Cheap data analytics using cold storage devices

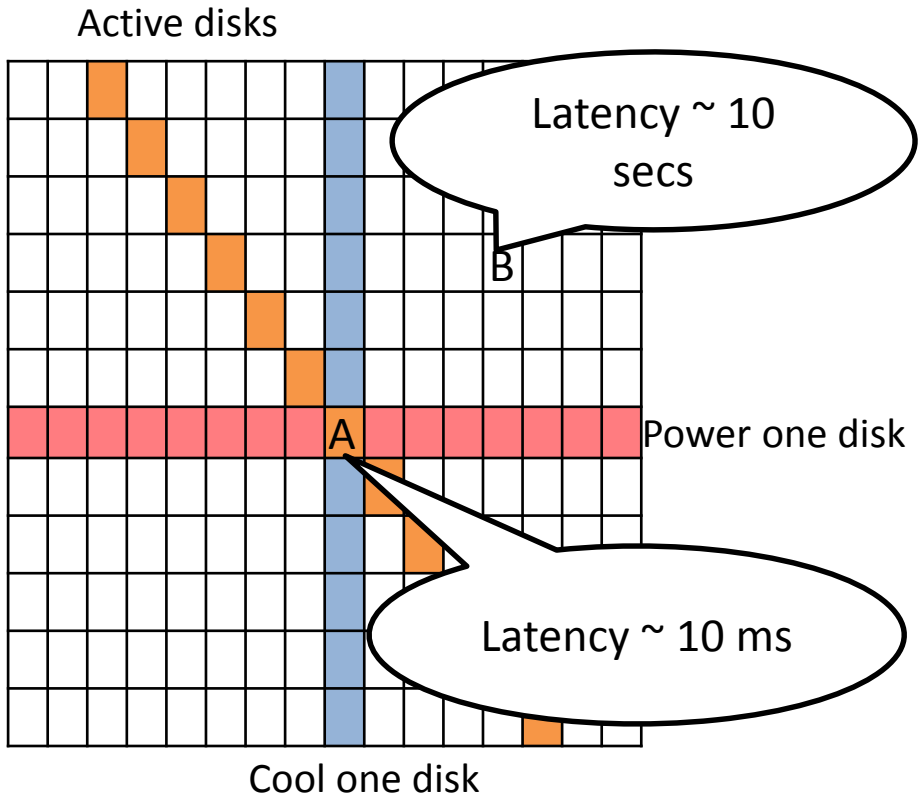
Renata Borovica-Gajic, Raja Appuswamy, and Anastasia Ailamaki

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

Proliferation of cold data

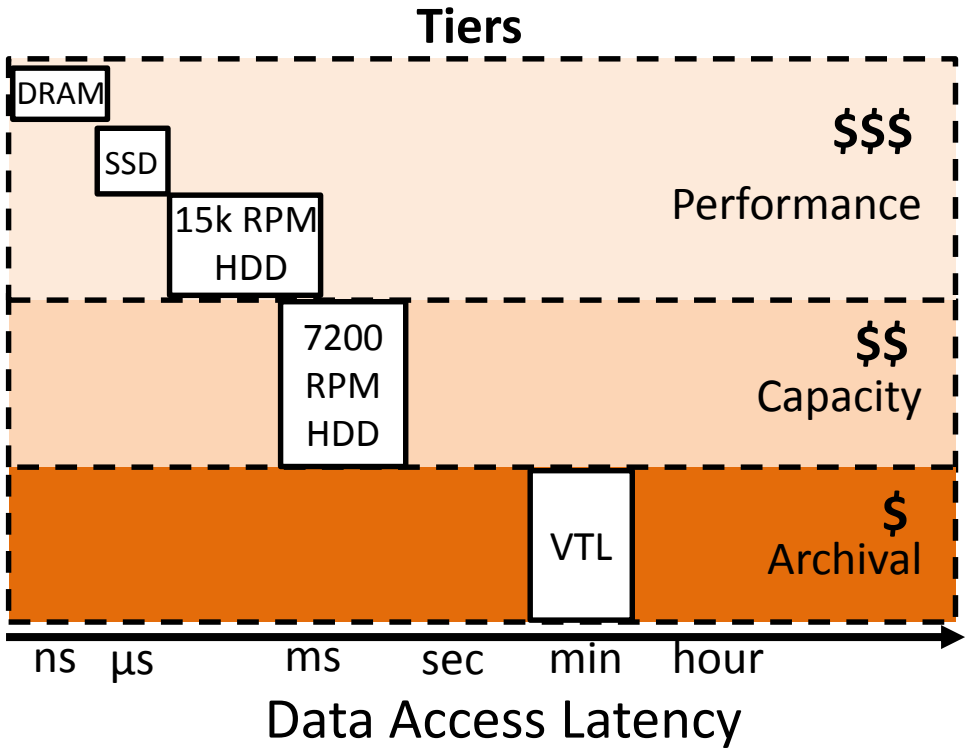
"80% enterprise data is cold with 60% CAGR" [Horison]
"cold data: an incredibly valuable piece of the analysis pipeline" [Intel]

