Adaptive Replication and Partitioning in Data Systems

Data Systems Group

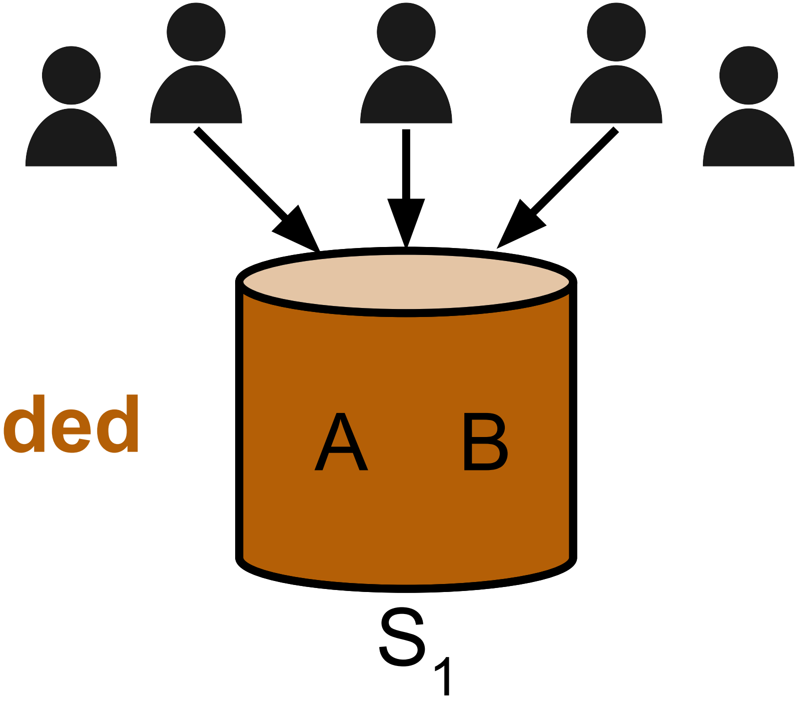
UNIVERSITY OF WATERLOO

Brad Glasbergen,
Michael Abebe,
Khuzaima Daudjee

Middleware 2018

BARK.AI

Single Node Architecture



Overloaded

Single Node Architecture

How to scale beyond a single node?

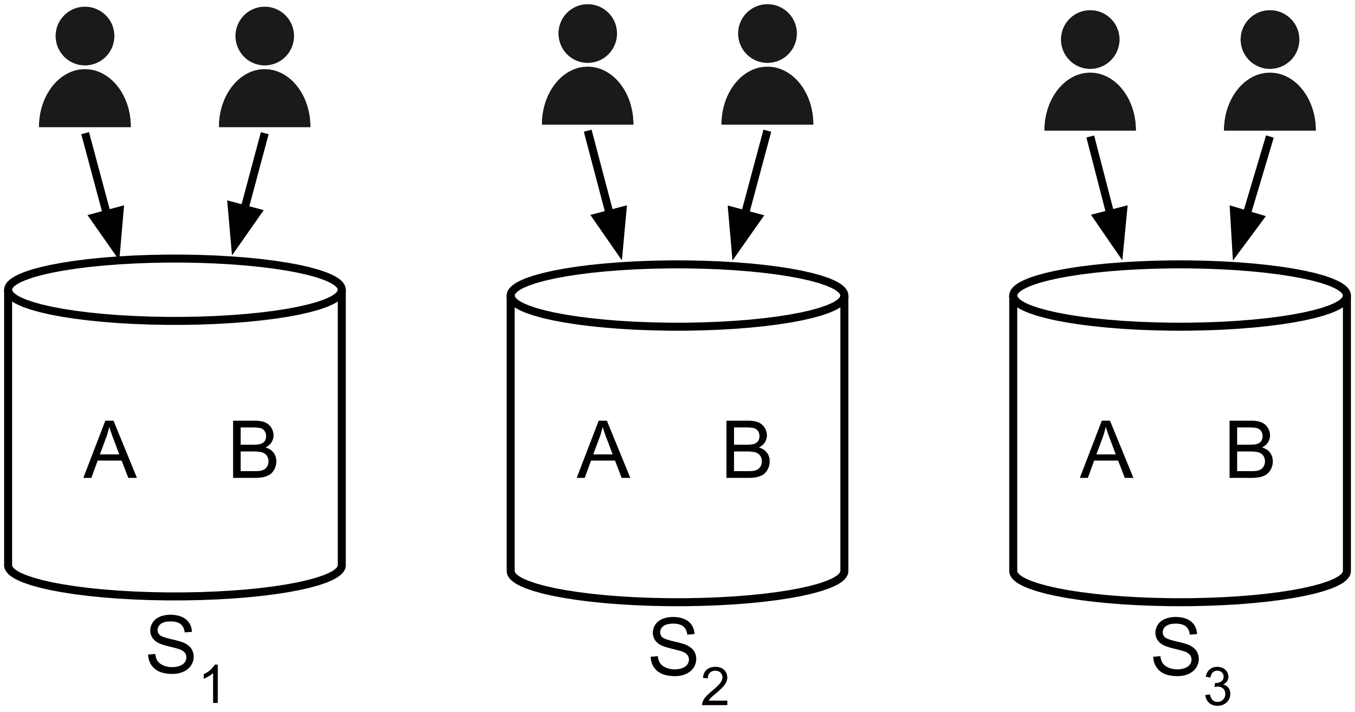
Replicate and partition

Single Node Architecture

How to scale beyond a single node?

