
Moirai: Optimizing Placement of Data and Compute in Hybrid Clouds

Ziyue Qiu 

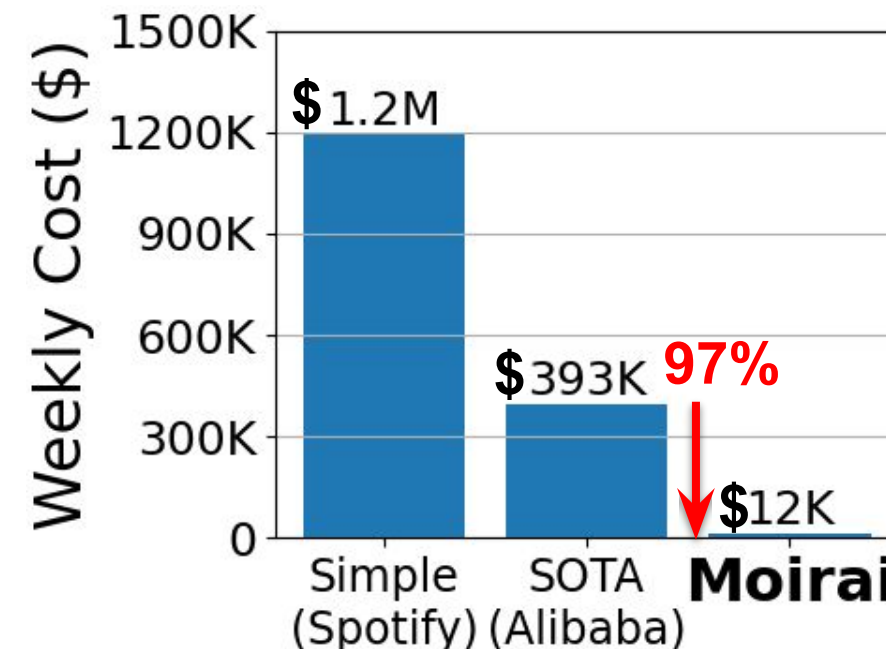
Hojin Park  Jing Zhao  Yu-Kai Wang  Arnav Balyan 
Gurmeet Singh  Yangjun Zhang  Suqiang (Jack) Song 
Gregory R. Ganger  George Amvrosiadis 

 **Carnegie Mellon University**
Parallel Data Laboratory



Optimizing Cost-efficiency in Hybrid Clouds

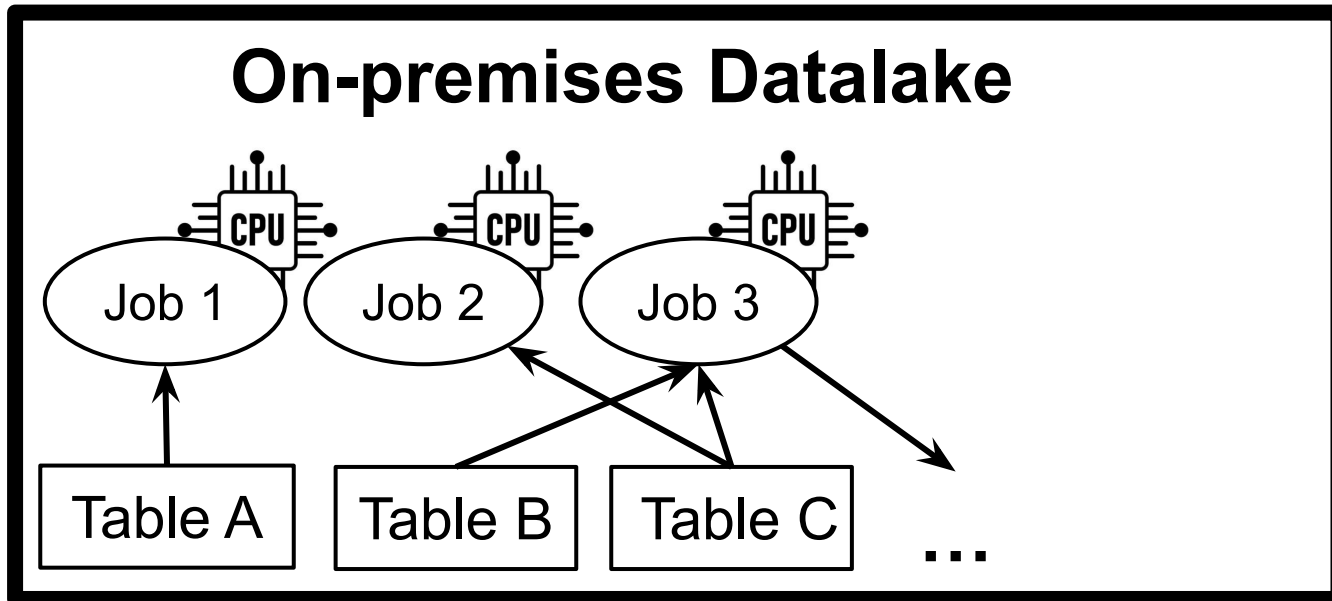
- Problem: Cross-site data movement can be costly
 - More than millions of dollars per week
- Our solution: ***Moirai*** saves **97%** hybrid cloud costs
 - A framework to place data and jobs across sites
 - Find and use job-data dependencies
 - Fine-grained joint job+data optimization
 - Evaluated with 4-month Uber trace
 - 97% lower cost than state-of-the-art



