Cheap data analytics using cold storage devices

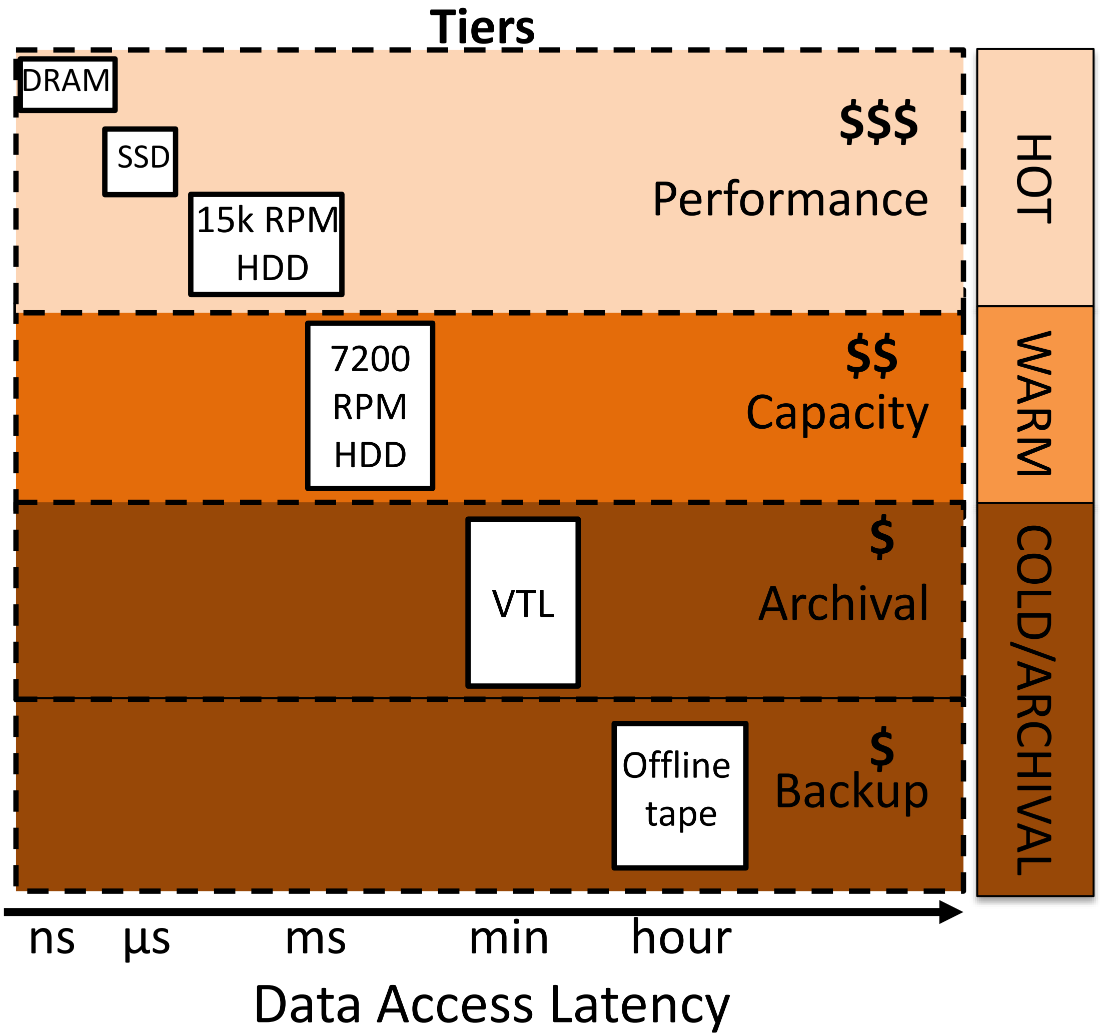
Data analytics for a penny

Renata Borovica-Gajic*

THE UNIVERSITY OF MELBOURNE

*joint work with Raja Appuswamy and Anastasia Ailamaki

Enterprise database storage tiering



Real-time analytics

Batch analytics

Compliance and verification

Disaster and recovery

Reduces capital and operational expenses

Proliferation of cold data

“**Enterprise data** is growing at a rate of **40% to 60% per year** and is projected to grow 50-fold – from under one zettabyte in 2010 to 40 zettabytes by 2020. “ [GigaOM]

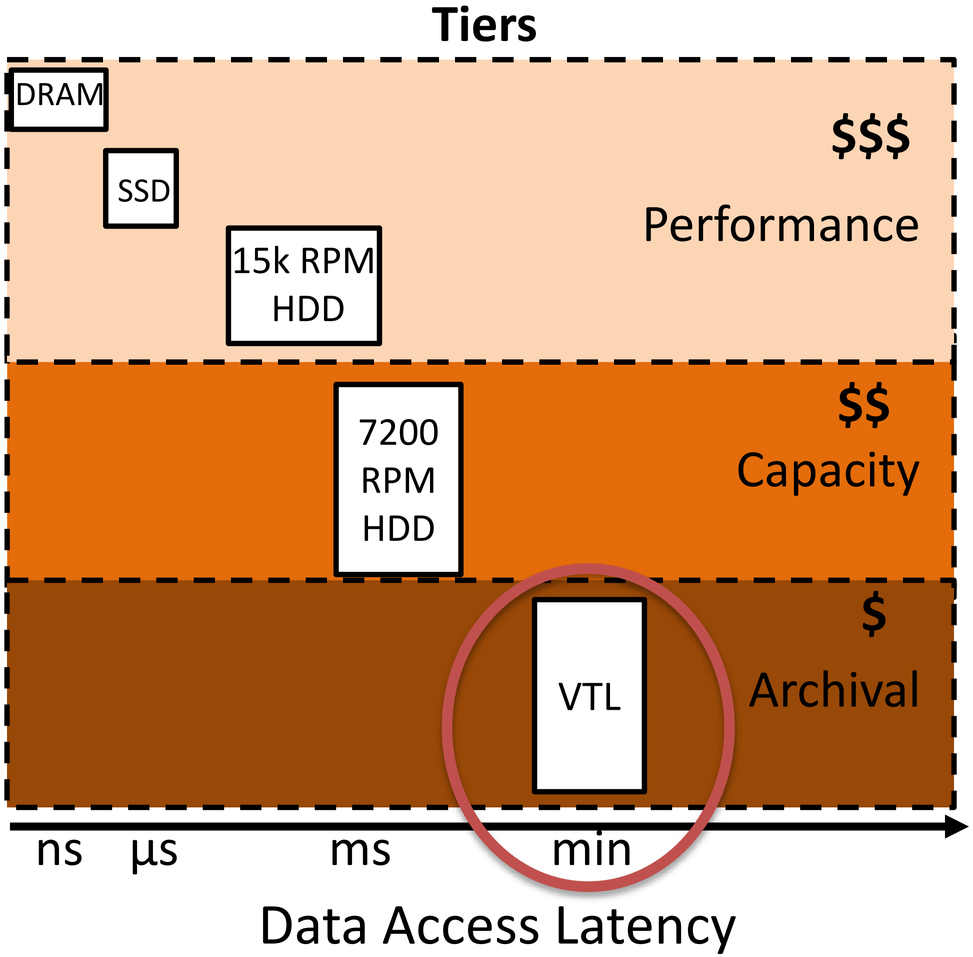
“**Archival data** presently represents approximately 43-60% of all data stored online, making it the **largest category** and at **> 60% CAGR** is the fastest growing data classification segment. “

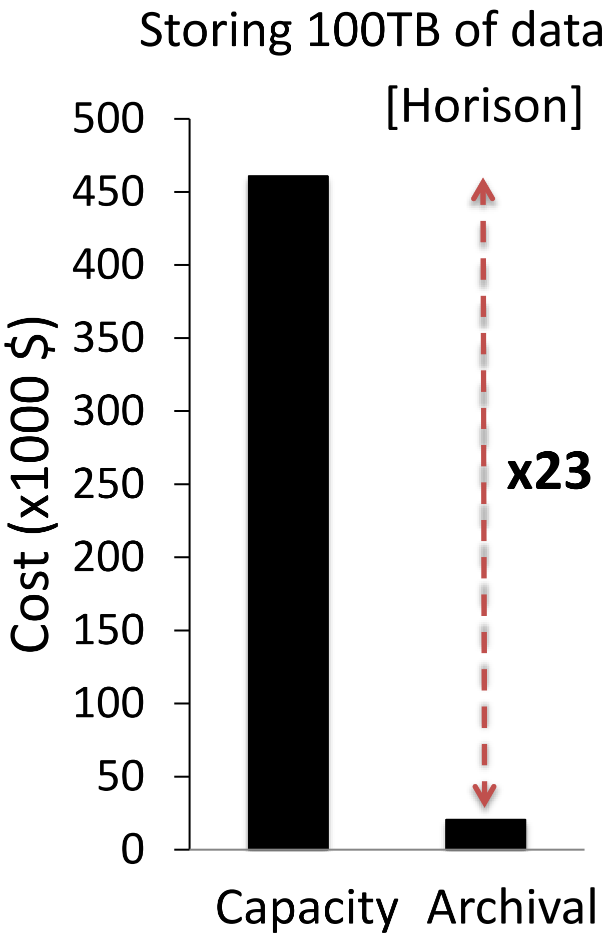
[Horison]

“Although **cold data** is infrequently accessed, it is still **incredibly valuable**. Businesses are increasingly investing in “big data” analytics to identify customer and operational trends, and to gain business insights. Cold storage must therefore provide the performance and capabilities required to **enable analysis.** “ [Intel]

Where should we store cold data?

Cold data in the storage hierarchy

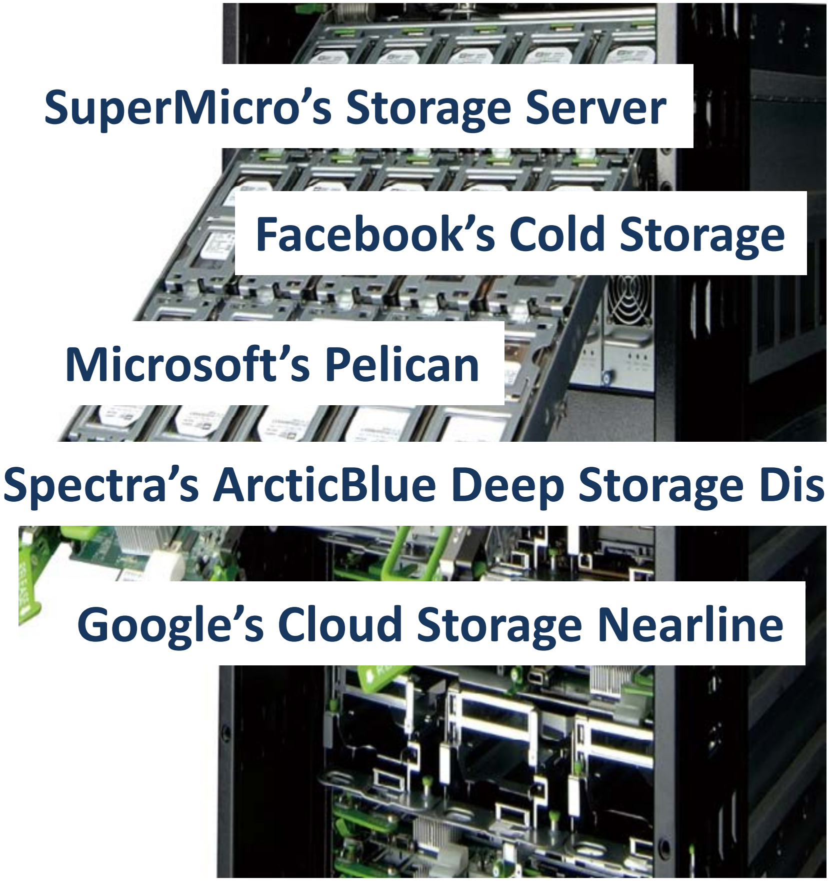




Capacity too expensive… archival too slow…

Cold Storage Devices – hardware for cold data

[Wiwynn]