Cold Storage Devices (CSD) to the rescue

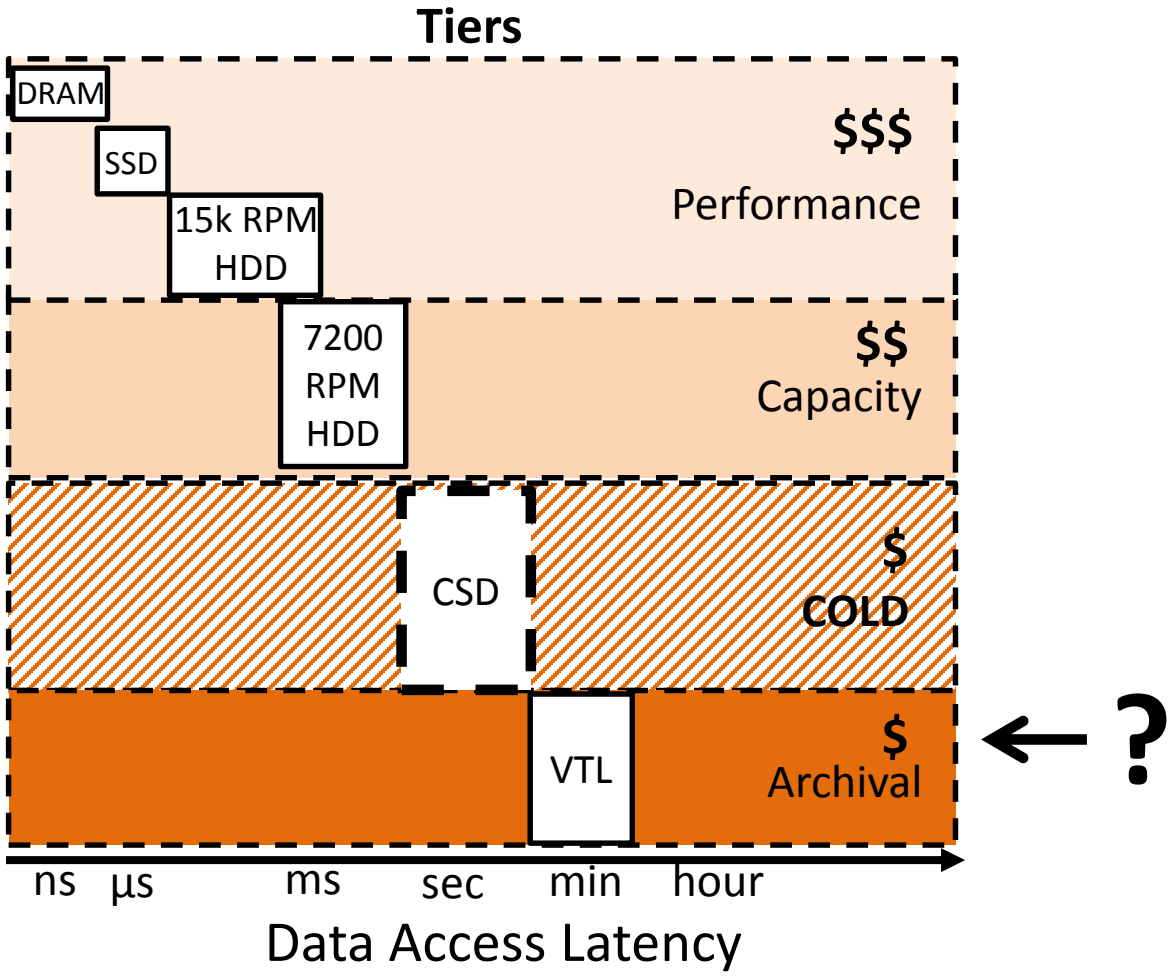


PB of storage at cost ~ tape and latency ~ disks

CSD in the storage tiering hierarchy

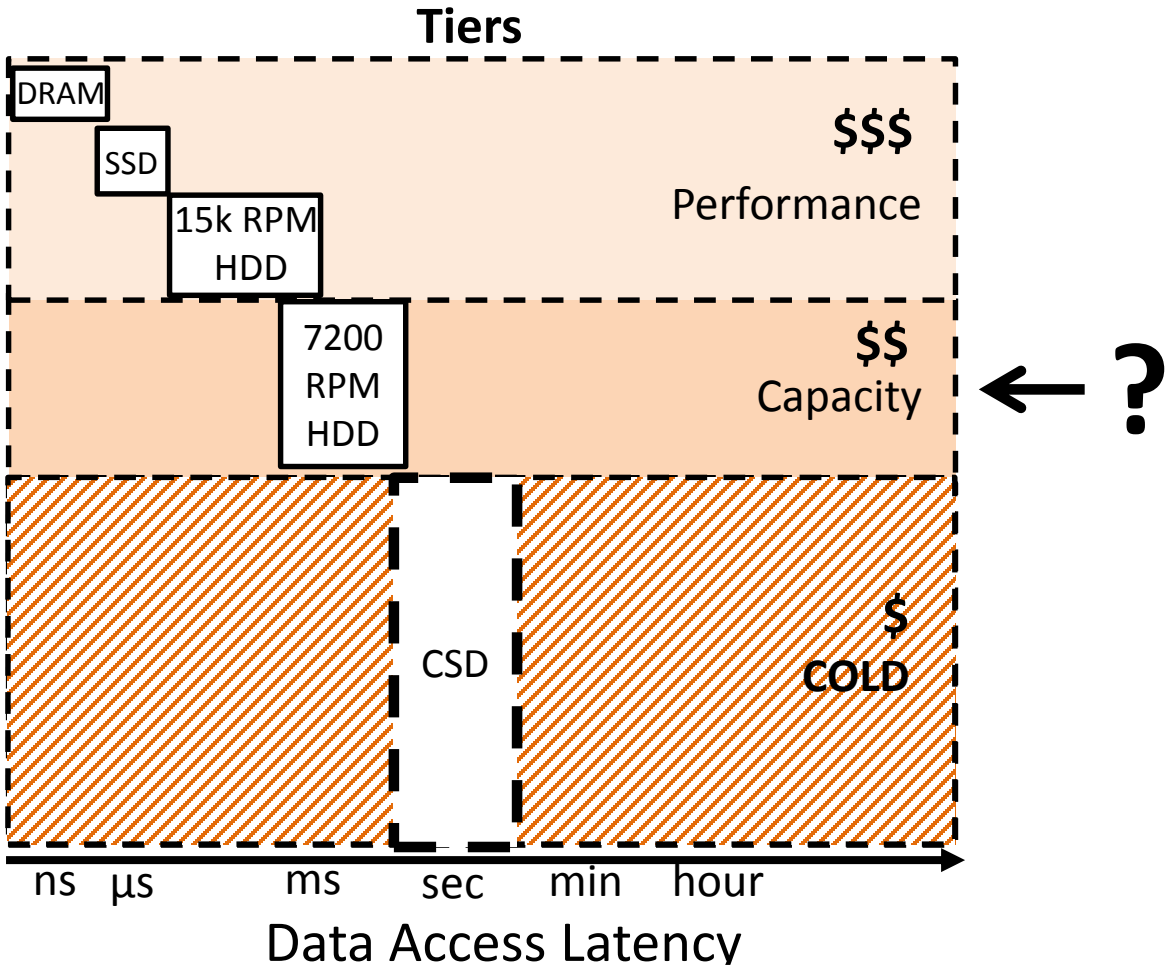