Suboptimal Placement Leads to Inefficiencies

Before cloud adoption

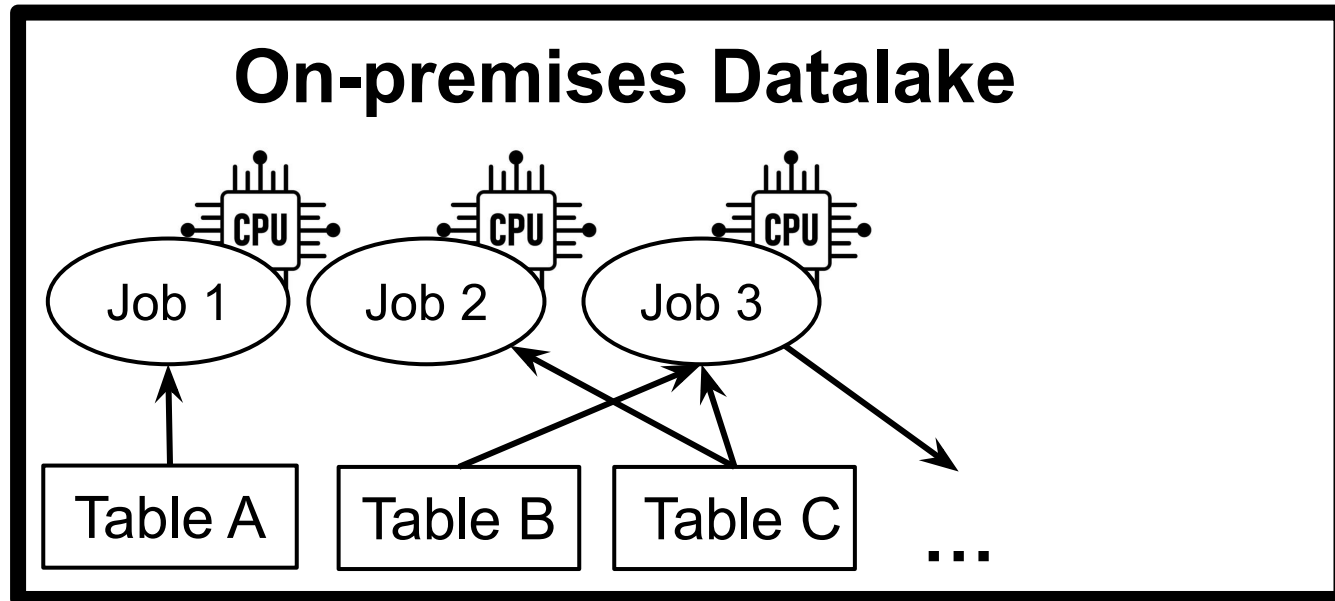
Uber



- Over 1M tables, 300PB stored
- 4M jobs/week
- Reads and writes **900PB/week**

Suboptimal Placement Leads to Inefficiencies

