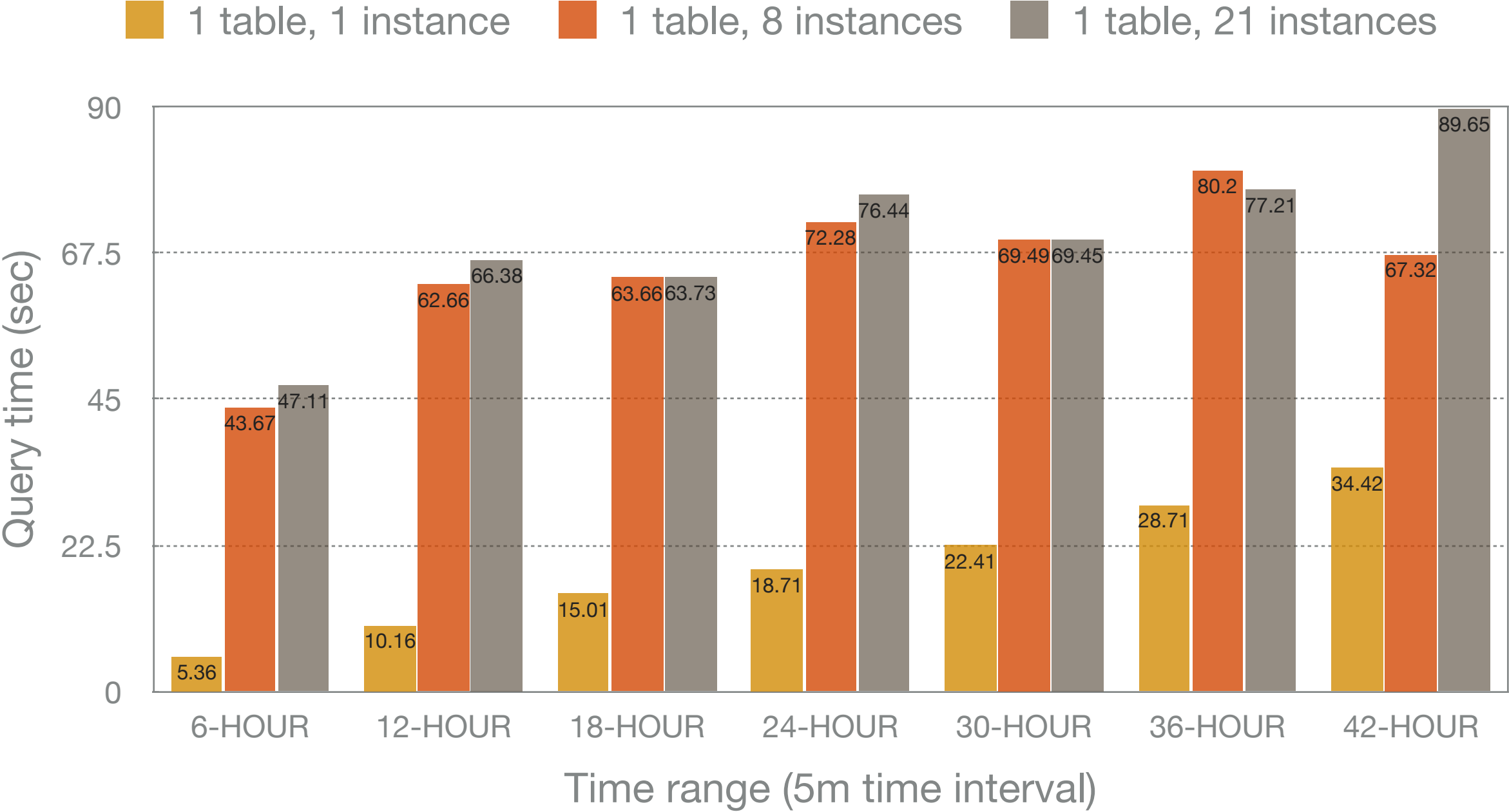
Dr. Tommy Dang, Assistant Professor, CS Dept, TTU