CSD in the storage tiering hierarchy



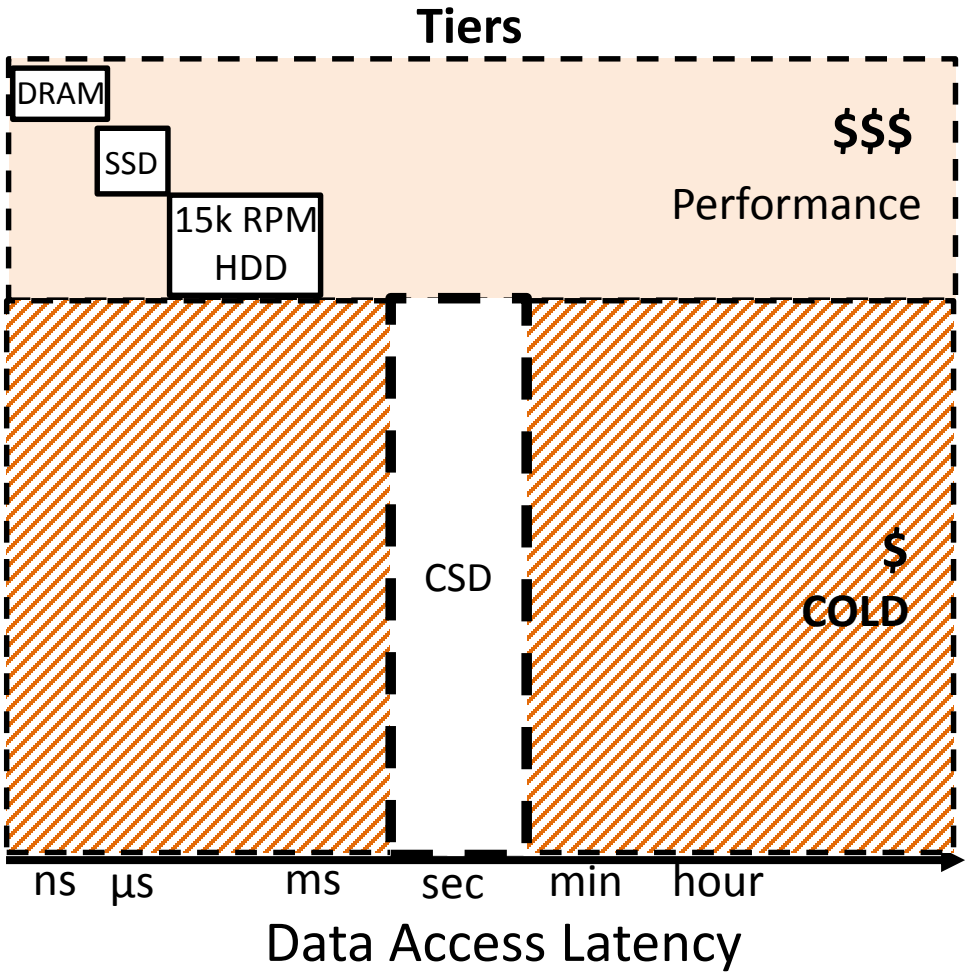
Can we shrink tiers to further save cost?

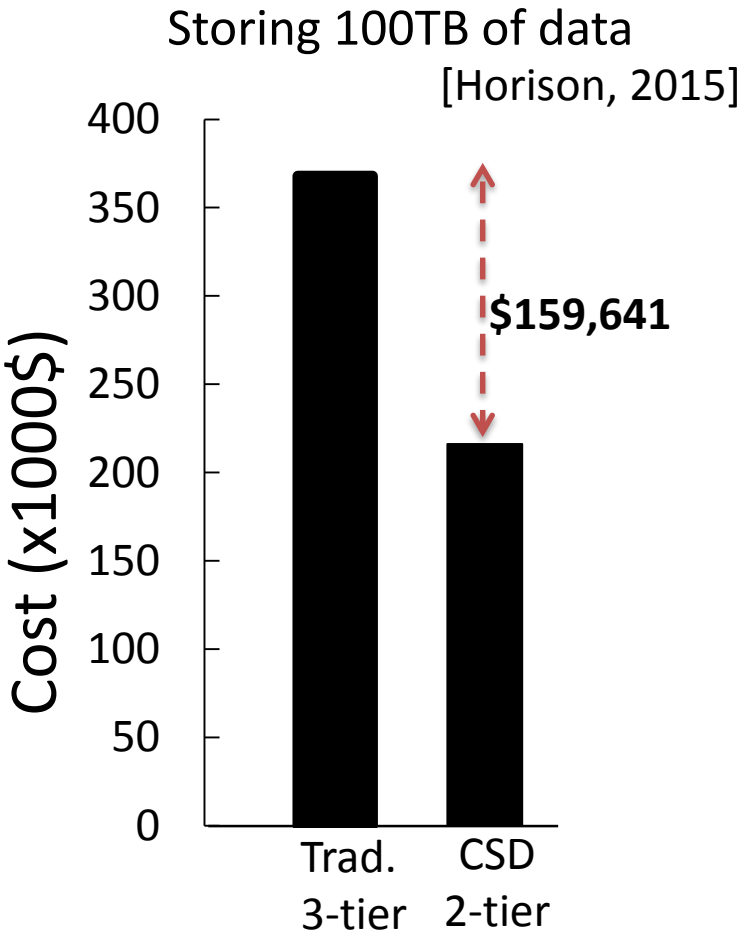
CSD in the storage tiering hierarchy



Can we shrink tiers to further save cost?