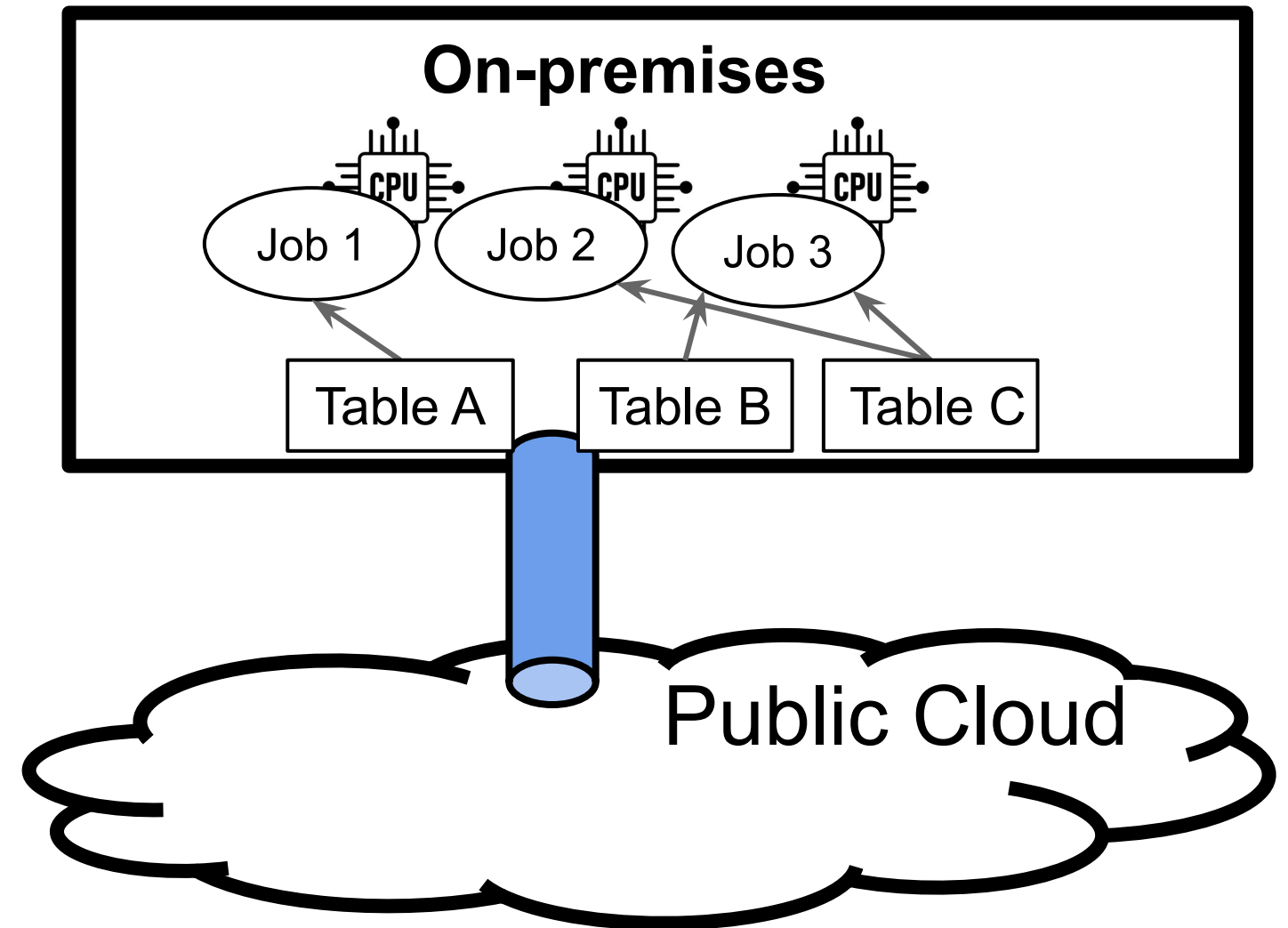
Before cloud adoption



- Over 1M tables, 300PB stored
- 4M jobs/week
- Reads and writes **900PB/week**

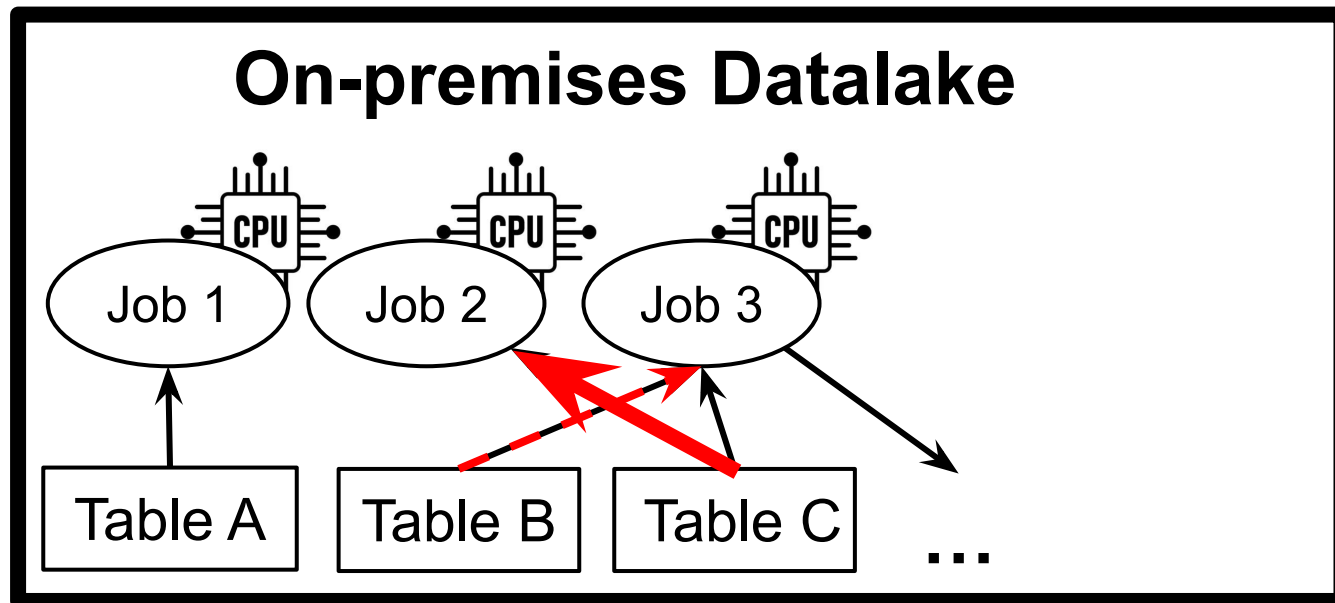
Uber

Hybrid Cloud Datalake



Suboptimal Placement Leads to Inefficiencies