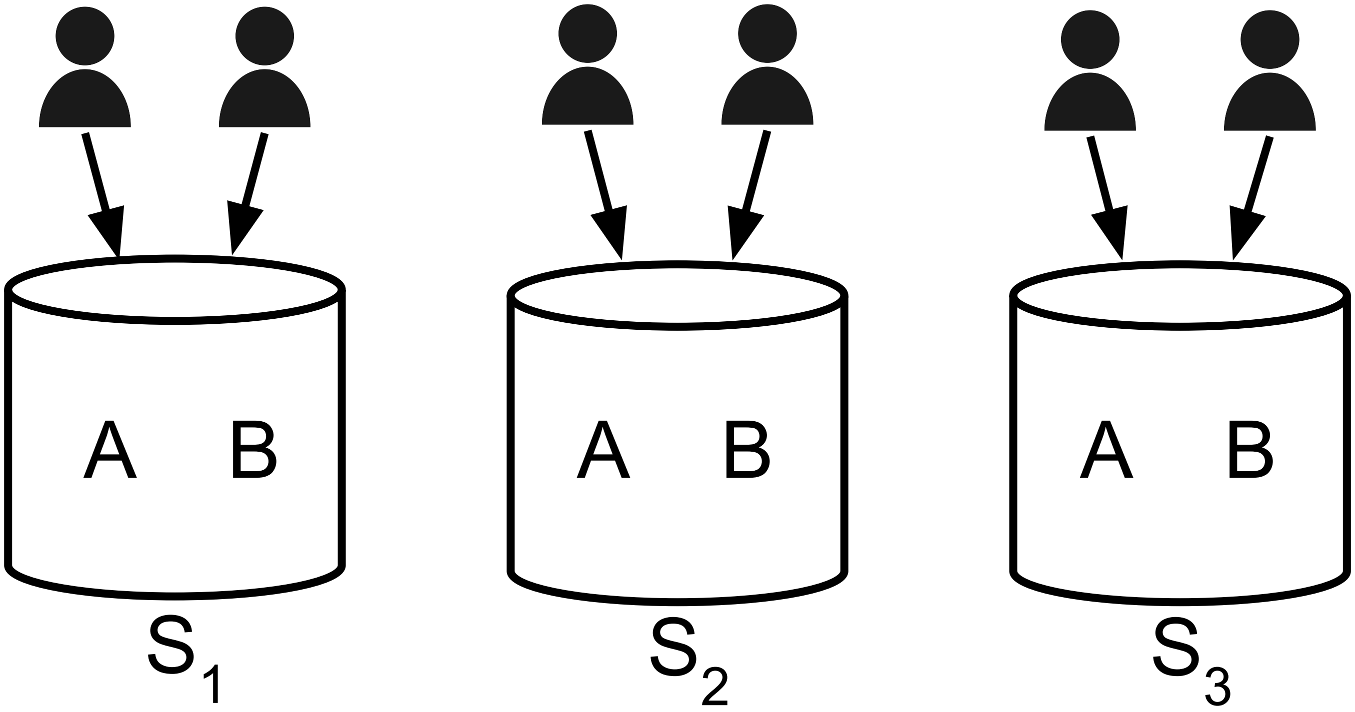
Replicate and partition

Replicated Architecture



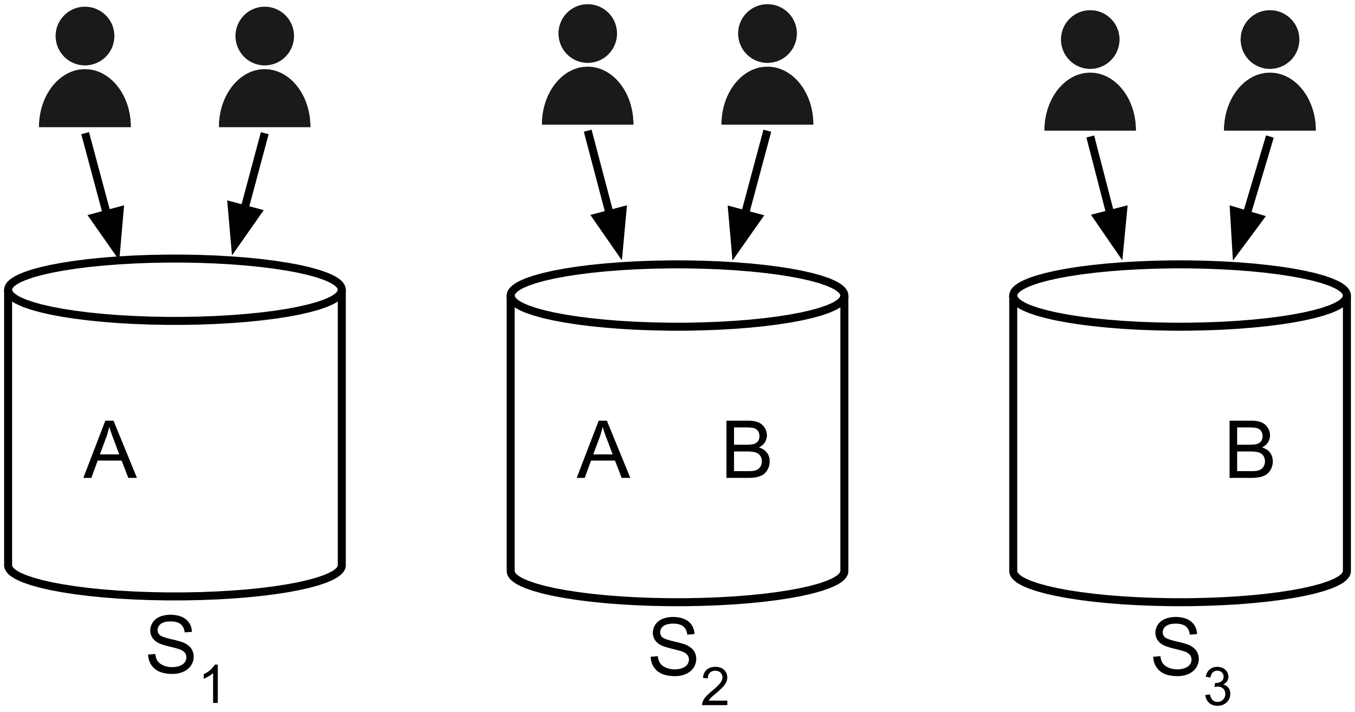
Handle more requests

Replicated Architecture



Cost of coordination

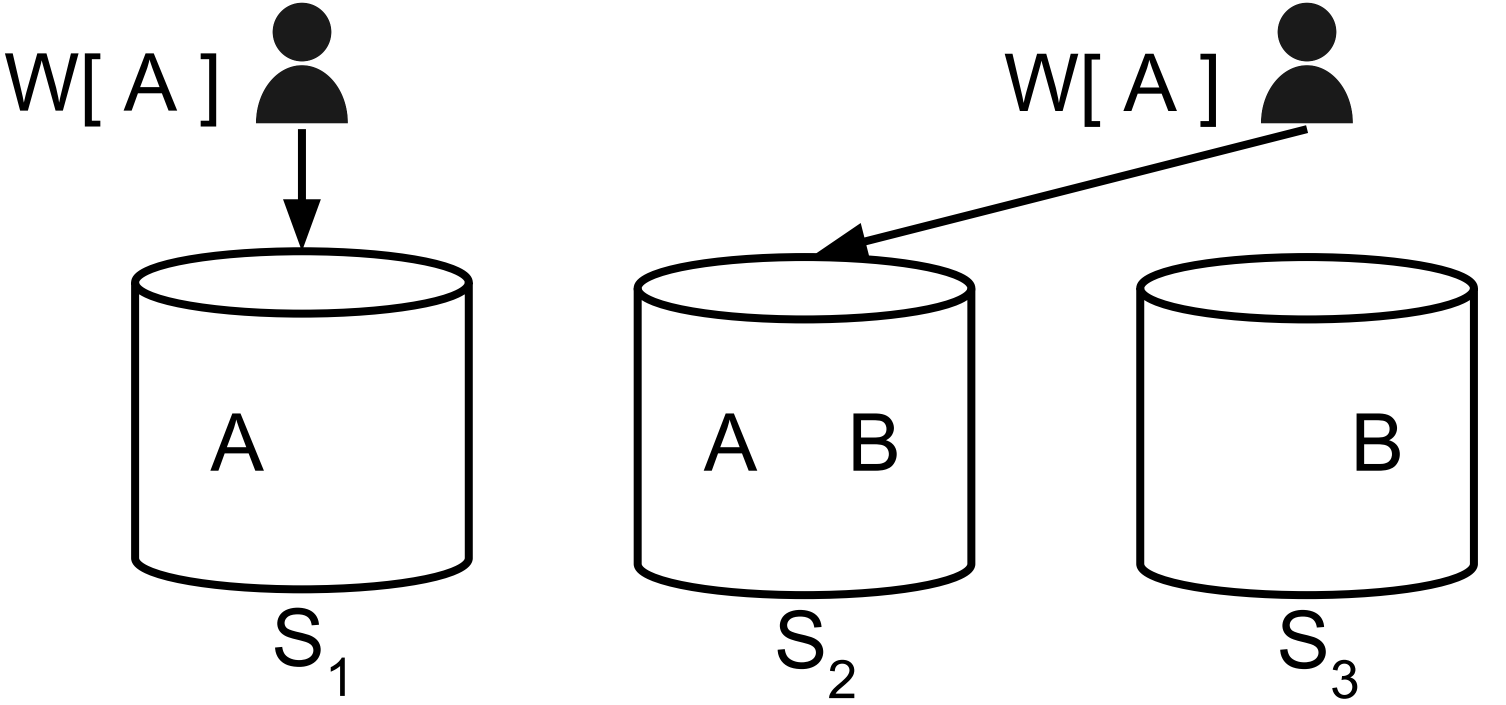
Replicated Architecture



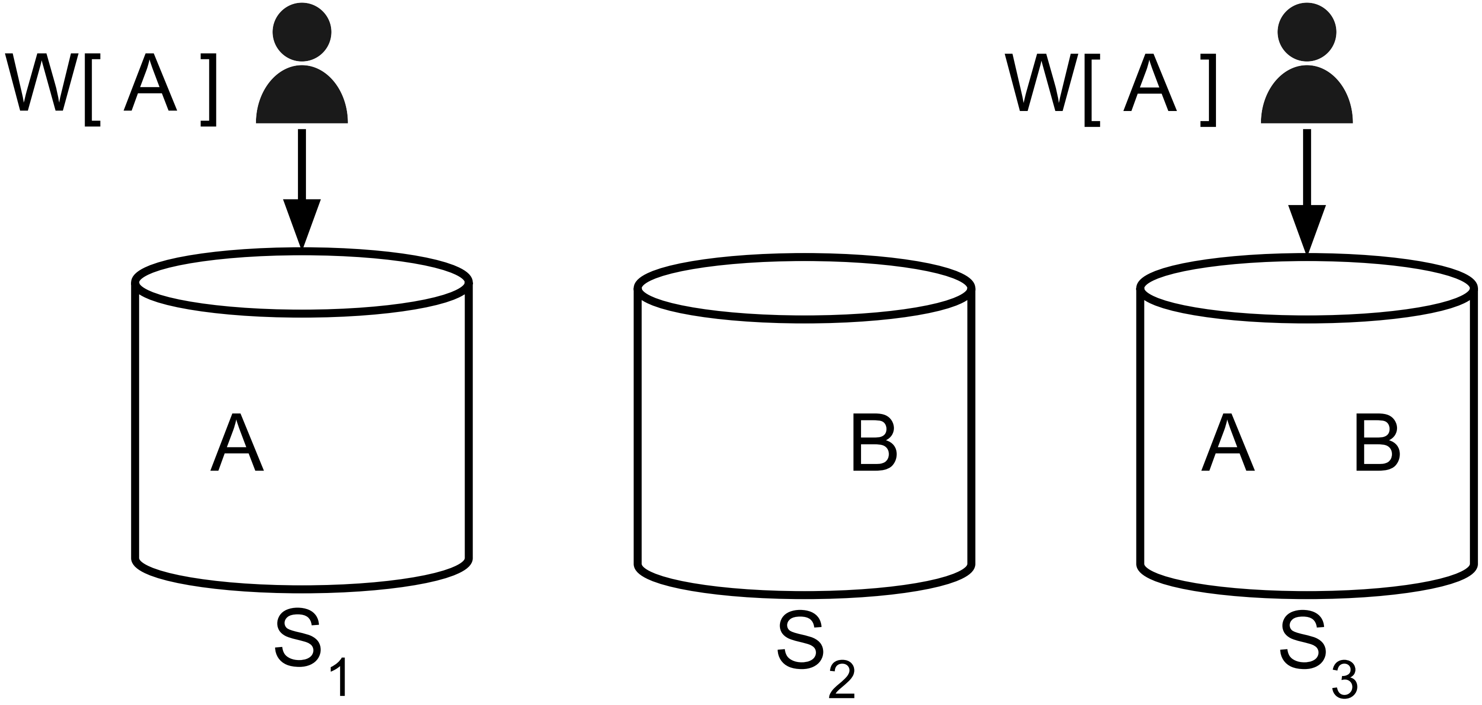
How **many** replicas?

UNIVERSITY OF WATERLOO

Replicated Architecture

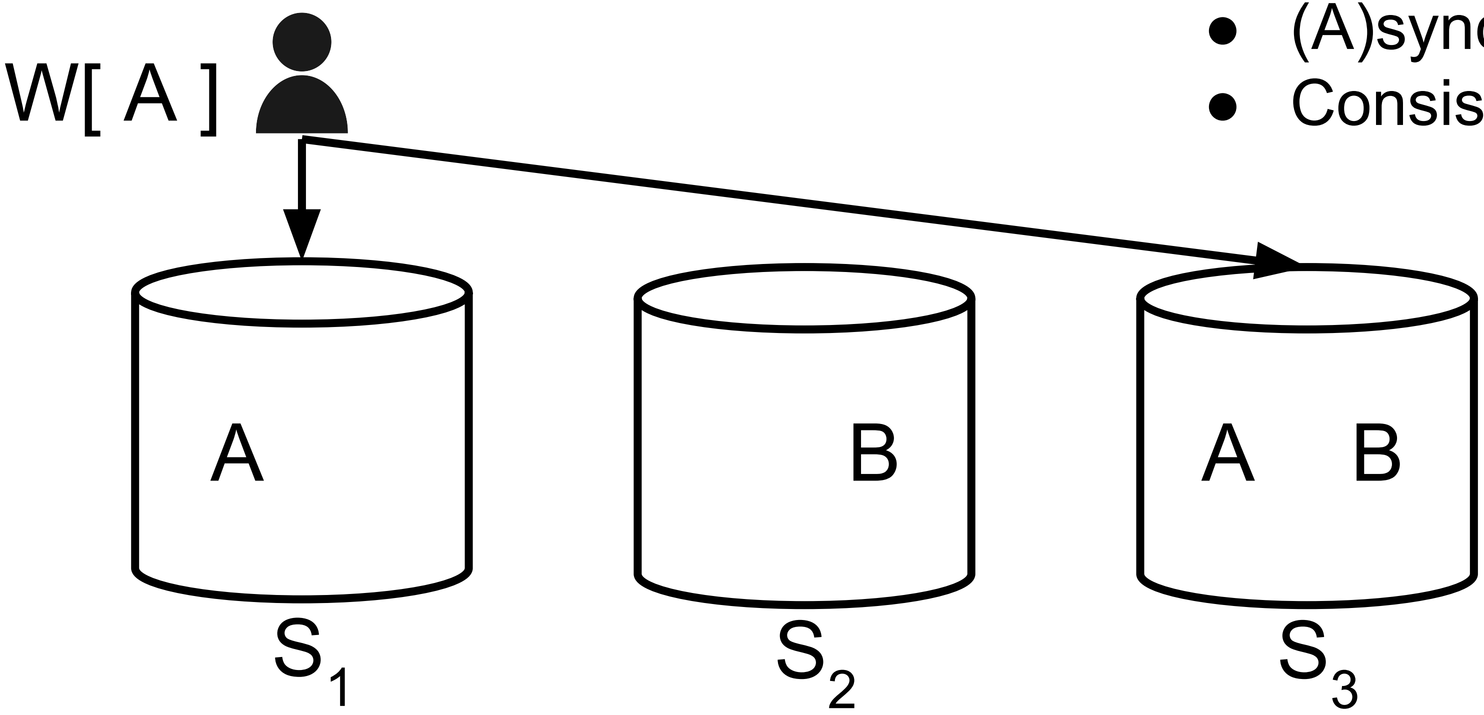


Replicated Architecture



Where to **place** replicas?

Replicated Architecture



- (A)synchronous
- Consistency

How to **propagate** updates?

Replication Decisions

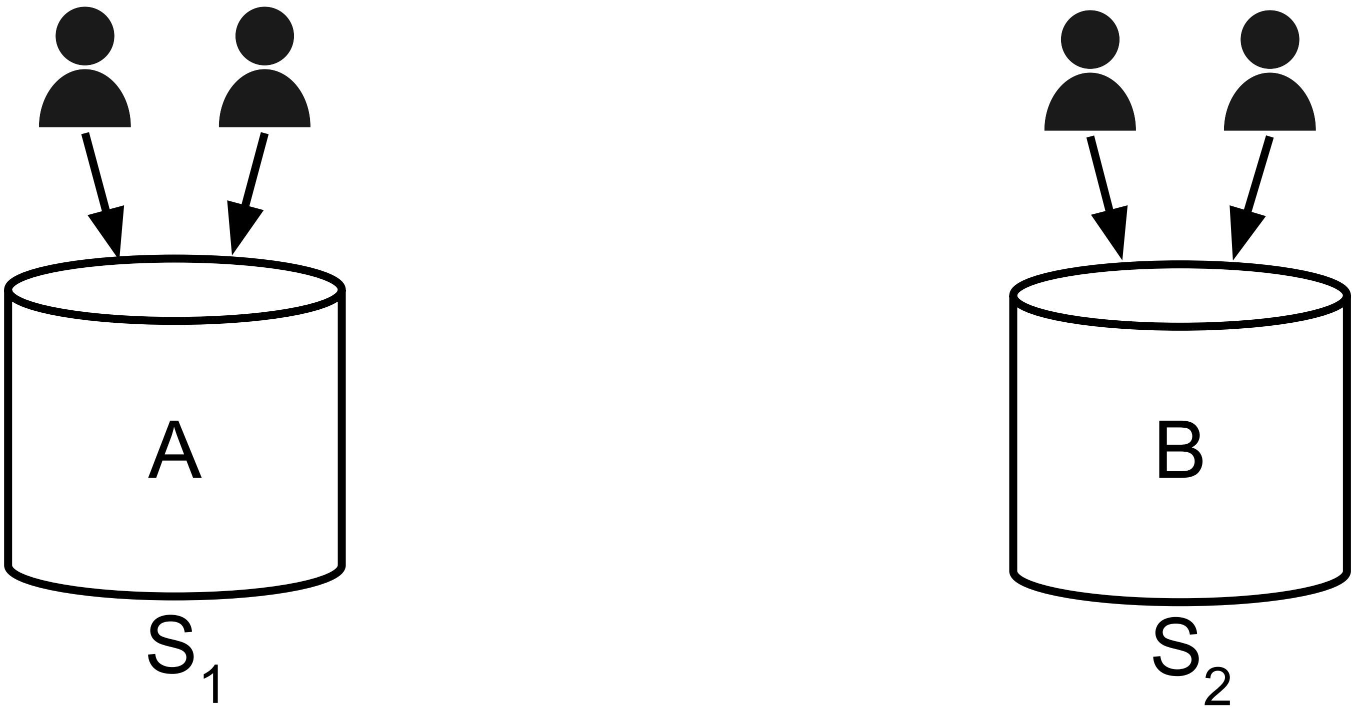
- **How many replicas?**
- **Where to place replicas?**
- **How to propagate updates?**

Single Node Architecture

How to scale beyond a single node?