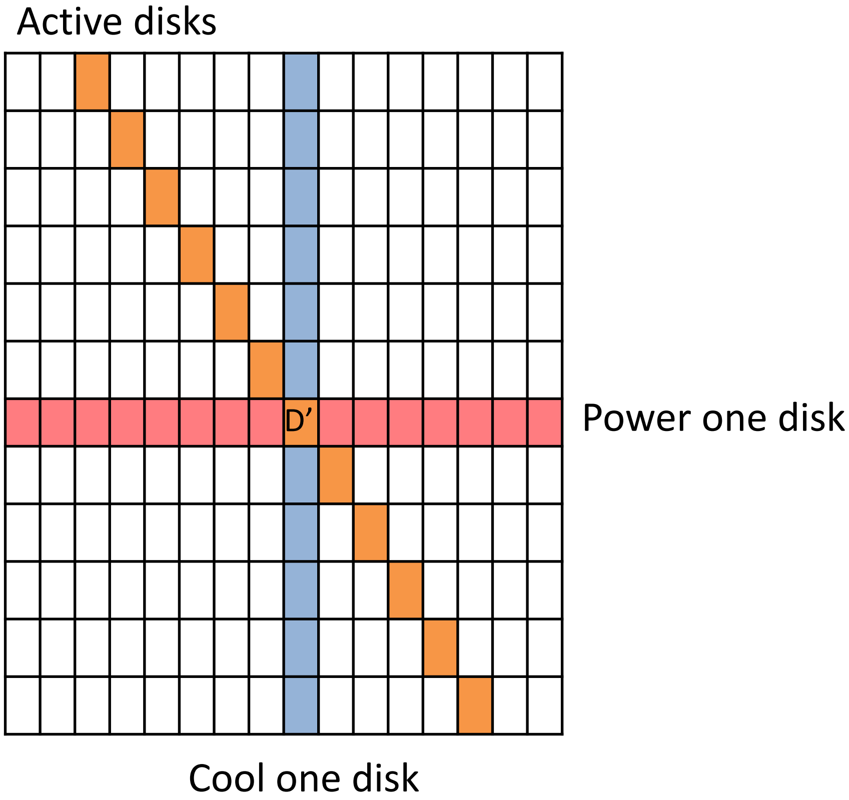
SuperMicro’s Storage Server

Facebook’s Cold Storage

Microsoft’s Pelican

Spectra’s ArcticBlue Deep Storage Disk

Google’s Cloud Storage Nearline

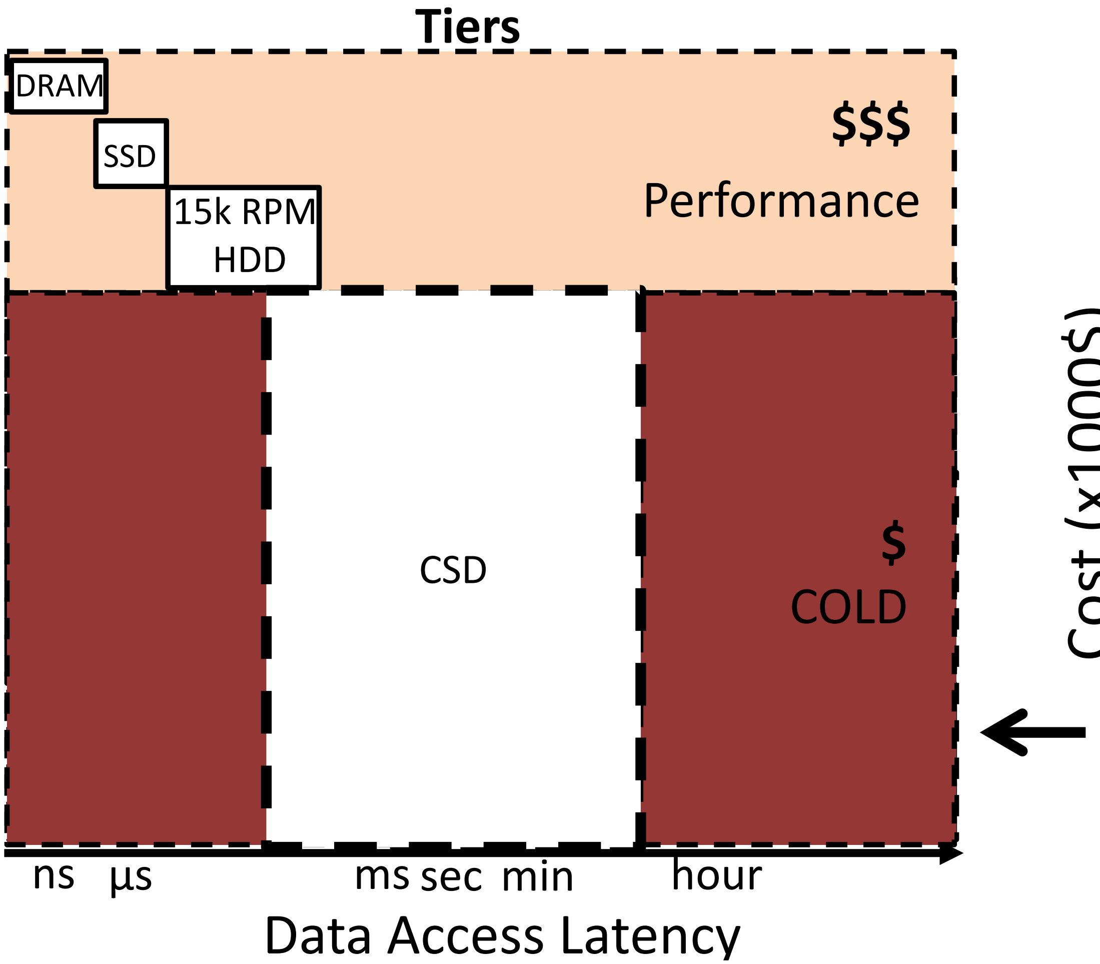


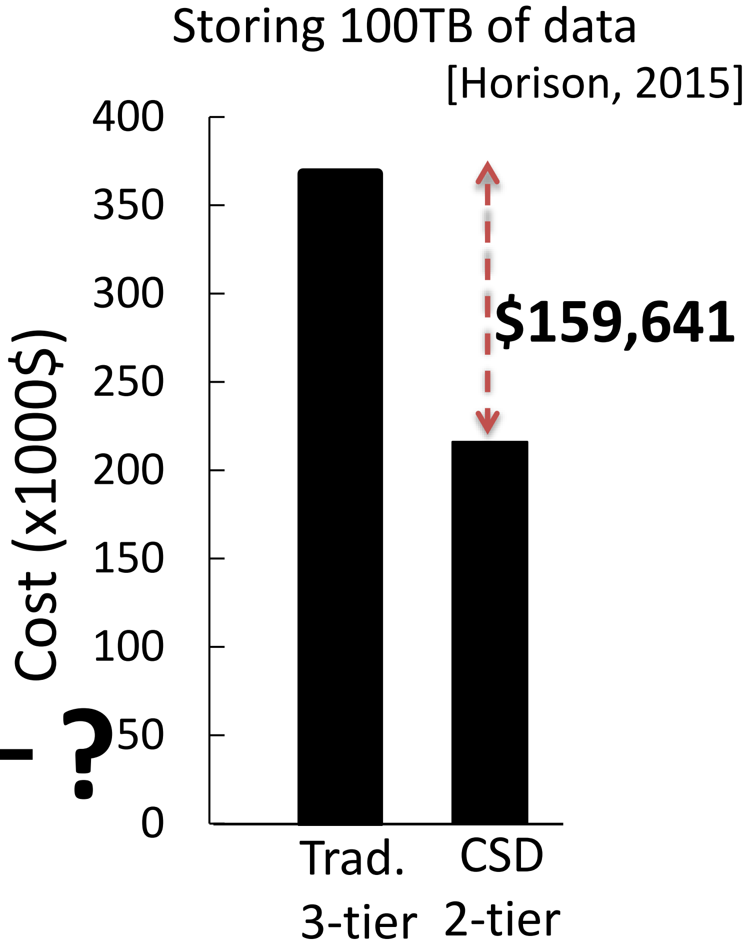
Rack-scale PB-size storage based on
High density HDD organized in **MAID**

Constraints on number of active disks
Group switch latency **~10sec**

TCO of tape + HDD latency, BUT handful of disks at a time

Cold storage in the tiering hierarchy



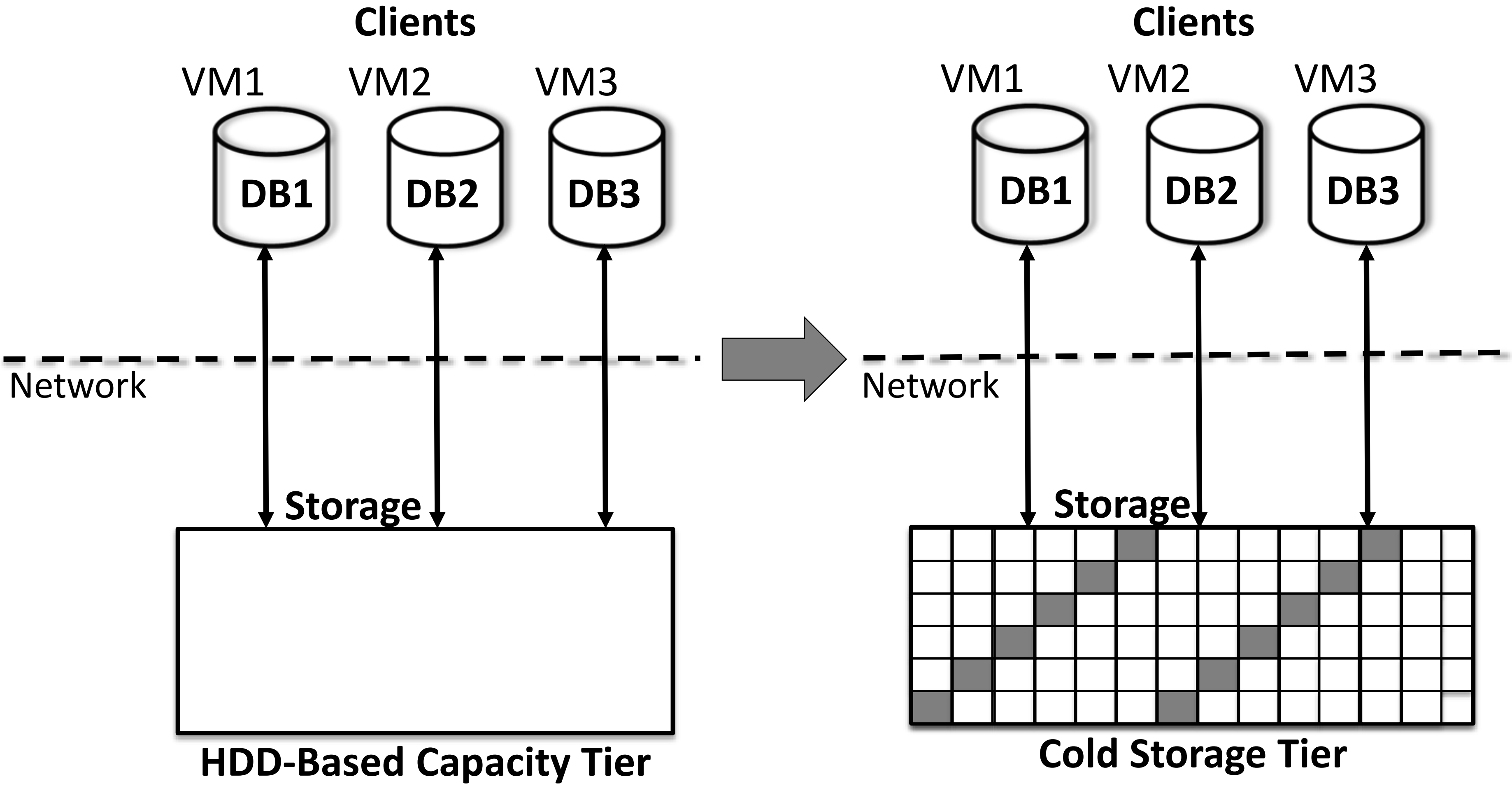


Can cold storage tier (CST) subsume archival and capacity tiers?

CST offers significant cost savings (40%)