CSD in the storage tiering hierarchy



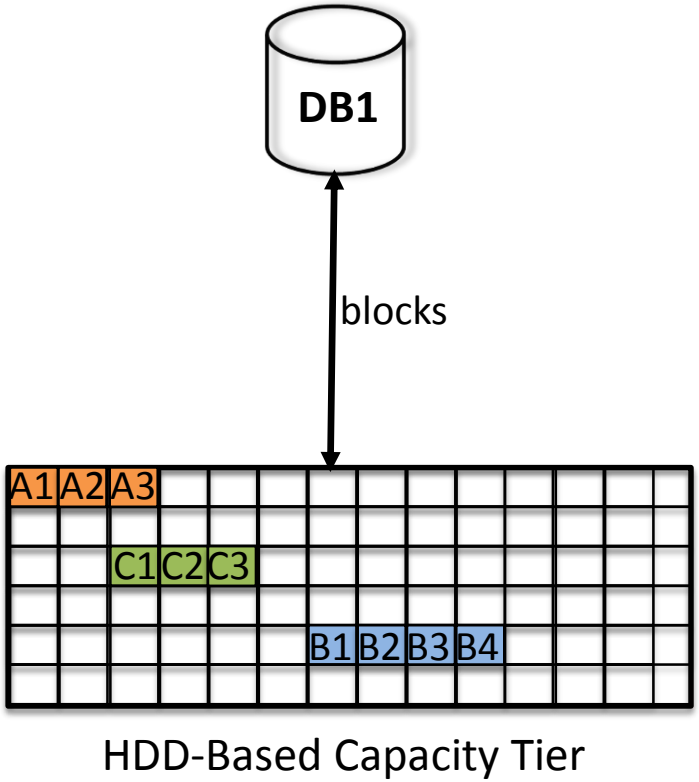


CSD offers significant cost savings (40%)

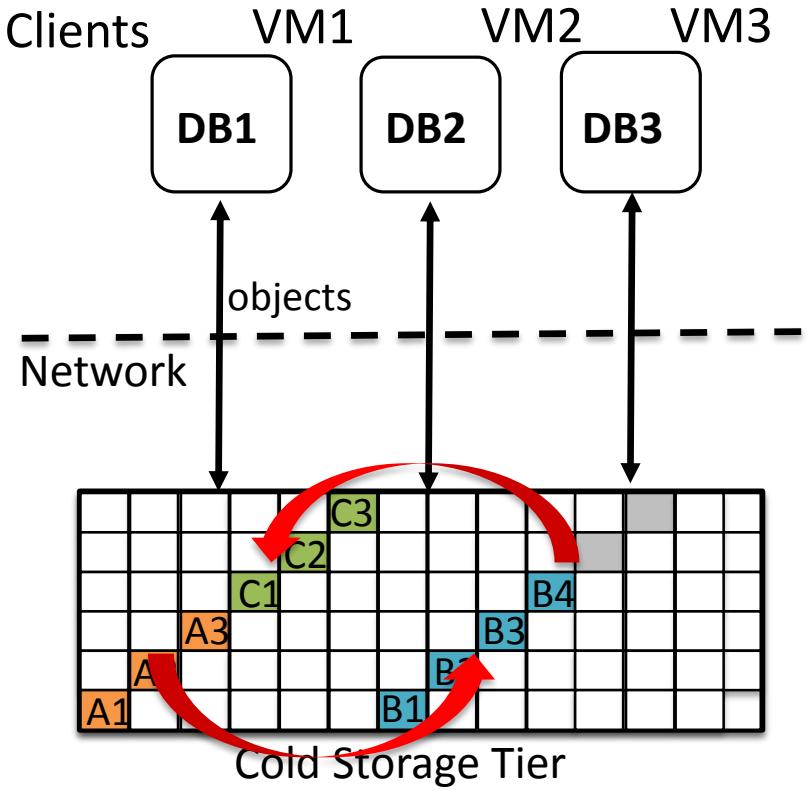
But… can we run queries over CSD?