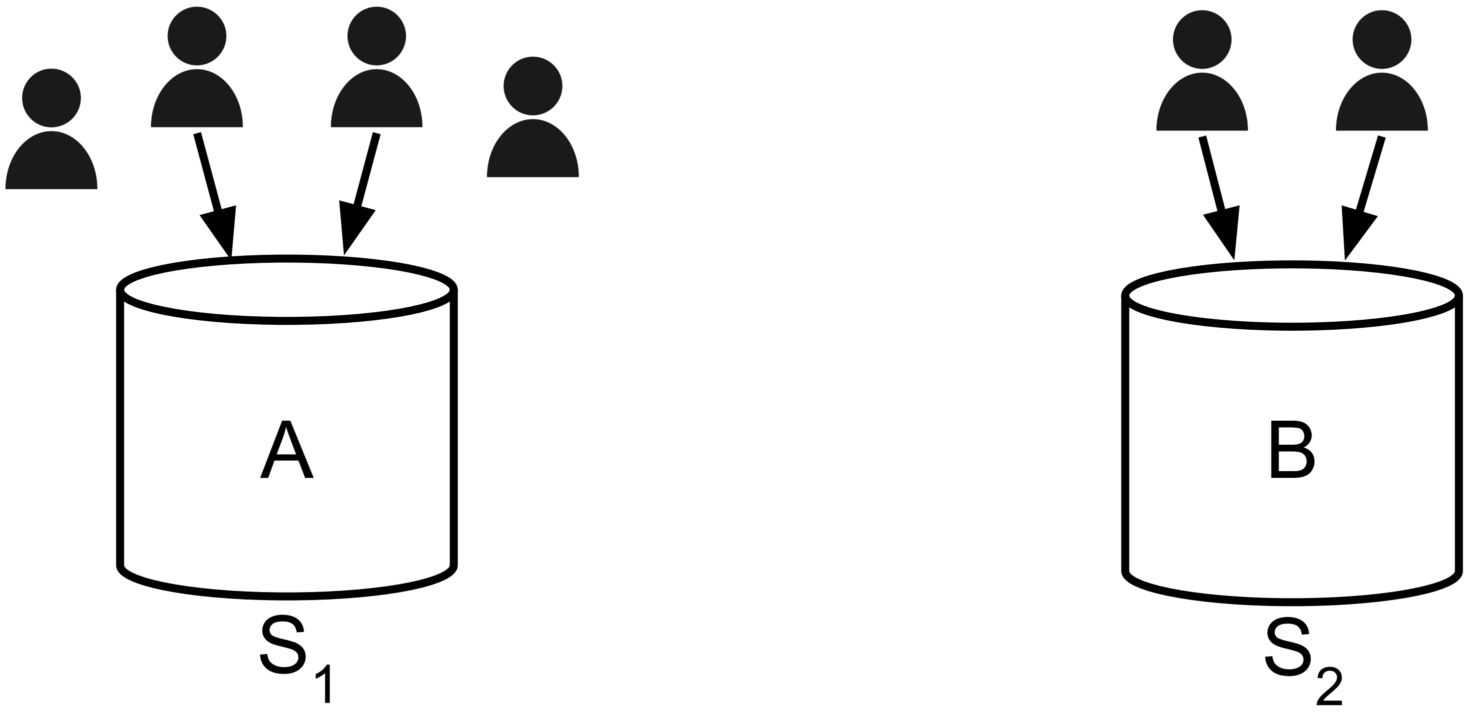
Replicate and partition

Partitioned Architecture

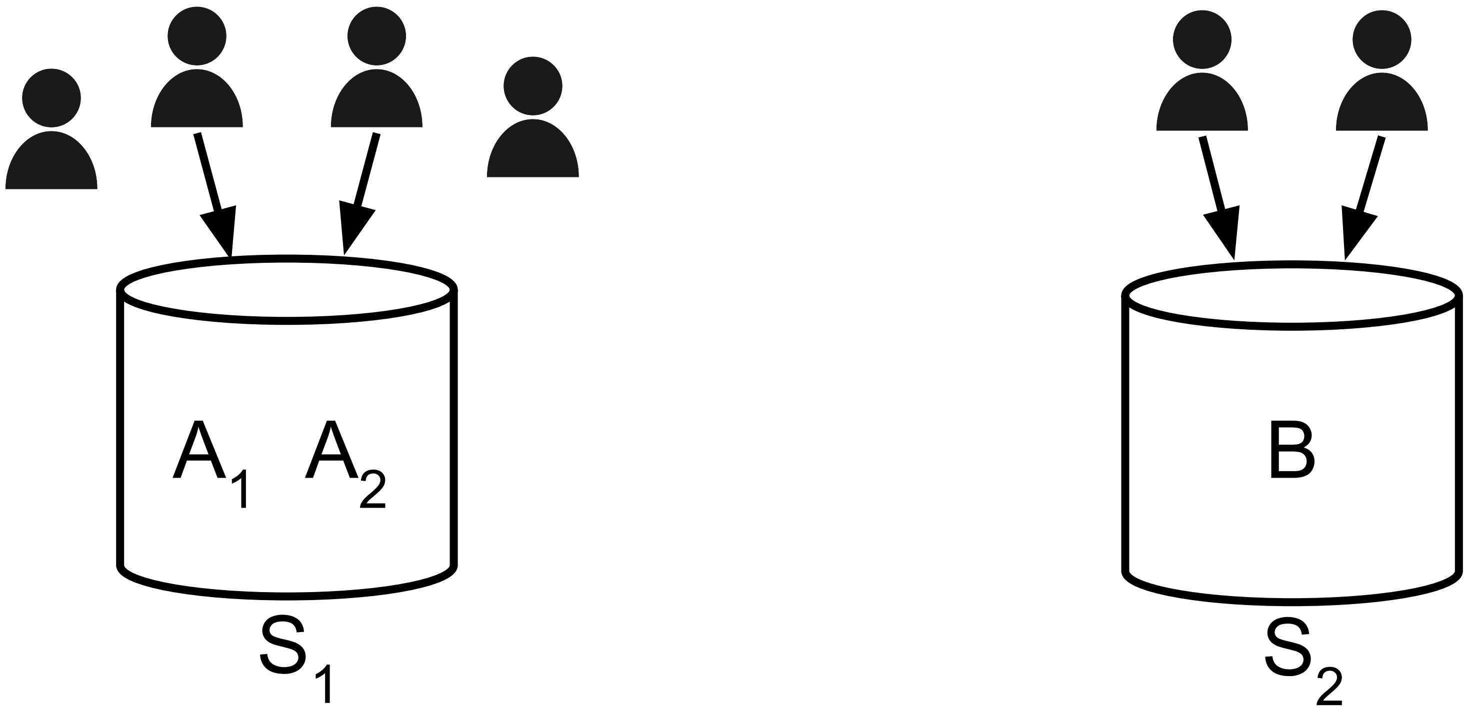


Distributes requests

Partitioned Architecture

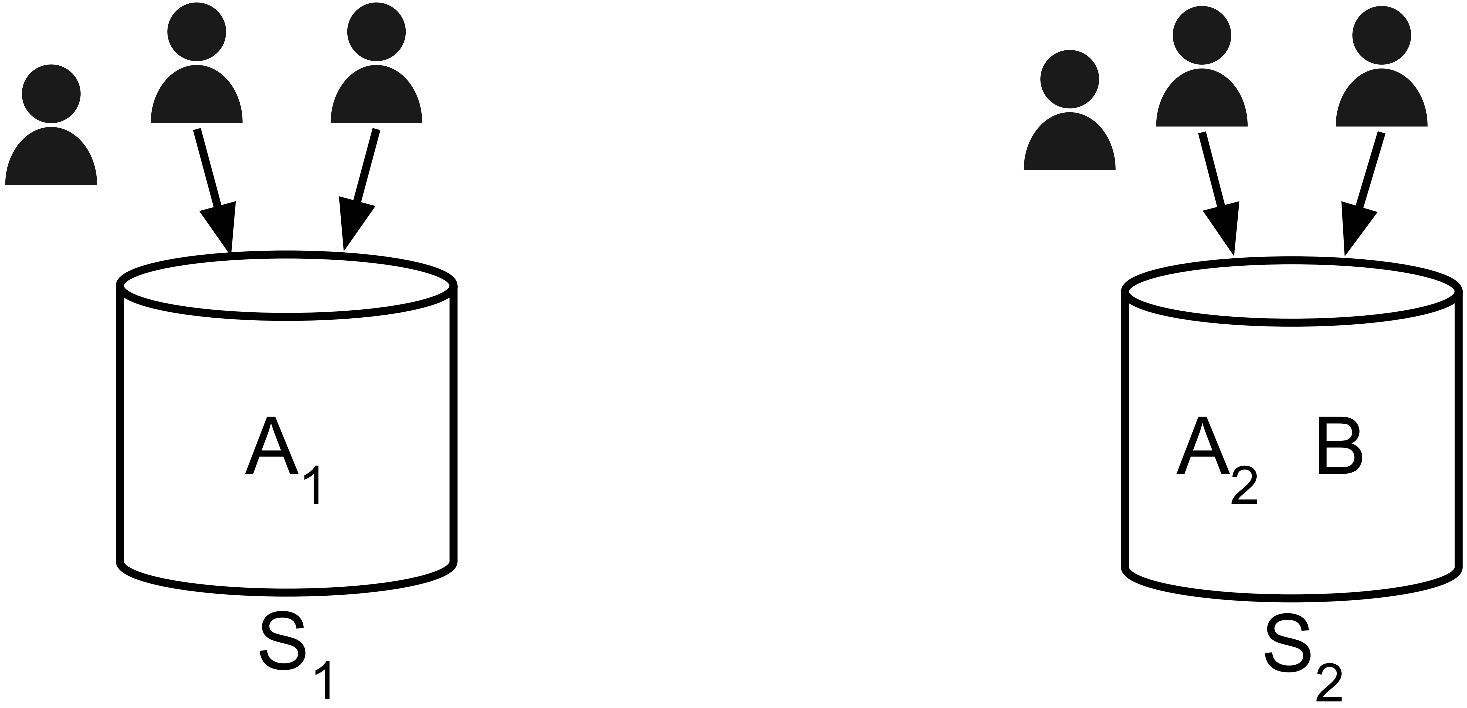


Partitioned Architecture



How to **form** partitions?

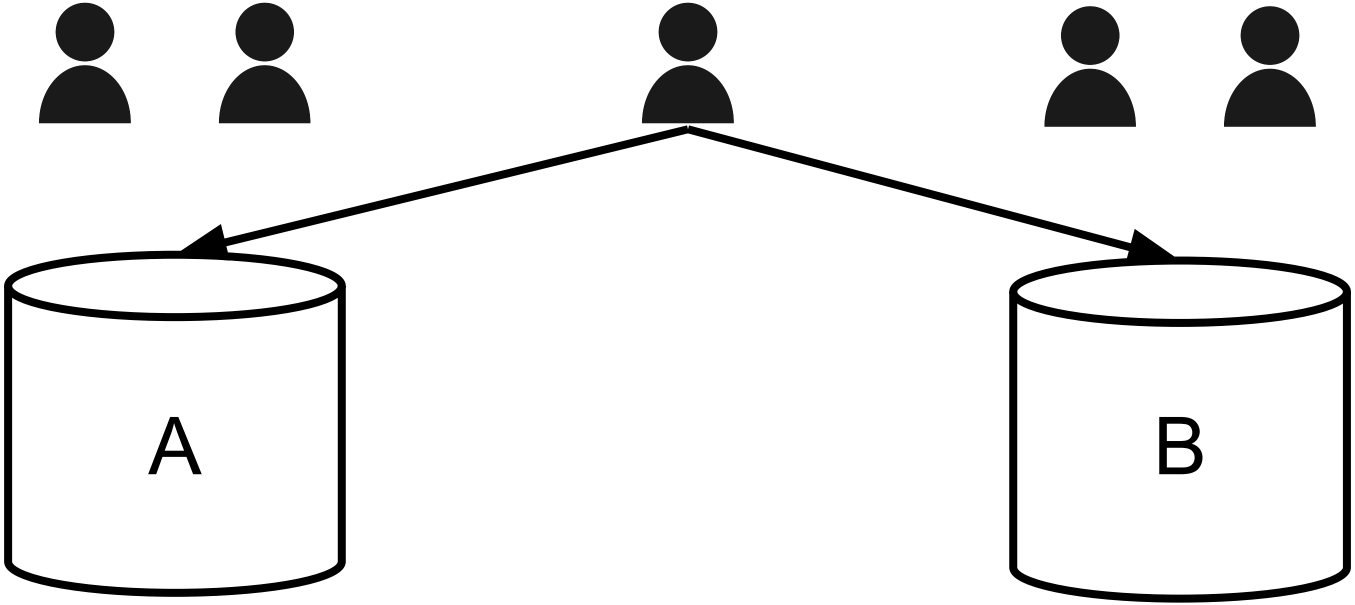
Partitioned Architecture



Where to **place** partitions?

UNIVERSITY OF WATERLOO

Partitioned Architecture



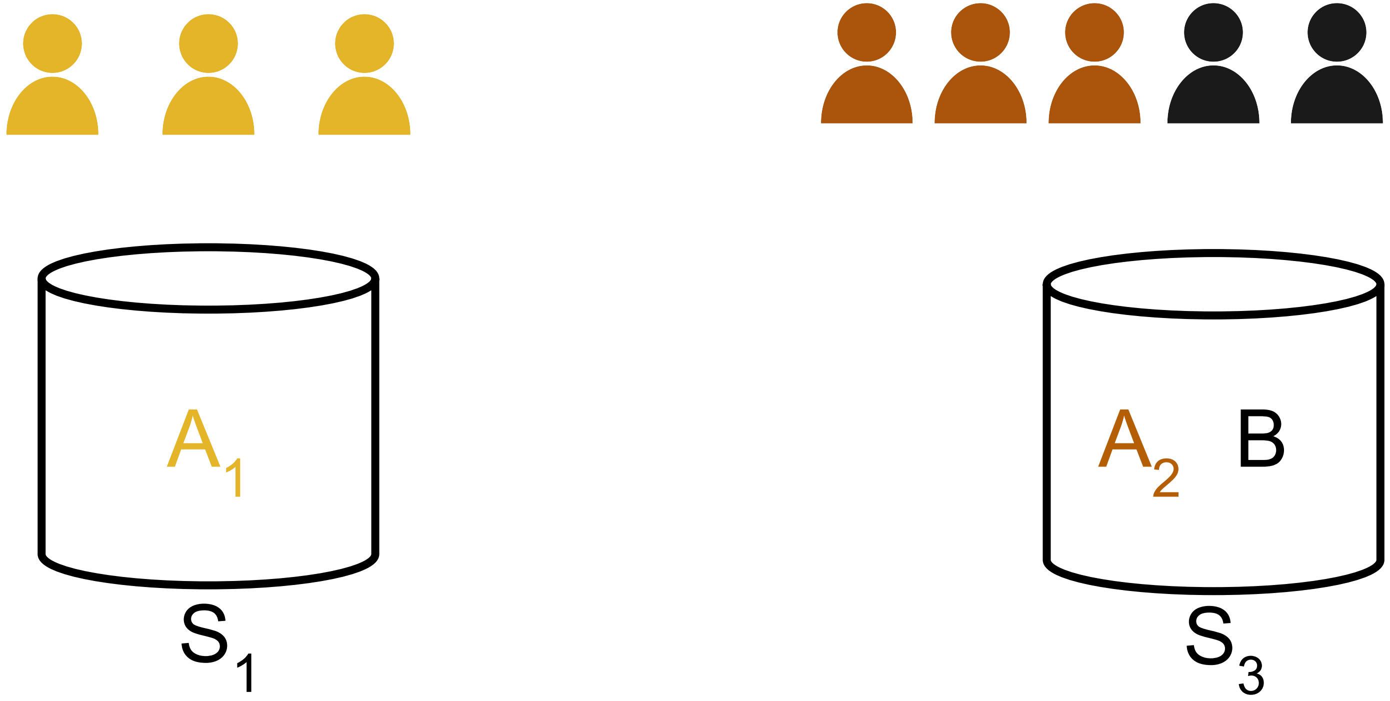
How to **execute**
multi-partition operations?

UNIVERSITY OF WATERLOO

Partitioning Decisions

- How to form partitions?
- Where to place partitions?
- How to execute multi-partition operations?

Where to place partitions?



