OMNI: OCS Middleware for Network Improvement

Sri Pramodh Rachuri[1], Mihir Kestur[1], Stefan Schmid [2], Anshul Gandhi[1]

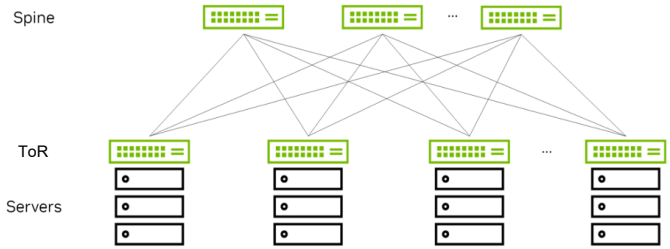
[1]PACE Lab, Stony Brook University, USA
[2]TU Berlin, Germany

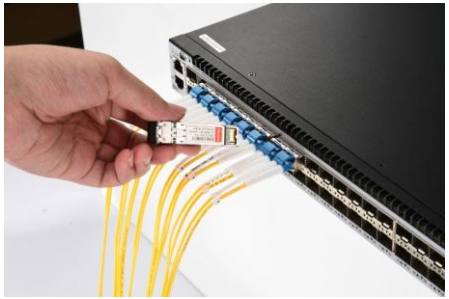
The content in this slide deck is prepared using the information from our paper- OMNI, MASCOTS 2026

srachuri@cs.stonybrook.edu
pramodh.rachuri.dev

Background – Datacenter Networks

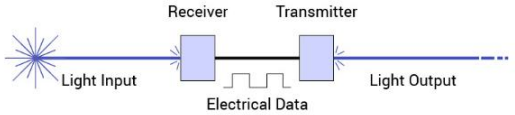
Clos network topology





Electrical Packet Switching (EPS)

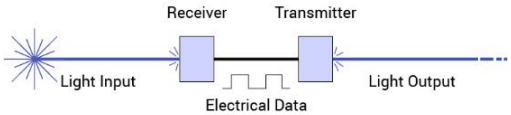
- Latency of O-E-O conversions
- Electric routing latency delays
- Shared bandwidth
- Always available connectivity



Background – Datacenter Networks

Electrical Packet Switching (EPS)

- Latency of O-E-O conversions
- Electric routing latency delays
- Shared bandwidth
- Always available connectivity



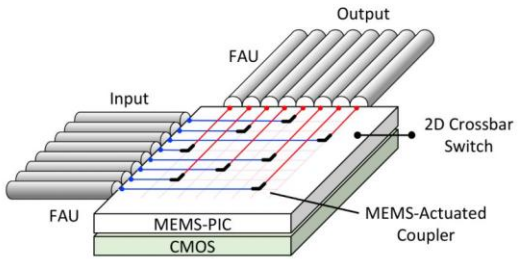
Optical Circuit Switching (OCS)

- Physical optical path deflection
- 10x-100x theoretical bandwidth capacity and lower latency
- Contention-free optical circuits



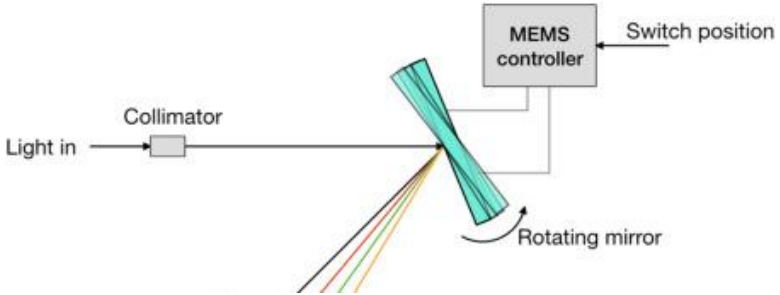
Background – Our OCS assumptions

- Dedicated Paths
  - Contention-free optical connections
- Directional Links
  - Independent upload and download
- Bijective Mapping
  - O(n!) search space



Background – OCS Challenges

- Input/output Selection
  - Limited simultaneous optical connections
- Reconfiguration Delay
  - Switching delay between different outputs causing overheads
- Dynamic network util
  - Unpredictable workload patterns



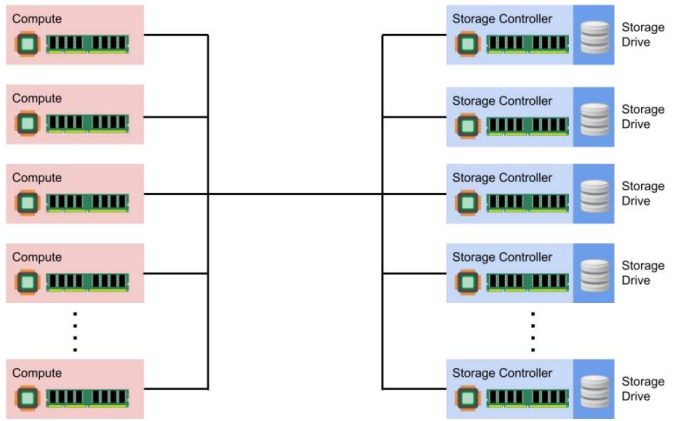
Background – Hybrid OCS

- Multiple connections
  - Dual NIC
- Electrical Packet Switching (EPS)
  - Always available
  - Electronic routing
  - Low bandwidth + High latency
- Optical Circuit Switching (OCS)
  - Intermittent
  - Circuit switching
  - High bandwidth + Low latency

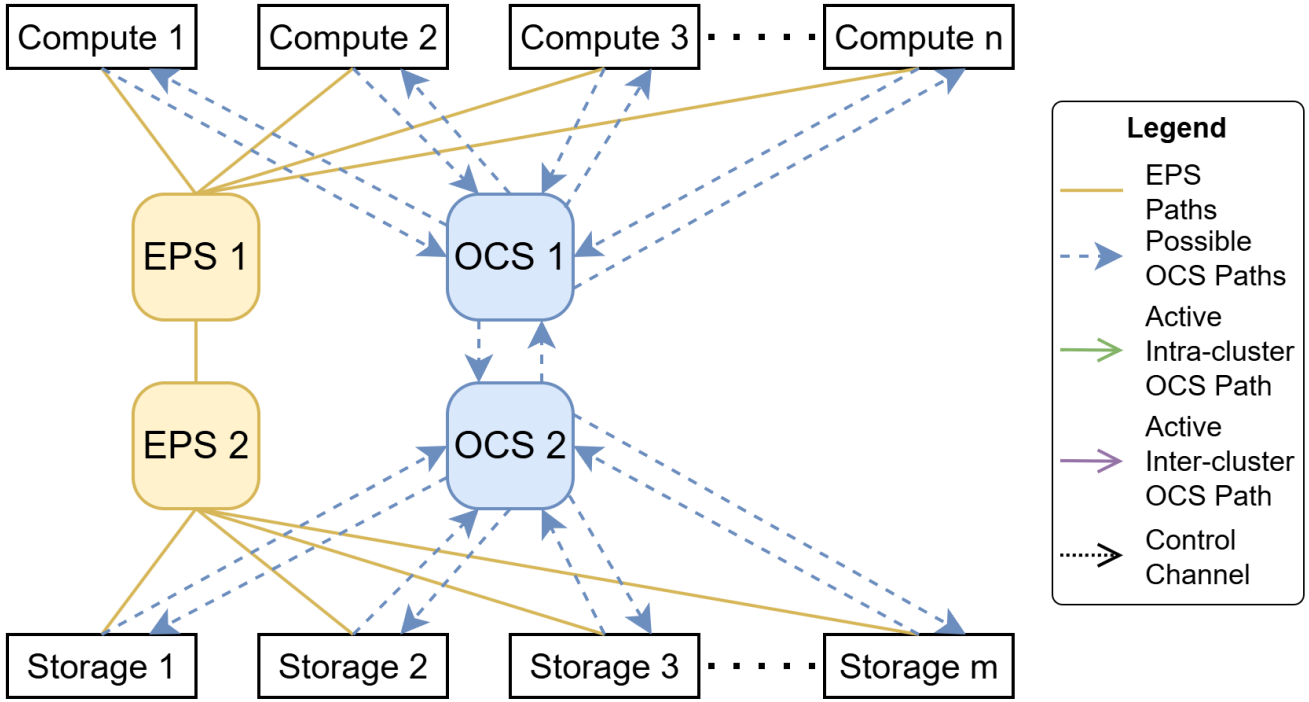


Background – Disaggregated Infrastructure

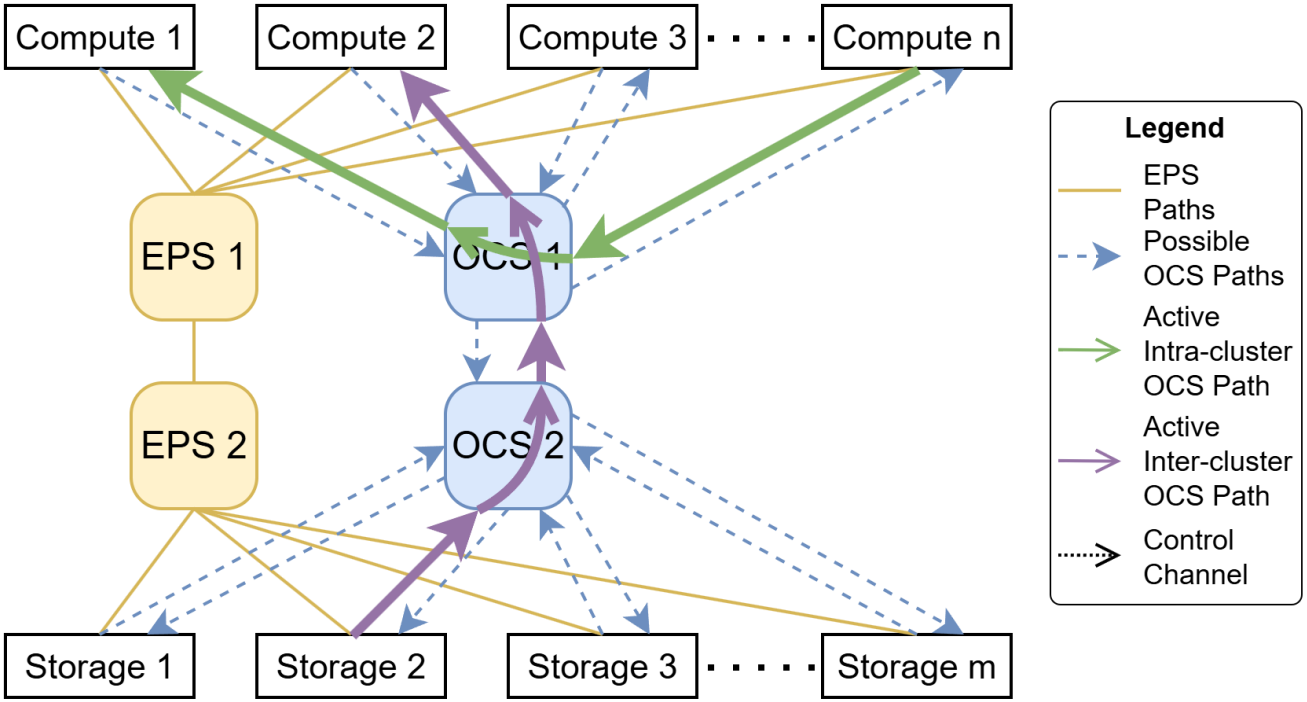
- Separated pools
  - Compute pool – Apache Spark
    - High compute + Low storage
    - Ephemeral
  - Storage pool – HDFS
    - Low compute + High storage
    - Persistent
- Pools connected over network
  - High inter-pool data transfers
  - Bottleneck