Cold storage in enterprise data center

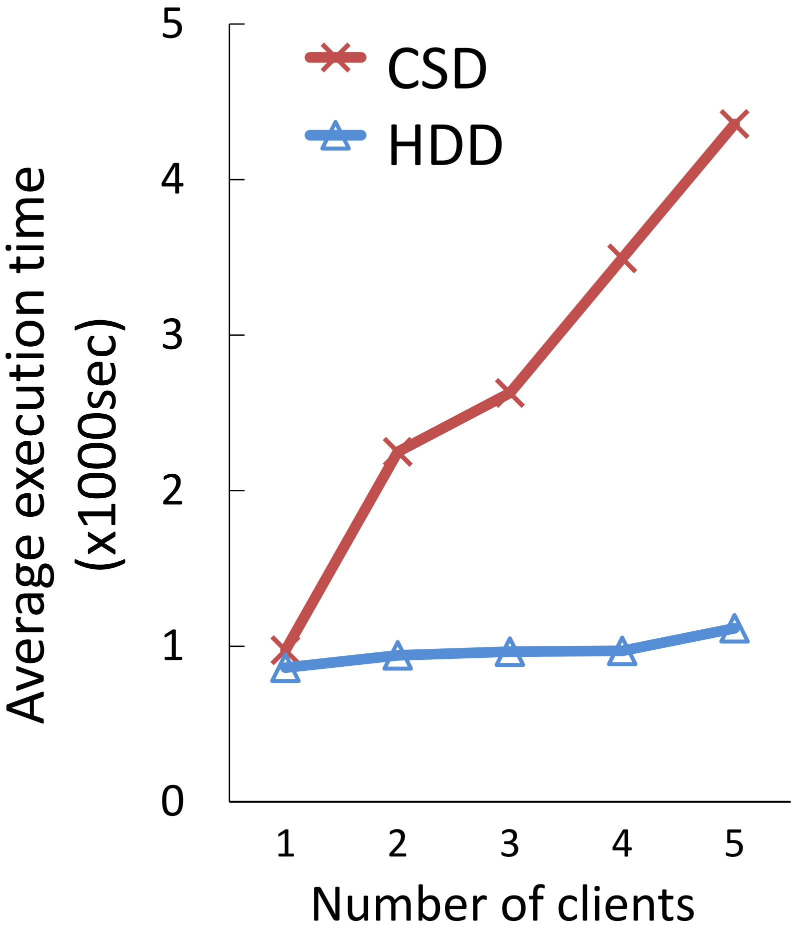
Virtualized enterprise data center

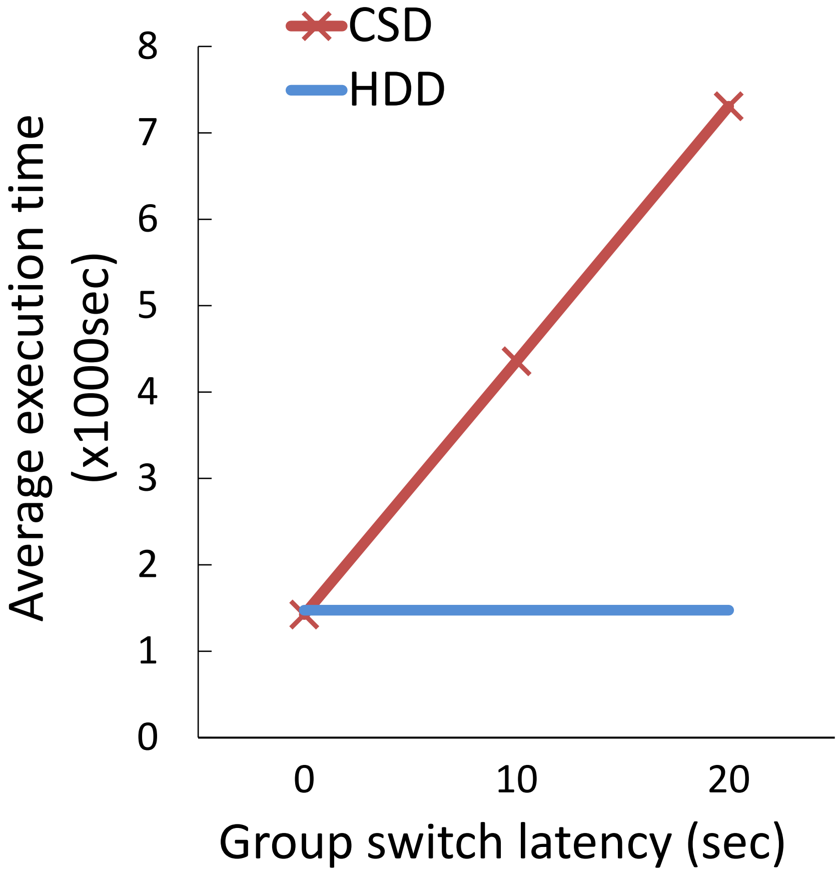


THE UNIVERSITY OF MELBOURNE

Querying data on CSD

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





High group switch latency severely degrades performance

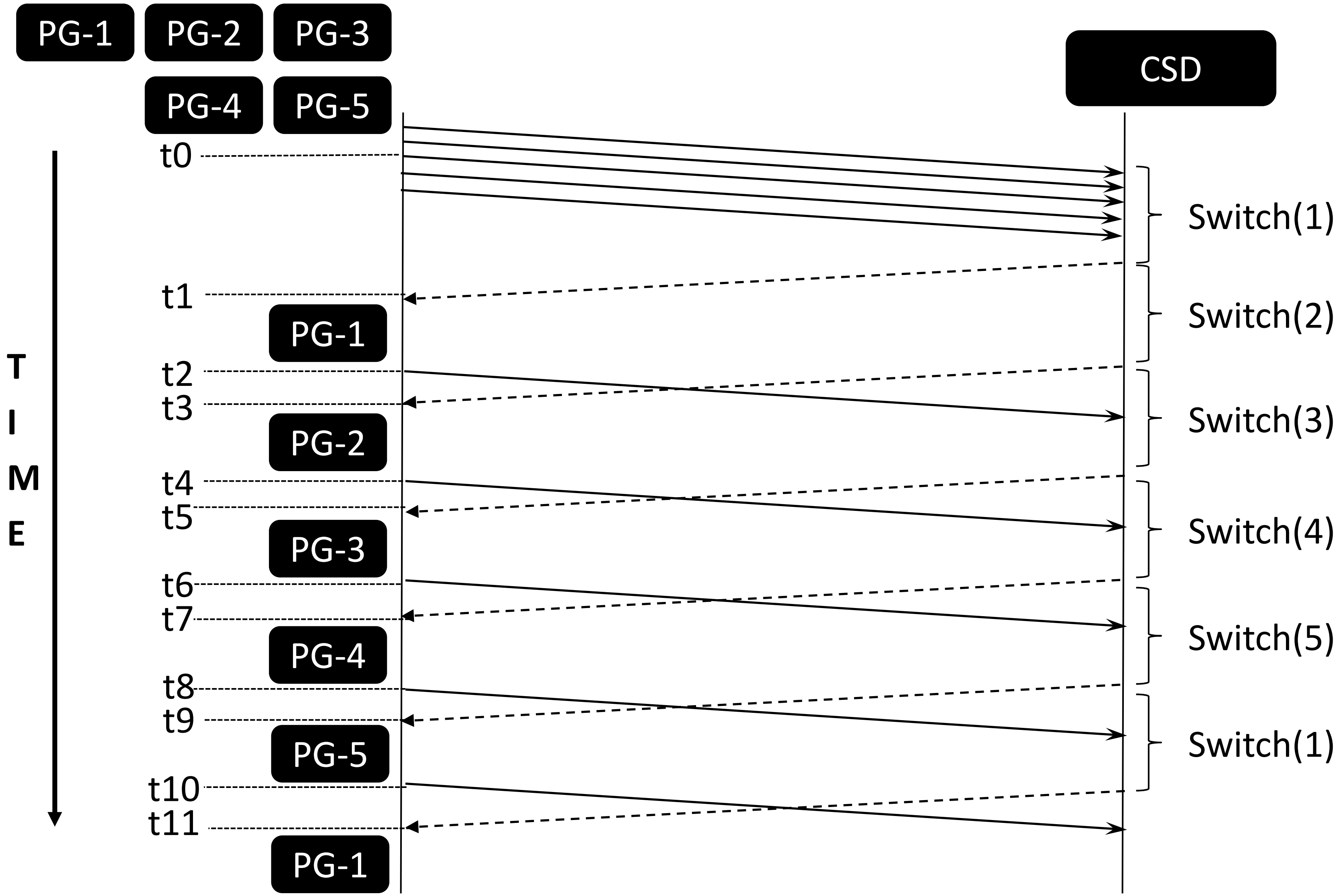
Cold storage device pitfalls

- **Non-uniform access latency**
  - Same as “warm” storage if data is on the spun-up disk
  - Otherwise 4 orders of magnitude slower (10s vs. ~ms)

- **Shared storage**
  - Each CSD hosts several DBs (by virtualizing storage)
  - DBs do not control data layout (data spread across disks)
  - CSD balances multi-tenancy & data locality

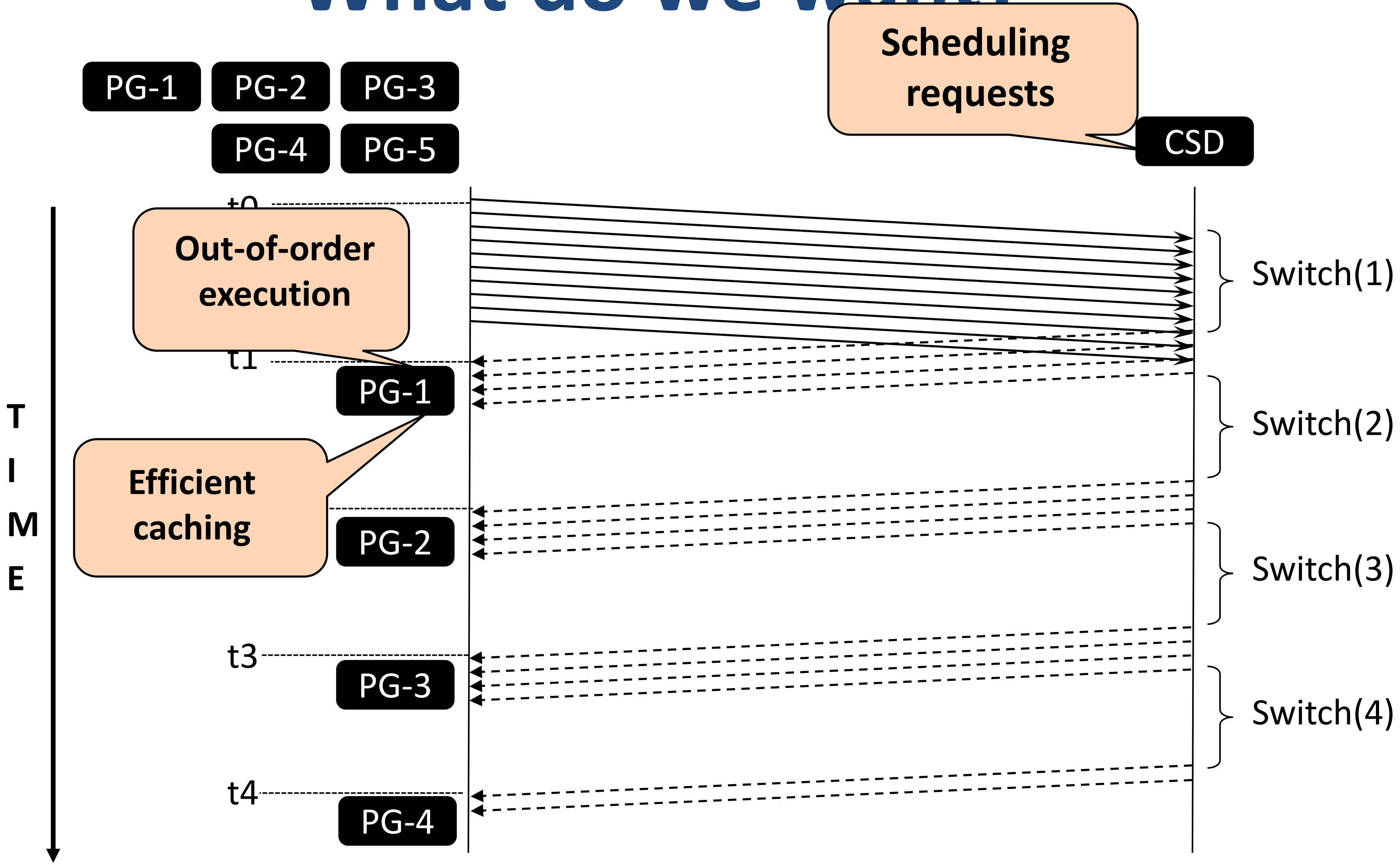
Poor & unpredictable DB performance due to lack of control

Why does performance suffer?



Pull-based execution model incompatible with CSD

What do we want?



From pull-based execution to push-based execution to minimize group switches

Need for the paradigm change

1. Data access has to be **hardware-driven** to minimize group switches

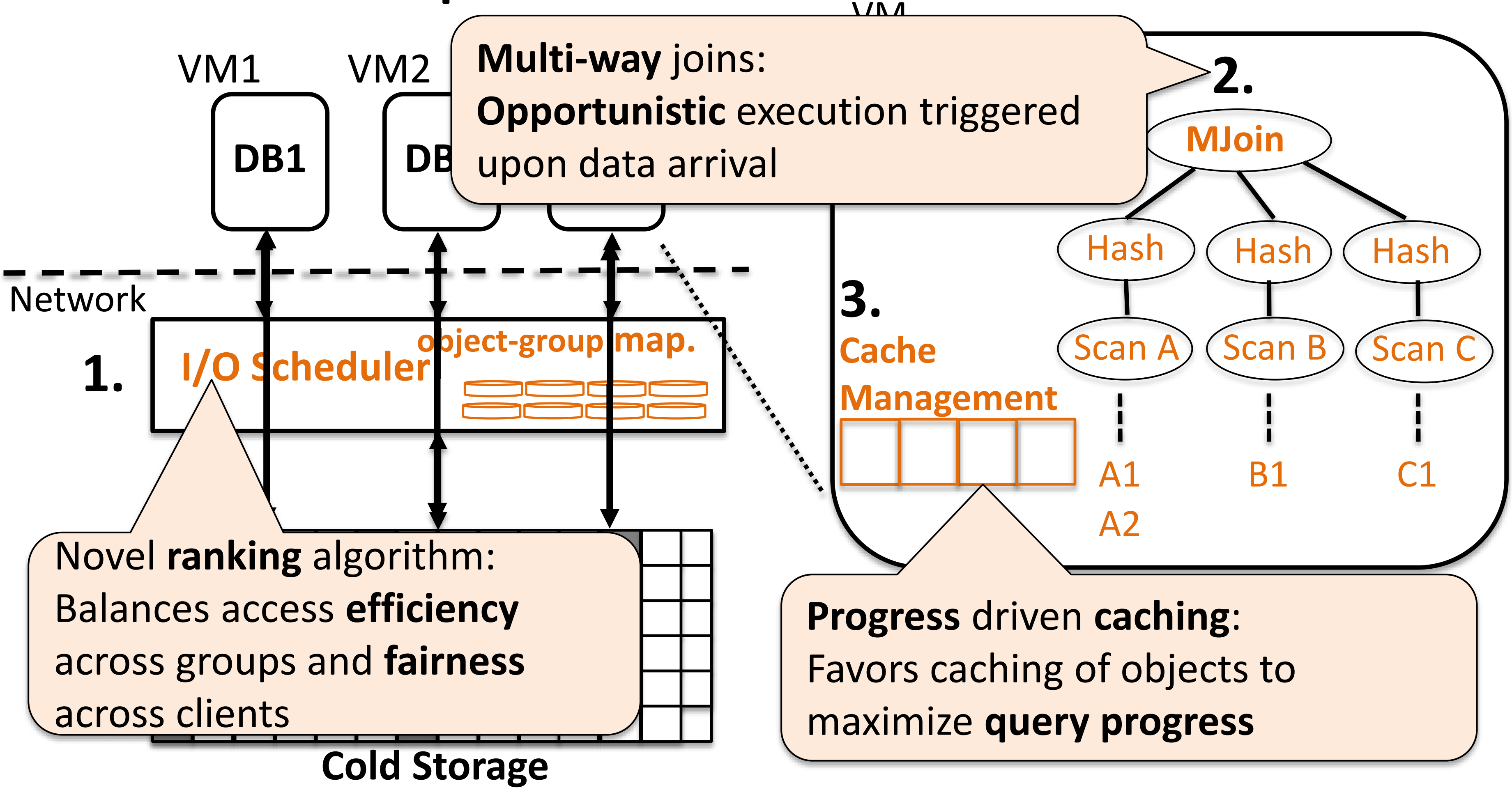
2. Query execution engine has to process data pushed from storage in **out-of-order** (unpredictable) manner

3. Reduce data round-trips to cold storage by **smart** data **caching**

Skipper to the rescue

[VLDB'16]

Virtualized enterprise data center



Multi-way joins:
Opportunistic execution triggered upon data arrival

Novel **ranking** algorithm:
Balances access **efficiency** across groups and **fairness** across clients

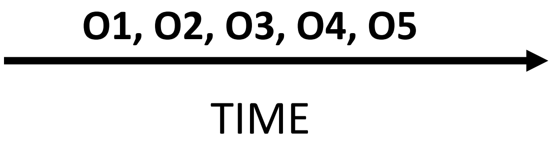
Progress driven **caching**:
Favors caching of objects to maximize **query progress**

Out-of-order execution with efficient cache and I/O scheduling policies

Rank-based scheduling

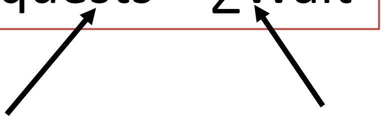
Which group to switch to ?