# TIMESCALEDB



Query metrics (1 table) of 240 nodes from TimeScaleDB concurrently

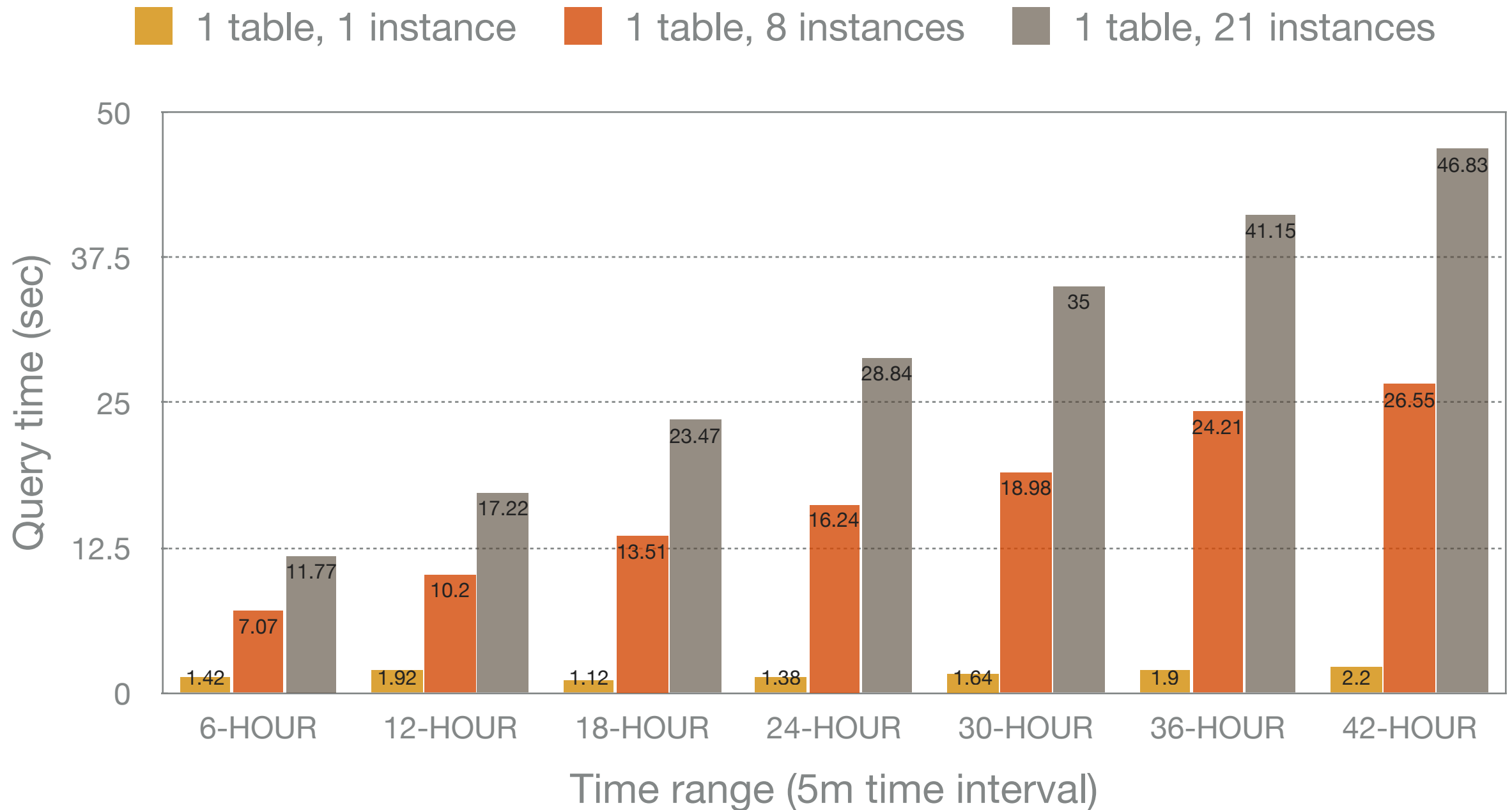
# TIMESCALEDB



Query metrics (1 table) of 240 nodes from TimeScaleDB concurrently

```
SELECT time_bucket_gapfill('{interval}', timestamp) as time, {aggregate}(value) from slurm.{metric} WHERE nodeid = {host_id} AND timestamp >= '{start}' AND timestamp < '{end}' GROUP BY time ORDER BY time;
```