Query execution over CSD

Traditional setting



Virtualized enterprise data center



Uniform access ✓ **Control layout** ✓

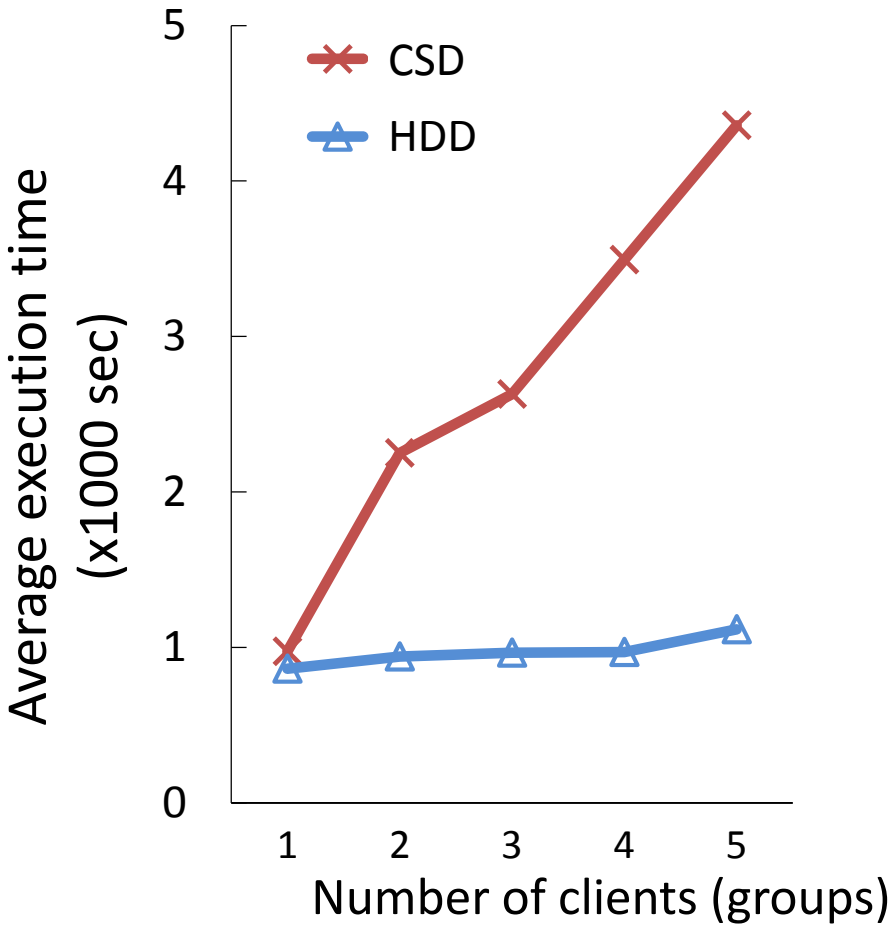
Static (pull-based) execution ✓

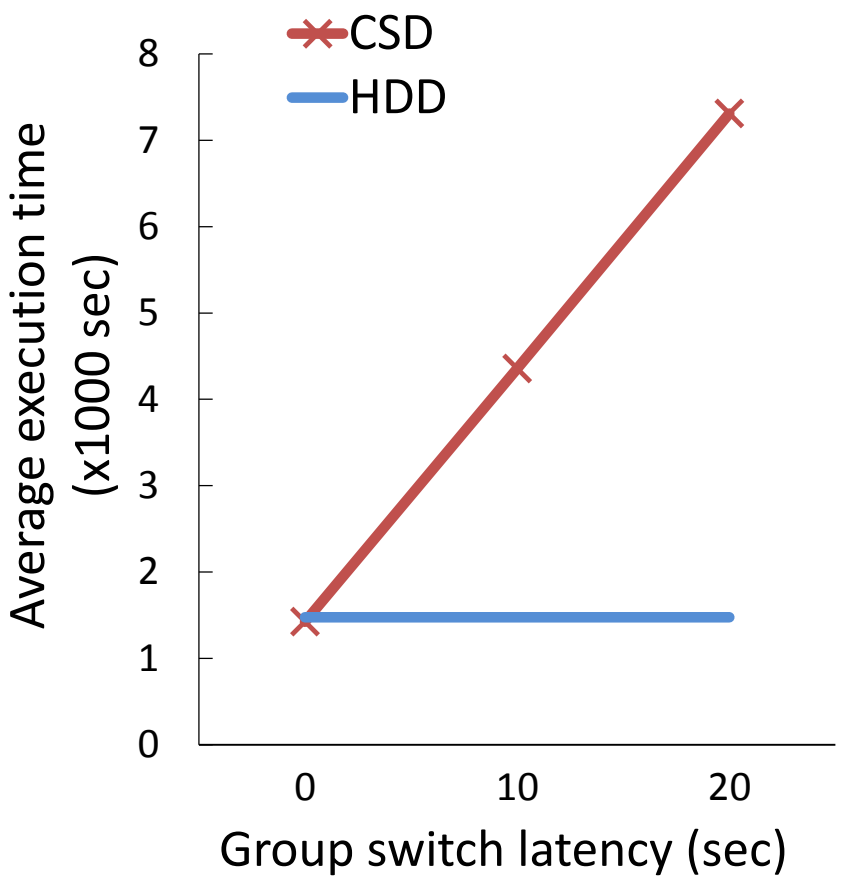
Uniform access ✗ **Control layout** ✗

Pull-based execution will trigger unwarranted group switches

What this means for an enterprise datacenter…

Setting: multitenant enterprise datacenter, clients: PostgreSQL , TPCH 50, Q12, CSD: shared, layout: one client per group