Group	Table objects
G1	O1 (Client1), O3 (Client3)
G2	O2 (Client2), O4 (Client4)
G3	O5 (Client5)

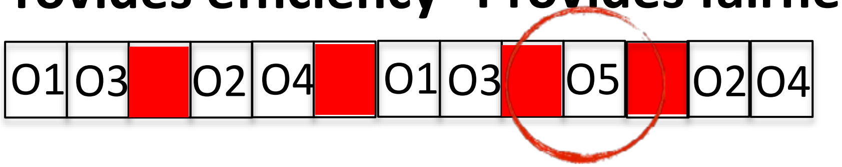


Rank-based scheduling

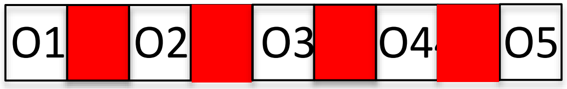
Rank(G) = #Requests + ∑Wait



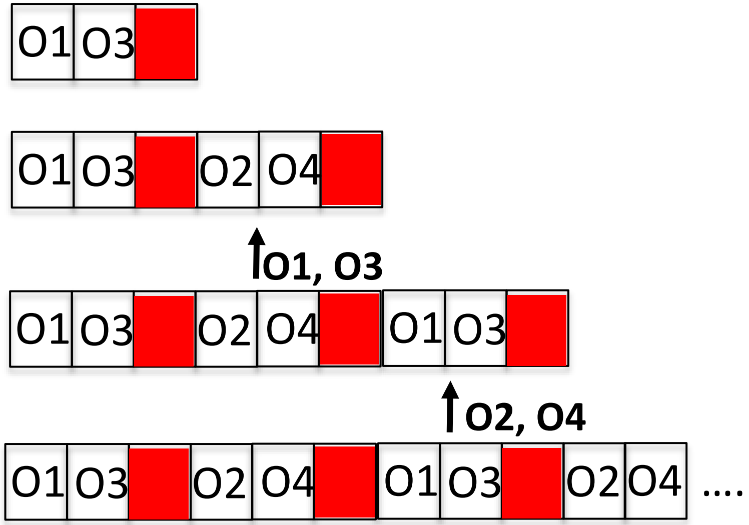
Provides efficiency **Provides fairness**



FCFS – Fair, but inefficient



Max-requests: Efficient, not fair



Client5 STARVES

Balances efficiency and fairness

Multi-way joins in PostgreSQL

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

VM: PostgreSQL

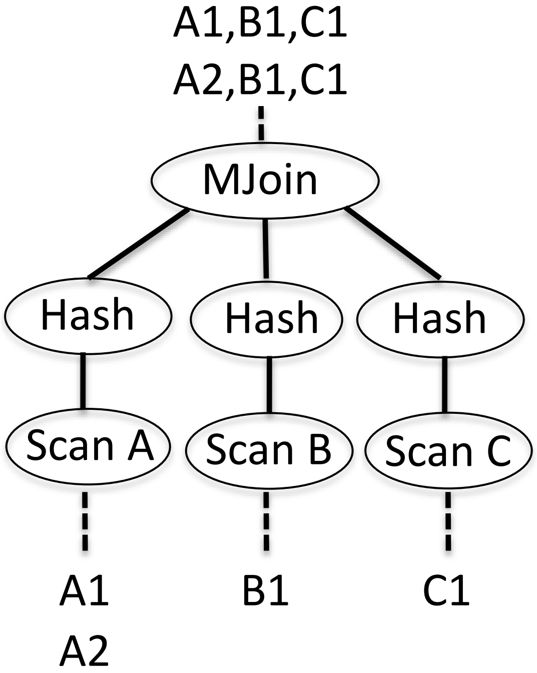
State Manager

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

Join Execution



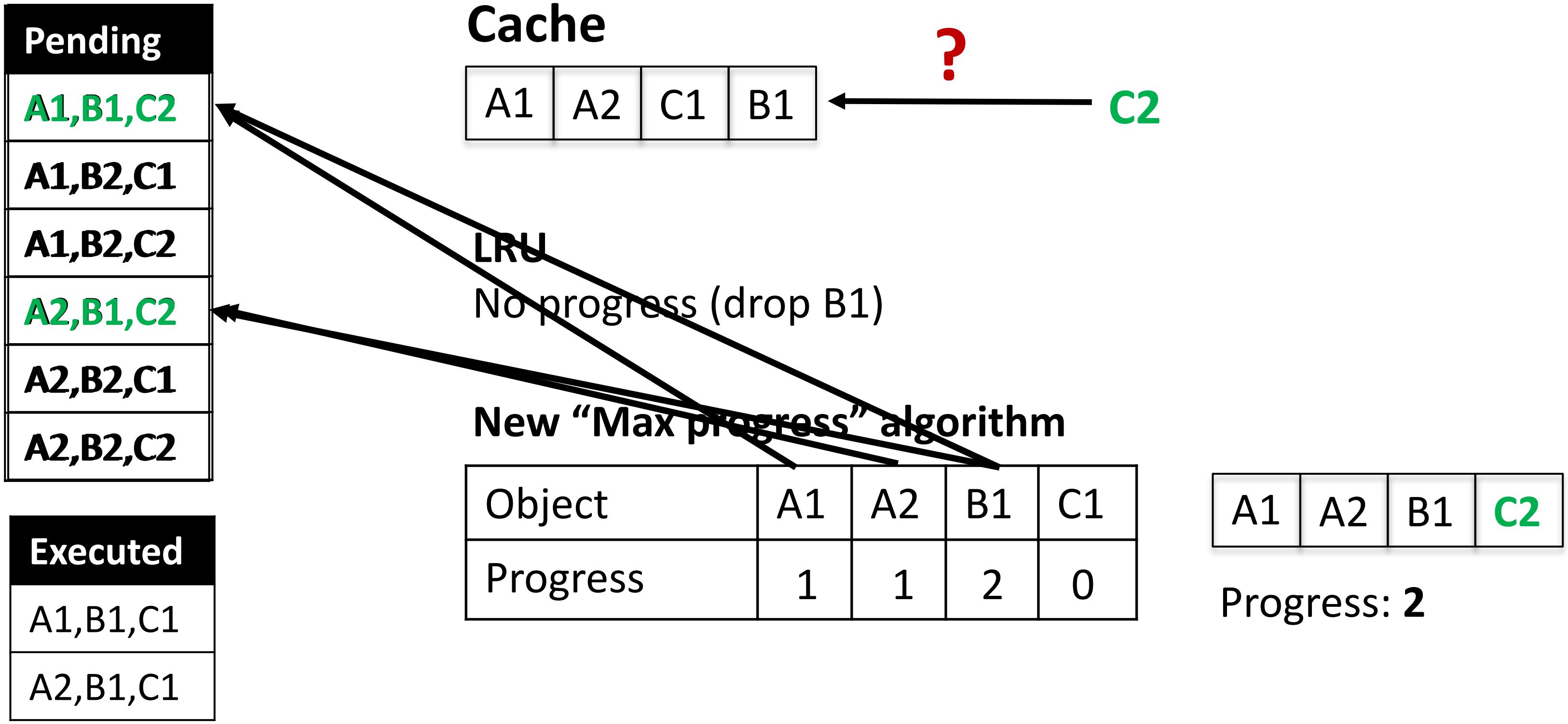
Cache Manager

A1	A2	C1	B1

Enable out-of-order opportunistic execution

Progress-driven caching

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



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

Executed
A1,B1,C1
A2,B1,C1

Cache

A1	A2	C1	B1

LRU

No progress (drop B1)

New "Max progress" algorithm

Object	A1	A2	B1	C1
Progress	1	1	2	0

A1	A2	B1	C2

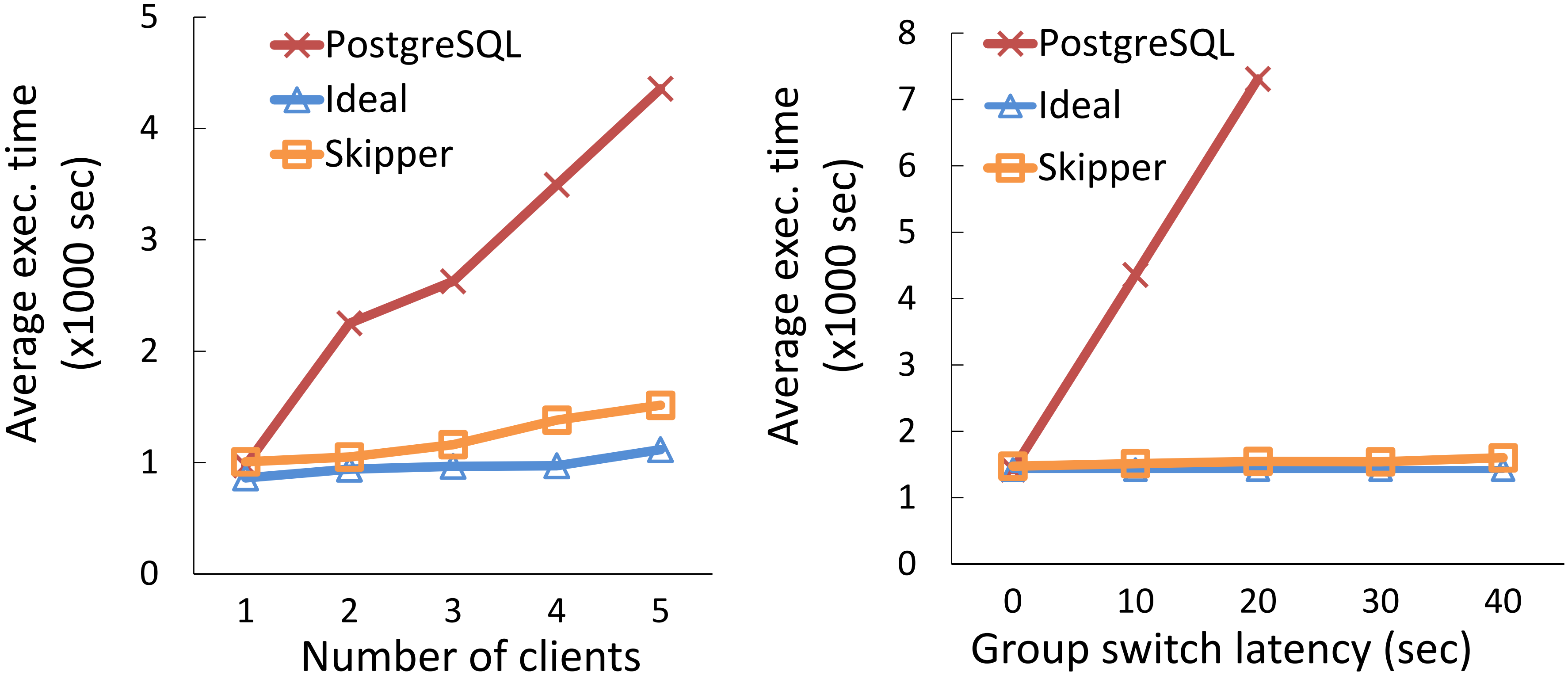
Progress: **2**

Minimizes data roundtrips, maximizes query progress

Skipper in action

[VLDB'16]

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

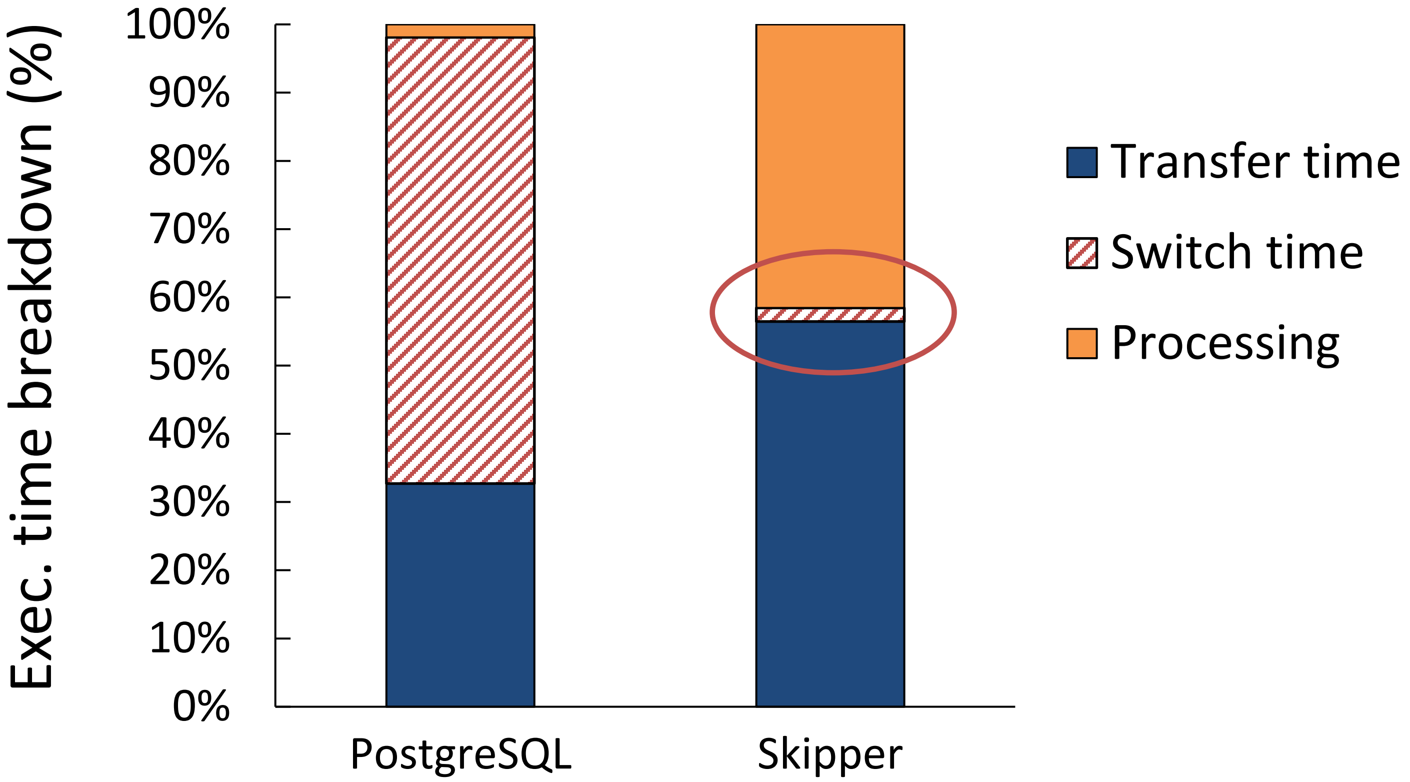


Skipper approximates HDD-based capacity by 20% avg.