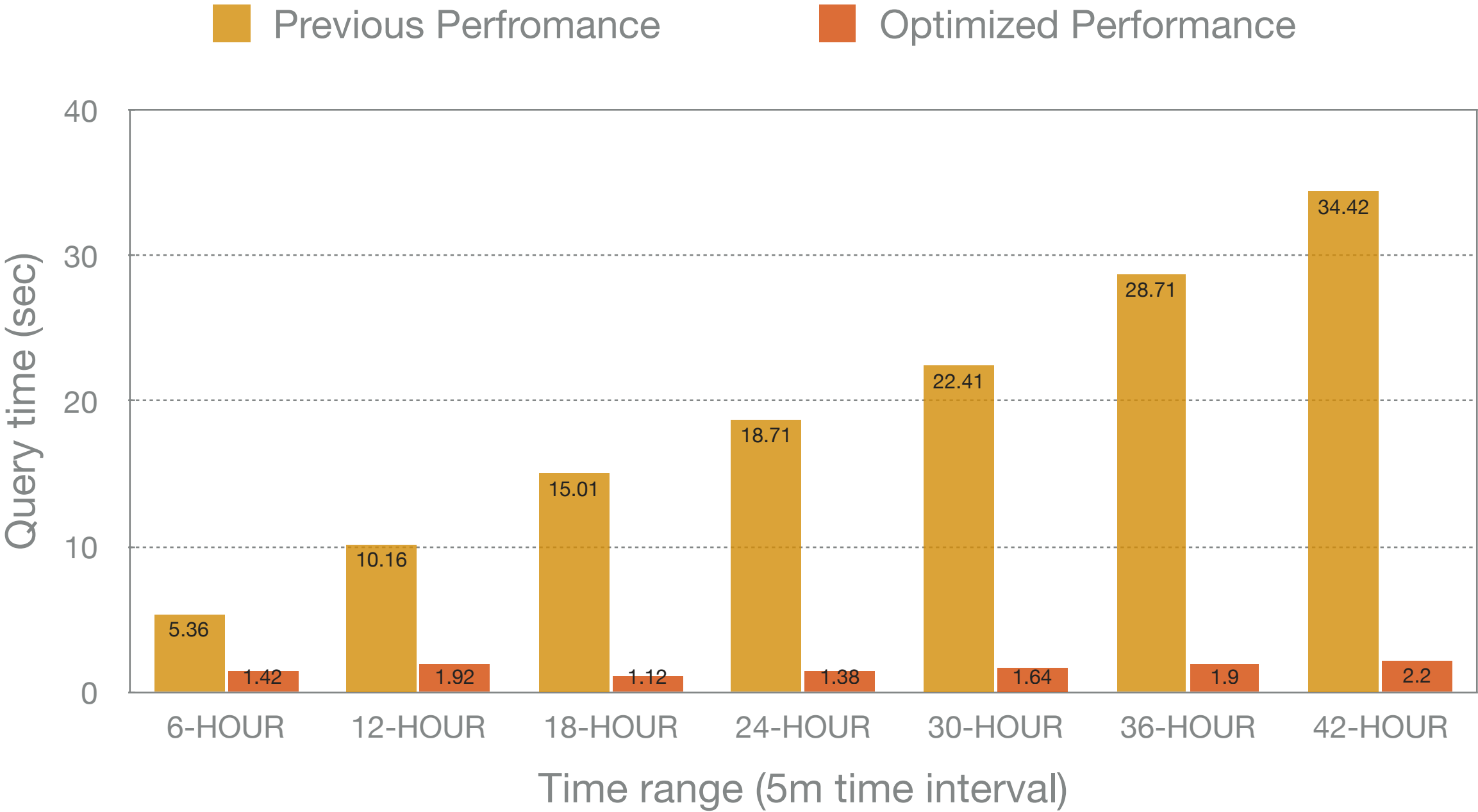
# TIMESCALEDB - OPTIMIZED



Query metrics (1 table) of 240 nodes from TimeScaleDB

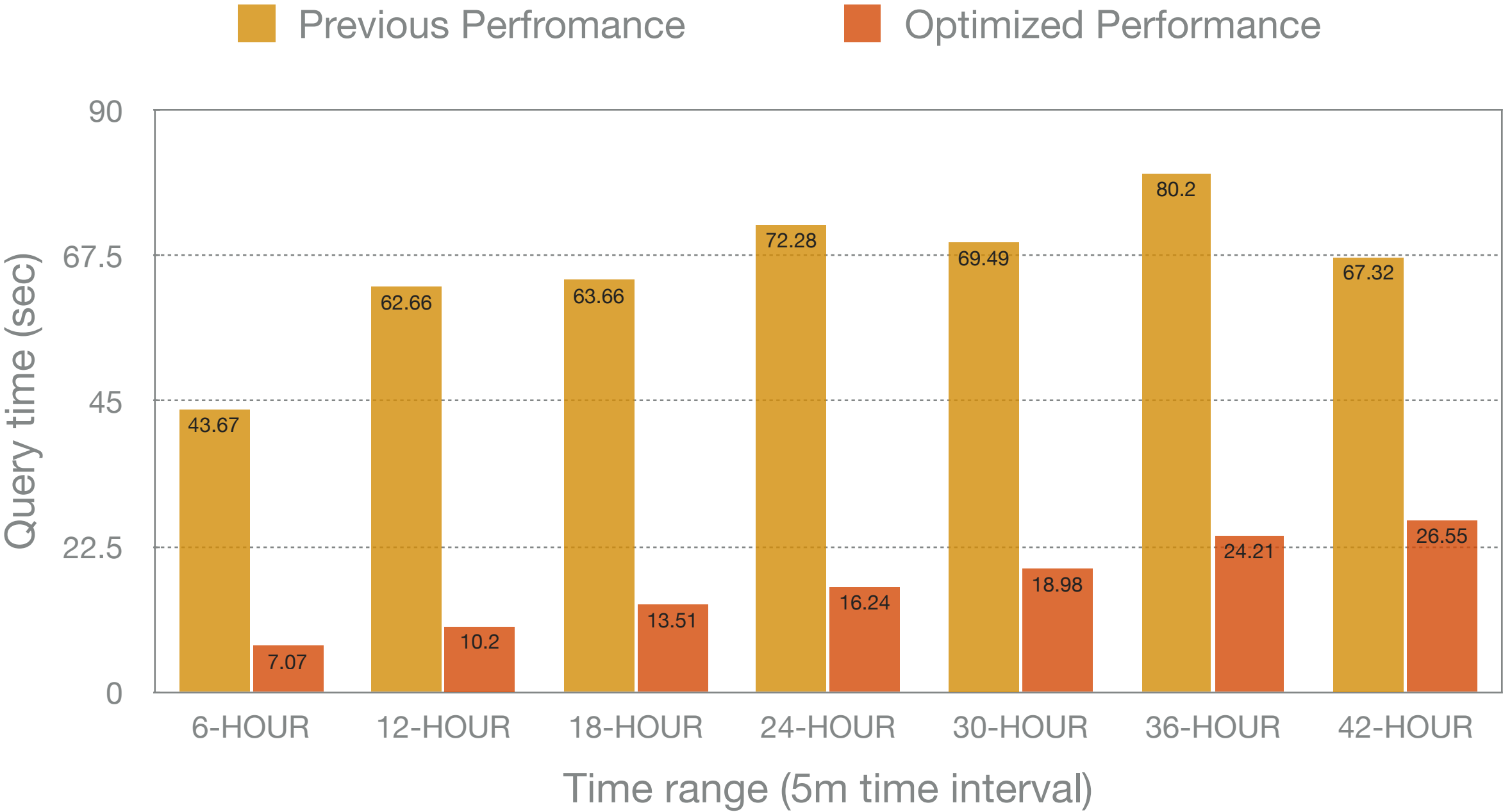
```
SELECT time_bucket_gapfill('{interval}', timestamp) as time, nodeid, array_agg(jobs) as jobs, array_agg(cpus) as cpus from slurm.{metric} WHERE timestamp >= '{start}' AND timestamp < '{end}' GROUP BY time, nodeid, jobs, cpus ORDER BY time;
```