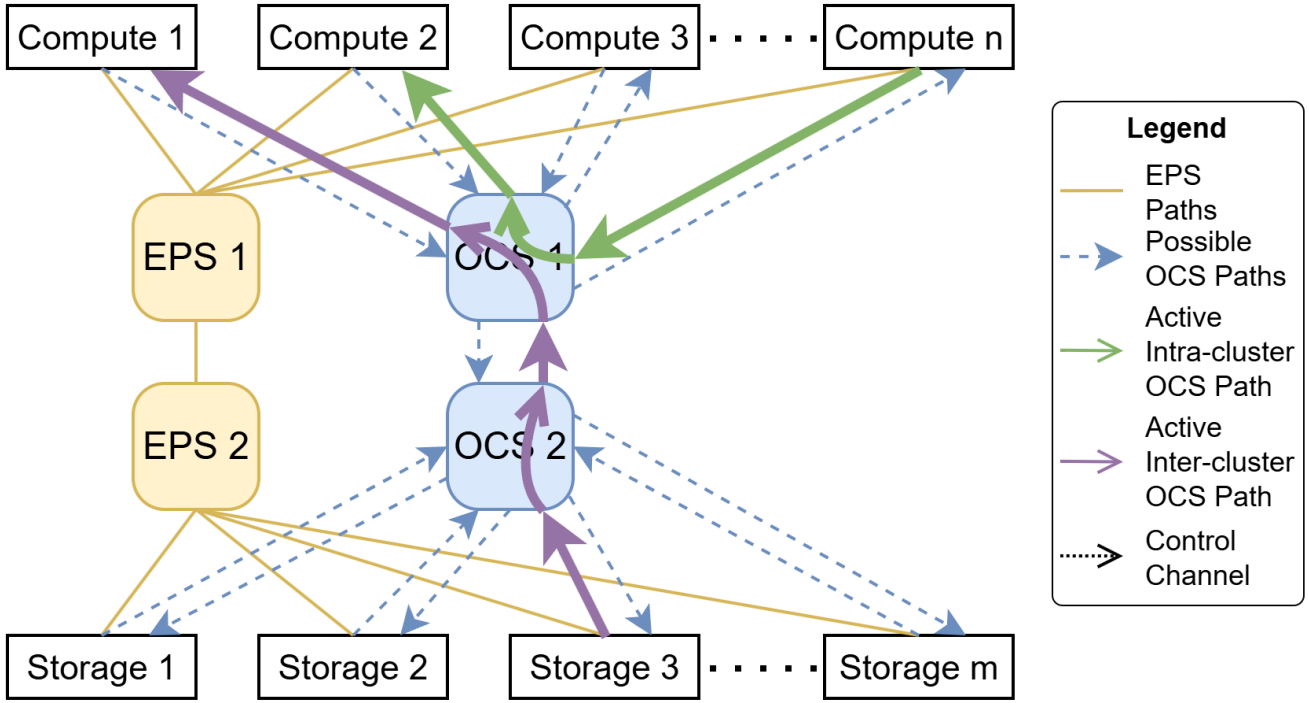
System Overview



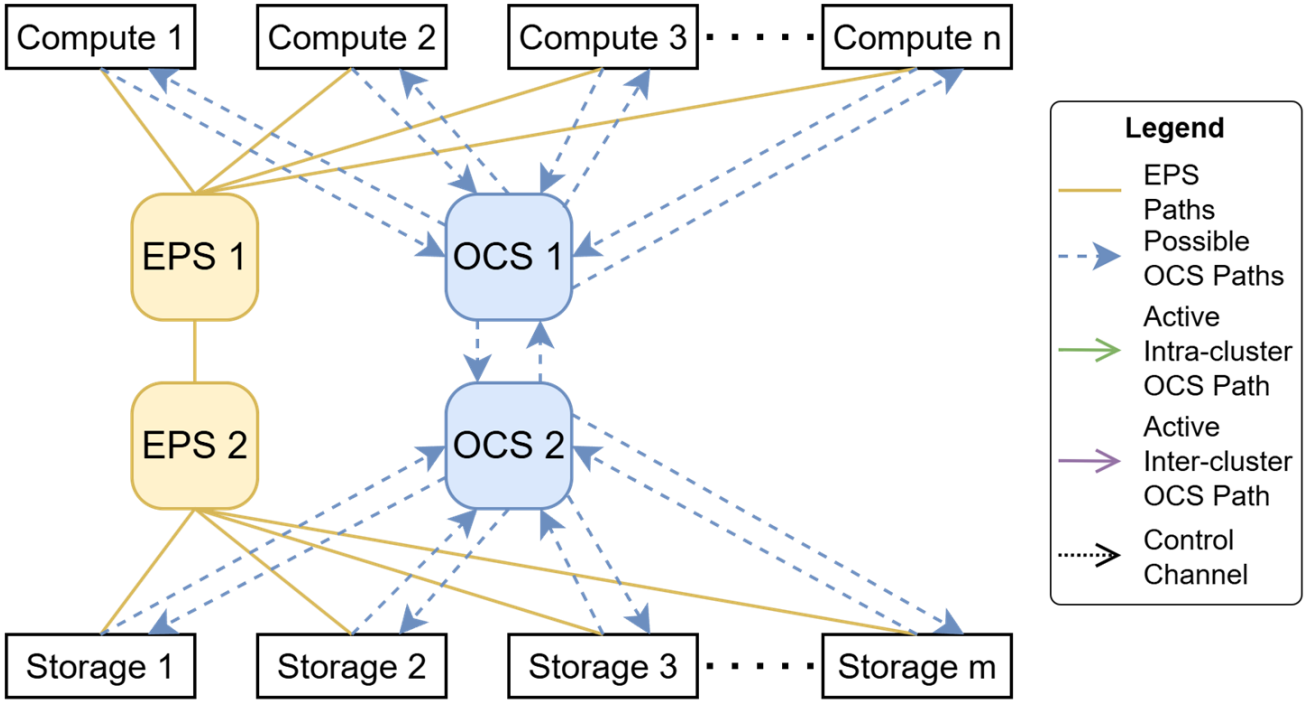
System Overview



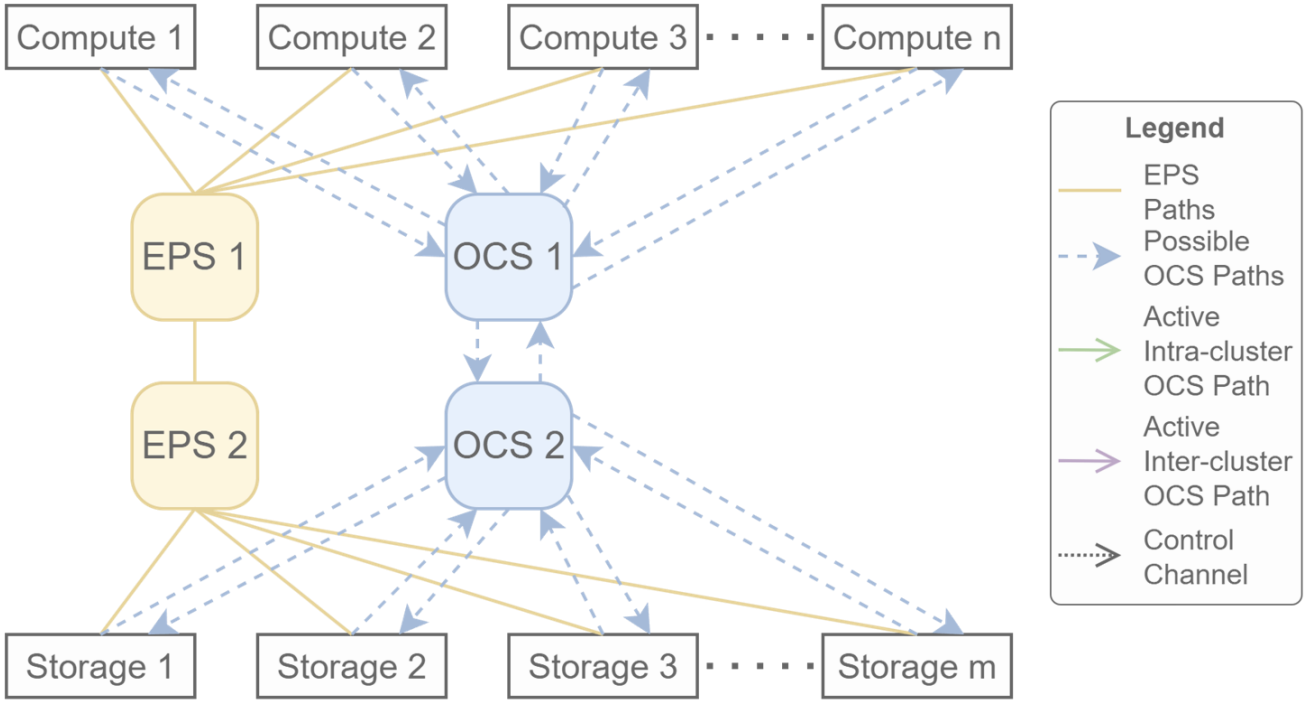
System Overview



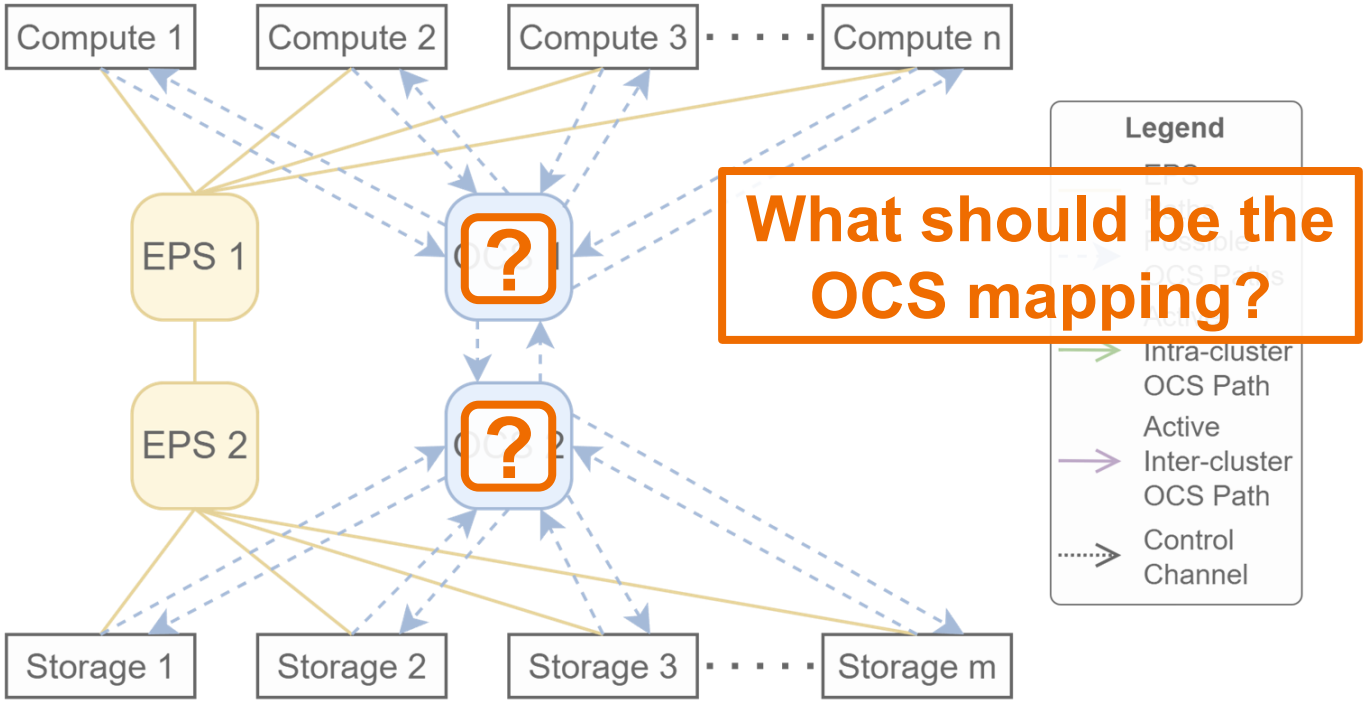
Problem Statement



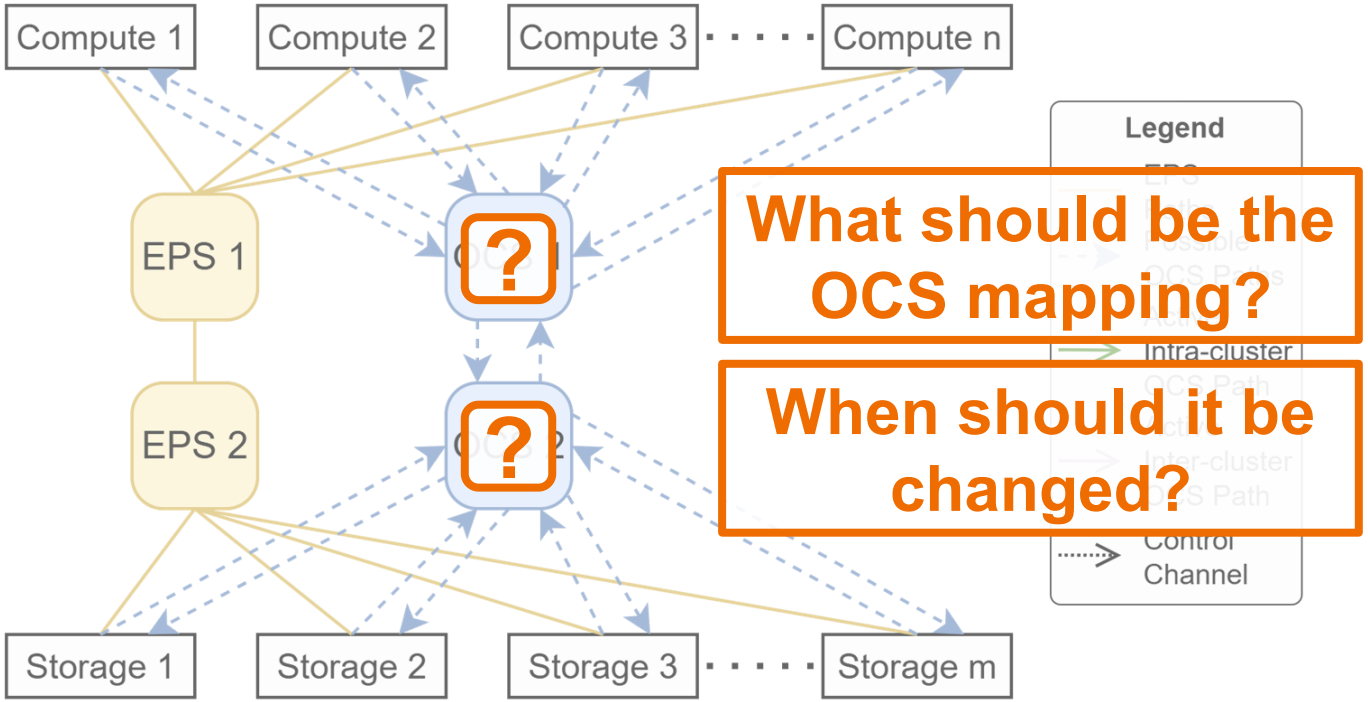
Problem Statement



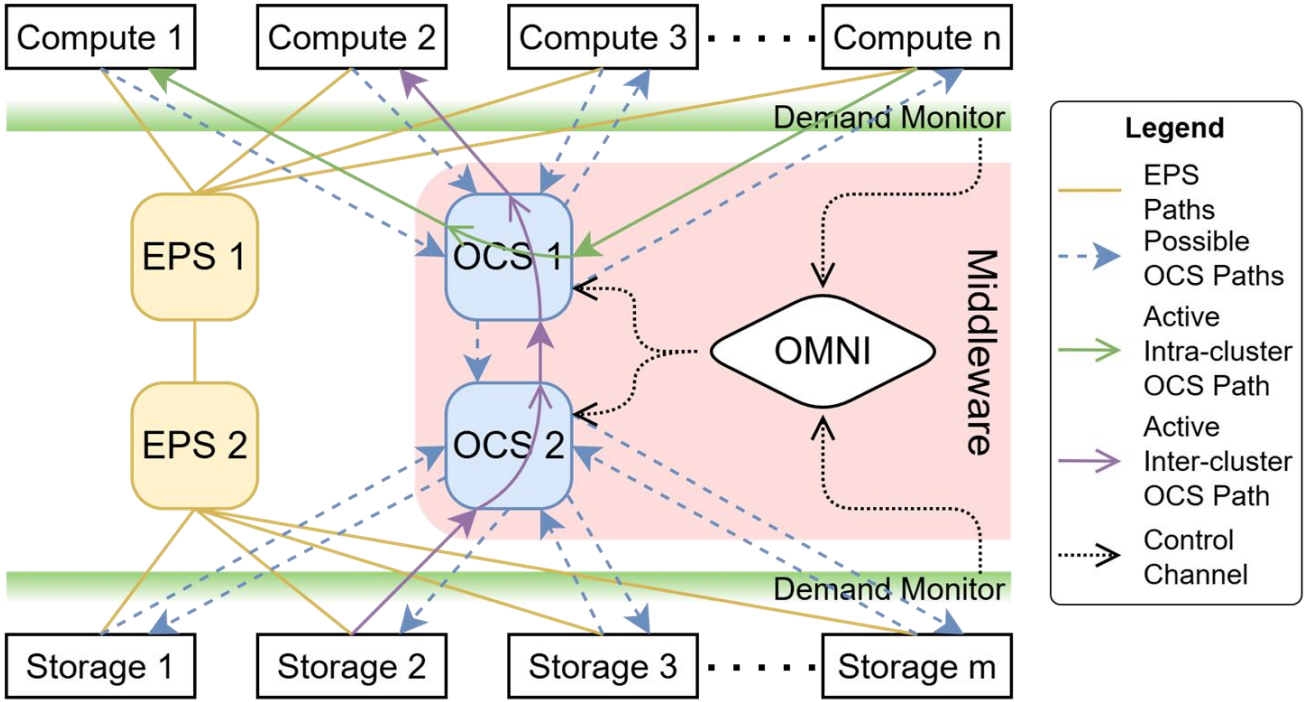