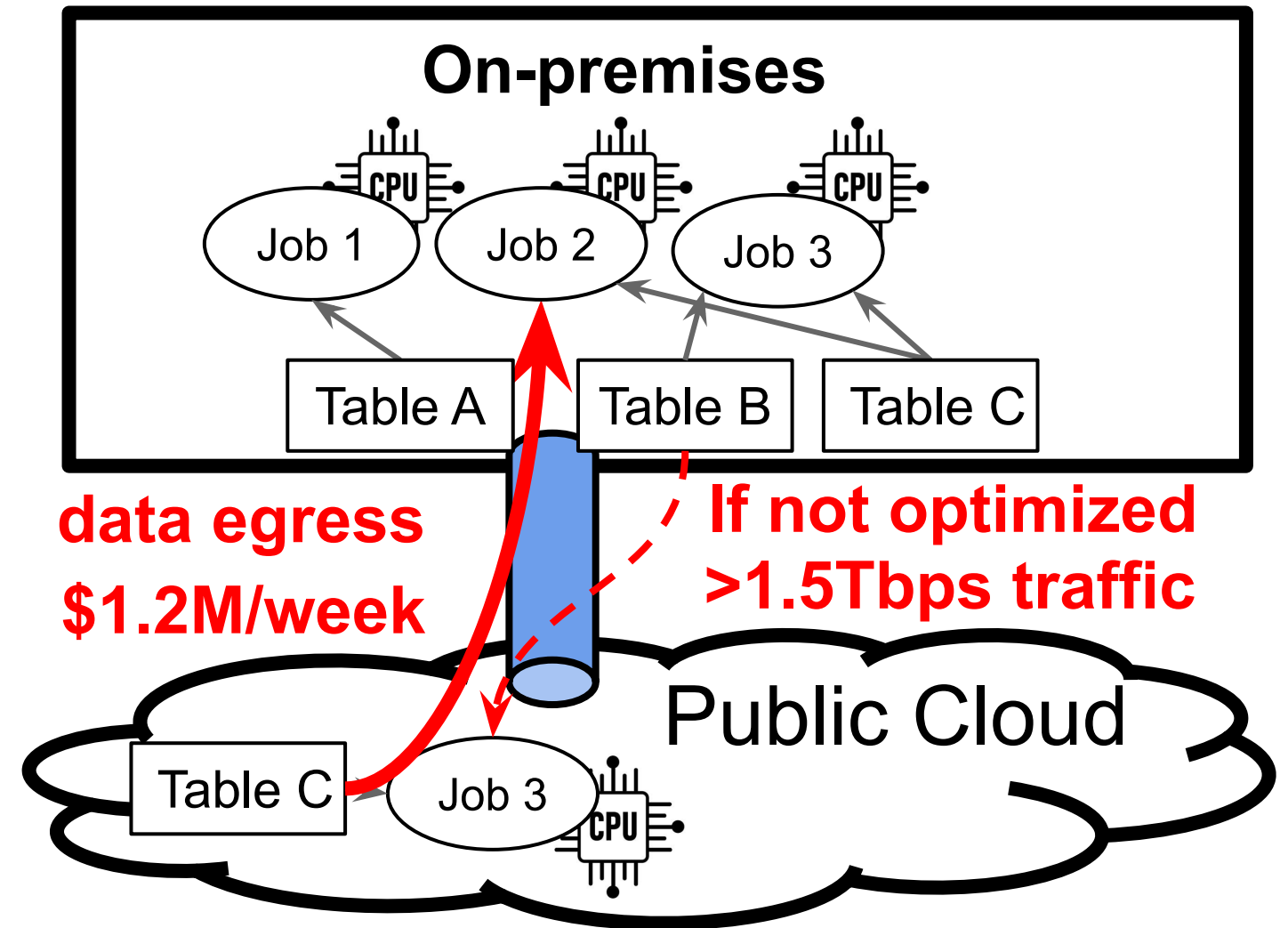
Before cloud adoption



- Over 1M tables, 300PB stored
- 4M jobs/week
- Reads and writes **900PB/week**

Uber

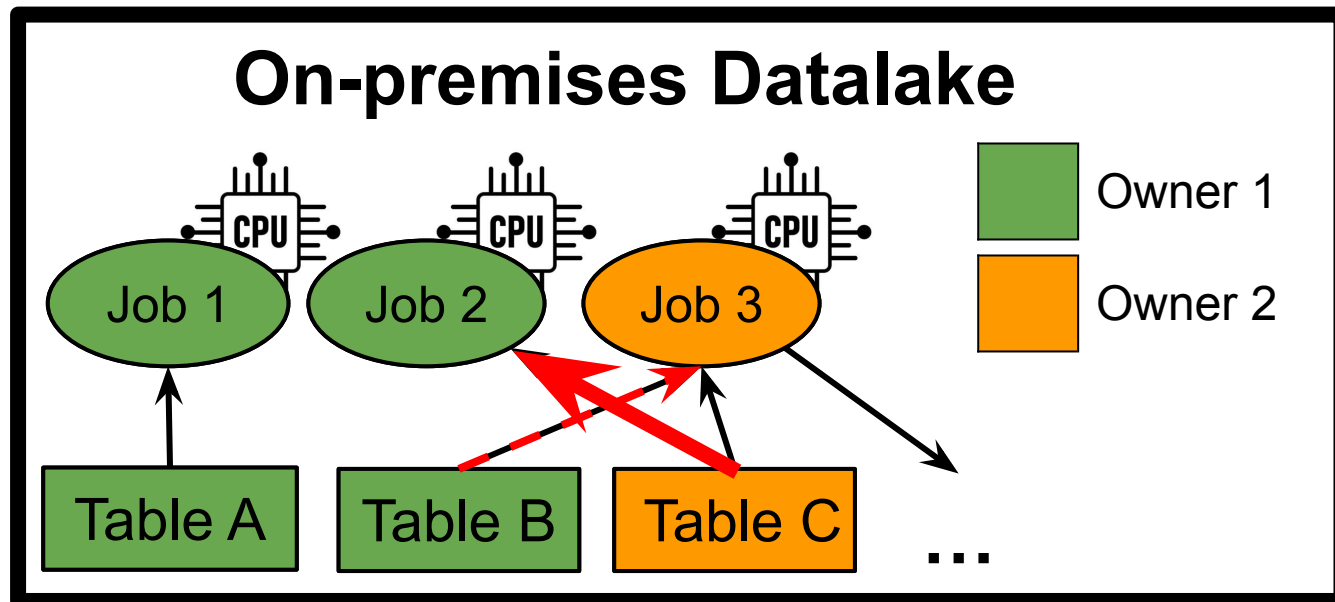
Hybrid Cloud Datalake



Placement should consider the scale and egress costs.