# IMPROVEMENT



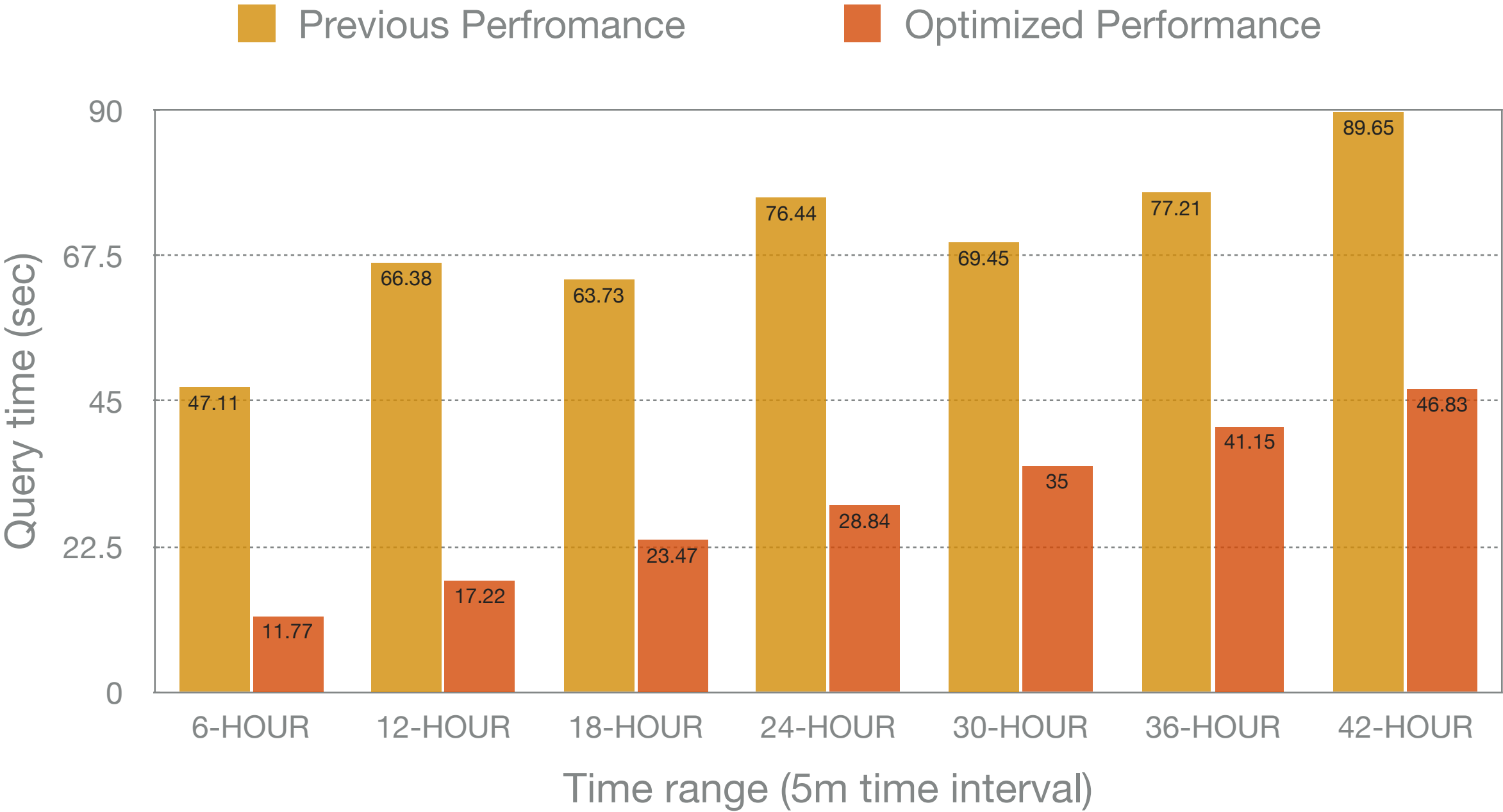
Query metrics (1 table, 1 instance) of 240 nodes from TimeScaleDB

# IMPROVEMENT



Query metrics (1 table, 8 instances) of 240 nodes from TimeScaleDB

# IMPROVEMENT



Query metrics (1 table, 21 instances) of 240 nodes from TimeScaleDB

# DISCUSSION

---

## Previous SQL

```
SELECT time_bucket_gapfill('{interval}', timestamp) as time, {aggregate}(value)
from slurm.{metric} WHERE nodeid = {host_id} AND timestamp >= '{start}' AND
timestamp < '{end}' GROUP BY time ORDER BY time;
```

## Optimized SQL

```
SELECT time_bucket_gapfill('{interval}', timestamp) as time, nodeid,
array_agg(jobs) as jobs, array_agg(cpus) as cpus from slurm.{metric} WHERE
timestamp >= '{start}' AND timestamp < '{end}' GROUP BY time, nodeid, jobs,
cpus ORDER BY time;
```


- ▶ The node id constraint in the SQL degrades the query performance
- ▶ Instead of filtering nodes by TimescaleDB, a better approach is to query all the data using only time constraints and then filter the nodes by ourself.



# DISCUSSION

---

- ▶ The size of the collected data has reached to **2.3 T** (From March 17 to April 29.), consuming **39%** of the data storage disk (5.9T) on Hugo.
- ▶ <https://medium.com/timescale/optimizing-queries-timescaledb-hypertables-with-partitions-postgresql-6366873a995d>, **the query overhead will increase** with the entire data size, even the query itself is the same.



**QUESTIONS?/COMMENTS?**