Problem Statement



Problem Statement

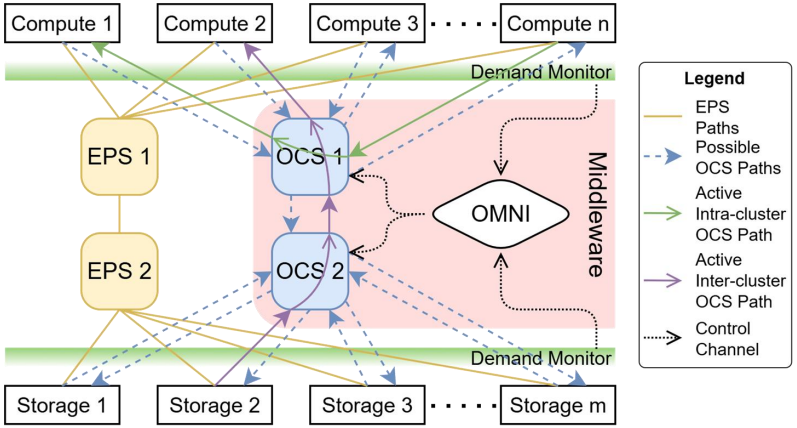


Introducing – OMNI



Introducing – OMNI

- Demand Monitors
  - On each node
  - Outgoing bandwidth per destination
  - Matrix of size $(m+n)\times(m+n)$
- Demand Matrix fed into OMNI middleware at intervals
- OMNI runs an algo to decide mapping
- OMNI reconfigures OCS switches