UNIVERSITY OF WATERLOO

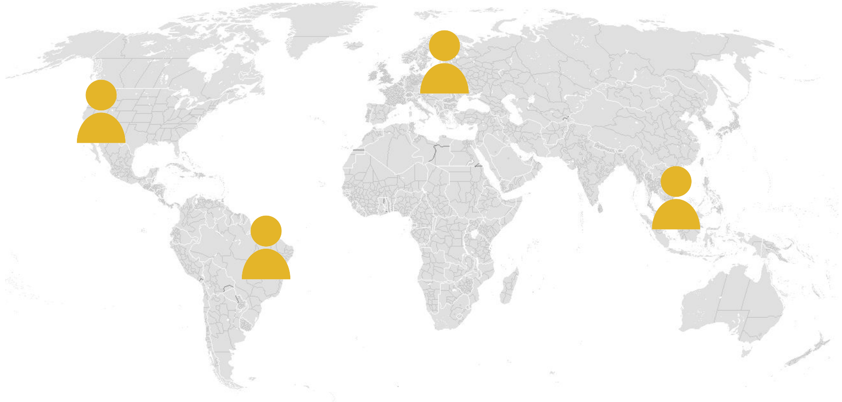
Static Decisions

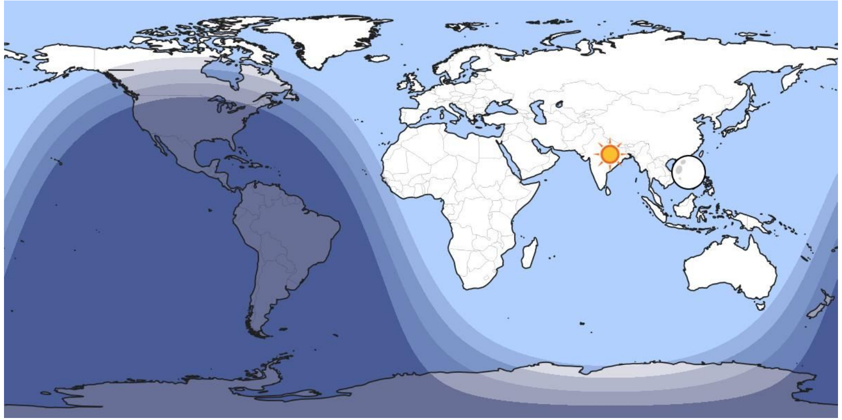
How to **make** a partitioning or replication **decision** when access patterns **change**?

Why do access patterns change?

UNIVERSITY OF WATERLOO

Why do accesses change?





Humans have **follow-the-sun cycles**

Why do accesses change?

reddit

Load bursts

Why do accesses change?

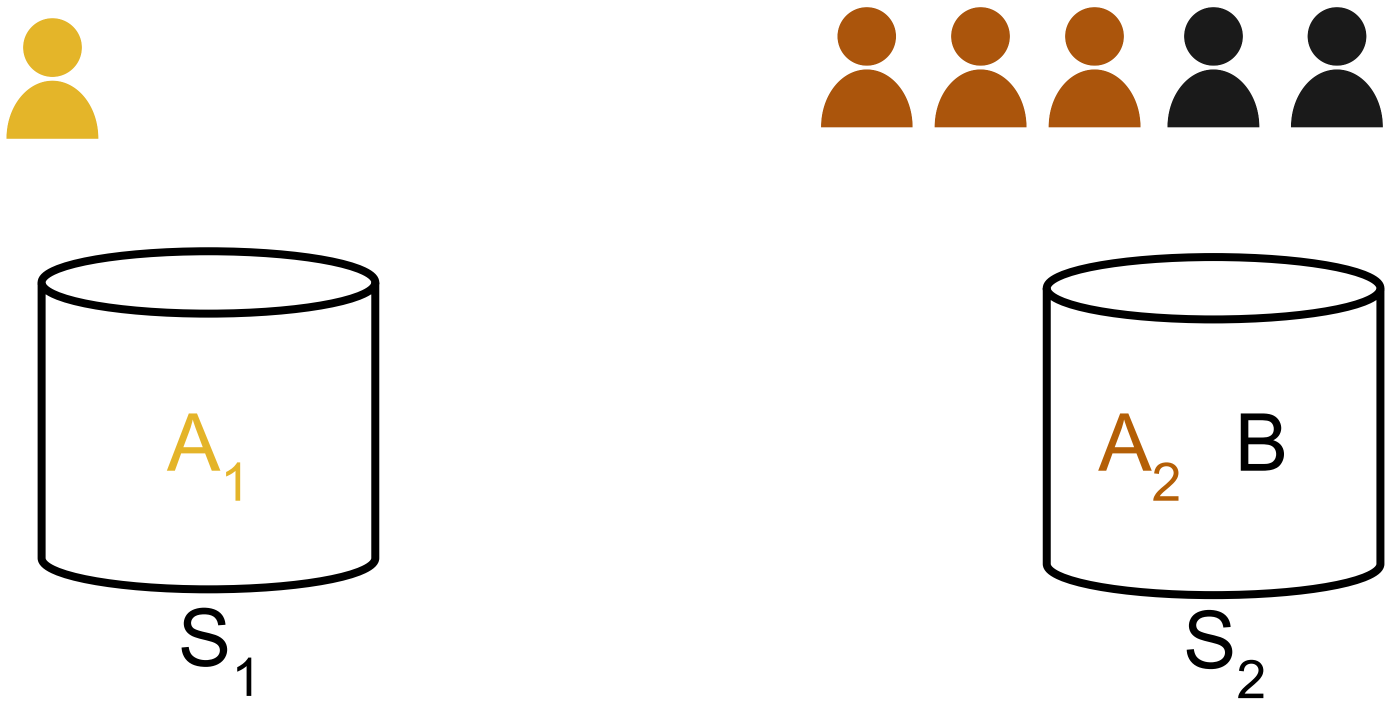
Shifting hot-spots

Static Decisions

How to **make** a partitioning or replication **decision** when access patterns **change**?

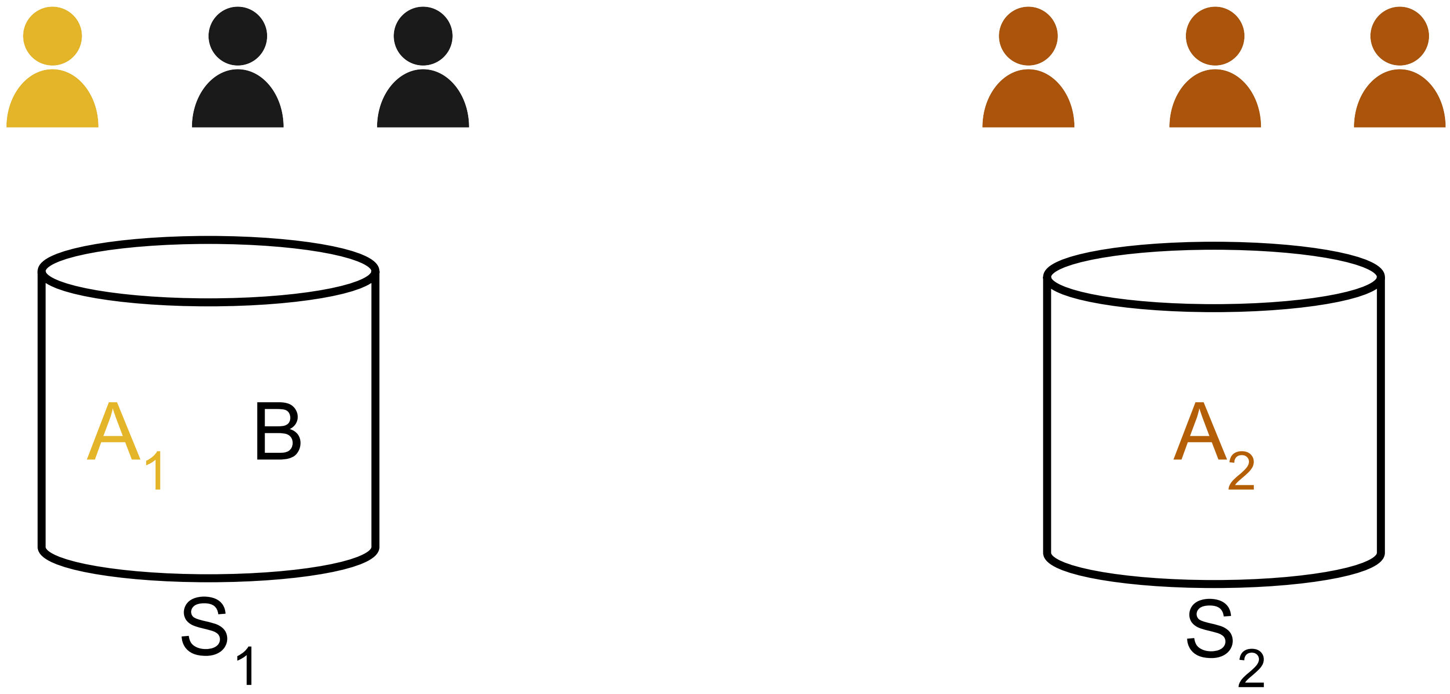
Adaptively replicate and partition

Where to place partitions?

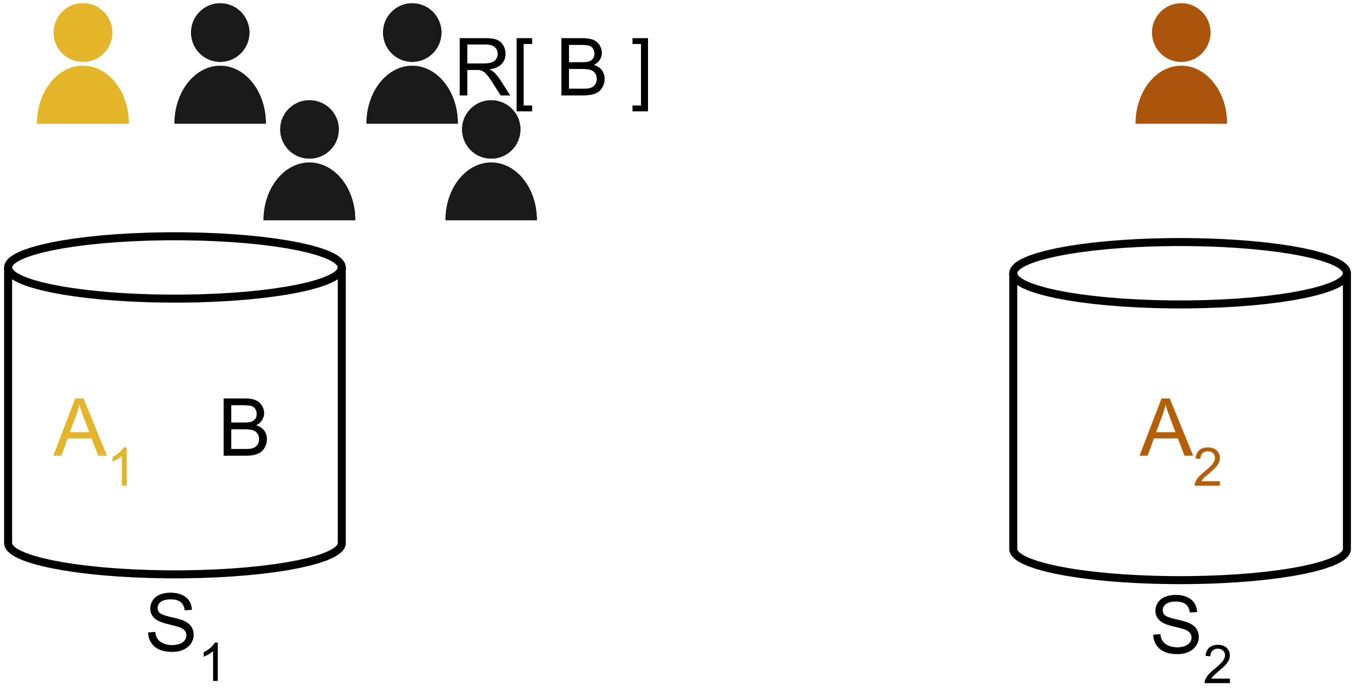


UNIVERSITY OF WATERLOO

Where to place partitions?

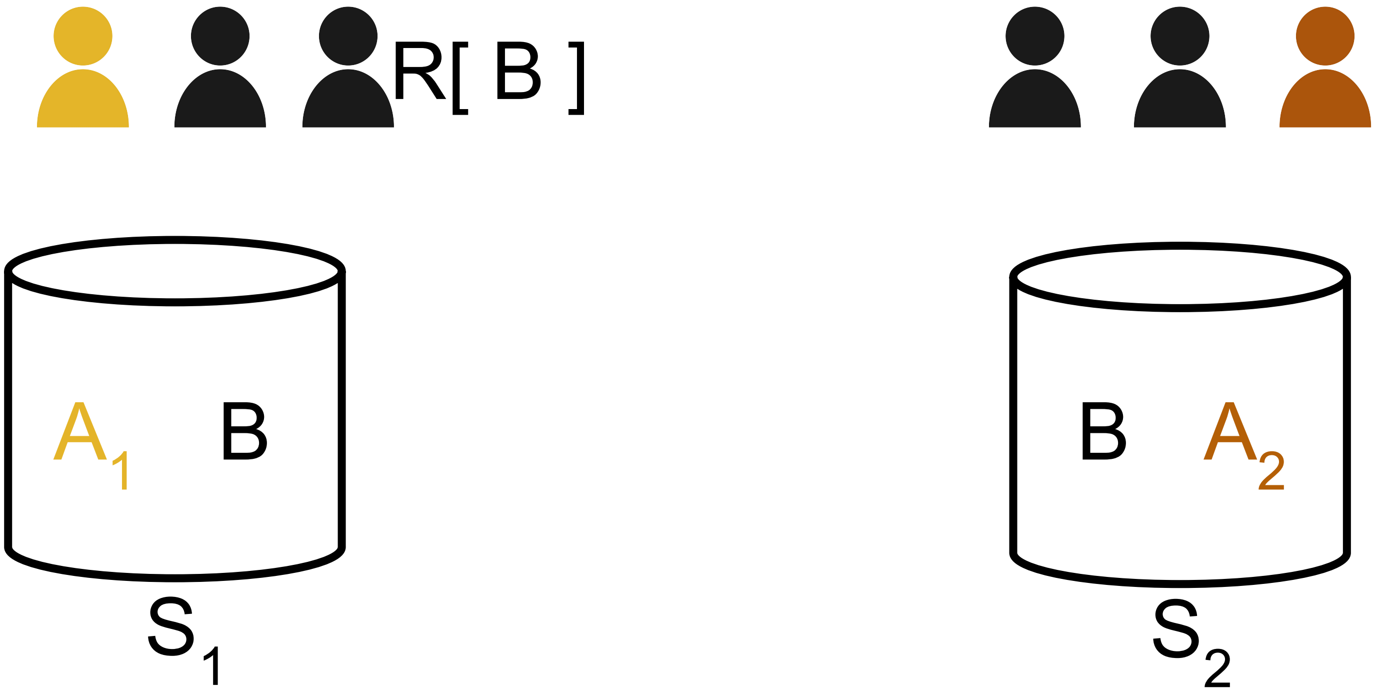


How many replicas?