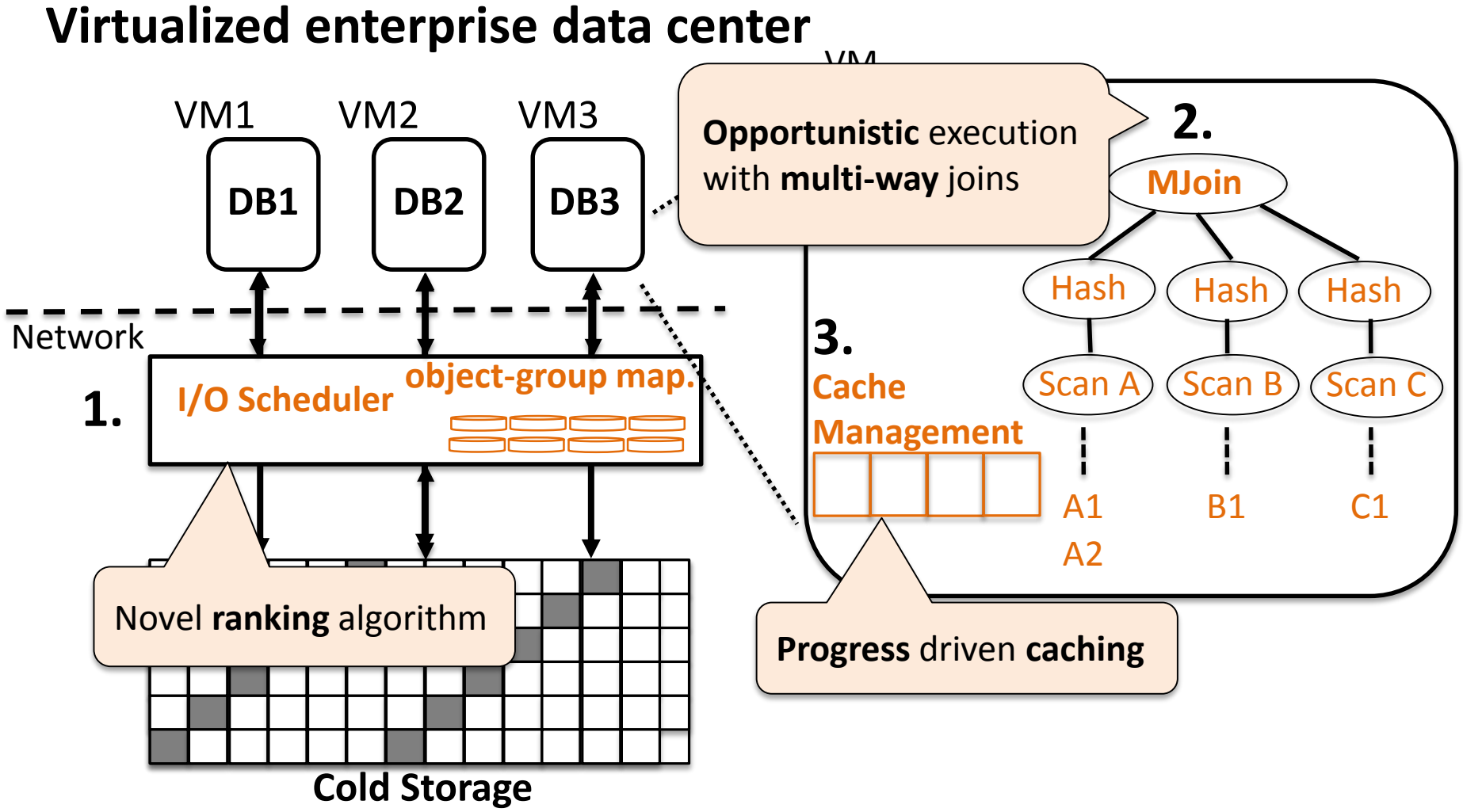


Lost opportunity: CSD relegated to archival storage

Need hardware-software codesign

1. Data access has to be **hardware-driven** to minimize group switches
2. Query execution engine has to process data pushed from storage in **out-of-order** (unpredictable) manner
3. Reduce data round-trips to cold storage by **smart** data **caching**

Skipper to the rescue



Multi-way joins in PostgreSQL

Setting: Query AxBxC, A:A1, A2; B: B1,B2; C:C1, C2;

VM: PostgreSQL

State Manager

Subplans:

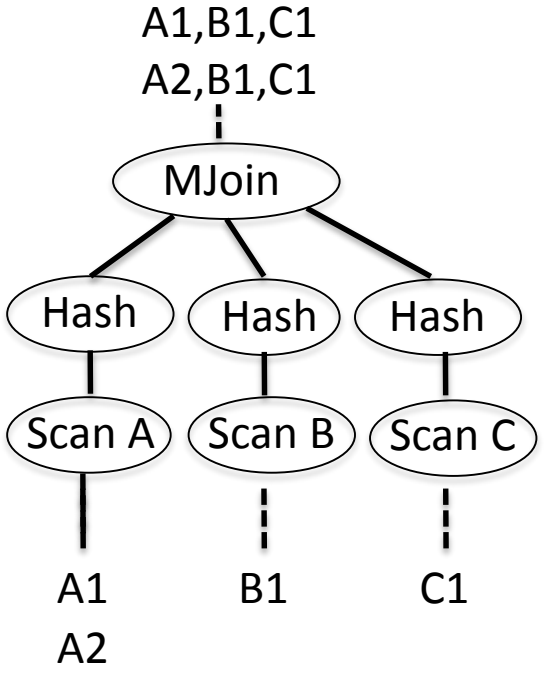
Pending
~~A1,B1,C1~~
~~A1,B1,C2~~
~~A1,B2,C1~~
~~A1,B2,C2~~
~~A2,B1,C1~~
~~A2,B1,C2~~
~~A2,B2,C1~~
~~A2,B2,C2~~