Skipper is resilient to group switch latency

Minimizing group switches

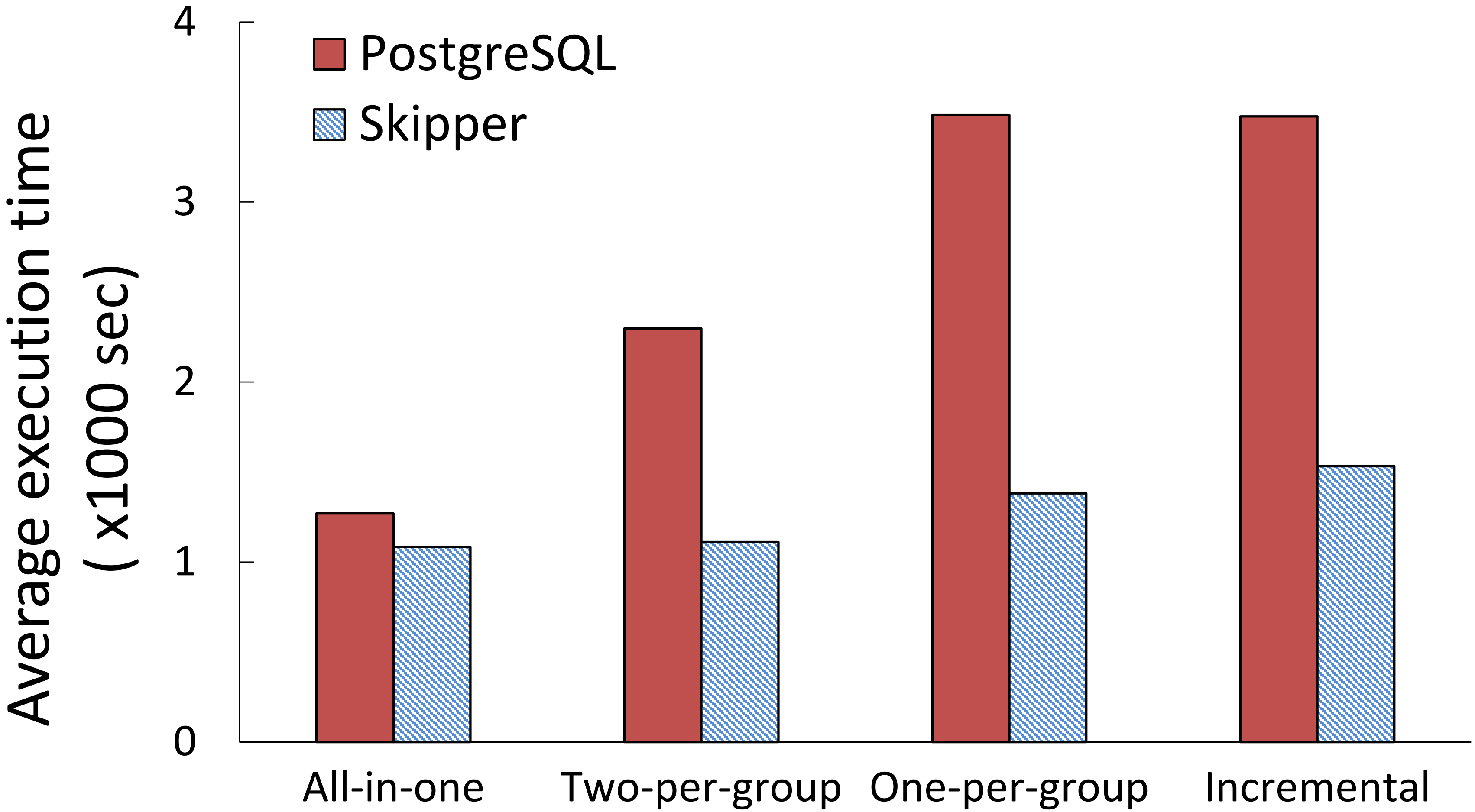
Setting: multitenant enterprise datacenter, 5 clients: TPCH 50, Q12, CSD: shared, layout: one client per group



Skipper substantially reduces overhead of group switches

Layout sensitivity

Setting: 5 clients TPCH 50, Q12, CSD: shared, vary layout (span from 1 to 4 groups)



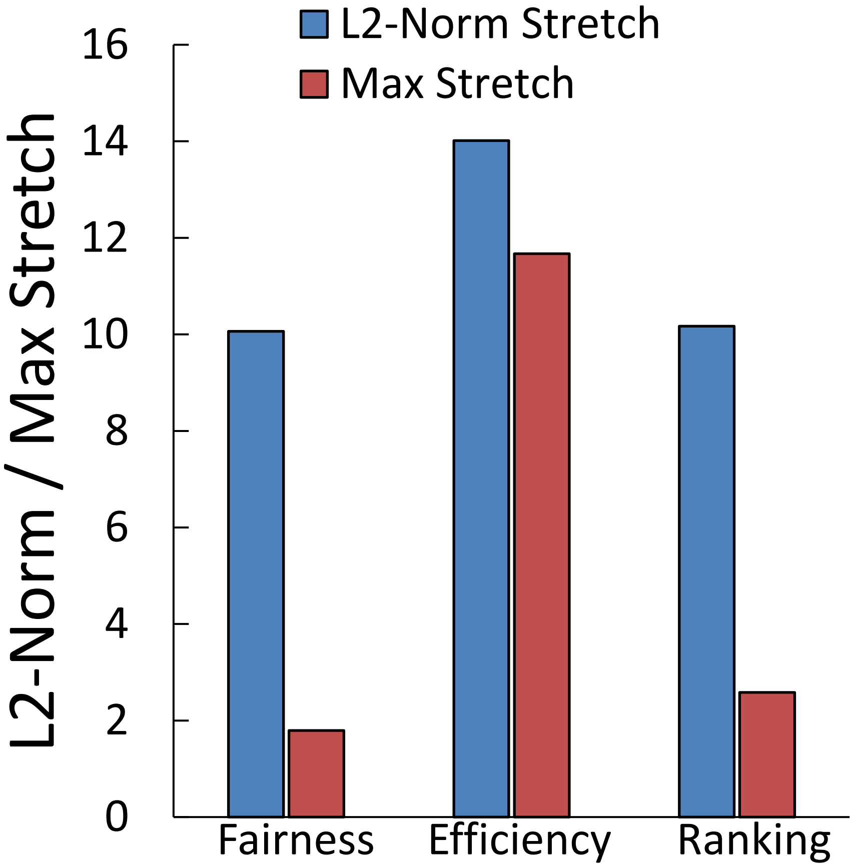
Skipper insensitive to CSD data layout

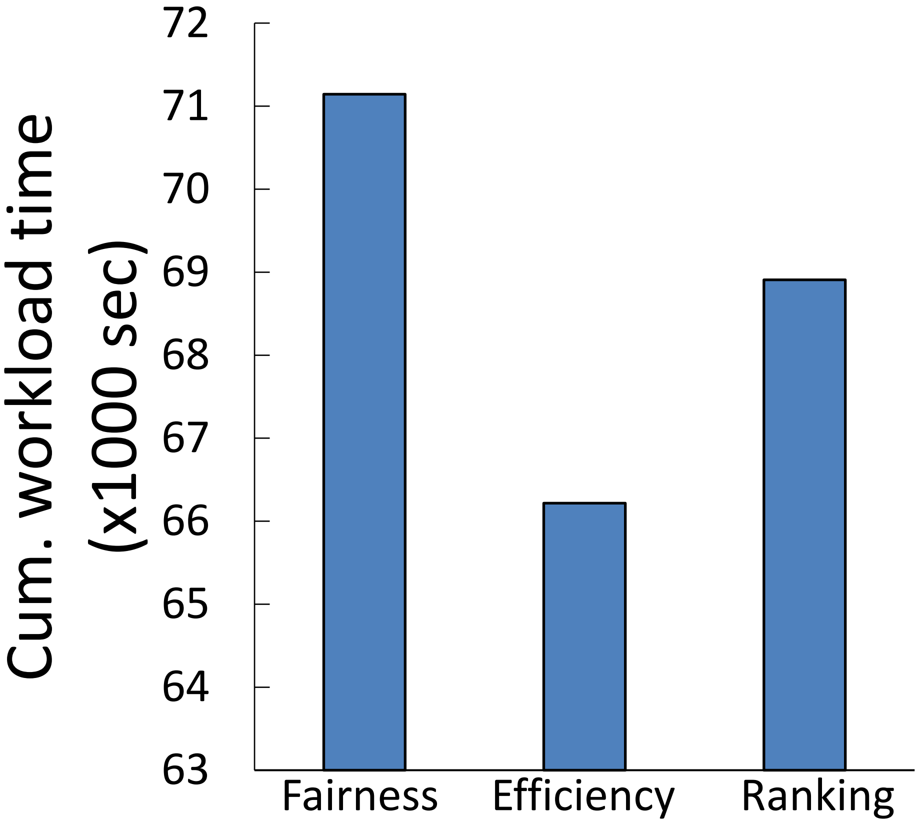
I/O Scheduling

Fairness vs efficiency…

Setting: 5 clients, each TPCH 50, Q12 x10, skewed layout: g1 and g2: 2 clients, g3: one client

$$stretch_i = observed\_time/ideal\_time\,; \quad l2 - norm = \sqrt{\sum_{i=1}^{n} stretch_i^2}$$





Ranking based I/O scheduling balances efficiency and fairness

Summary

- **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

What do HW trends tell us?

The five-minute rule thirty years later [CACM'19]

- Growing DRAM-HDD & shrinking DRAM-NVM intervals

Most performance critical data will sit in SSD/NVM

- Rapid improvements in SSD/NVM density

All randomly accessed data can sit in SSD/NVM

- Shrinking HDD—tape/CSD difference w.r.t $/TB scan

Can merge archival+capacity tier into cold storage tier

Sequential batch analytics can be hosted on new tier

Five-minute rule suggests impending consolidation in the storage hierarchy

Where to go from here

"It is not the strongest species that survive, nor the most intelligent, but the ones most responsive to change." Charles Darwin

Queries
[SIGMOD'12]
[VLDB'12]
[CACM'15]
[ICDE'21]
[ICDM'21]

Data
[DBTest'12]
[ICDE'15]
[VLDBJ'18]
[ADC'20]

Hardware
[VLDB'16]
[ADMS'17]
[CACM'19]

Fast response

DBMS System

Thank you!

Adaptive DBMSs for efficient data analysis

Questions?

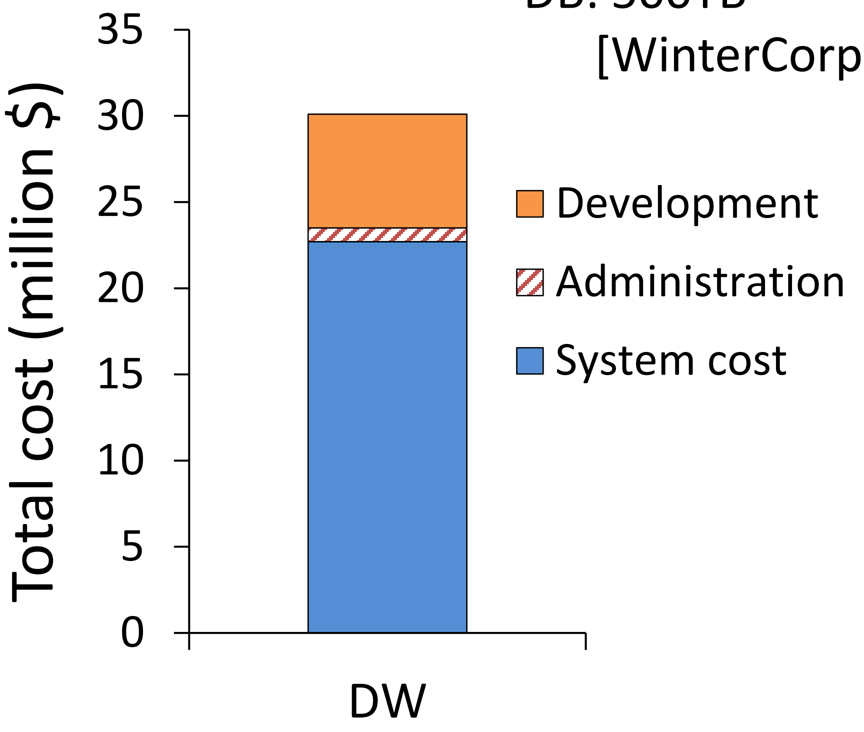
THANK YOU

Analyzing data is expensive

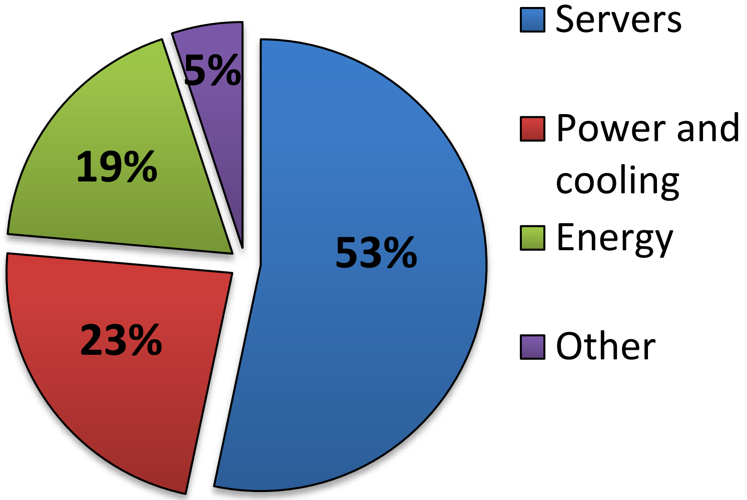
“Most firms estimate that they are only analyzing 12% of the data that they already have” [Forrester, 2014]

DB: 500TB

[WinterCorp, 2013]