Placement Is Challenging at Large Scale

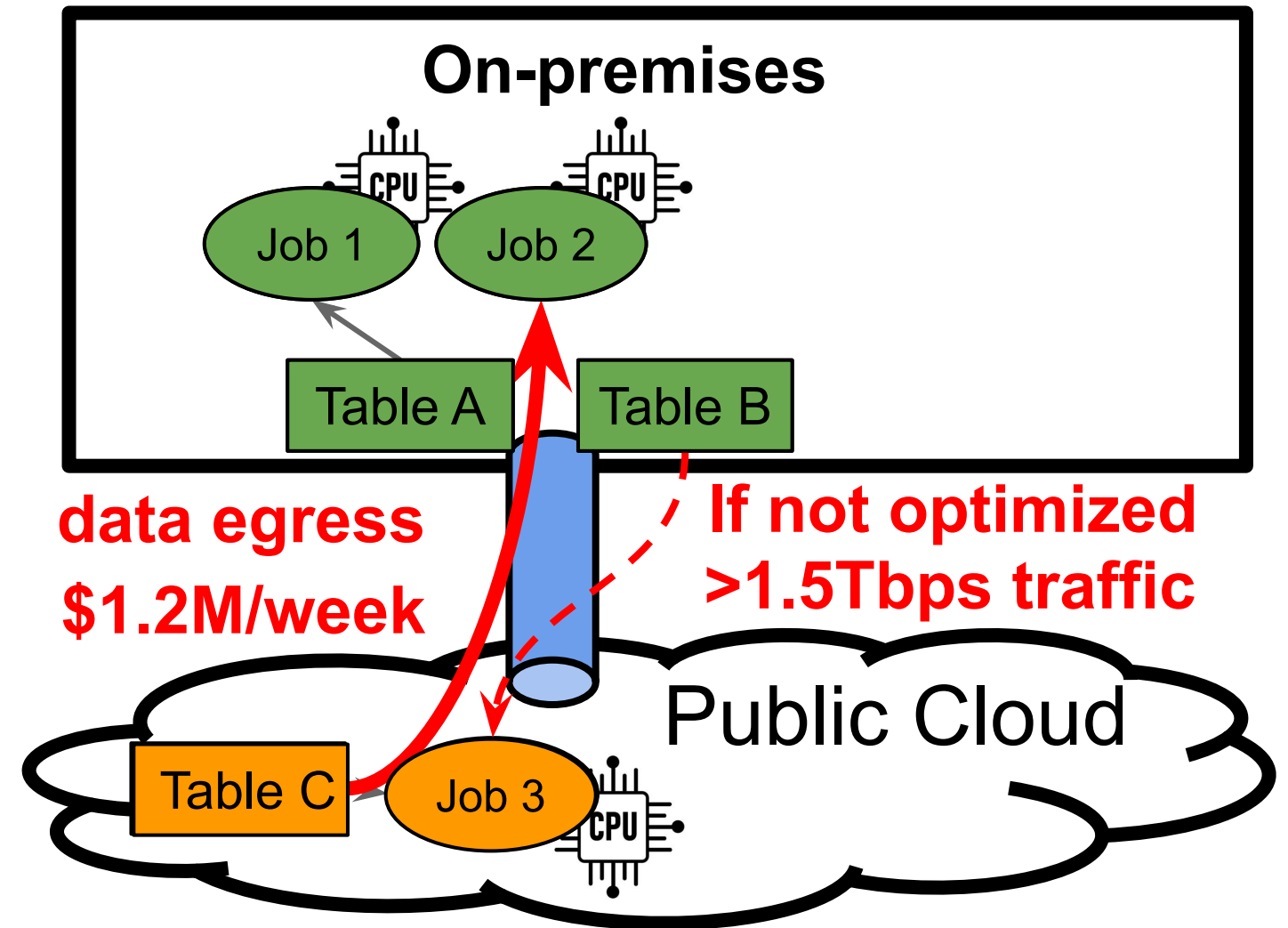
Before cloud adoption



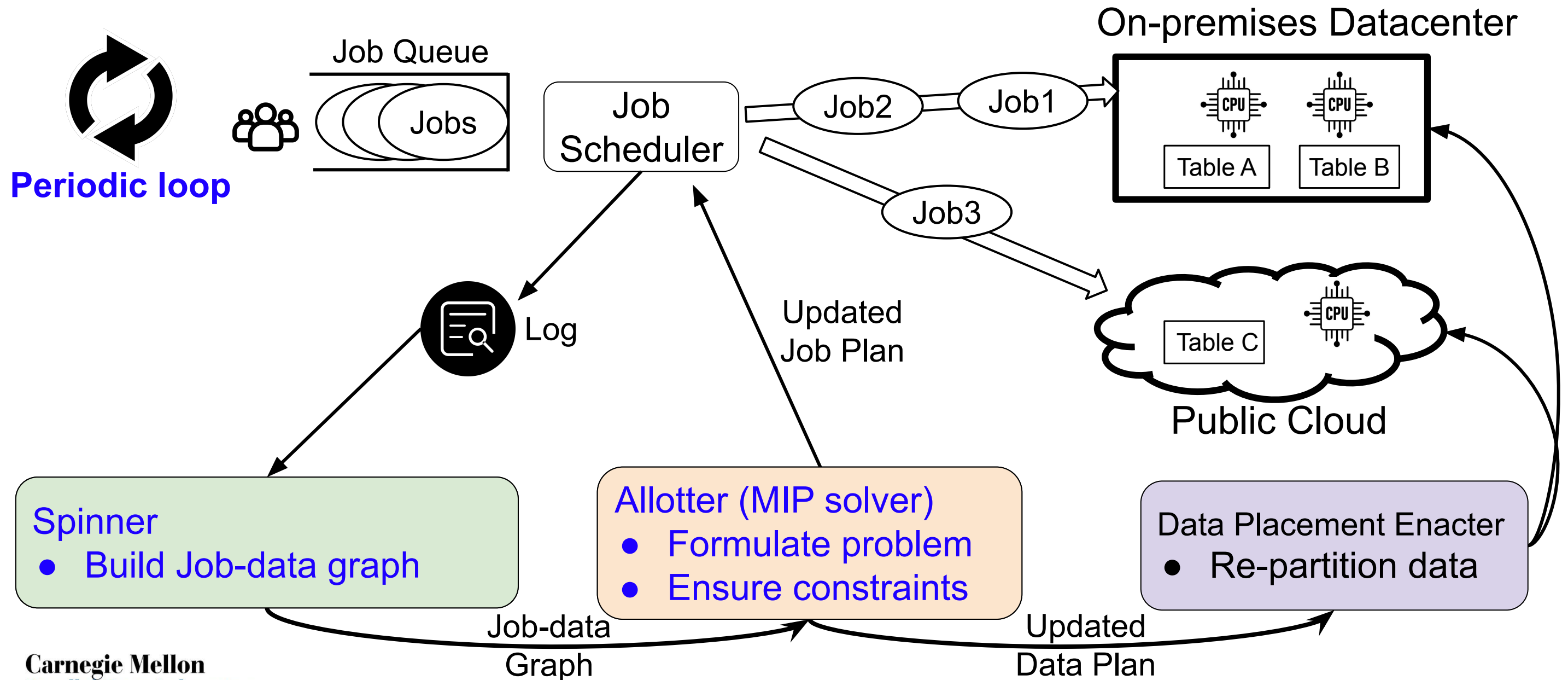
- Problem: one giant component
- Prior work: group by owner
- Challenge:
 - 10% reads are owner-local