Executed
A1,B1,C1
A2,B1,C1

Cache Manager

A1	A2	C1	B1

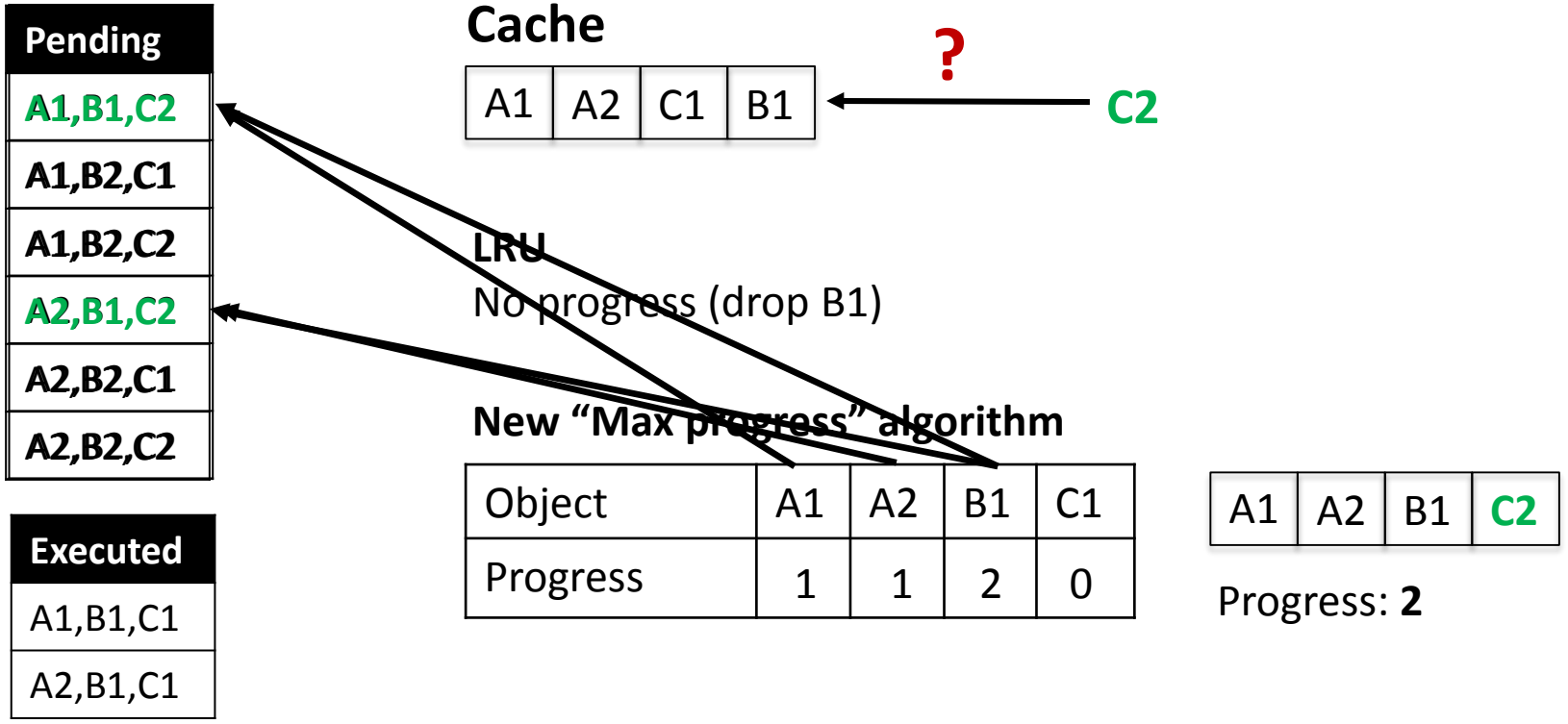
Join Execution



Enable out-of-order opportunistic execution

Progress driven caching

Setting: Query AxBxC, Cache size: 4, Cache full, Evict a candidate



Minimizes data roundtrips, maximizes query progress

Rank-based scheduling

Which group to switch to ?

Group	Table objects
G1	O1 (DB1), O3 (DB3)
G2	O2 (DB2), O4 (DB4)
G3	O5 (DB5)