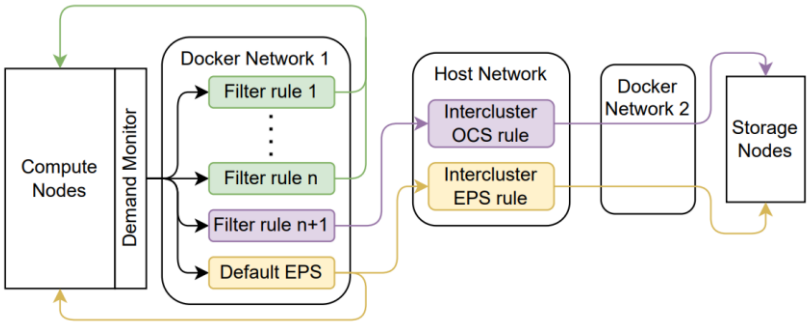
OMNI Algorithm

- Flatten DM into $\text{list}(\text{src}, \text{dst}, \text{demand}_{\text{src,dst}})$
- Sort by demand
  - Accelerated, Used – highest
  - Unaccelerated, used – high
  - Accelerated, unused – Zero
  - Unaccelerated, unused – Zero
- Accommodate in the order
  - If conflict, skip to next
  - Break if mapping filled

- Conflict detection
  - One-to-one mapping intra-cluster
  - Single link each direction inter-cluster
- Intra-cluster conflict
  - src in a tuple is already an src in accommodated links
  - Similar with dst
- Inter-cluster conflict
  - Direction already occupied by an accommodated links

Emulator and OMNI Implementation

- Nodes ran as docker containers
  - *cpuset* for resource isolation
- iptables for packet filtering and demand monitoring
- Linux Traffic Control for bandwidth control
- OMNI running asynchronously in Python on host
  - Can edit iptables filter rules



Experimental Methodology

- AWS EC2 c5n bare metal
  - 72 CPU cores, 192 GiB memory
- MapReduce
  - TPC-H – 22 Queries
  - Join, aggregate, shuffle etc.
  - 3 reruns
- DBGen Dataset size – 5 to 40 GB
- Bandwidth
  - EPS – 100 Mbps
  - OCS – 1 Gbps

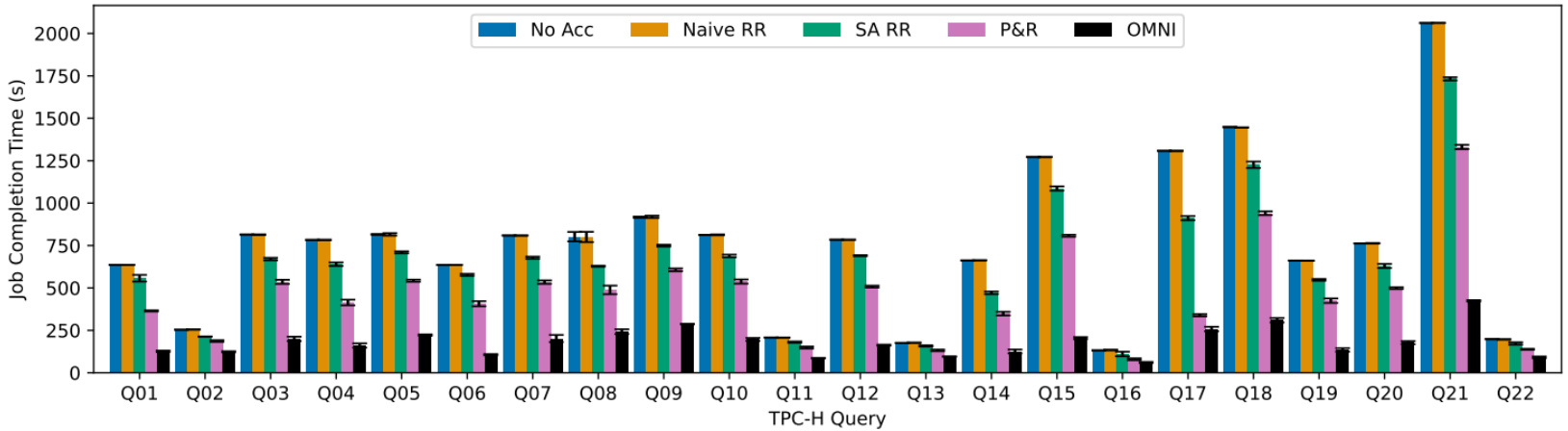
- Three cluster sizes

Config	#Compute Nodes	#Storage Nodes	#Cores per Node	Mem per Node
Small	5	2	8	20 GB
Default	10	5	4	10 GB
Large	20	10	2	5 GB

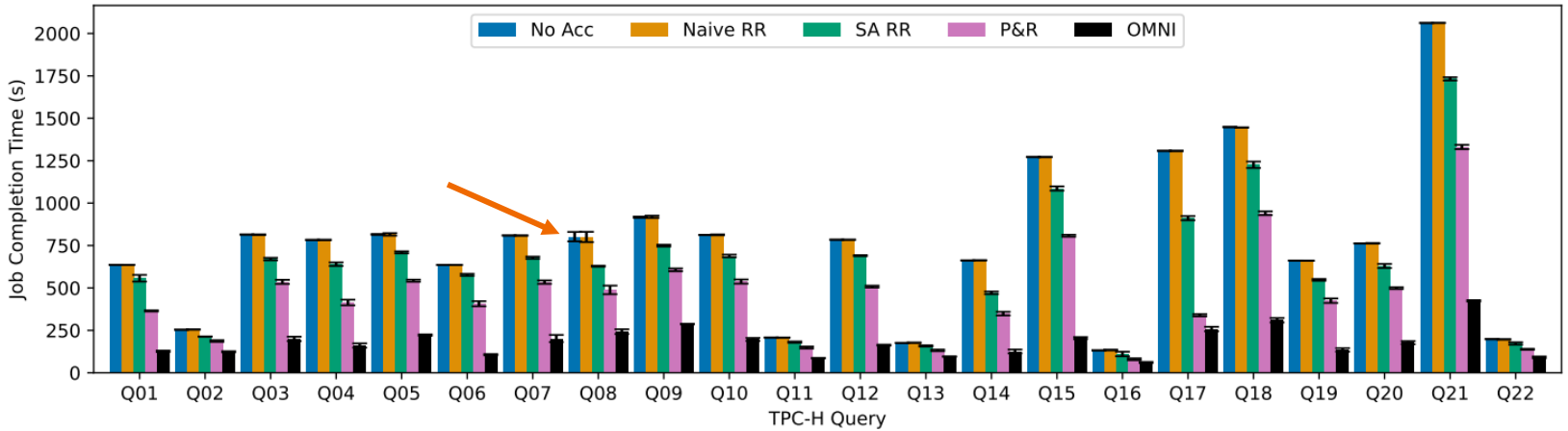
Experimental Methodology – Baseline Policies

- No acceleration (No Acc)
  - Use only EPS, no OCS paths
- Naïve Round Robin (Naïve RR)
  - Cycles through all OCS configs
  - Sequentially at every polling interval
- Stage Aware Round Robin (SA RR)
  - Access to MapReduce stage
  - Round Robin on subset that's best for the stage
  - E.g., Map stage – HDFS -> Spark
- Propose and Reject (P&R)
  - Based on prior work (Figiel *et al.*)
  - src nodes proposes to dst nodes
    - In order of demand
  - dst nodes accept 1 request
    - highest demand
    - Reject the rest
  - Rejected src node tries next most demanding dst…. So on
  - Designed for decentralized setup