UNIVERSITY OF WATERLOO

How many replicas?



UNIVERSITY OF WATERLOO

Road Map

- Adaptive Replication
- Adaptive Partitioning
- Outlook

Adaptive Replication

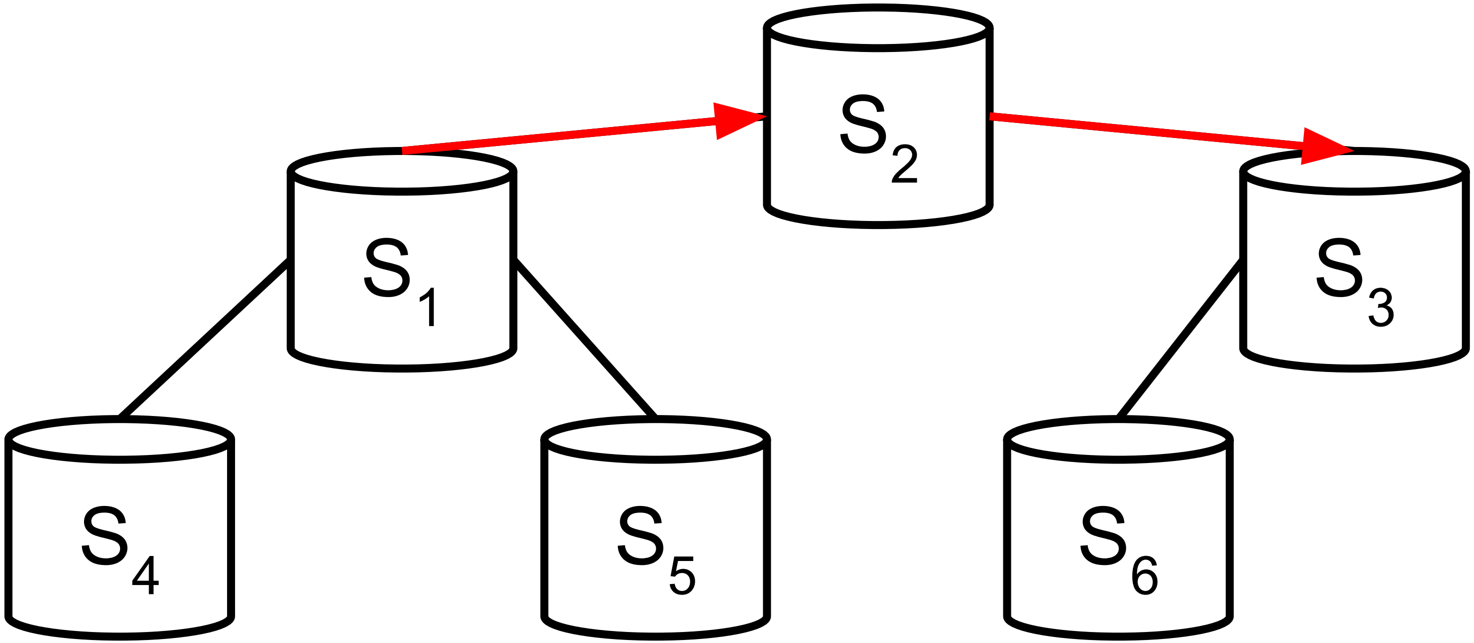
Replication Decisions

- **How many replicas?**
- **Where to place replicas?**
- **How to propagate updates?**

Adaptive Replication

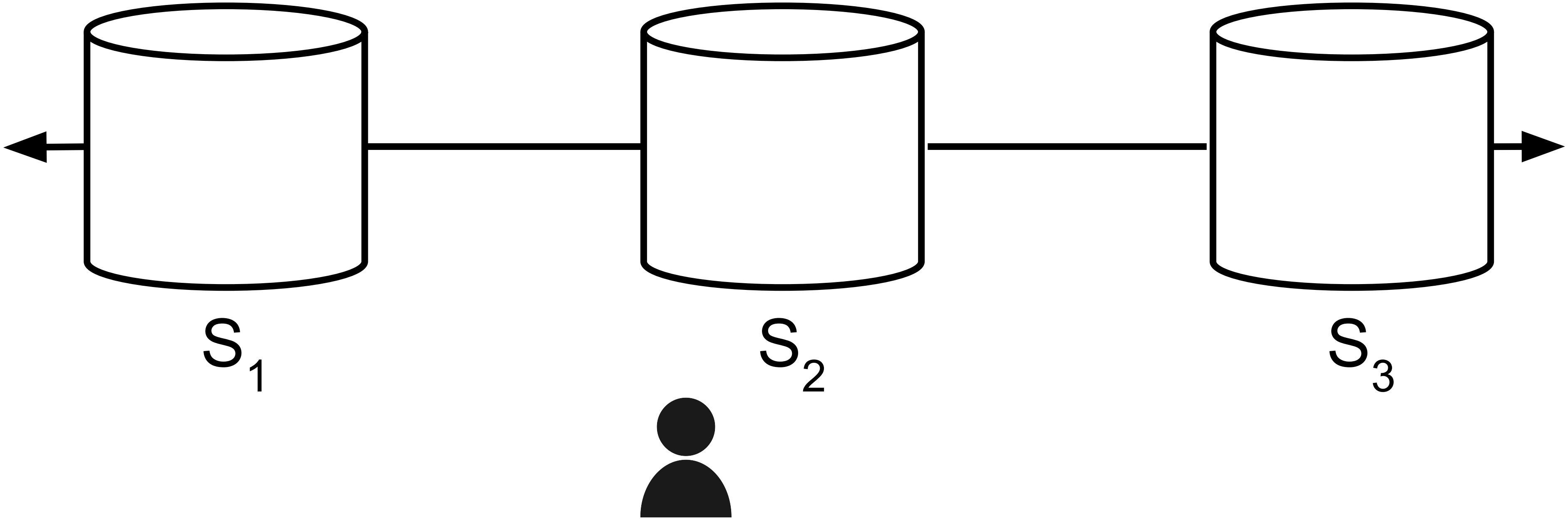
- Decentralized
- Geo-Distributed
- Caching
- Availability

Adaptive Replication (ADR)



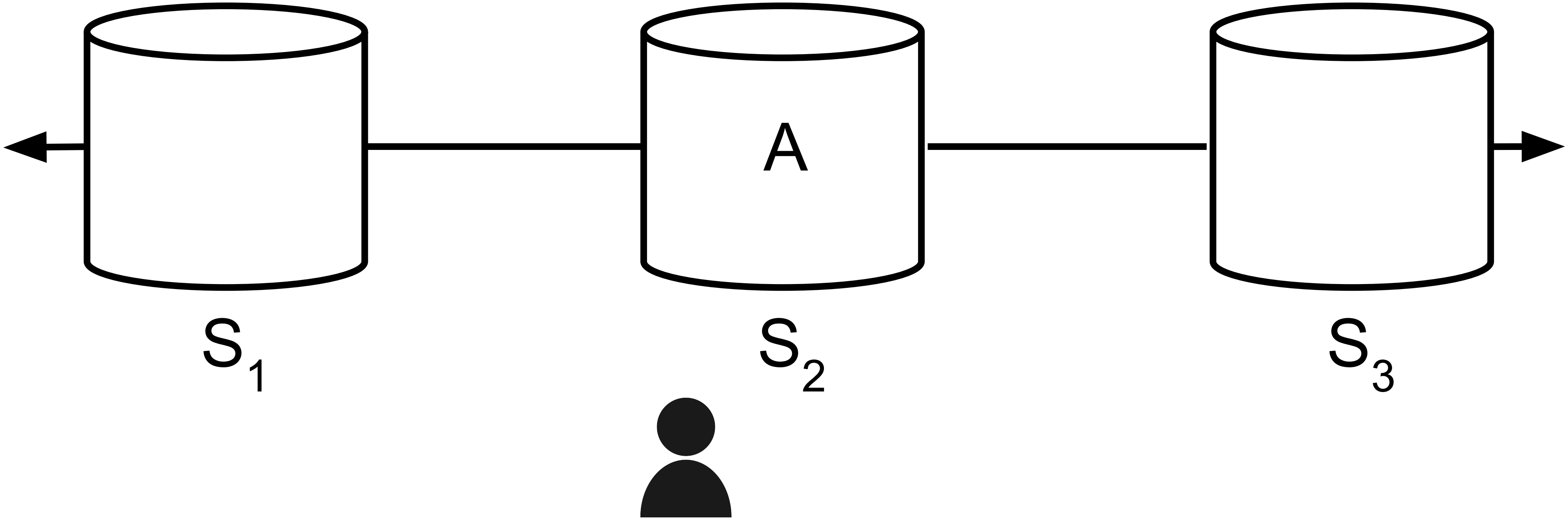
(Wolfson et al., TODS 1997)

Adaptive Replication (ADR)



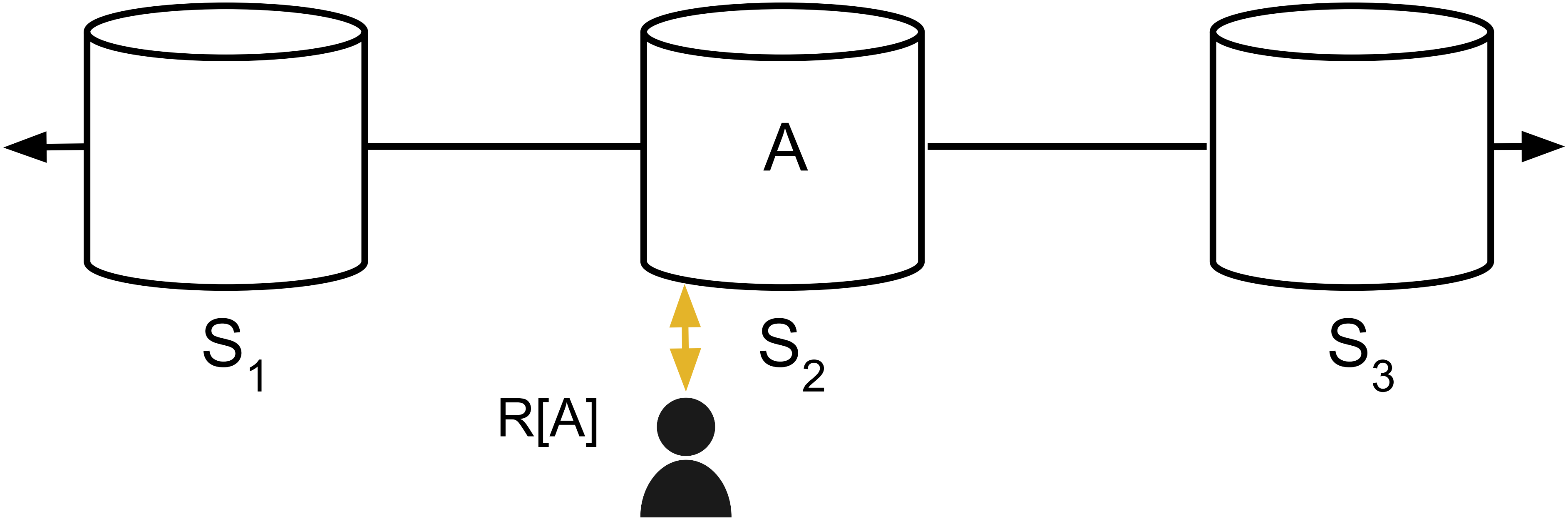
(Wolfson et al., TODS 1997)

Adaptive Replication (ADR)



(Wolfson et al., TODS 1997)

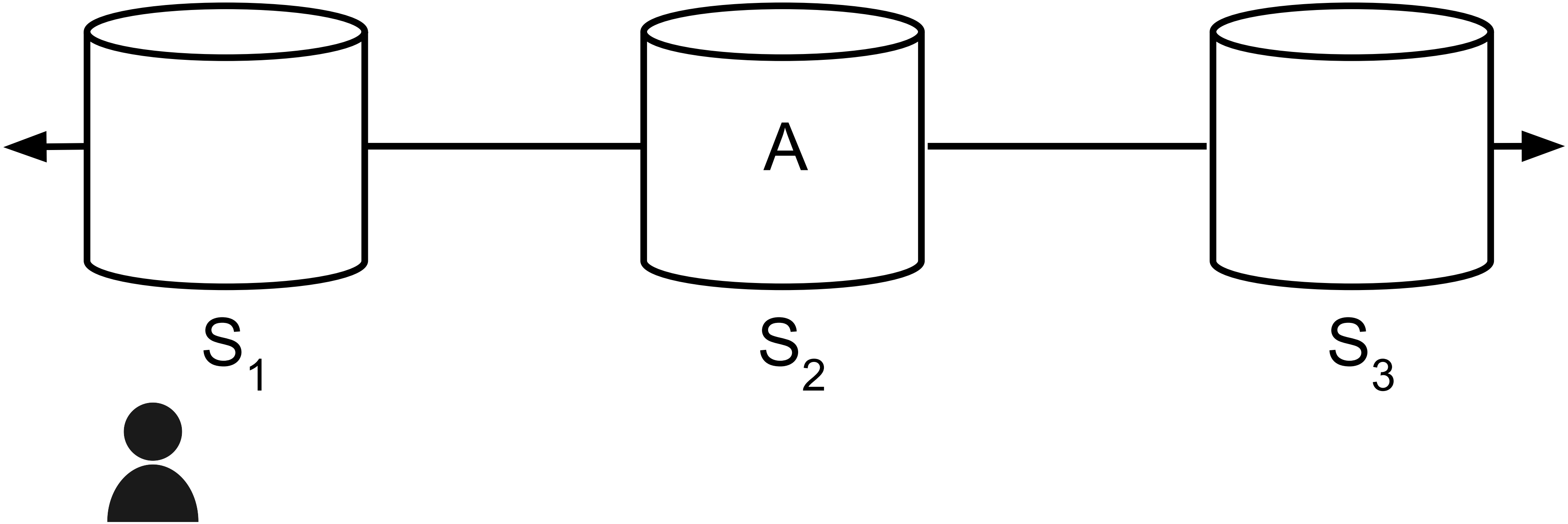
Local Read



(Wolfson et al., TODS 1997)