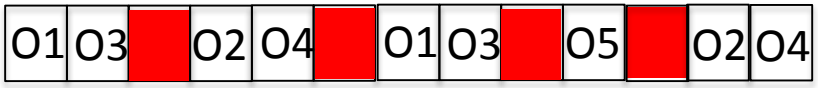
O1, O2, O3, O4, O5

TIME

New Ranking Algorithm

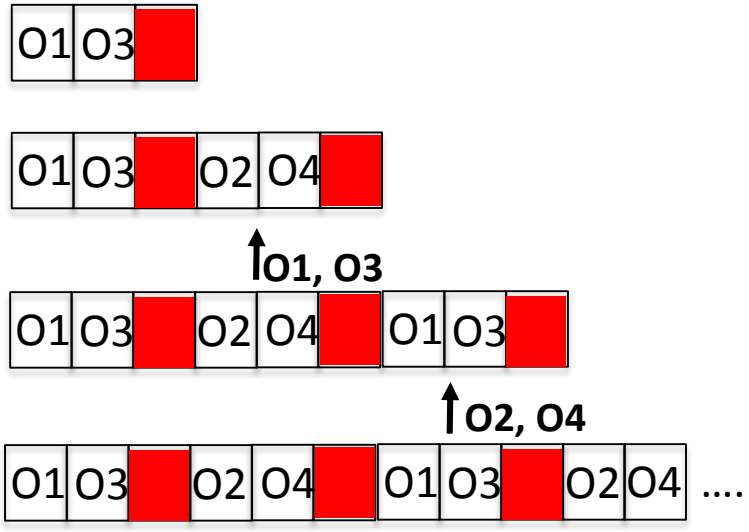
Rank(G) = #Requests + ∑Wait

Provides efficiency **Provides fairness**



FCFS – Fair but inefficient

Max-requests: Efficient, not fair

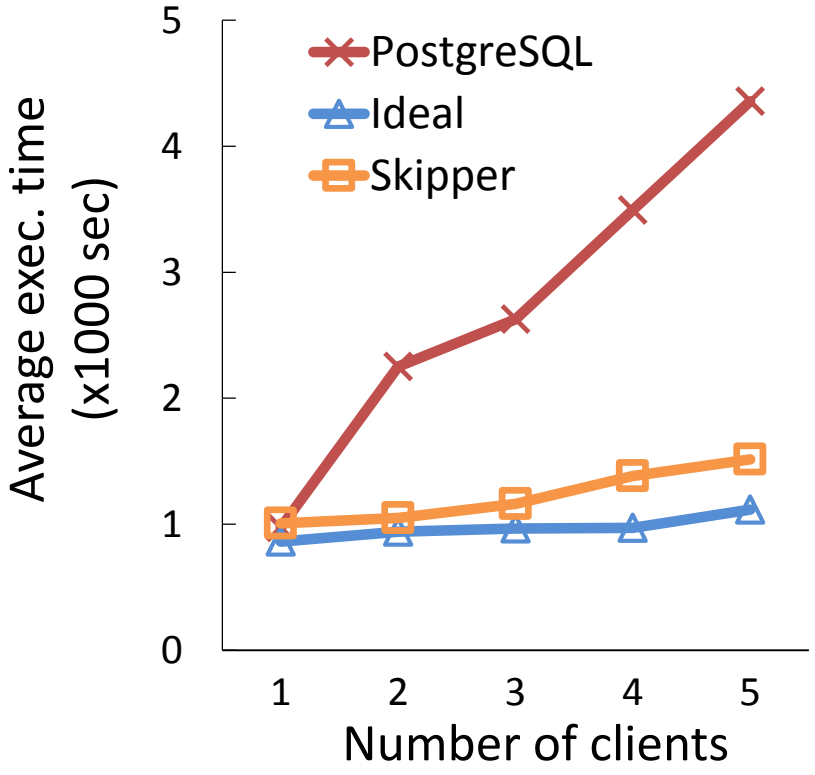


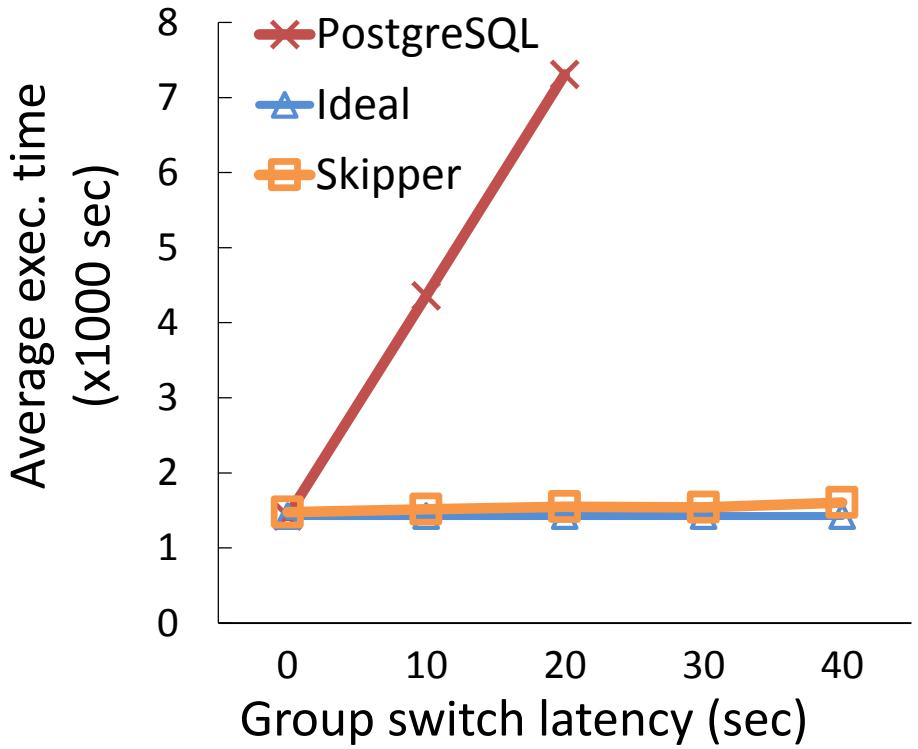
O5 STARVES

Balances efficiency and fairness

Skipper in action

Setting: multitenant enterprise datacenter, clients: TPCH 50, Q12, CSD: shared, layout: one client per group



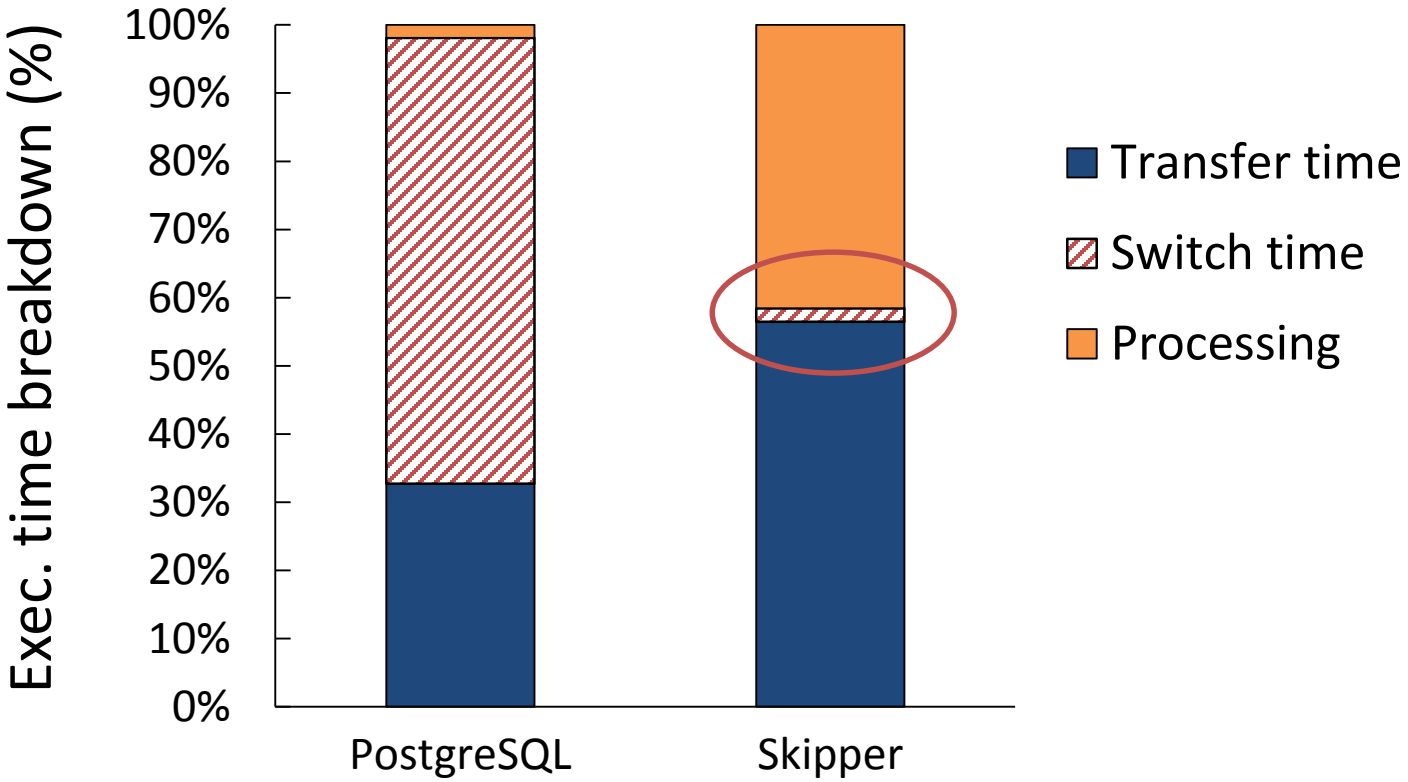


Skipper performs within 20% of HDD-based capacity tier

Skipper is resilient to group switch latency

Minimizing group switches

Setting: multitenant enterprise datacenter, 5 clients: TPCH 50, Q12, CSD: shared, layout: one client per group



Skipper substantially reduces overhead of group switches

Conclusions

- Cold storage can substantially reduce TCO
  - But DBMS performance suffers due to pull-based execution
- Skipper enables efficient query execution over CSD with
  - Out-of-order execution based on multi-way joins
  - Novel progress based caching policy
  - Rank based I/O scheduling
- Skipper makes data analytics over CSD as a service possible
  - Providers reduce cost by offloading data to CSD
  - Customers reduce cost by running inexpensive data analytics over CSD

Thank you!

DATA-INTENSIVE APPLICATIONS AND SYSTEMS