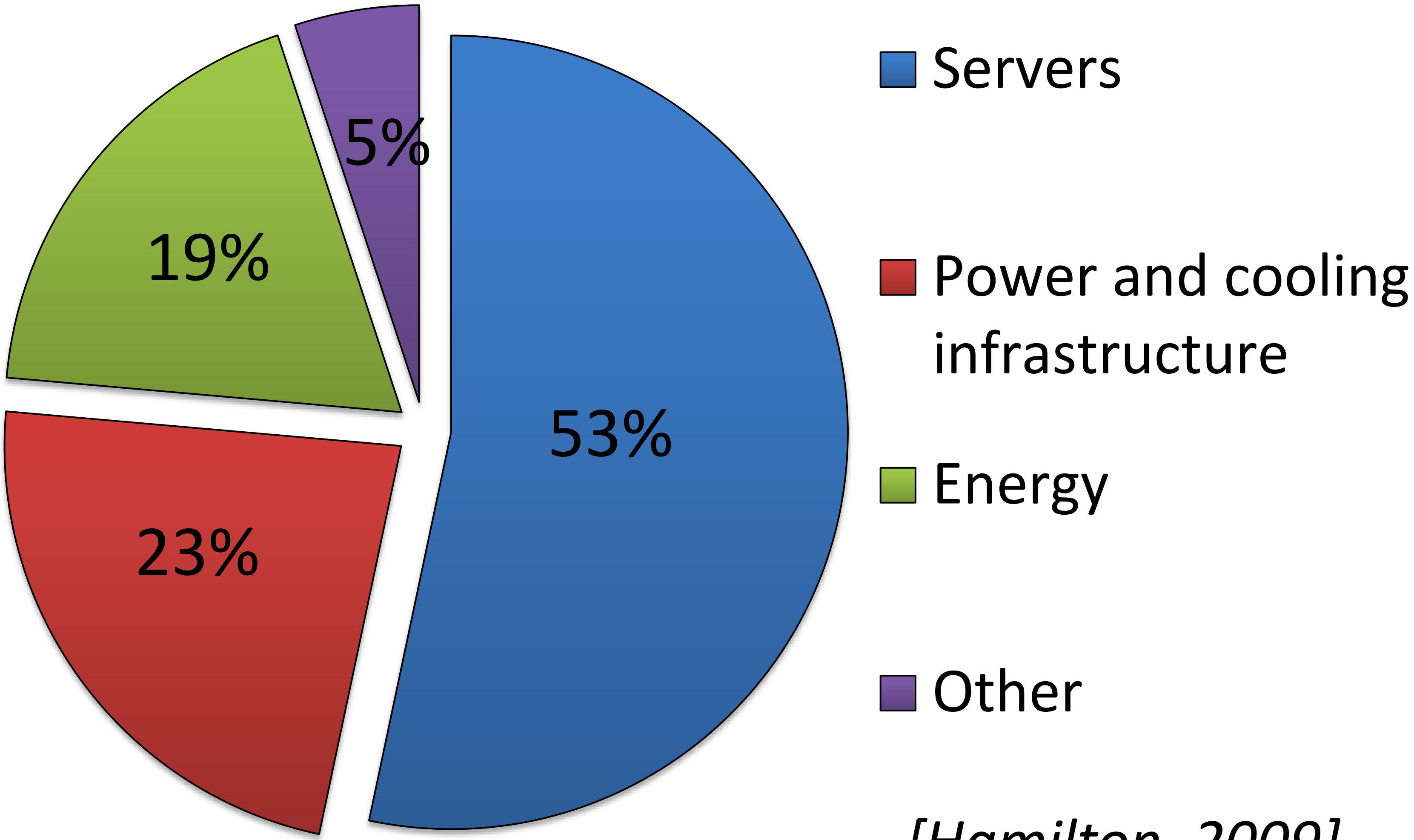


[Hamilton, 2009]



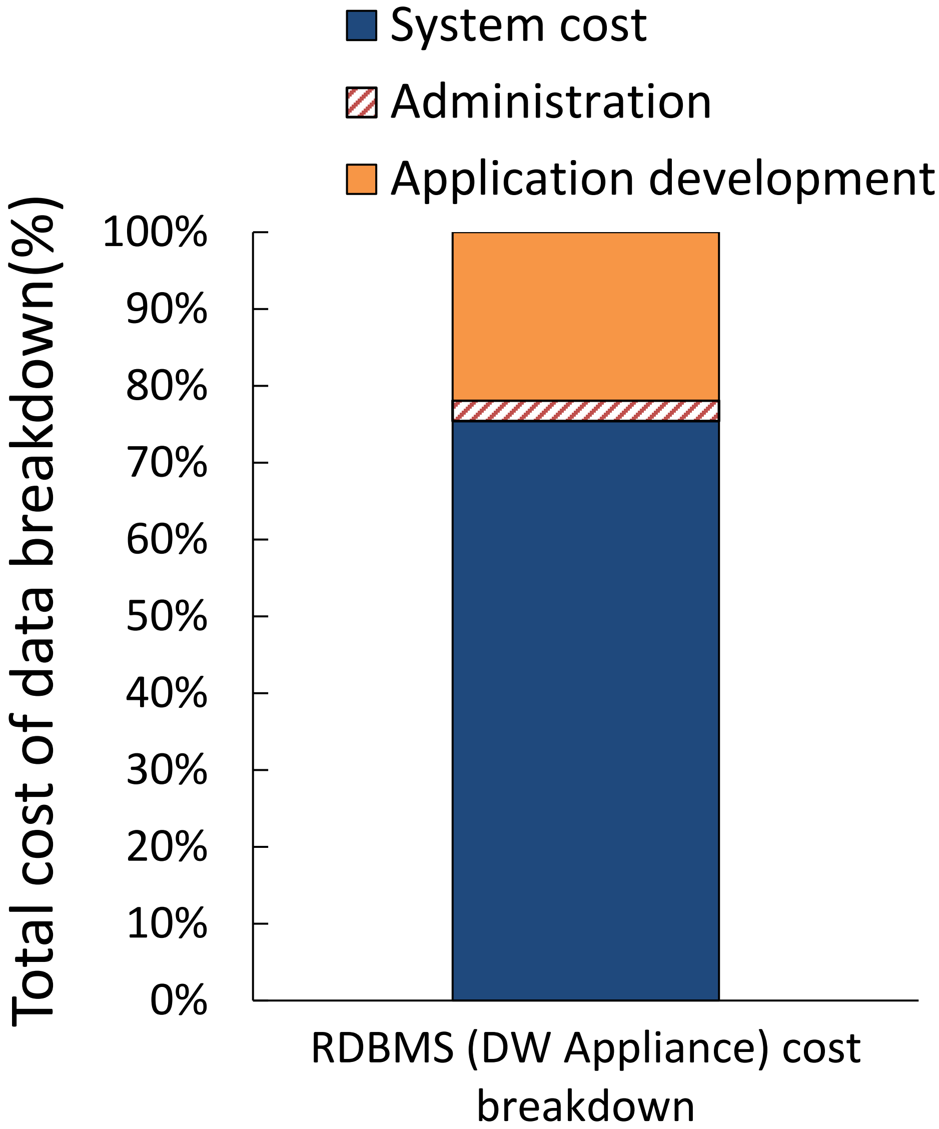
Infrastructure expensive for rarely accessed data

Monthly cost of a data center



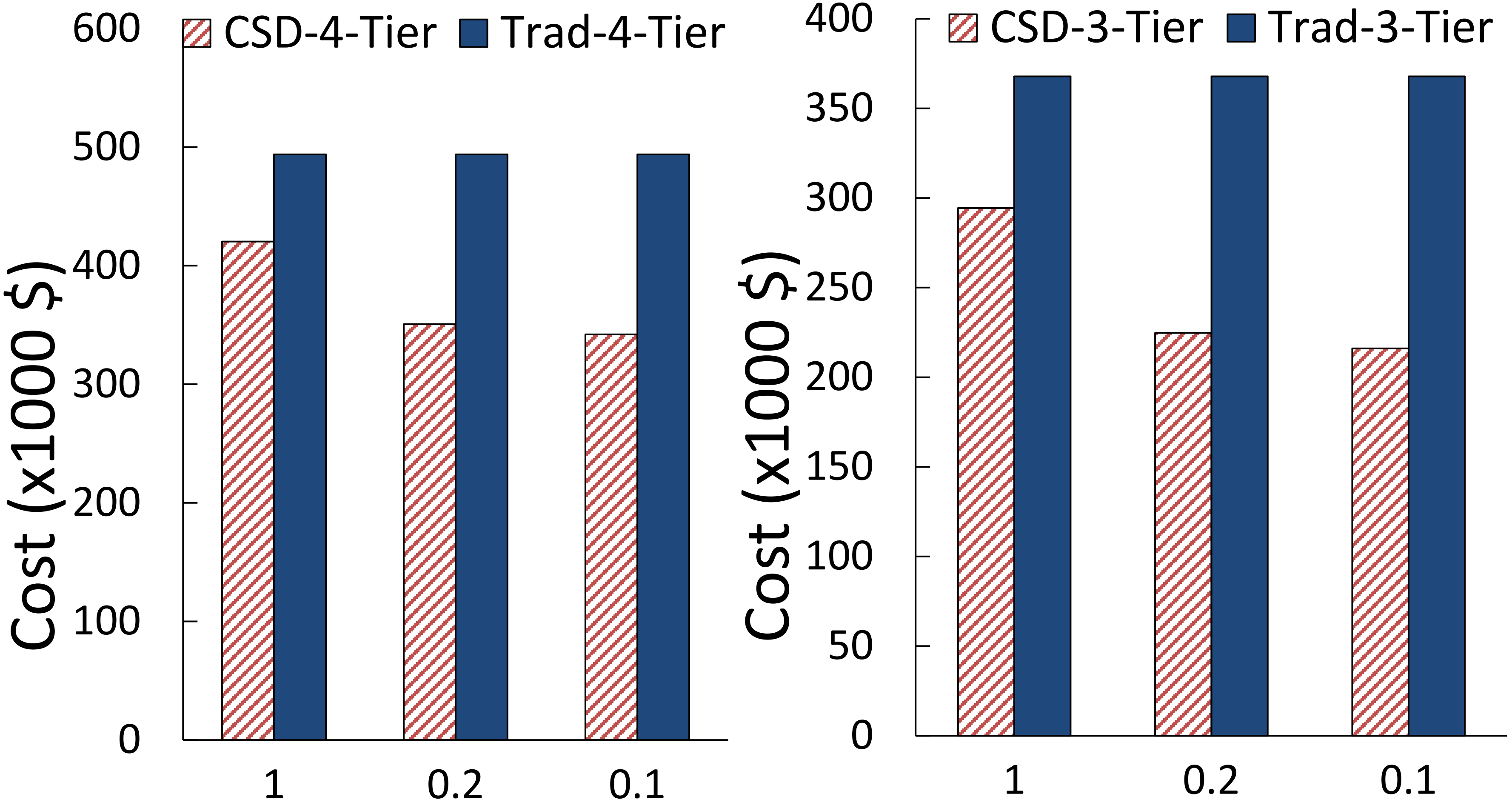
[Hamilton, 2009]

5 year TCOD for a data warehouse



Cost benefit of CSD

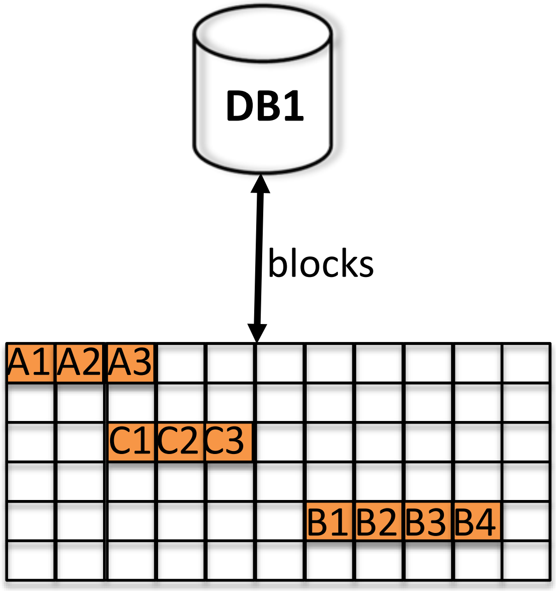
Setting: Horison, 100TB, 3 and 4-tier vs. CSD as capacity and archival



Substantial cost savings with CSD

Query execution over CSD

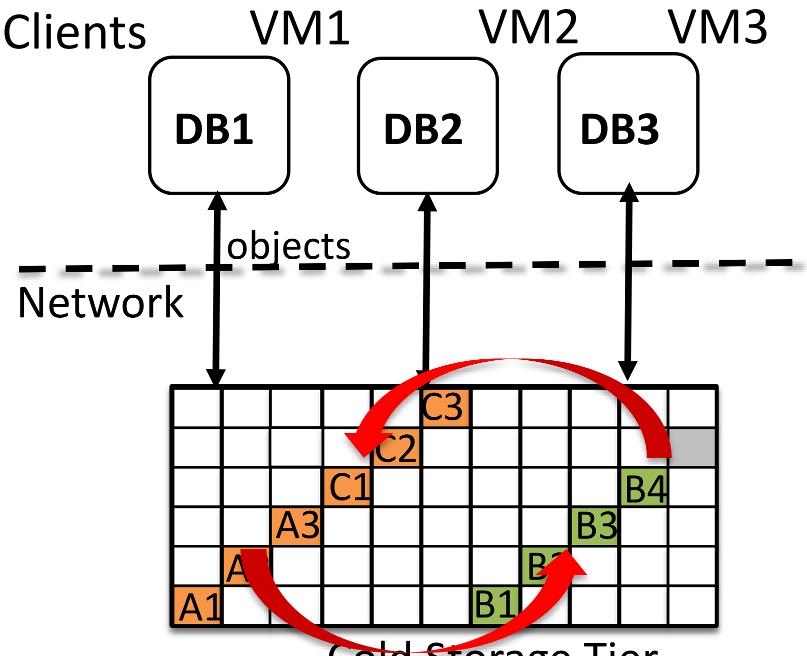
Traditional setting



HDD-Based Capacity Tier

- **Uniform access**
- **Control layout**
- **Static (pull-based) execution**

Virtualized enterprise data center



Cold Storage Tier

- **Non-uniform access**
- **No control over layout**

Pull-based execution triggers excessive group switches

Rank-based scheduling

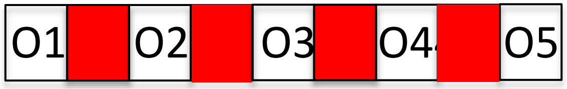
Which group to switch to ?