Uber

Hybrid Cloud Datalake

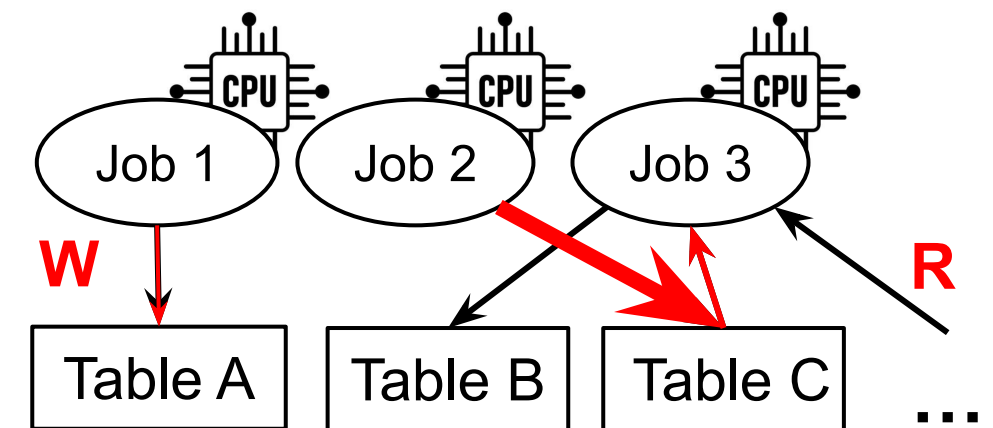


Moirai Architecture



Spinner: Dependencies Drive Placement

- Build the job–data graph from logs
 - Who reads/writes what
 - Table→Job = Read (R); Job→Table = Write (W)
- Weight edges by bytes
- Use the graph to drive placement
 - Co-locate Job 2 ↔ Table C

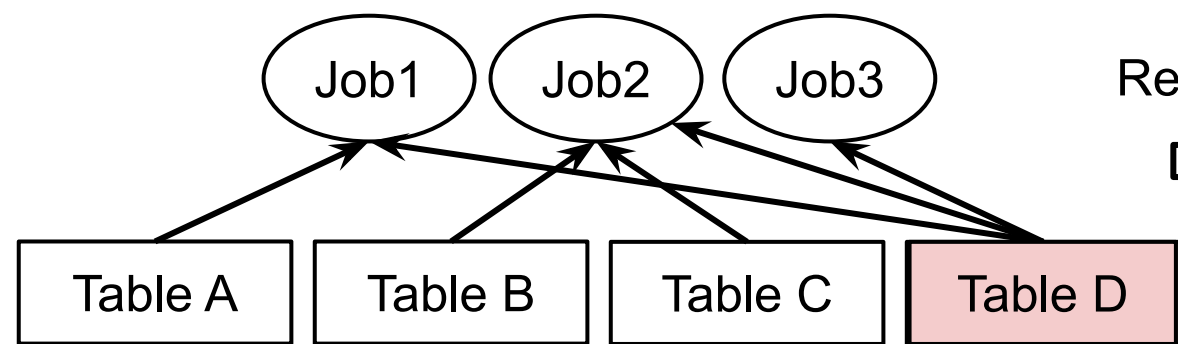


Allotter: “Hub” Pre-replication for Scale

- Solving for MIP placement at Uber scale: >7 days
- A few “hub” tables account for most dependencies
- Pre-replicating hub tables eliminates dependencies
 - -59% edges by replicating ~0.2% data, solved in 2hrs