Evaluation – Job Completion Time

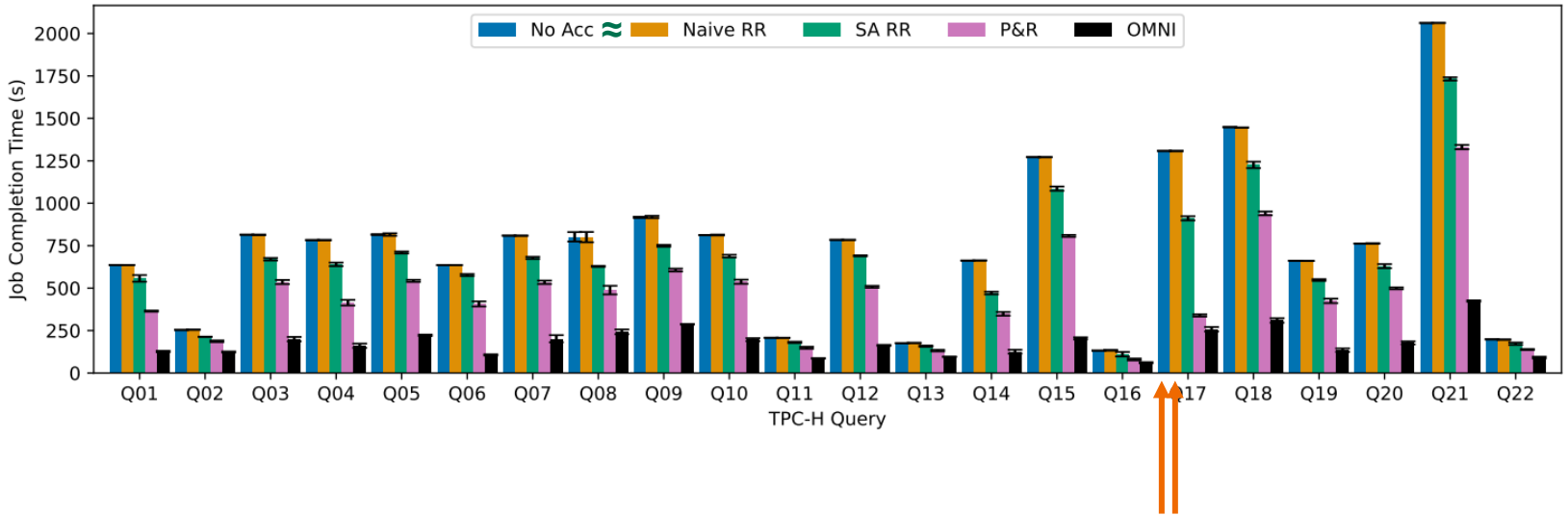


Evaluation – Job Completion Time



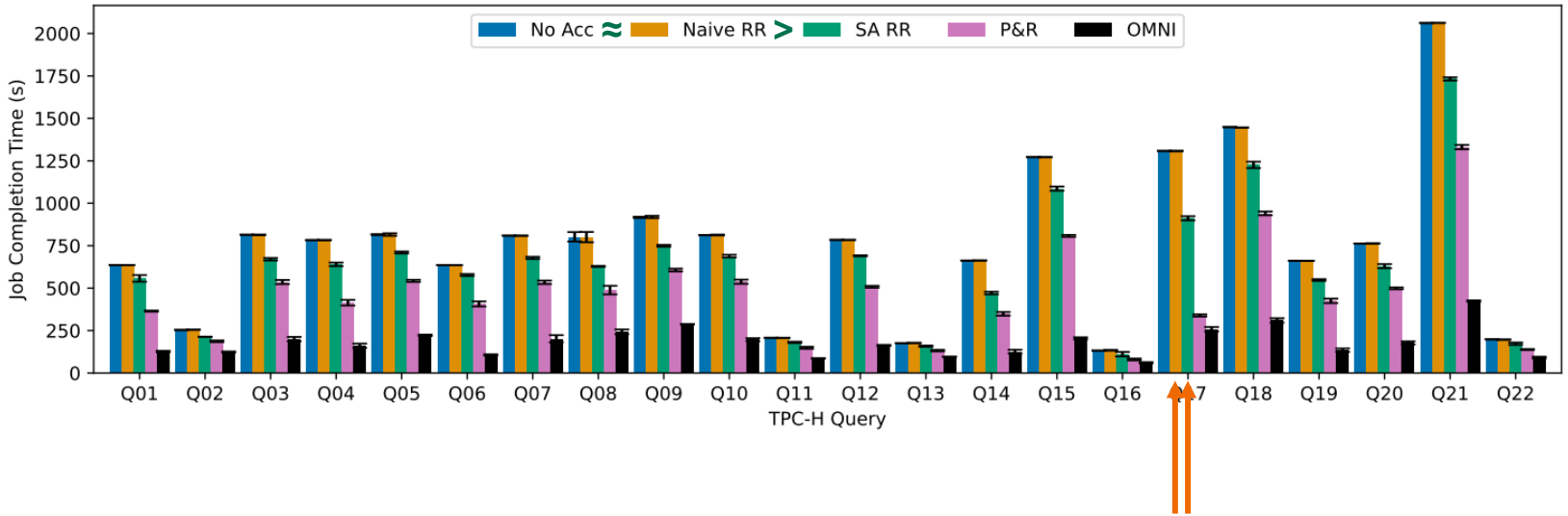
Evaluation – Job Completion Time

Naïve RR ≈ No Acc

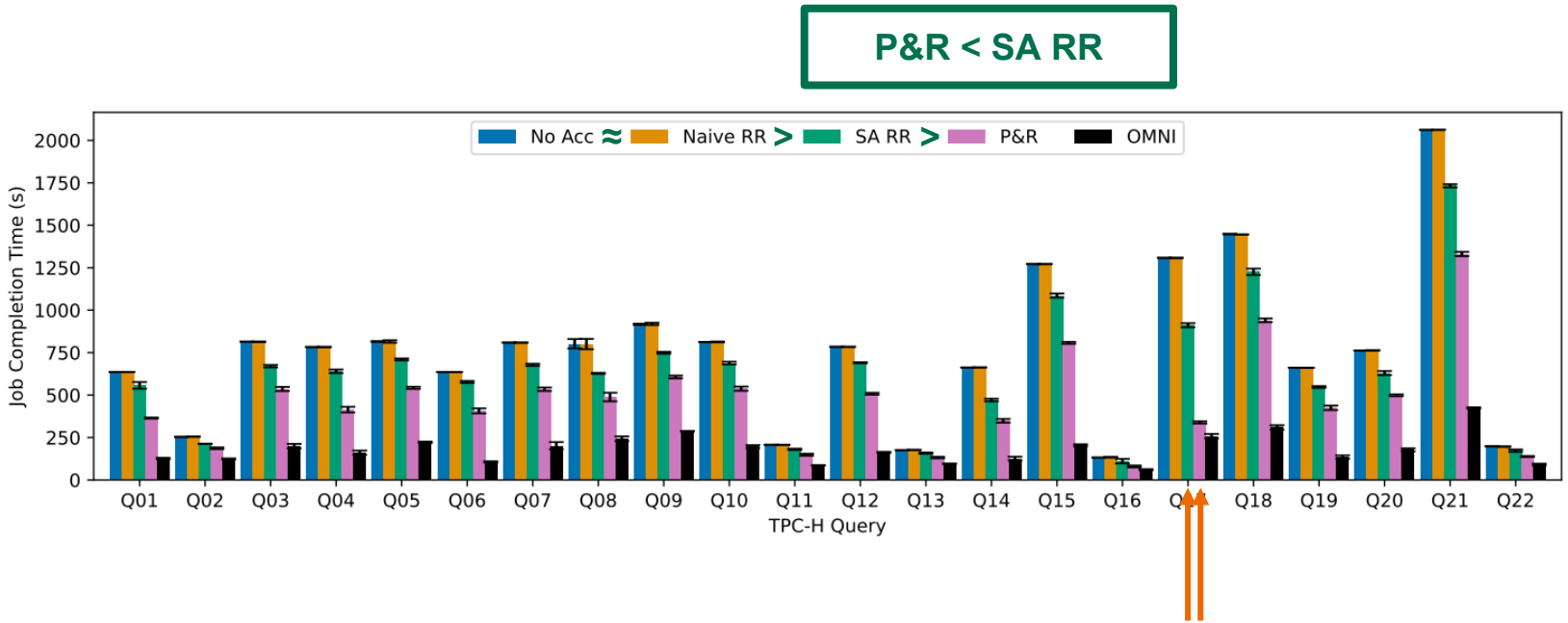


Evaluation – Job Completion Time

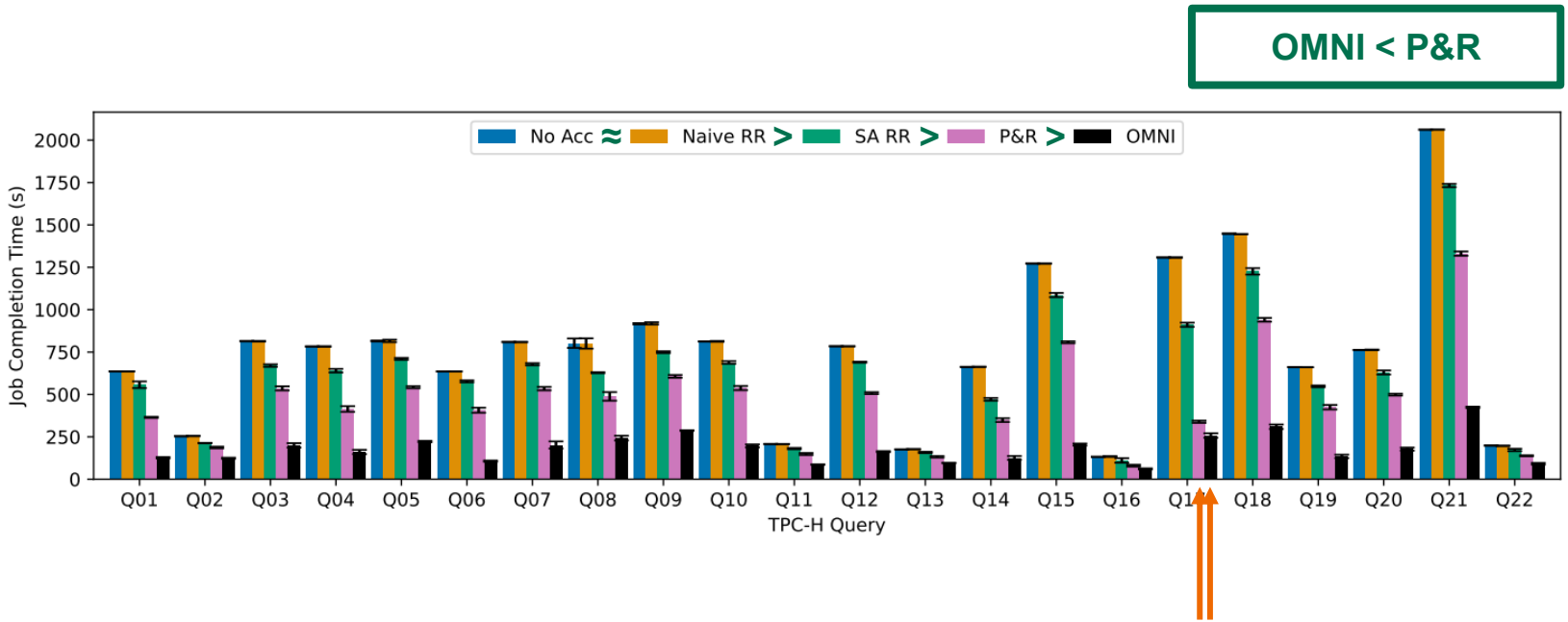
SA RR < Naïve RR



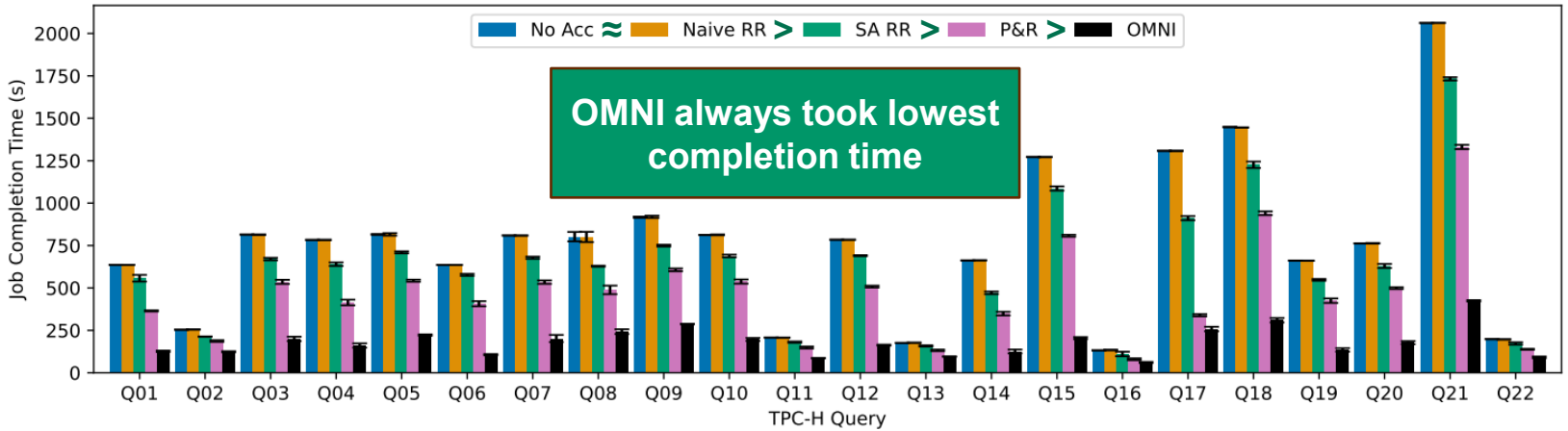
Evaluation – Job Completion Time



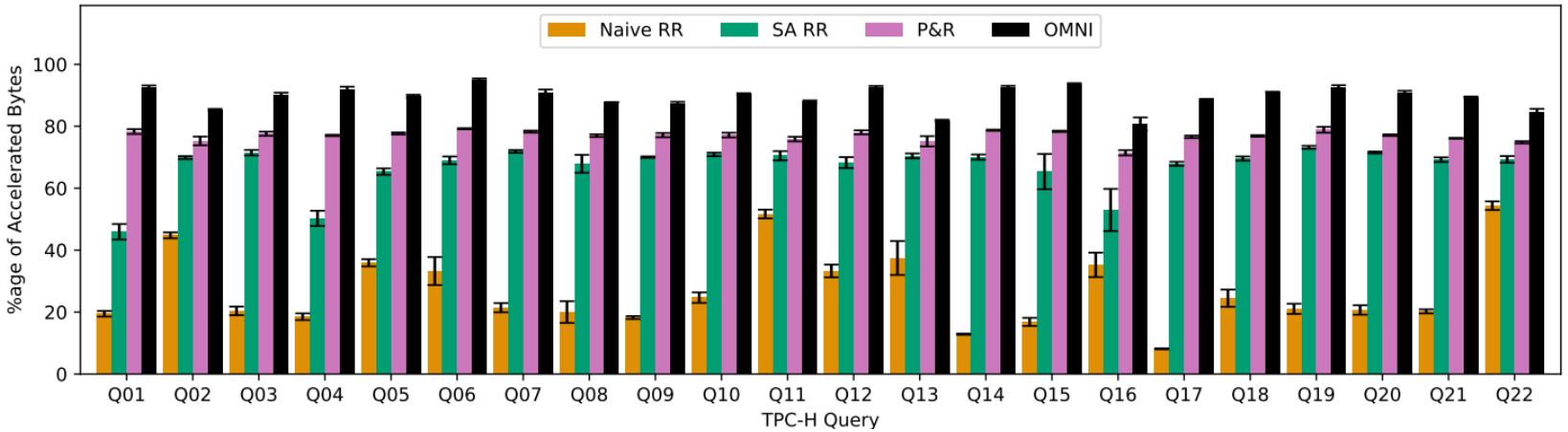
Evaluation – Job Completion Time



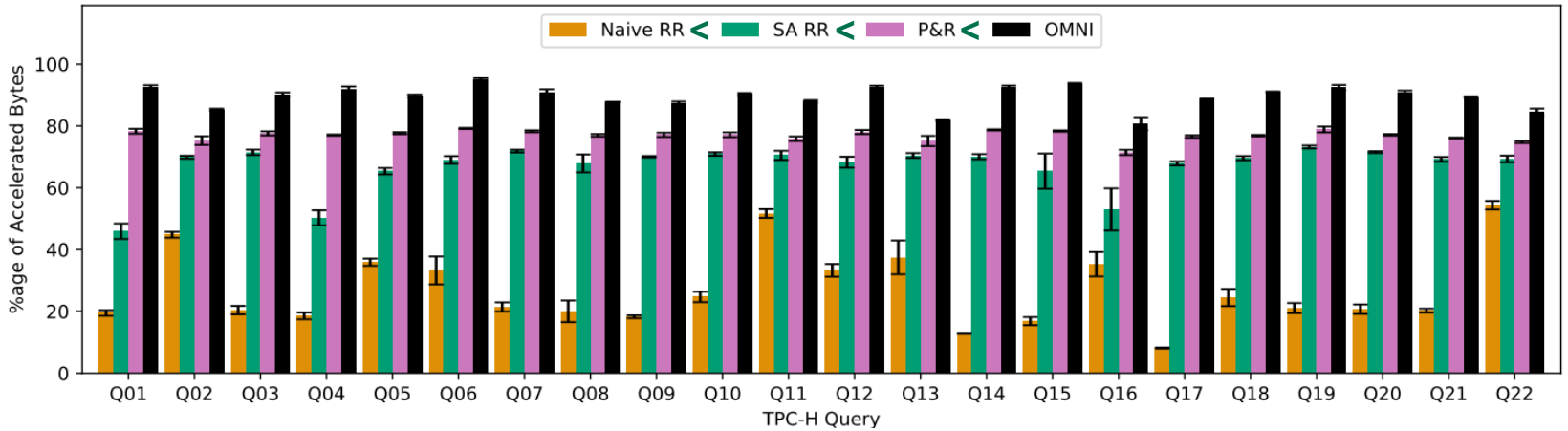