Group	Table objects
G1	O1 (Client1), O3 (Client3)
G2	O2 (Client2), O4 (Client4)
G3	O5 (Client5)

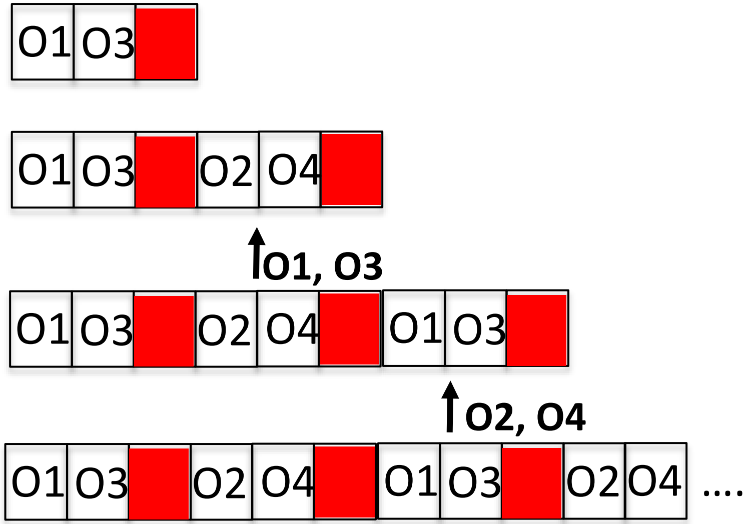
O1, O2, O3, O4, O5

TIME

FCFS – Fair, but inefficient



Max-requests: Efficient, not fair

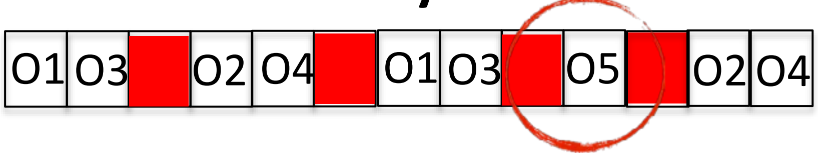


Client5 STARVES

Rank-based scheduling

Rank(G) = #Requests + ∑Wait

Provides efficiency **Provides fairness**



Balances efficiency and fairness

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

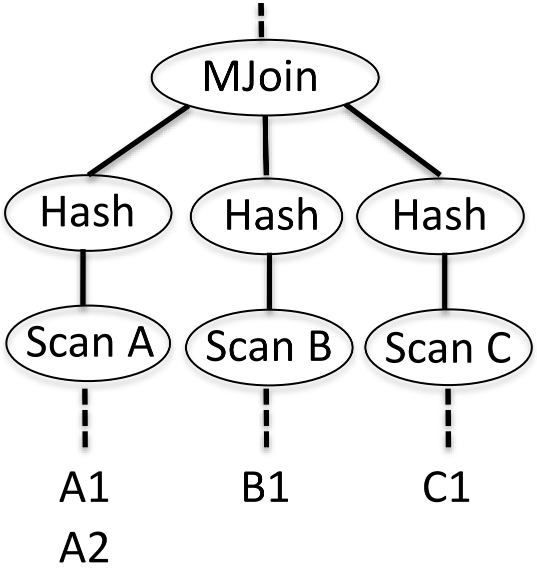
Executed
A1,B1,C1
A2,B1,C1

Cache Manager

A1	A2	C1	B1

Join Execution

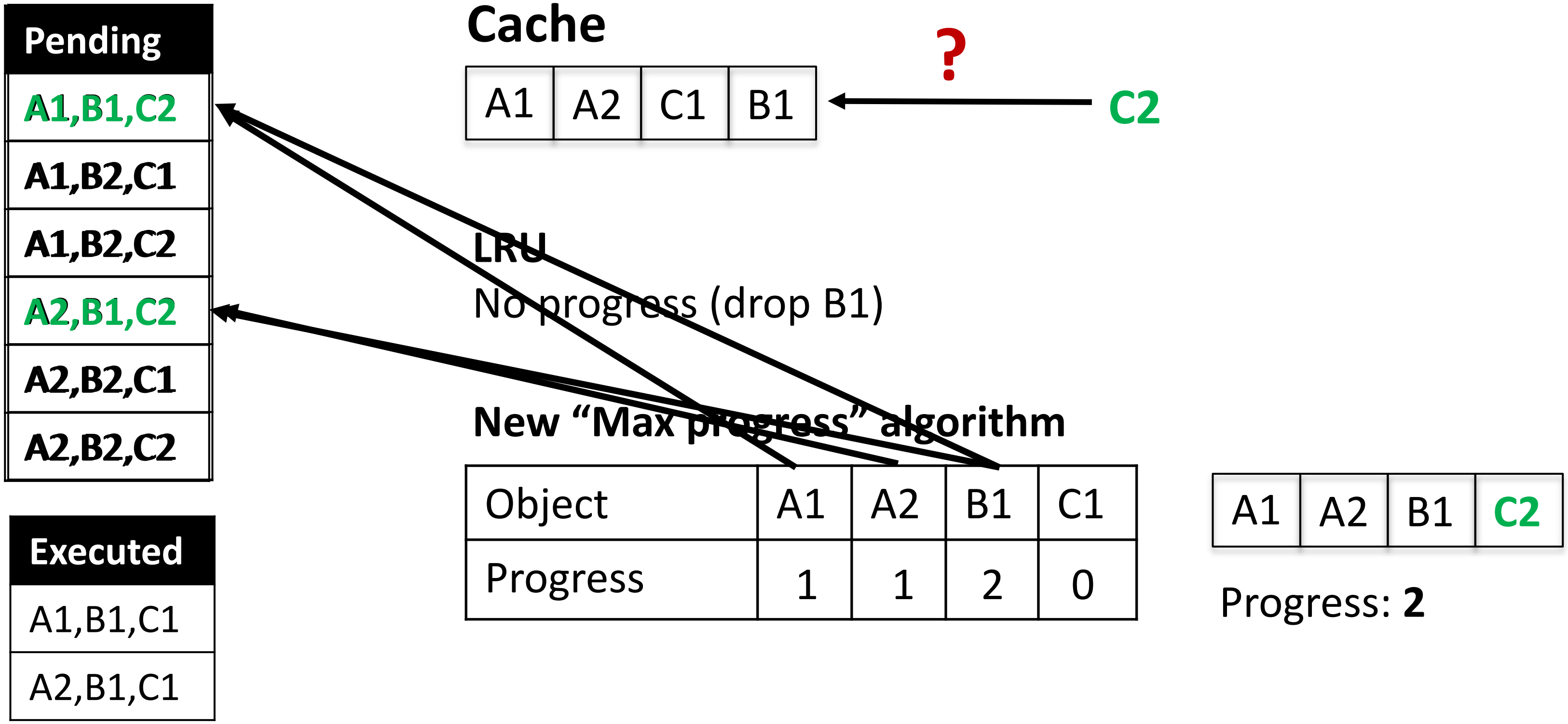
A1,B1,C1
A2,B1,C1



Enable out-of-order opportunistic execution

Progress driven caching

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



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

Executed
A1,B1,C1
A2,B1,C1

Cache

A1	A2	C1	B1

LRU

No progress (drop B1)

New "Max progress" algorithm

Object	A1	A2	B1	C1
Progress	1	1	2	0

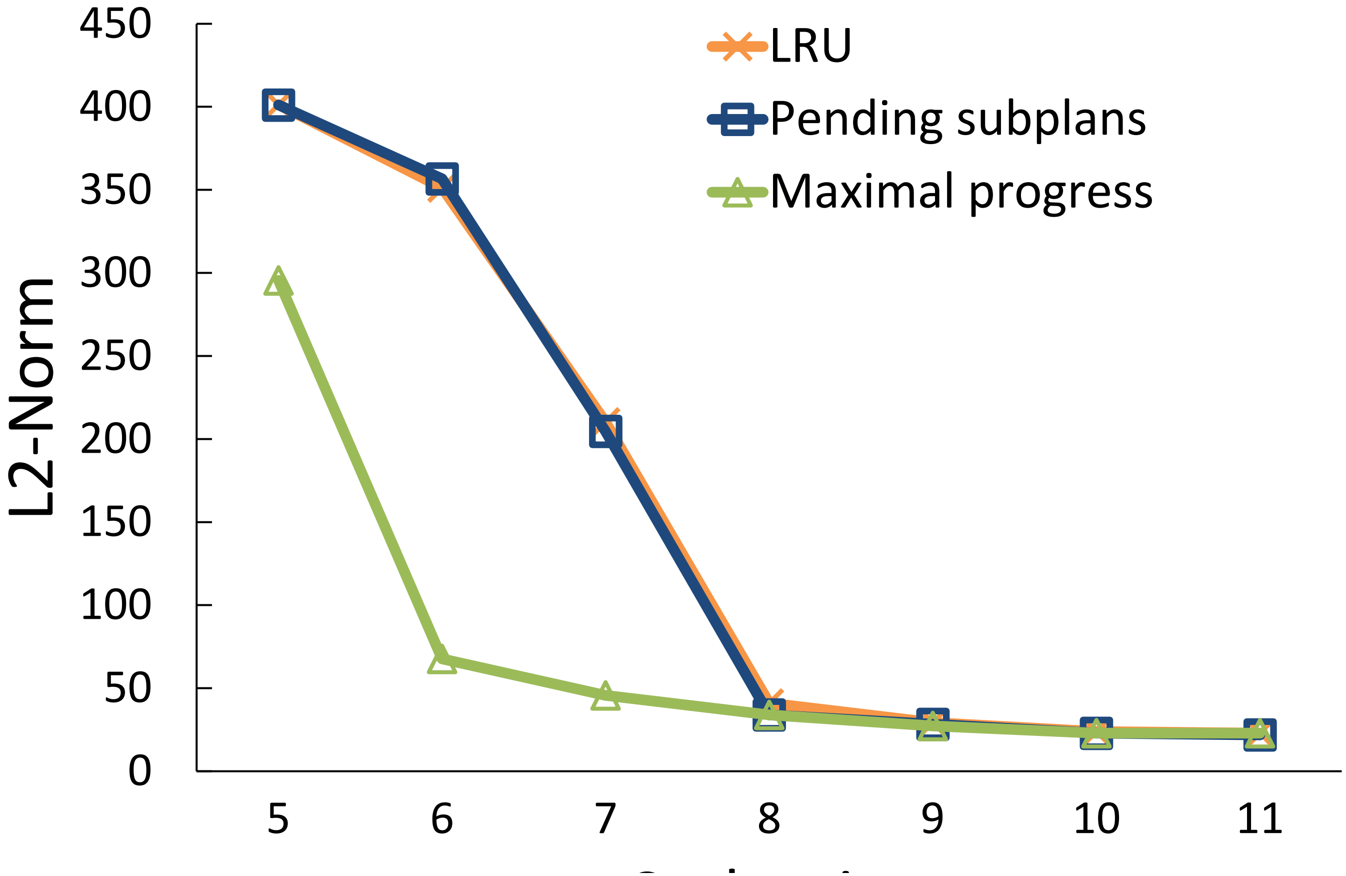
A1	A2	B1	C2

Progress: **2**

Minimizes data roundtrips, maximizes query progress

Caching algorithms

Setting: 10 clients, 20 tables each 1-5GB, 2-5 table joins
CSD: shared, layout: one client per group