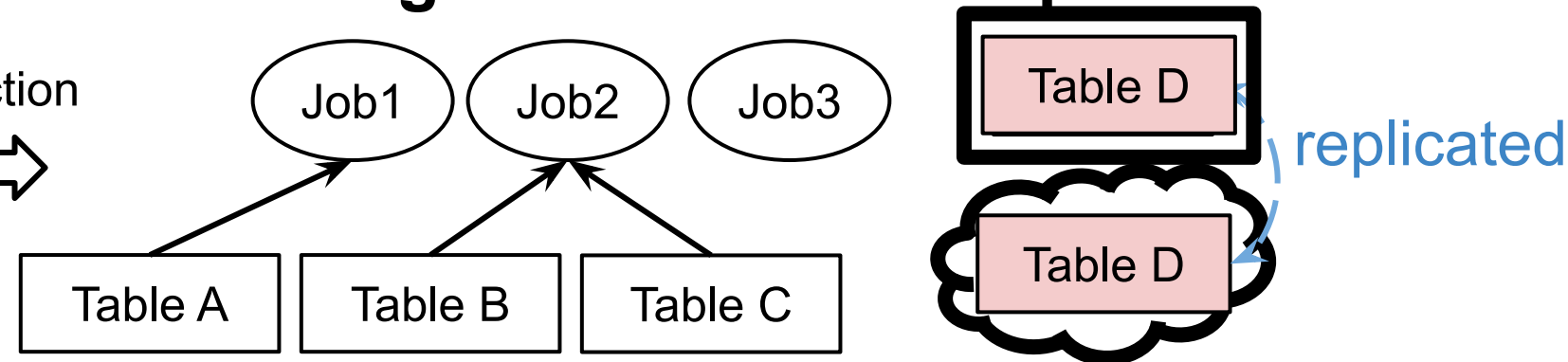
84X faster

Before: many edges, no clean split



Hub table

After: edges removed via replication





Experimental Setup

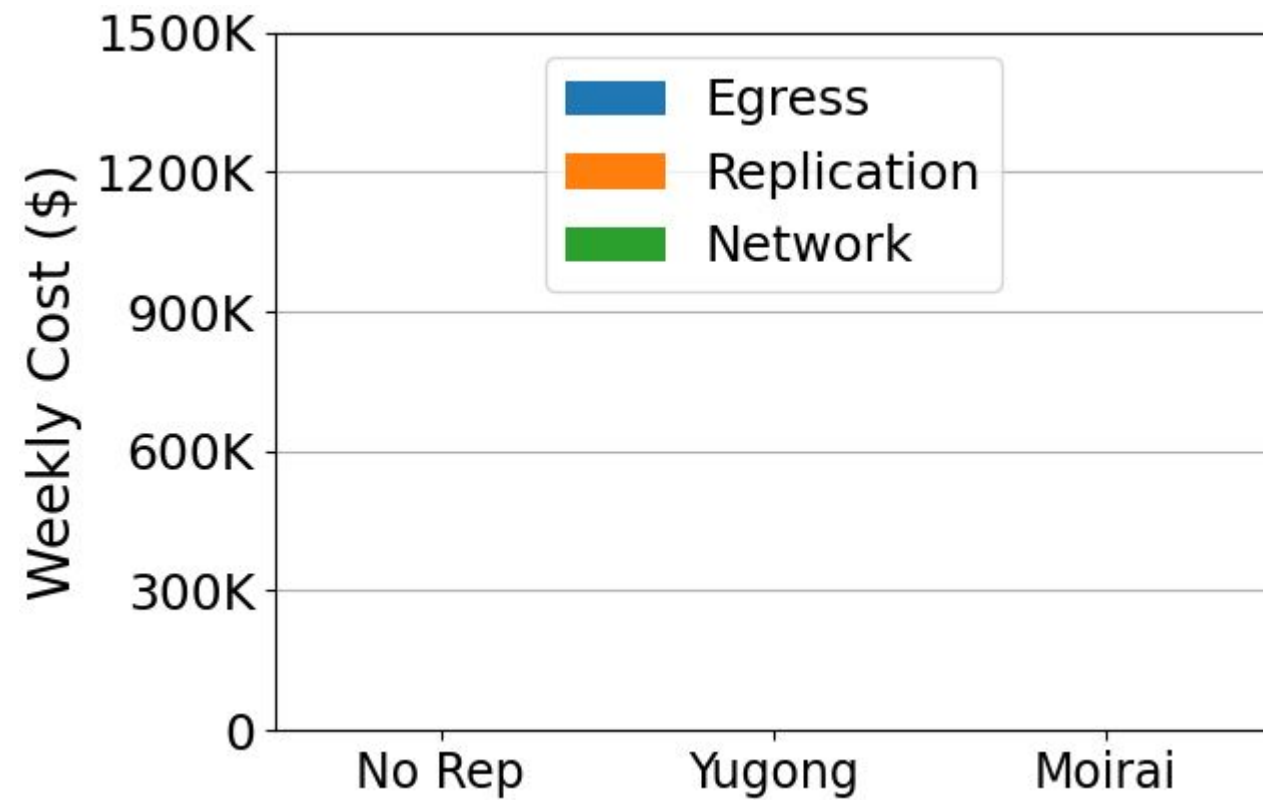
- Traces from Presto and Spark (open-source now)
 - First production dataset of job–data dependencies at scale
- Baselines:
 - *NoRep*: No Replication, splitting data randomly