Evaluation – Job Completion Time



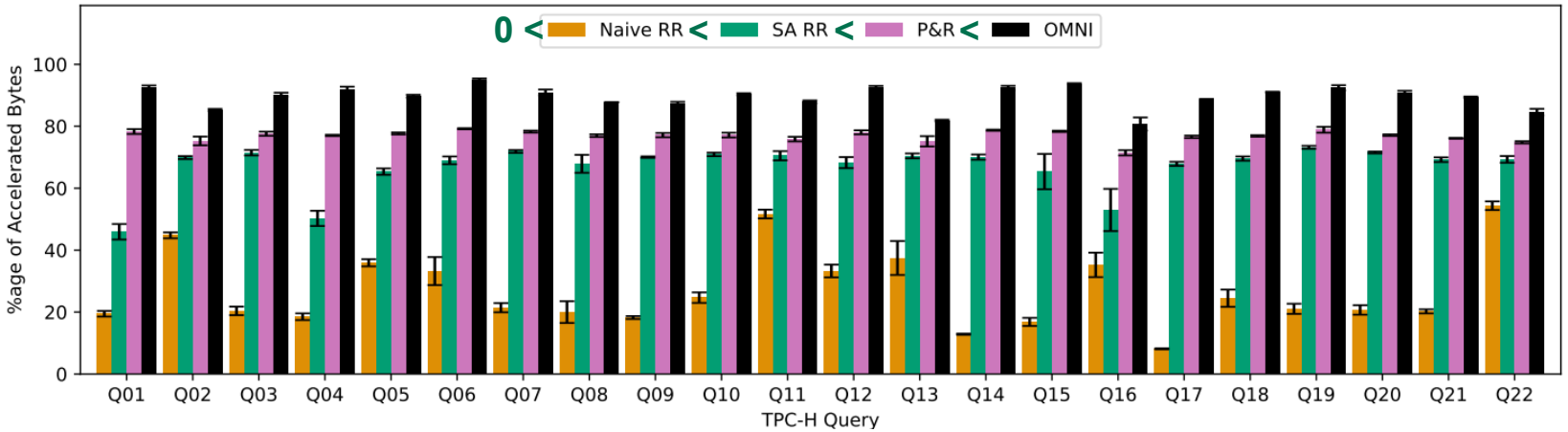
Evaluation – OCS Utilization



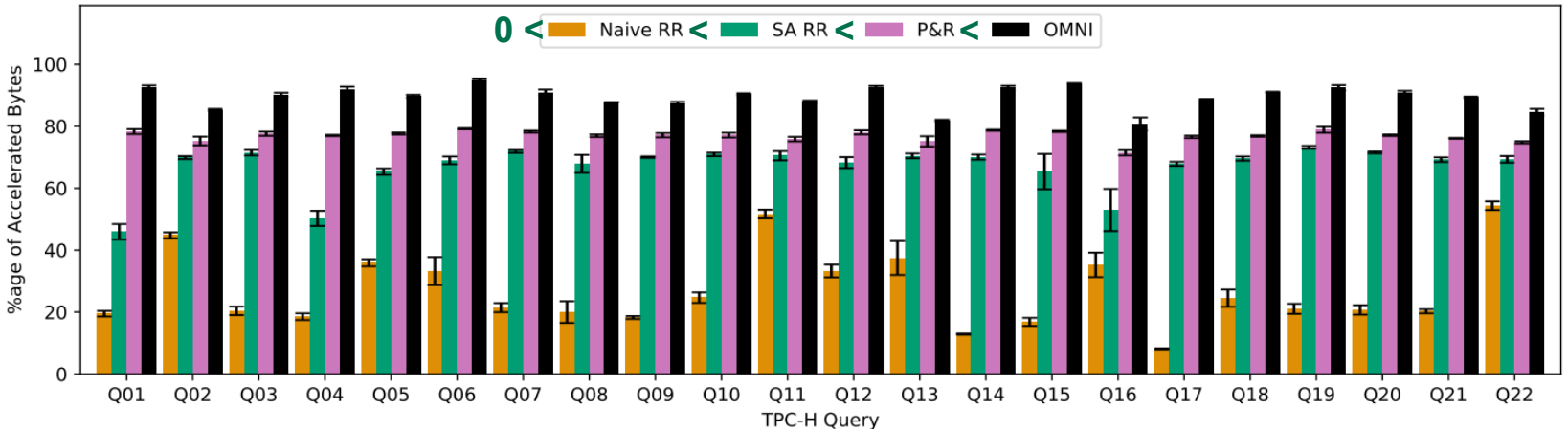
Evaluation – OCS Utilization



Evaluation – OCS Utilization

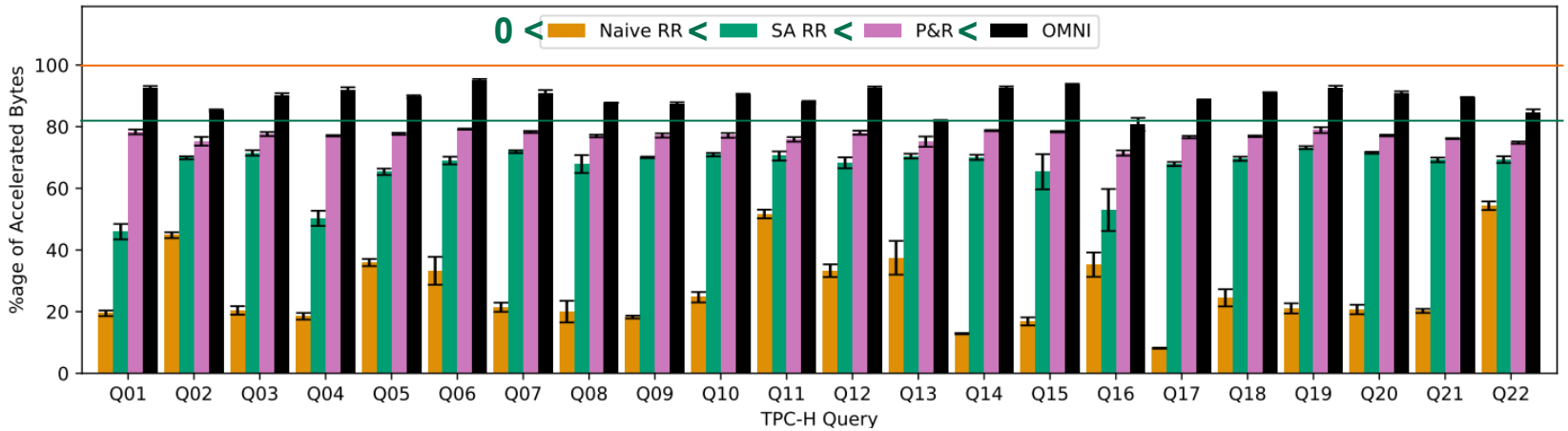


Evaluation – OCS Utilization



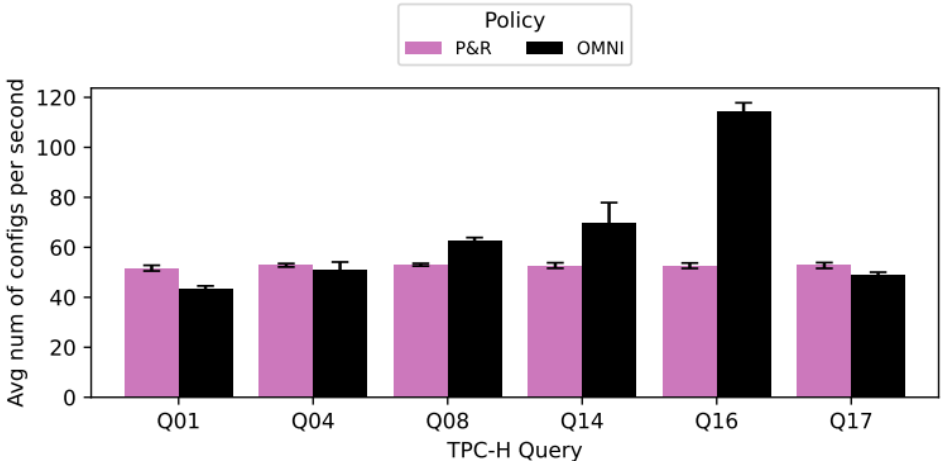
OMNI utilized OCS most
Mean 89%

Evaluation – OCS Utilization

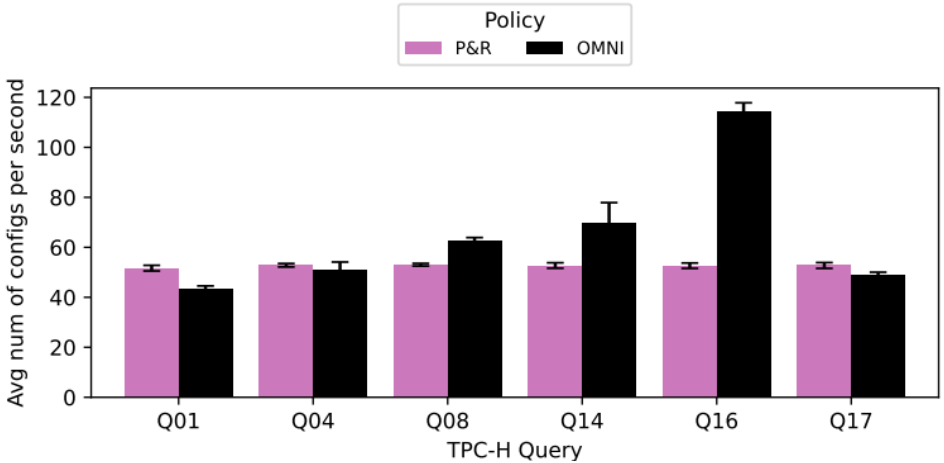


OMNI utilized OCS most
Mean 89%

Evaluation – Reconfig Overhead

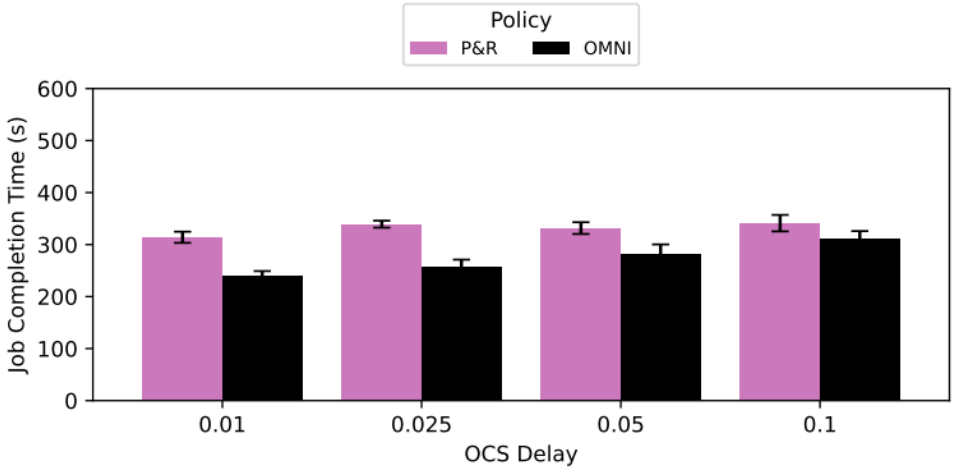