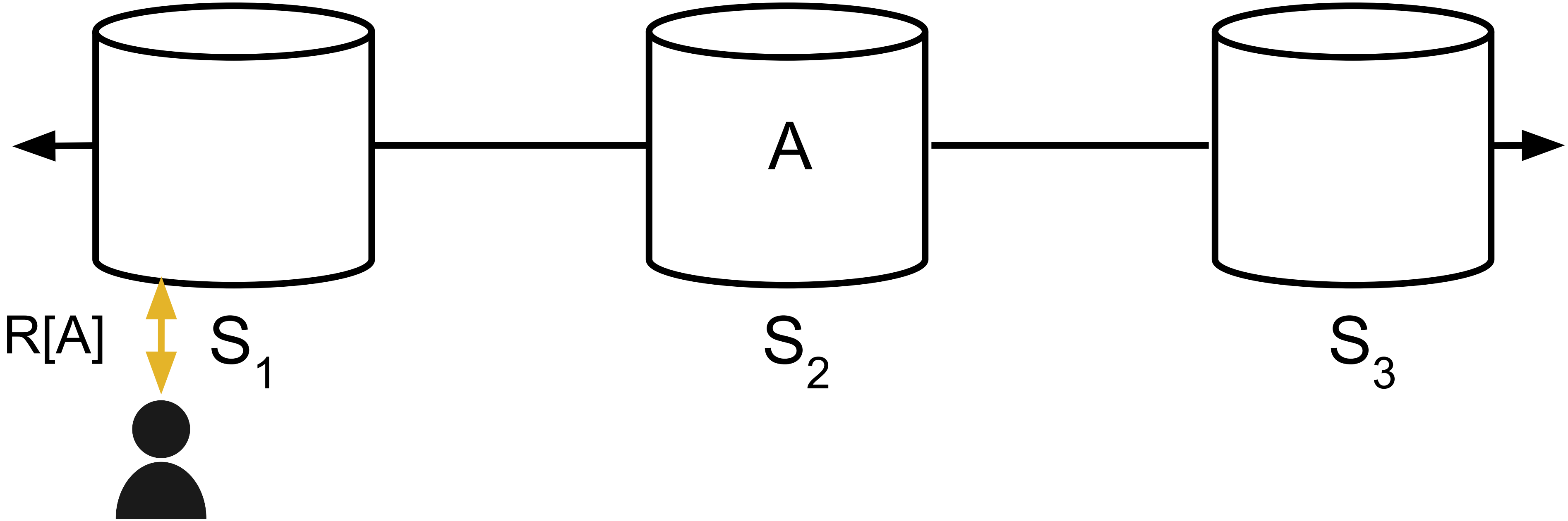
UNIVERSITY OF WATERLOO

Remote Read



(Wolfson et al., TODS 1997)

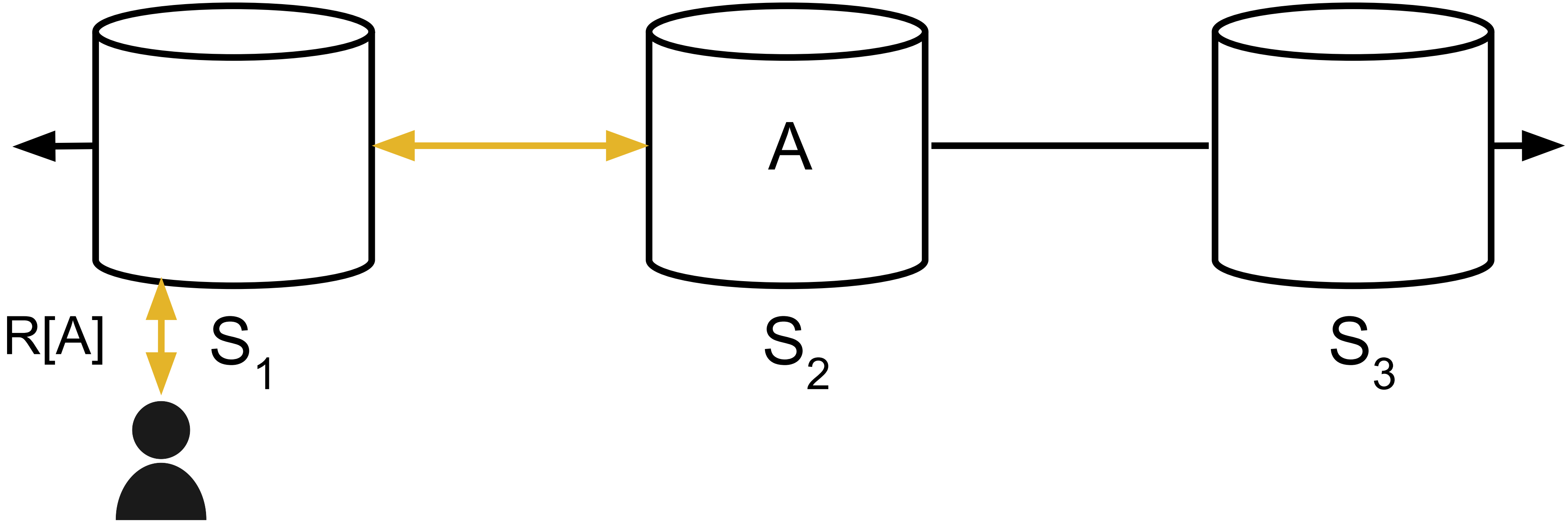
Remote Read



(Wolfson et al., TODS 1997)

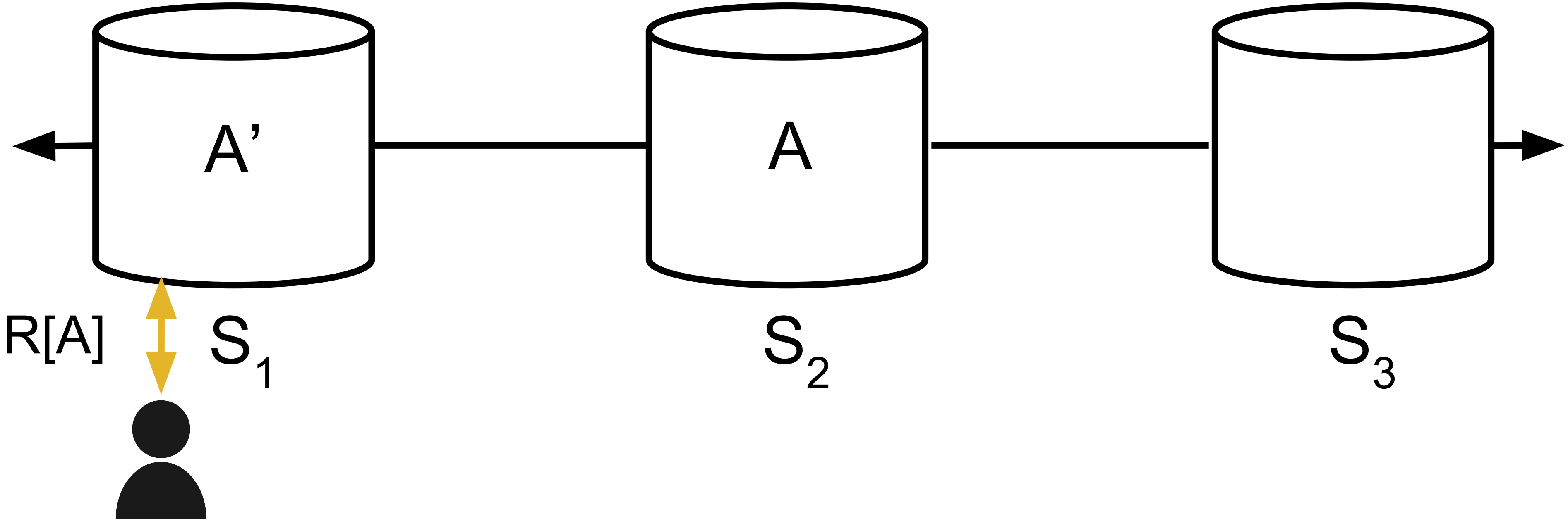
Remote Read

More Messages!



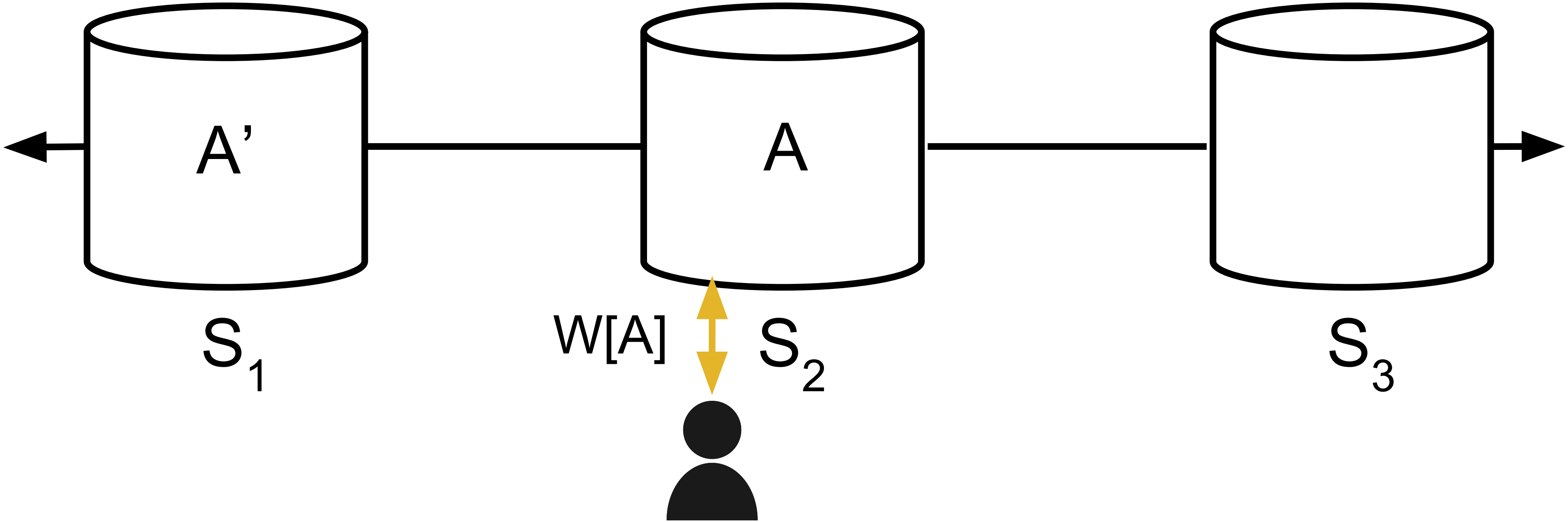
(Wolfson et al., TODS 1997)

Replication for Reads



(Wolfson et al., TODS 1997)

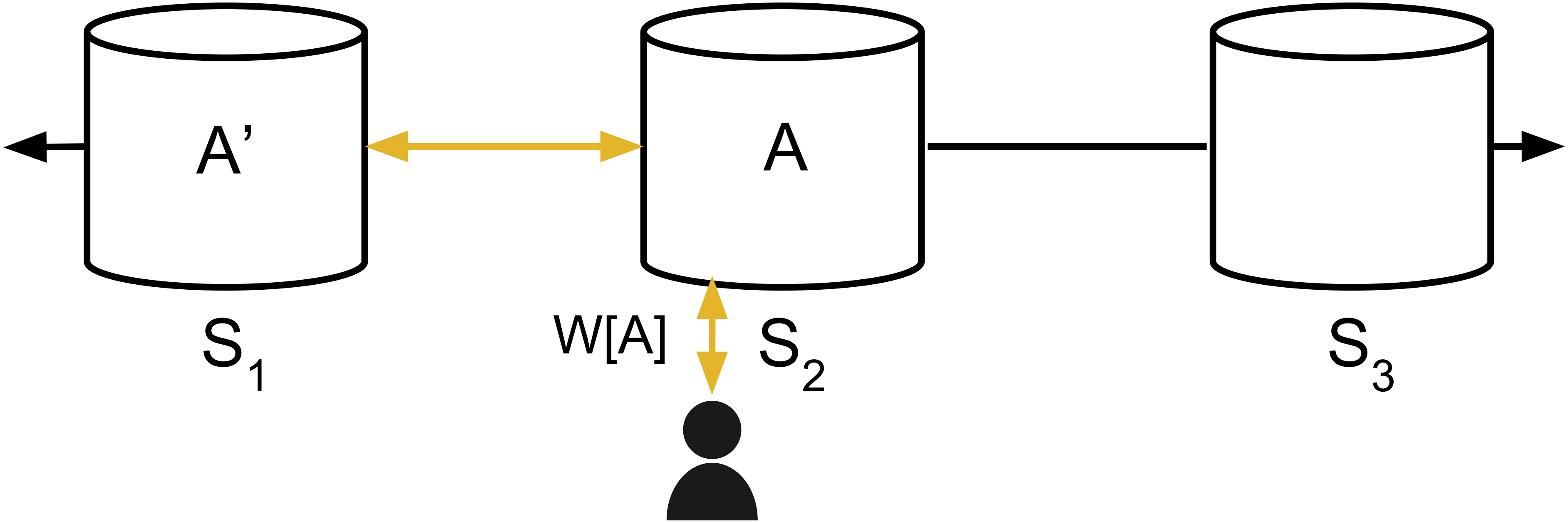
No Free Lunch



(Wolfson et al., TODS 1997)