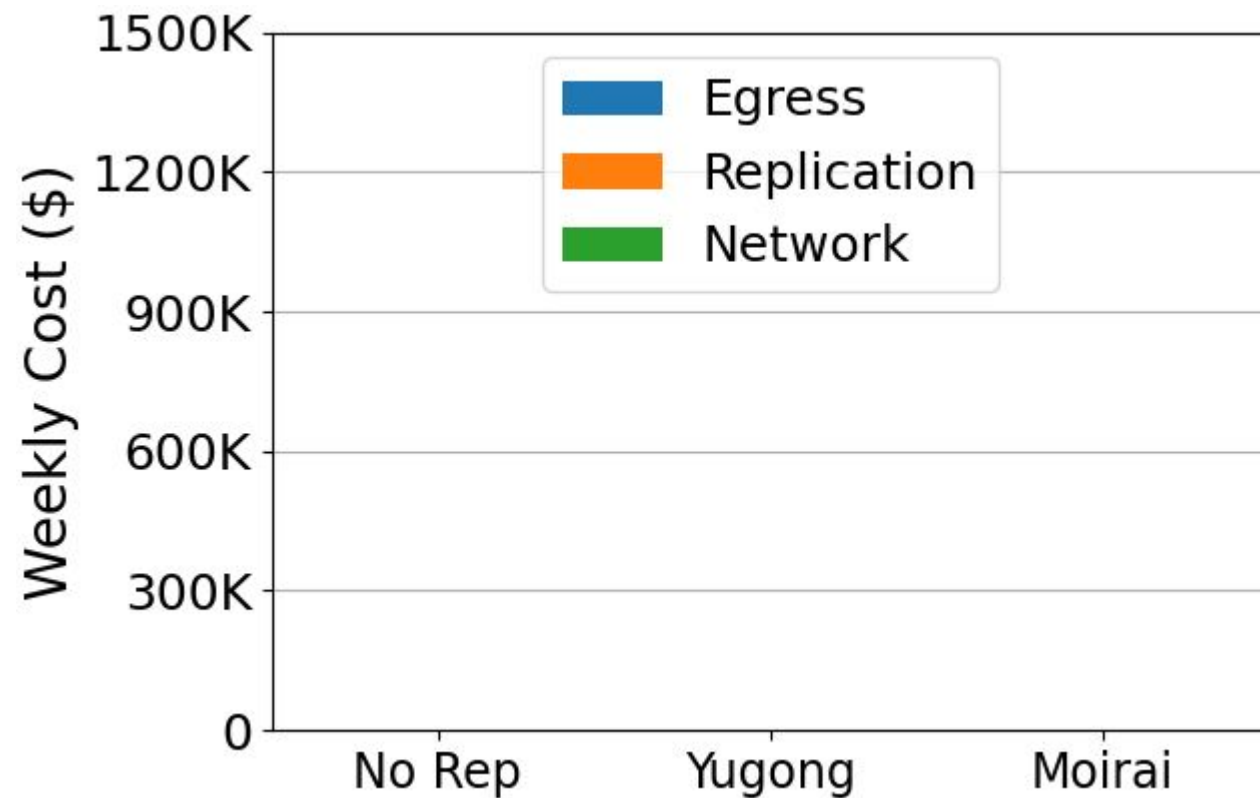
Experimental Setup

- Traces from Presto and Spark (open-source now)
 - First production dataset of job–data dependencies at scale
- Baselines:
 - *NoRep*: No Replication, splitting data randomly 
 - *Yugong*: state-of-the-art
 - Uses Mixed-Integer Programming model 
 - Makes decisions at the owner level
 - Owners don't always align with the actual data usage

Cost Reduction w/ Moirai



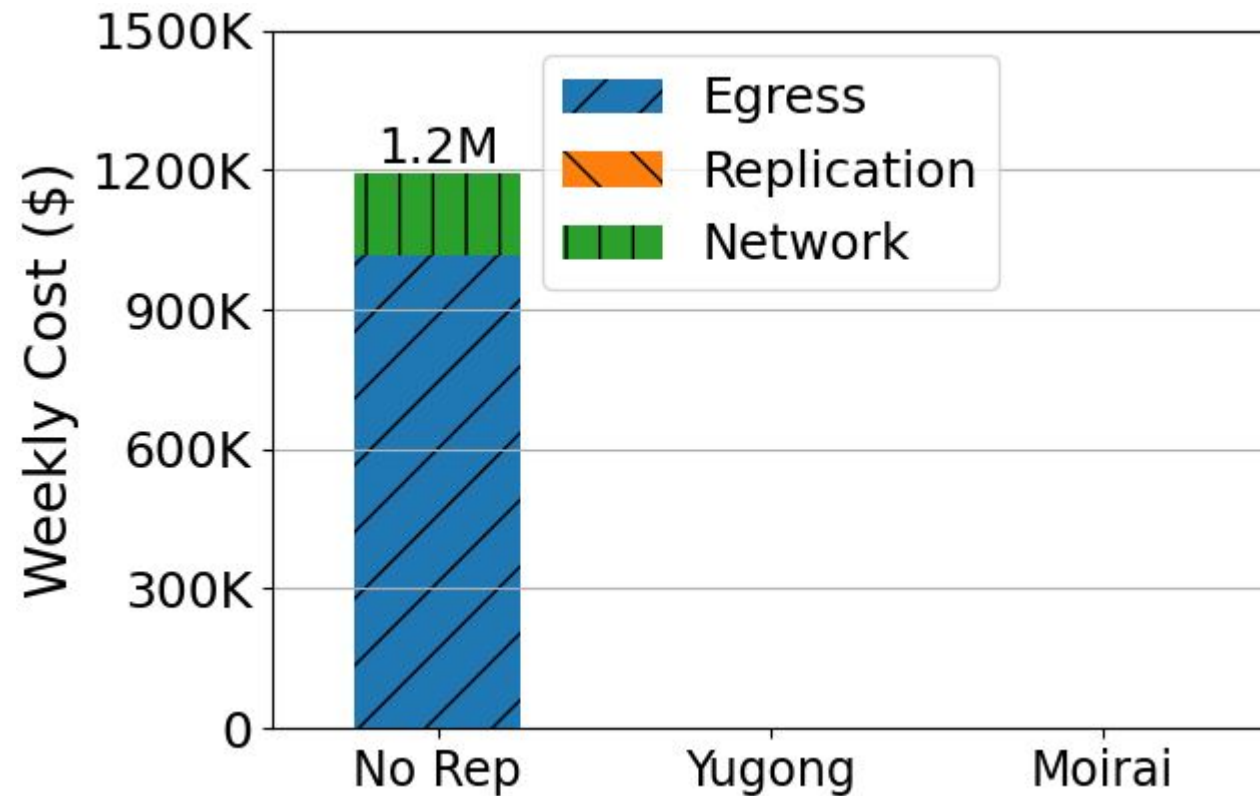
Cost Reduction w/ Moirai



- Egress: transfer out of cloud (/Byte)
- Replication: extra storage cost
- Network: dedicated link
- 50% job and data on-premises

Cost Reduction w/ Moirai

- No Rep shows significant egress costs




Cost Reduction w/ Moirai



- No Rep shows significant egress costs
- Yugong with MIP model reduces costs
 - Remains large improvement space
- Moirai: **97% cheaper than Yugong (SOTA)**
 - Fine-grained job-data optimization
- Moirai is a general-purpose framework
 - Works for multi-DC on-prem as well

Summary

- Moirai optimizes job and data jointly at Uber scale
 - By discovering and leveraging job-data dependencies
- Moirai reduces the costs by **97% over SOTA**
 - Optimization time is reduced to 2 hours (84 × faster)
- Publicly available traces and code here: 
 - <https://www.pdl.cmu.edu/Moirai/index.shtml>
- More details in the paper
 - Analytical model, routing for unseen jobs, deployment, ...