Evaluation – Reconfig Overhead

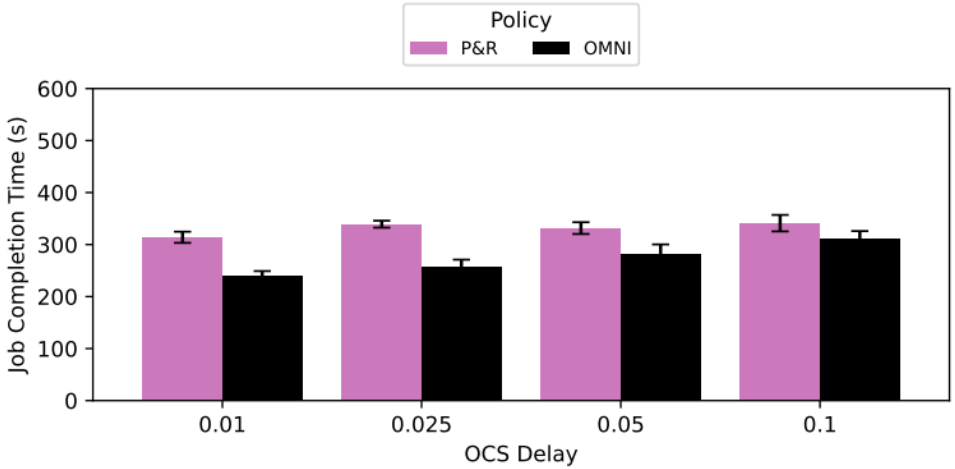


OMNI reconfigures more often

Evaluation

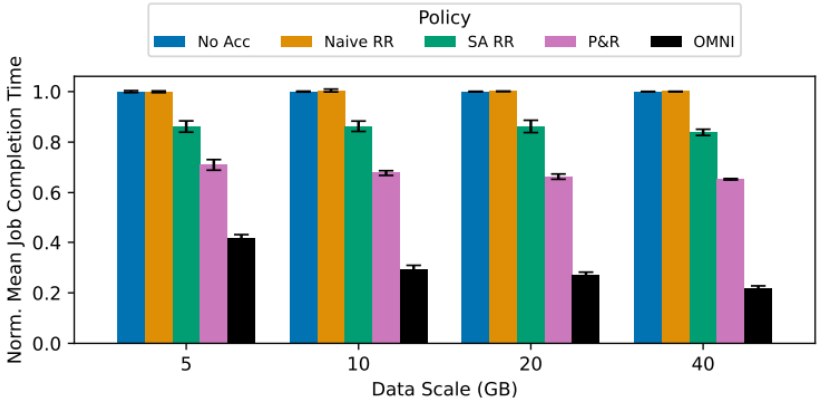

Policy
P&R
OMNI
Job Completion Time (s)
600
500
400
300
200
100
0
0.01
0.025
0.05
0.1
OCS Delay

Evaluation

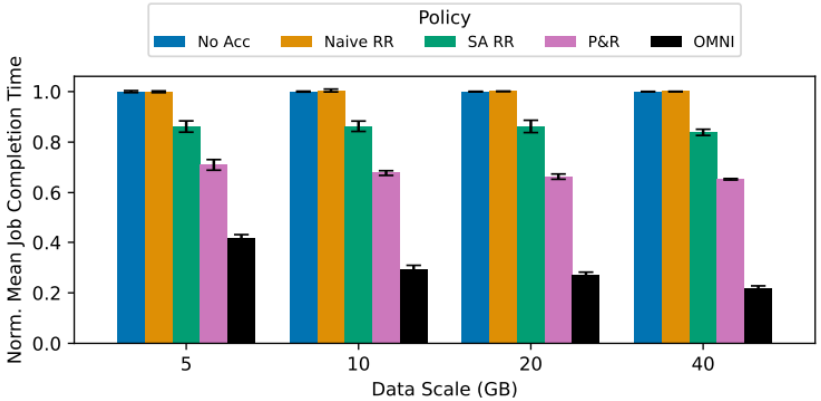


OMNI approaches P&R

Evaluation – Dataset Sizes and Cluster sizes

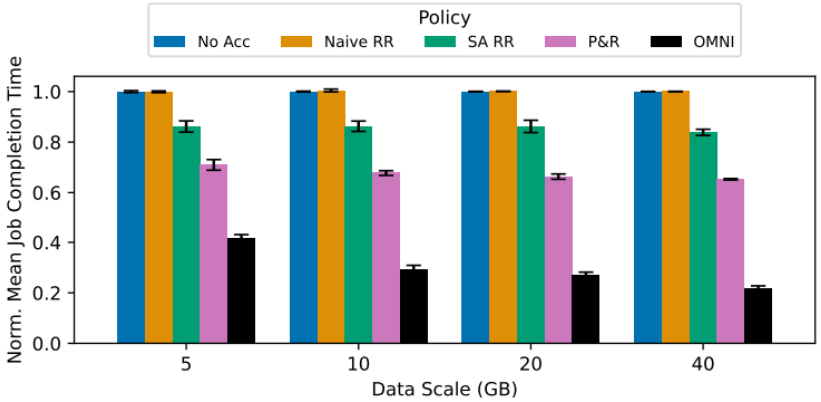


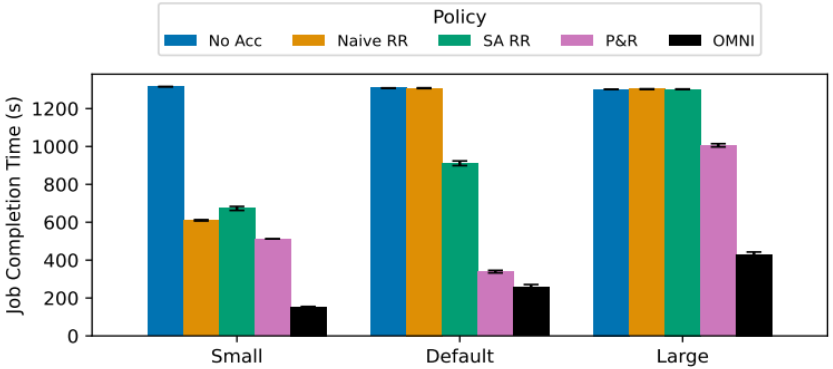
Evaluation – Dataset Sizes and Cluster sizes