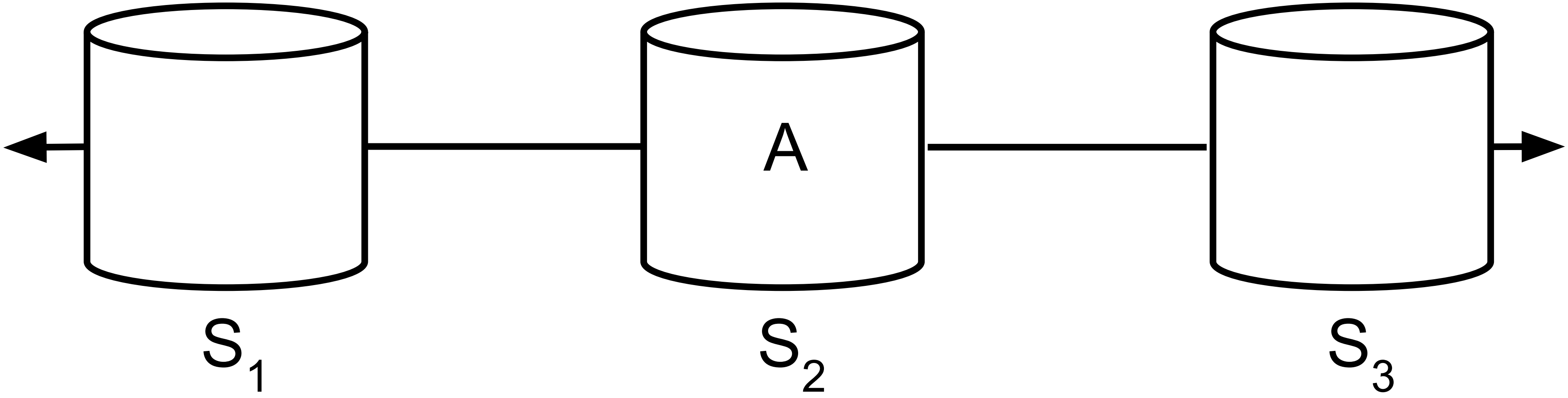
No Free Lunch

Reduces read cost,
Increases write cost



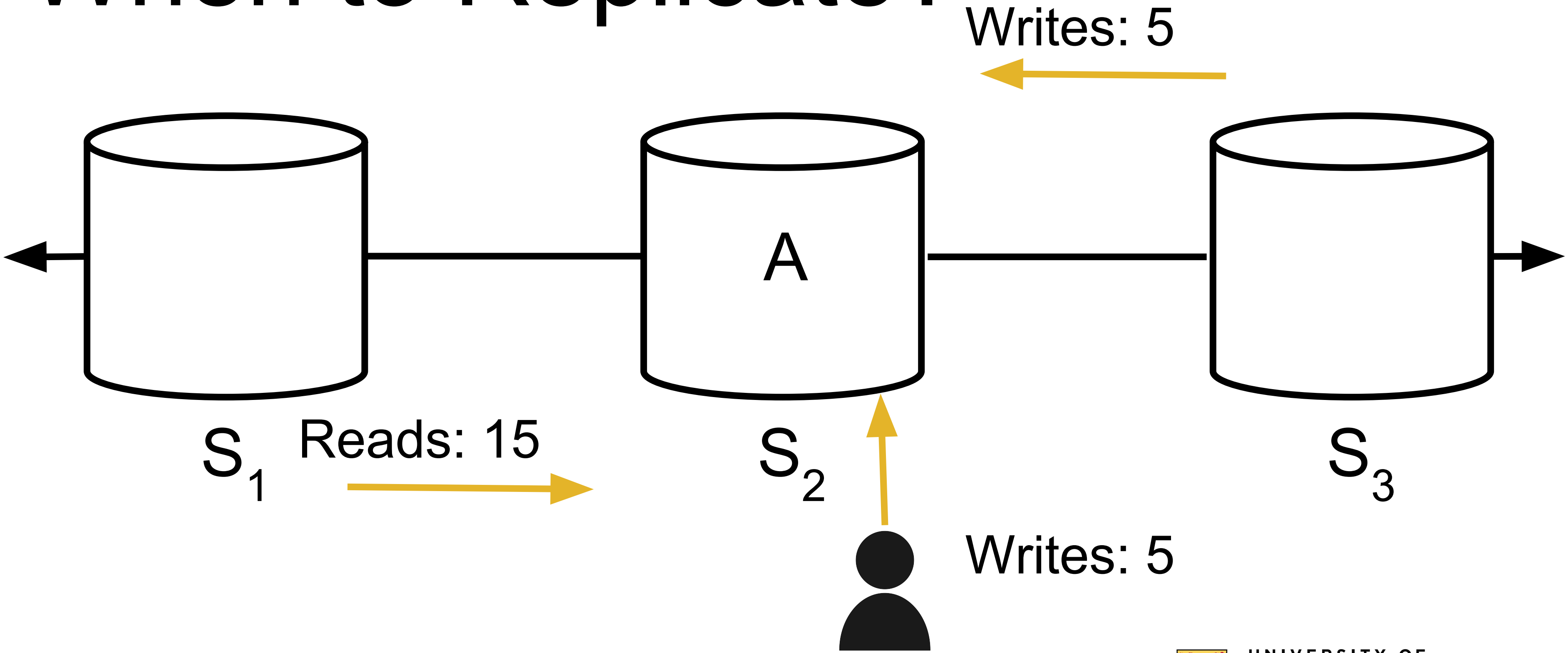
(Wolfson et al., TODS 1997)

When to Replicate?



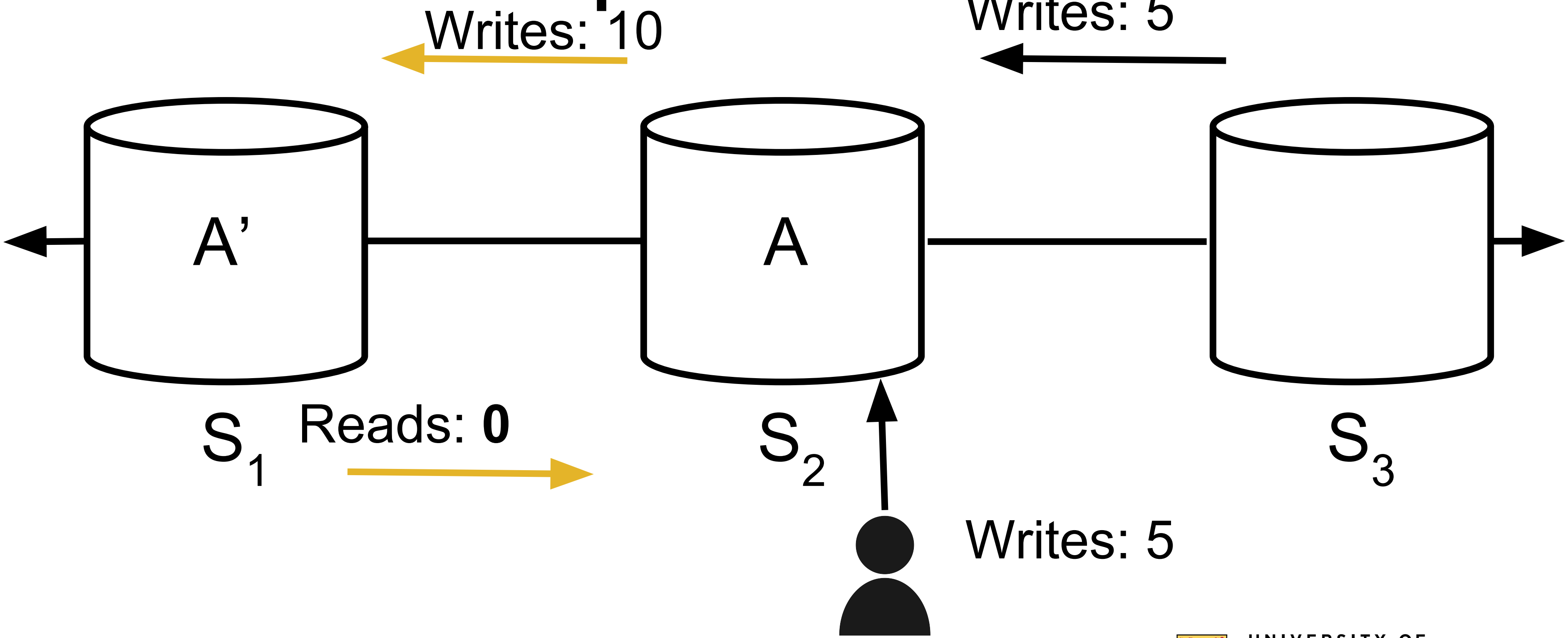
(Wolfson et al., TODS 1997)

When to Replicate?



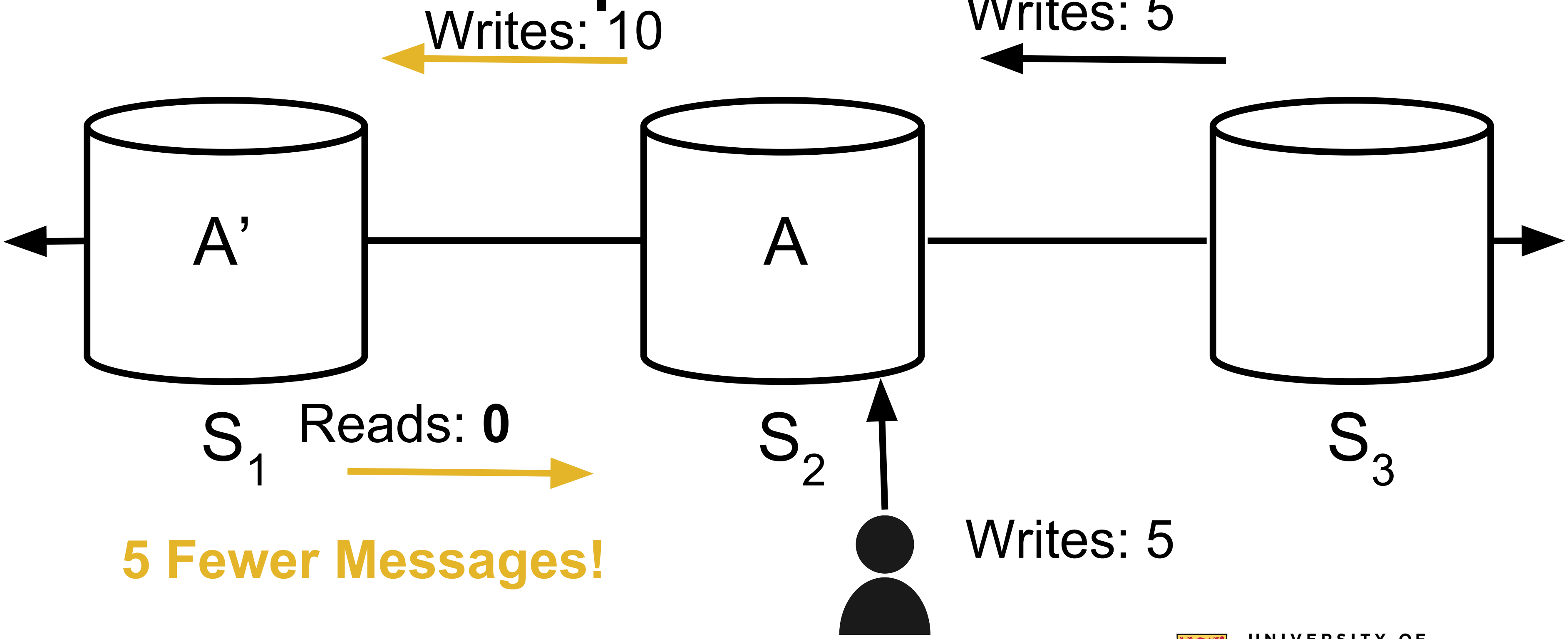
(Wolfson et al., TODS 1997)

When to Replicate?



(Wolfson et al., TODS 1997)

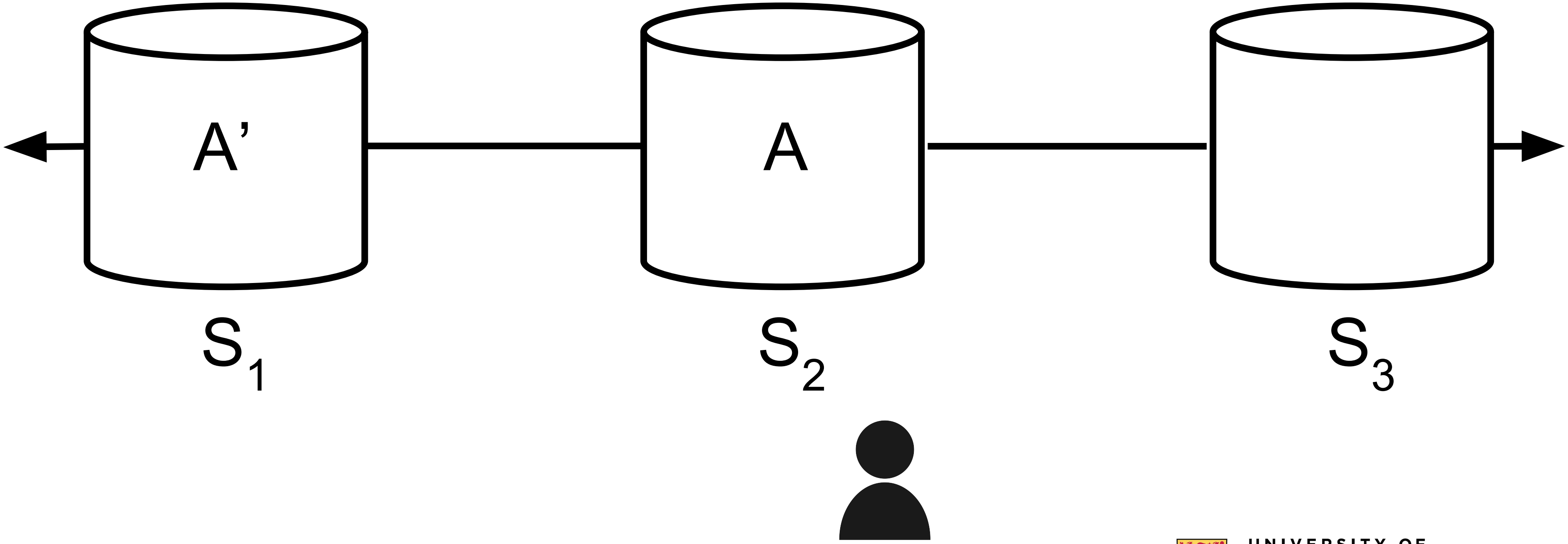
When to Replicate?