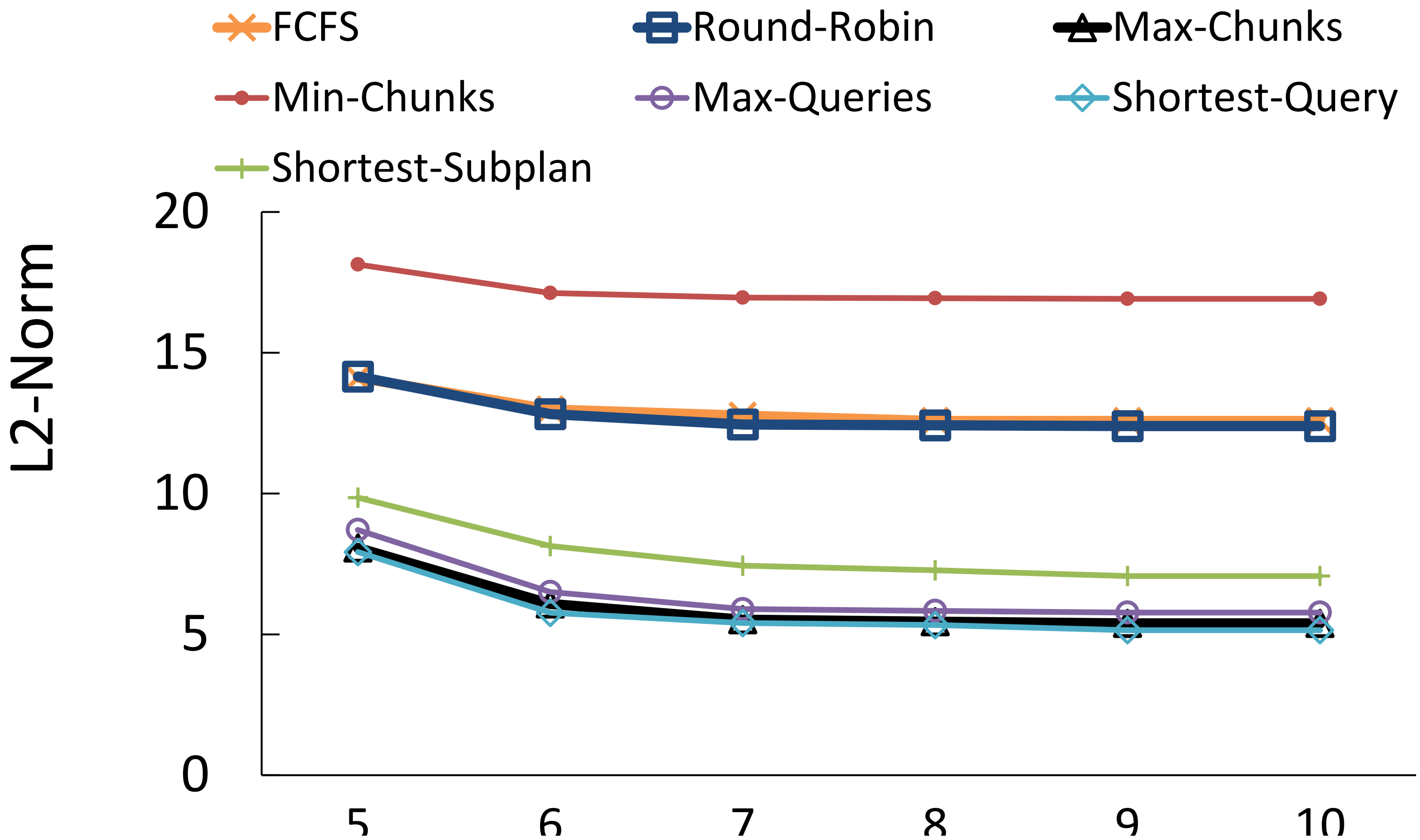


Maximal progress minimizes request reissue

Maximum efficiency algorithms

Setting: 10 clients, 20 tables each 1-5GB, 2-5 table joins

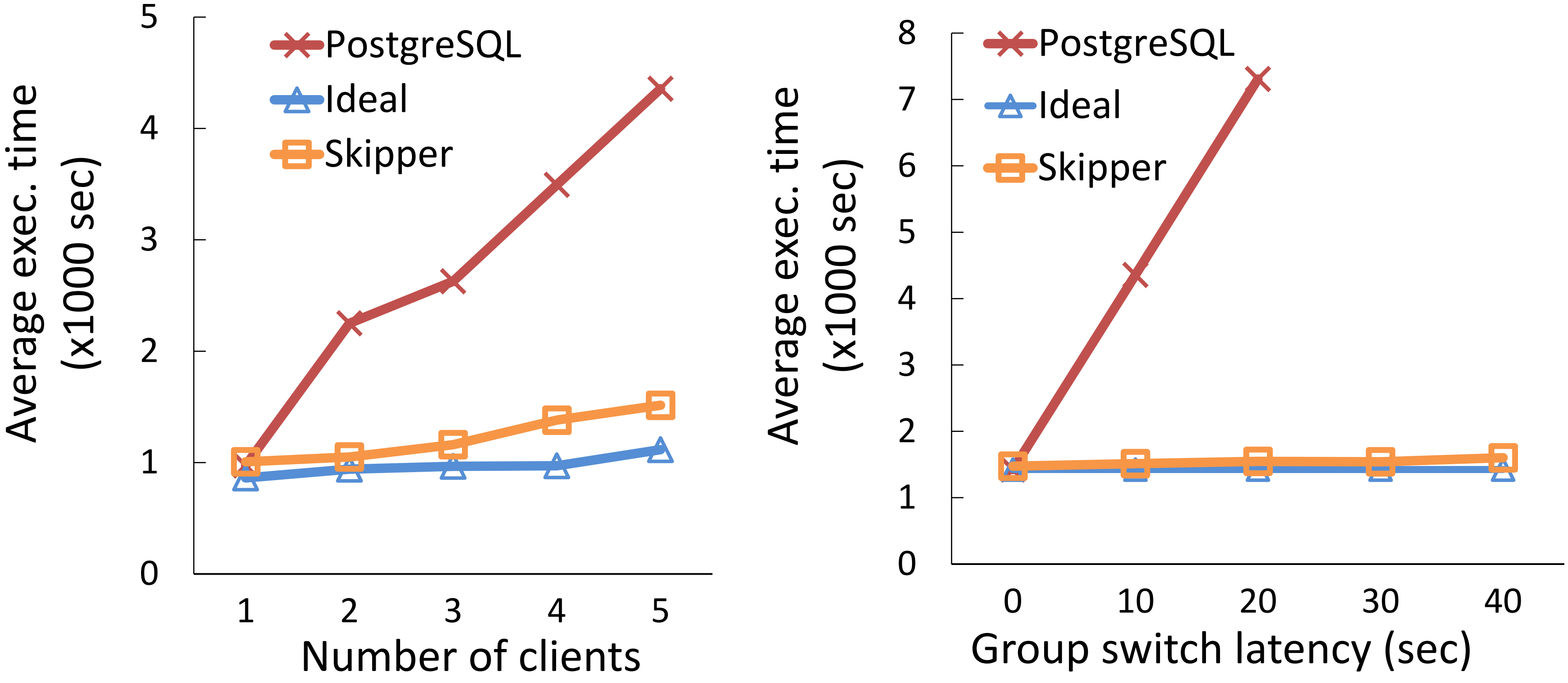
CSD: shared, layout: random per object



Max. queries in 20% of optimal in all layouts

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

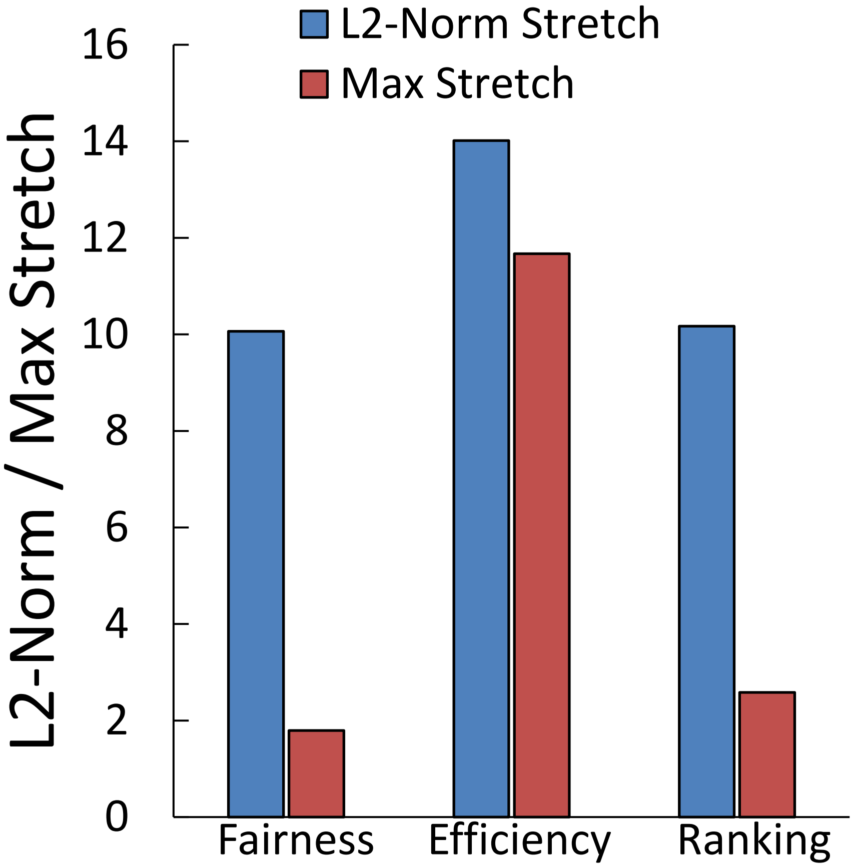
Skipper is resilient to group switch latency

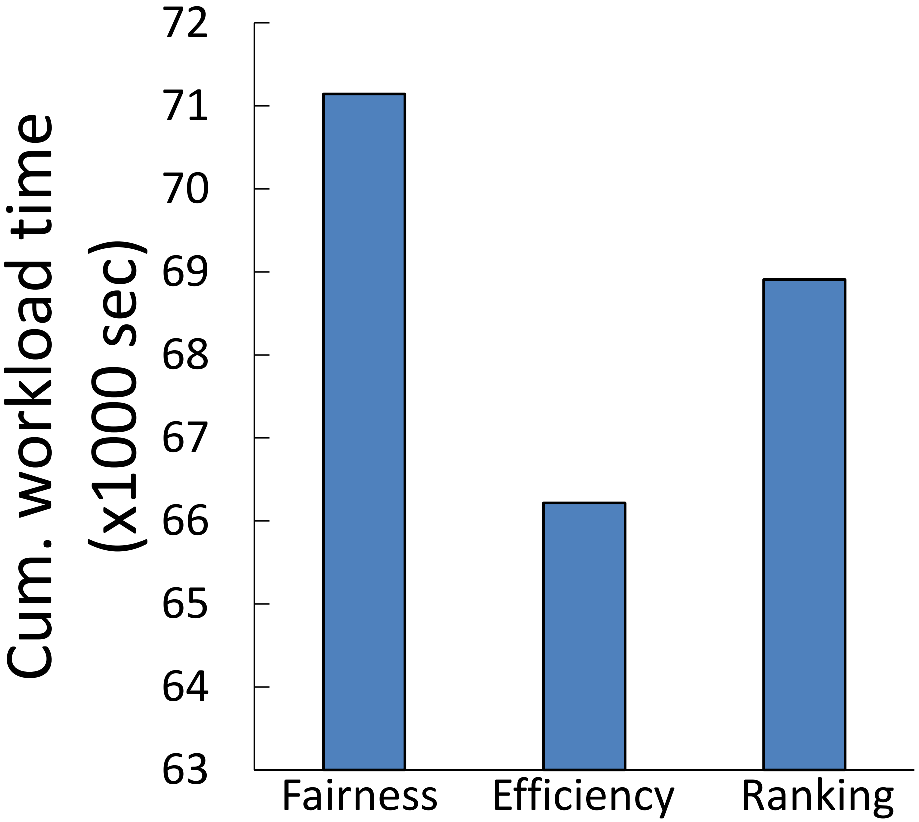
I/O Scheduling

Fairness vs efficiency…

Setting: 5 clients, each TPCH 50, Q12 x10, skewed layout: g1 and g2: 2 clients, g3: one client

$$stretch_i = observed\_time / ideal\_time \, ; \quad l2 - norm = \sqrt{\sum_{i=1}^{n} stretch_i^2}$$

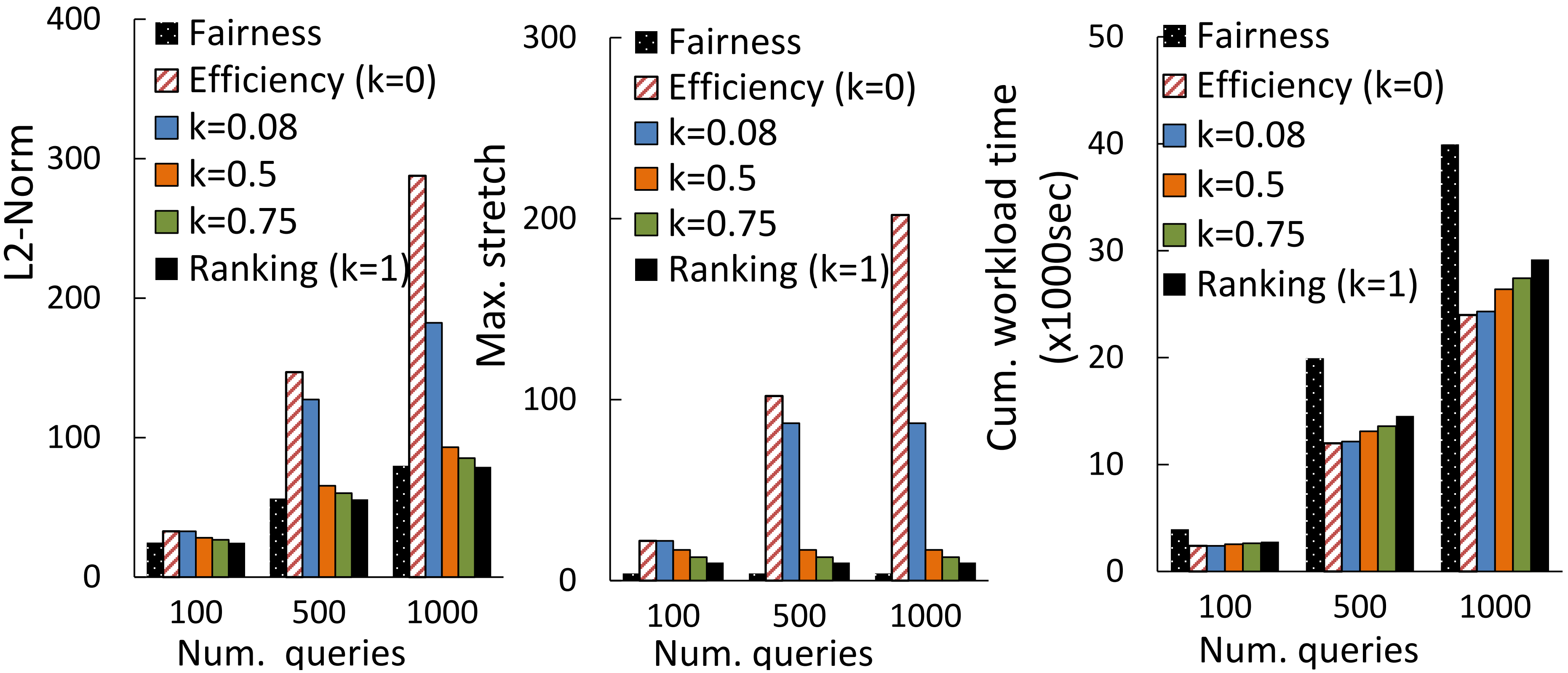




Rank-based I/O scheduling balances efficiency and fairness

Simulator: K-parameter variation

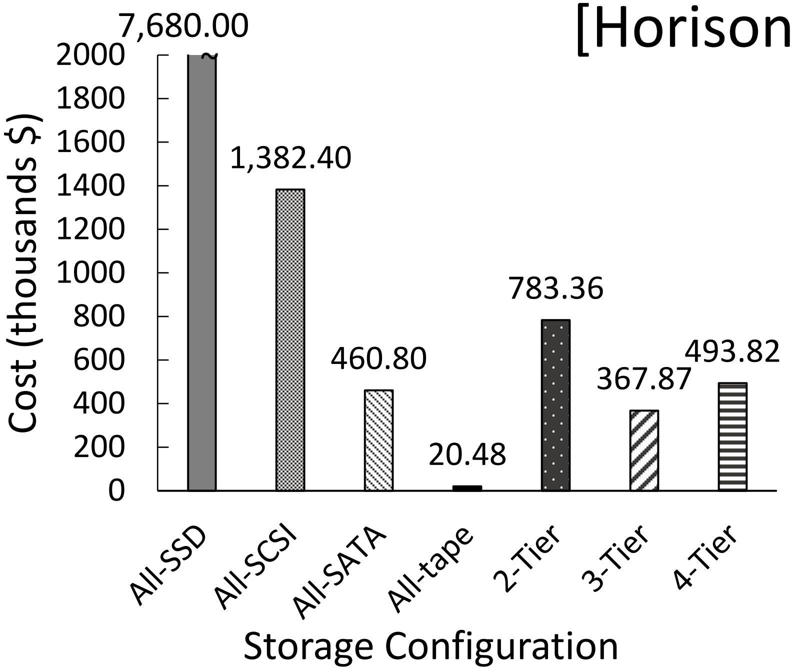
Setting: 10 clients, 20 tables 1-5GB, 2-5 table joins, 10-100 queries per client
CSD: shared, layout: power-law 80% clients in 20% groups



K in (0,1) performance between efficiency and fairness

Cold Storage in the storage tiering

[Horison, 2015]



CSD cost $/GB	Total cost k$/100TB	Savings k$
0.01	334.182	159.641
0.1	342.016	151.808
0.2	350.72	143.104
1	420.352	73.472