OMNI always took lowest completion time

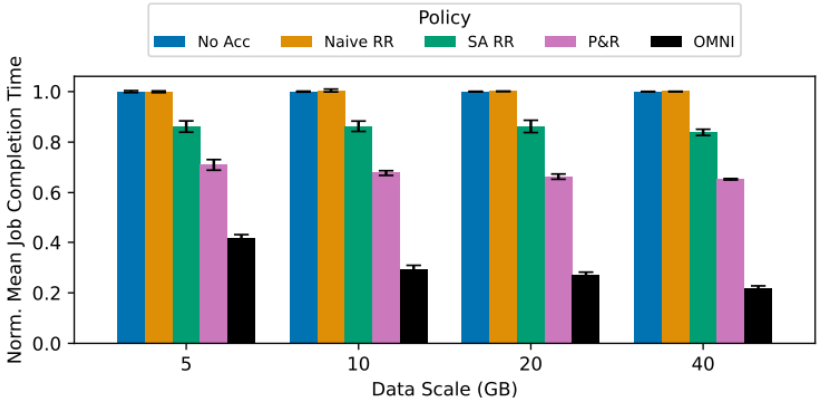
Evaluation – Dataset Sizes and Cluster sizes

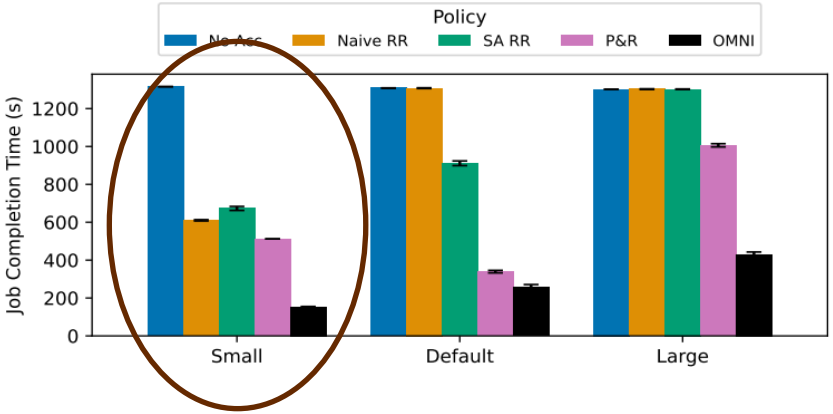




OMNI always took lowest completion time

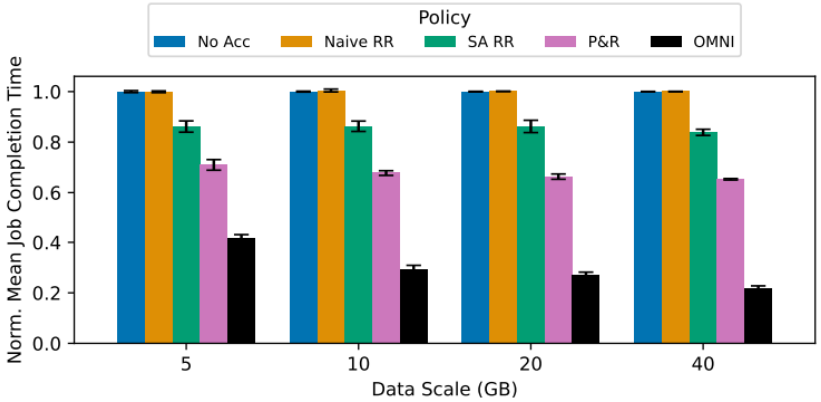
Evaluation – Dataset Sizes and Cluster sizes

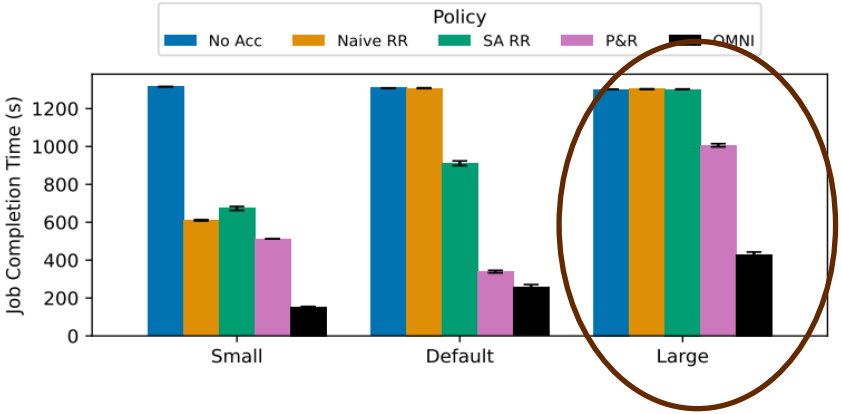




OMNI always took lowest completion time

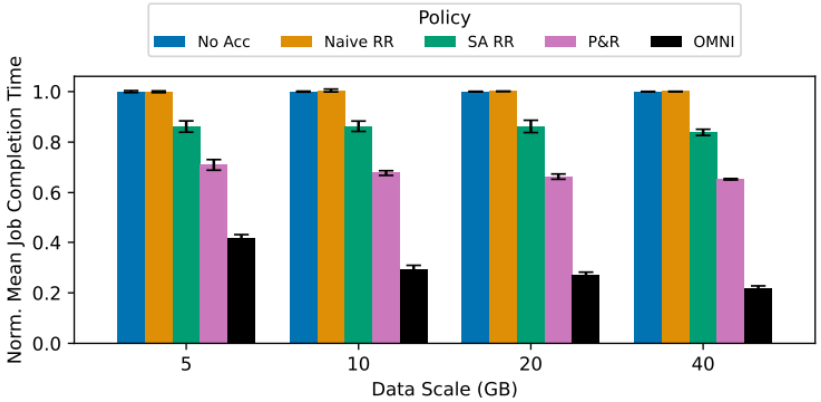
Evaluation – Dataset Sizes and Cluster sizes

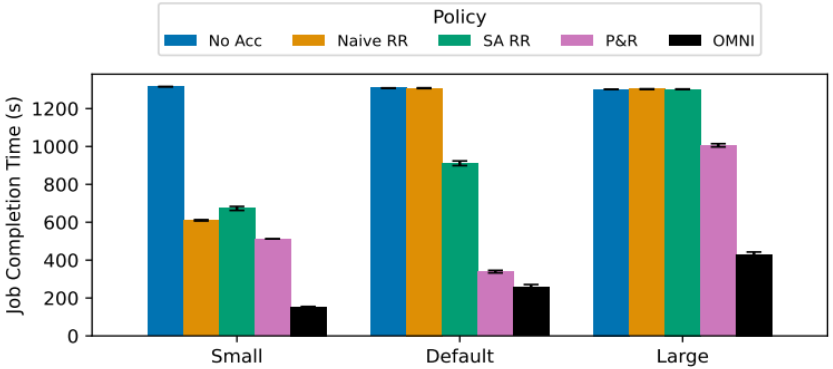




OMNI always took lowest completion time

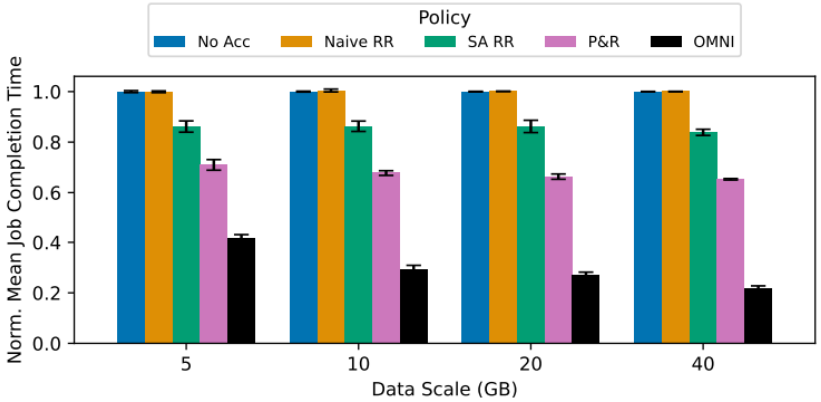
Evaluation – Dataset Sizes and Cluster sizes

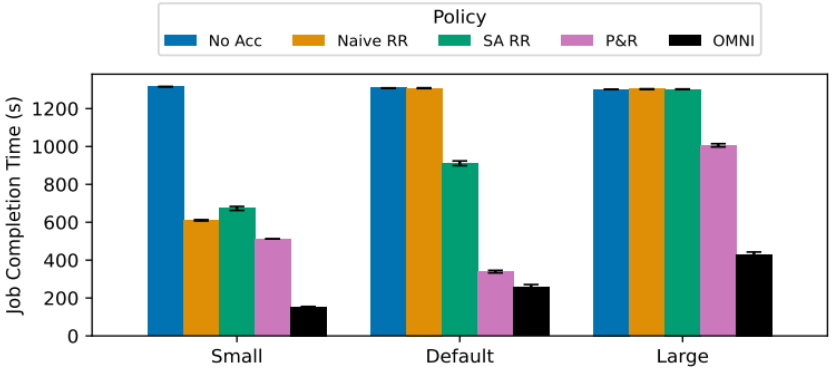




OMNI always took lowest completion time

Evaluation – Dataset Sizes and Cluster sizes





OMNI always took lowest completion time

OMNI (P&R too) maintained positions

Conclusion

- **Dynamic & bursty workloads** need **adaptive demand-aware** solutions.
- **OMNI** – **middleware** framework for managing **Hybrid OCS-EPS** networks.
  - **Greedy matching algo** that optimize both **intra-** and **inter-**cluster optical paths.

- **Job completion time reduction**
  - **as much as 72%** compared to baseline **non-accelerated** setups.
  - **average of 55%** compared to **existing** decentralized demand-aware **P&R**.
- **High OCS utilization**, **average of 89%** of workload data.

- **Wide applicability and scalability** across diverse TPC-H **queries**, variable **cluster sizes**, and growing **dataset scales**.