5 Fewer Messages!

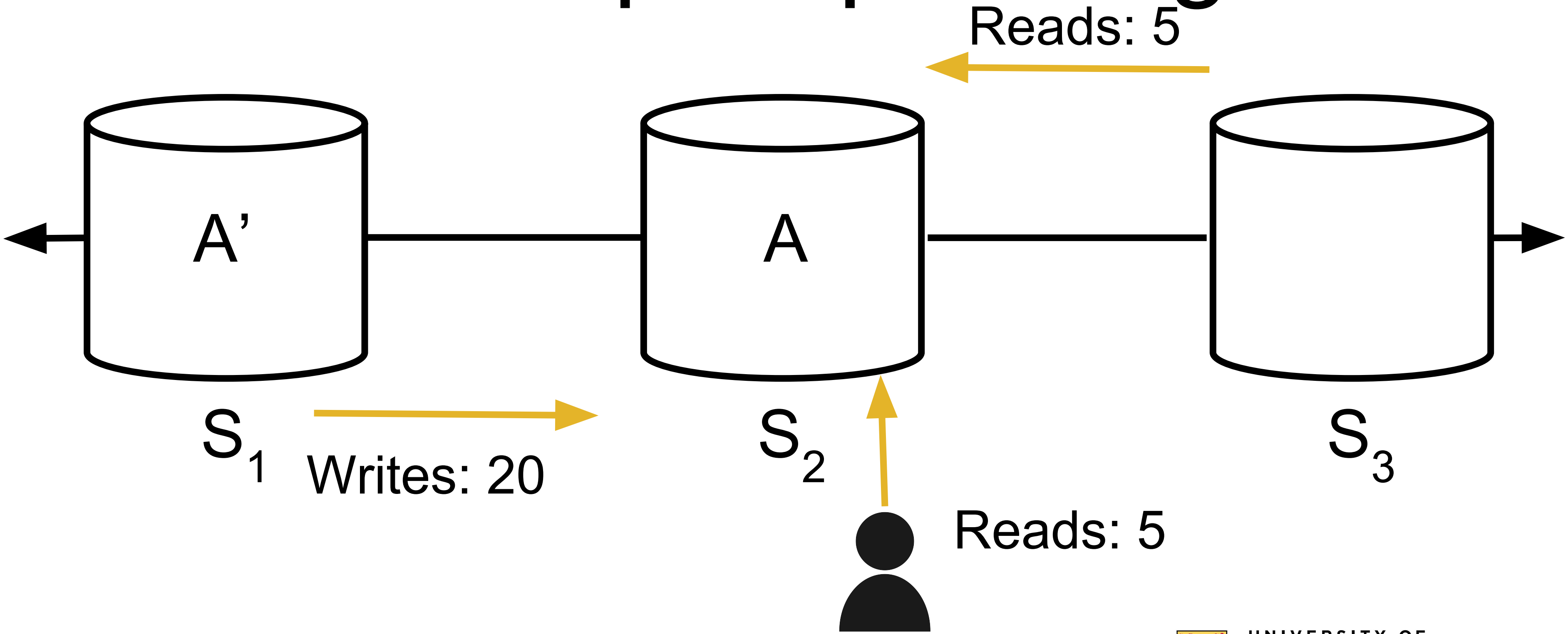
(Wolfson et al., TODS 1997)

When to Stop Replicating?



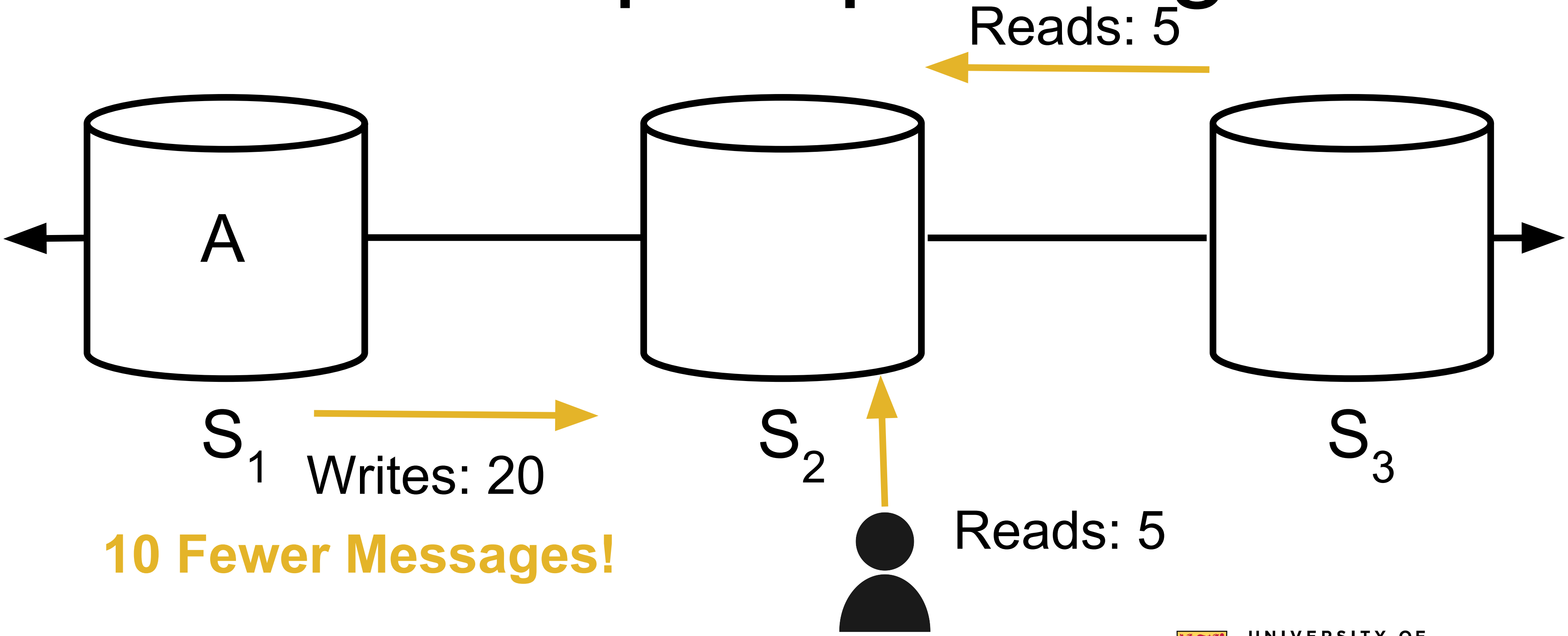
(Wolfson et al., TODS 1997)

When to Stop Replicating?



(Wolfson et al., TODS 1997)

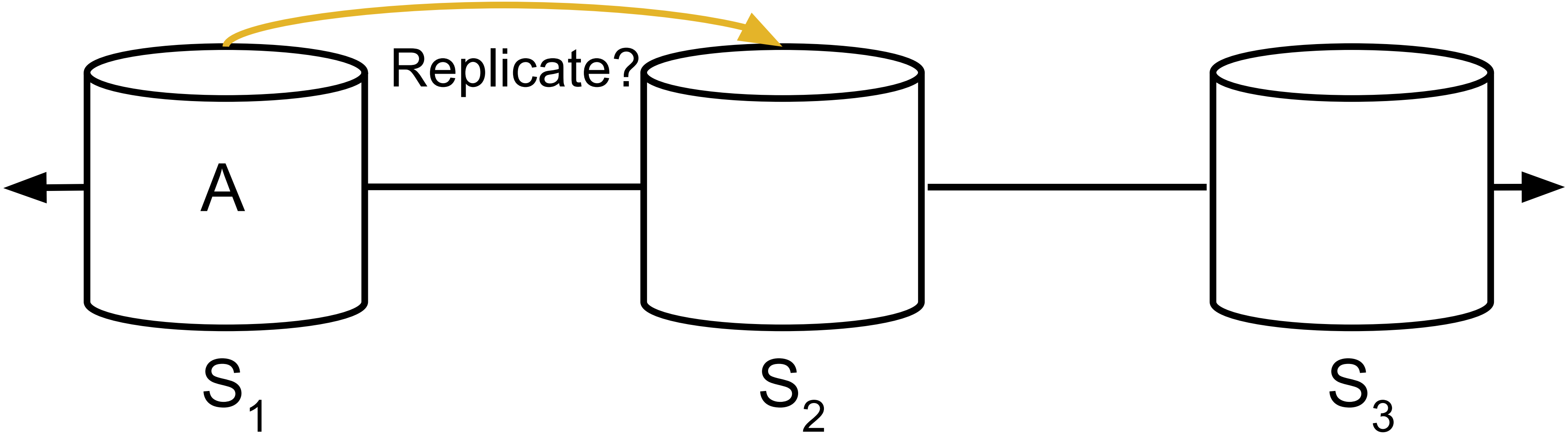
When to Stop Replicating?



10 Fewer Messages!

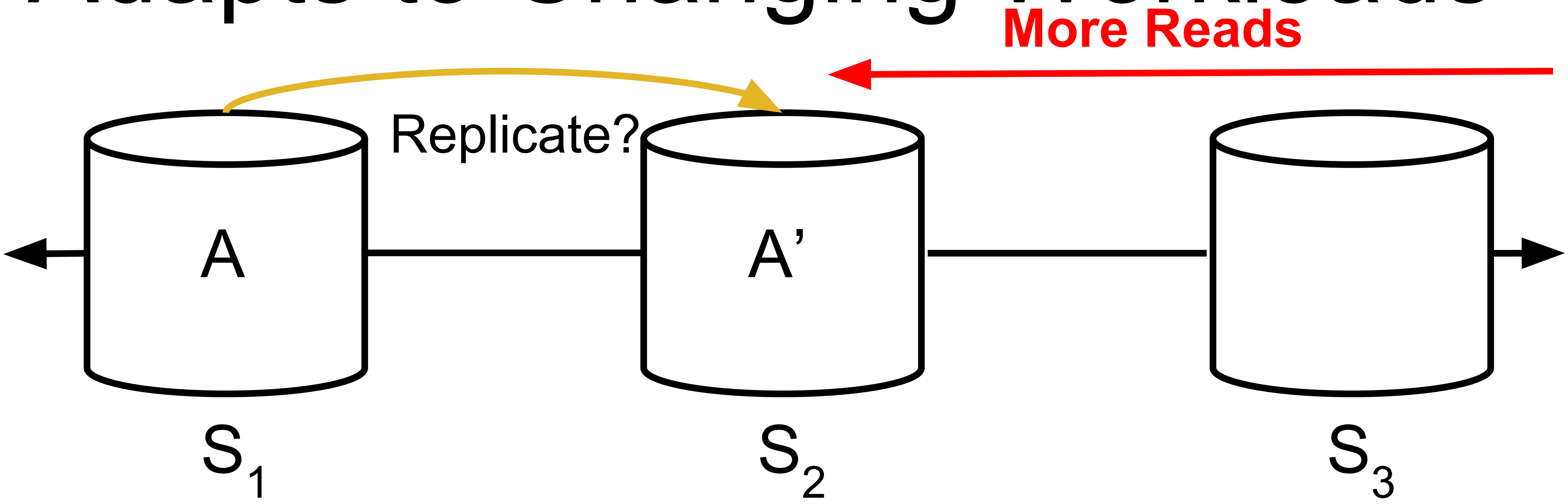
(Wolfson et al., TODS 1997)

Decentralized Decisions



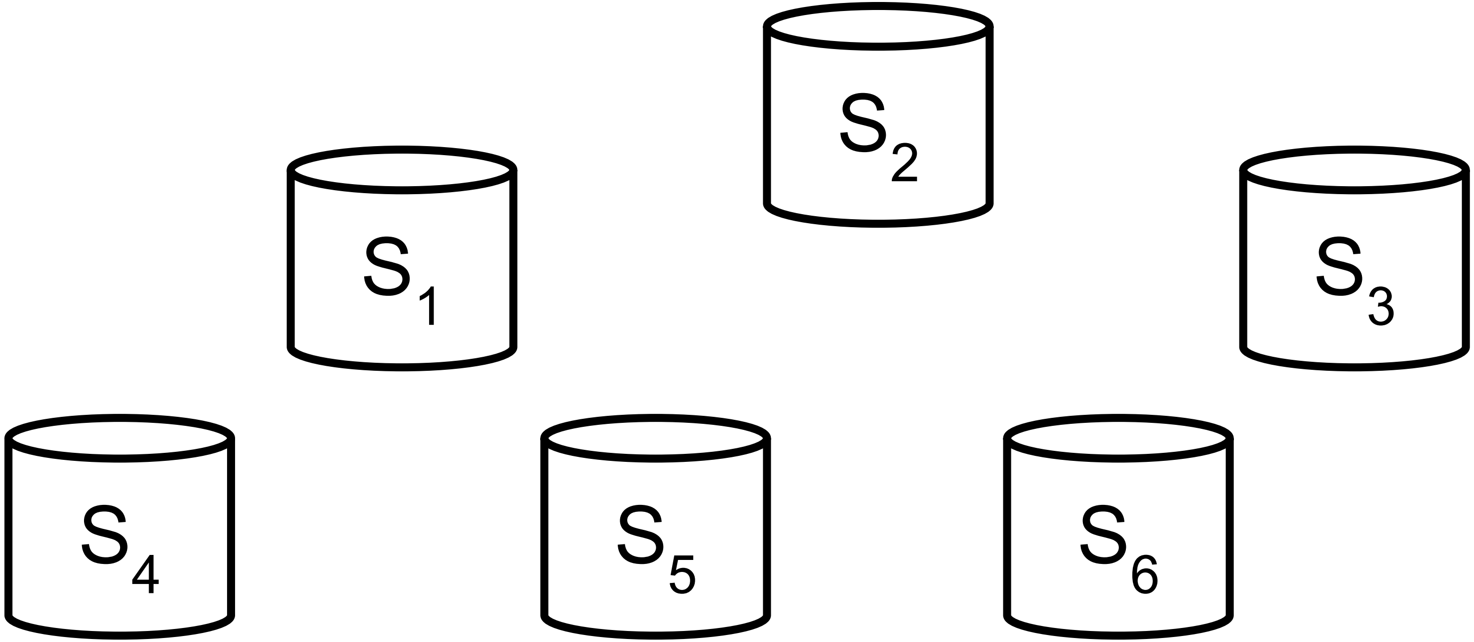
(Wolfson et al., TODS 1997)

Adapts to Changing Workloads



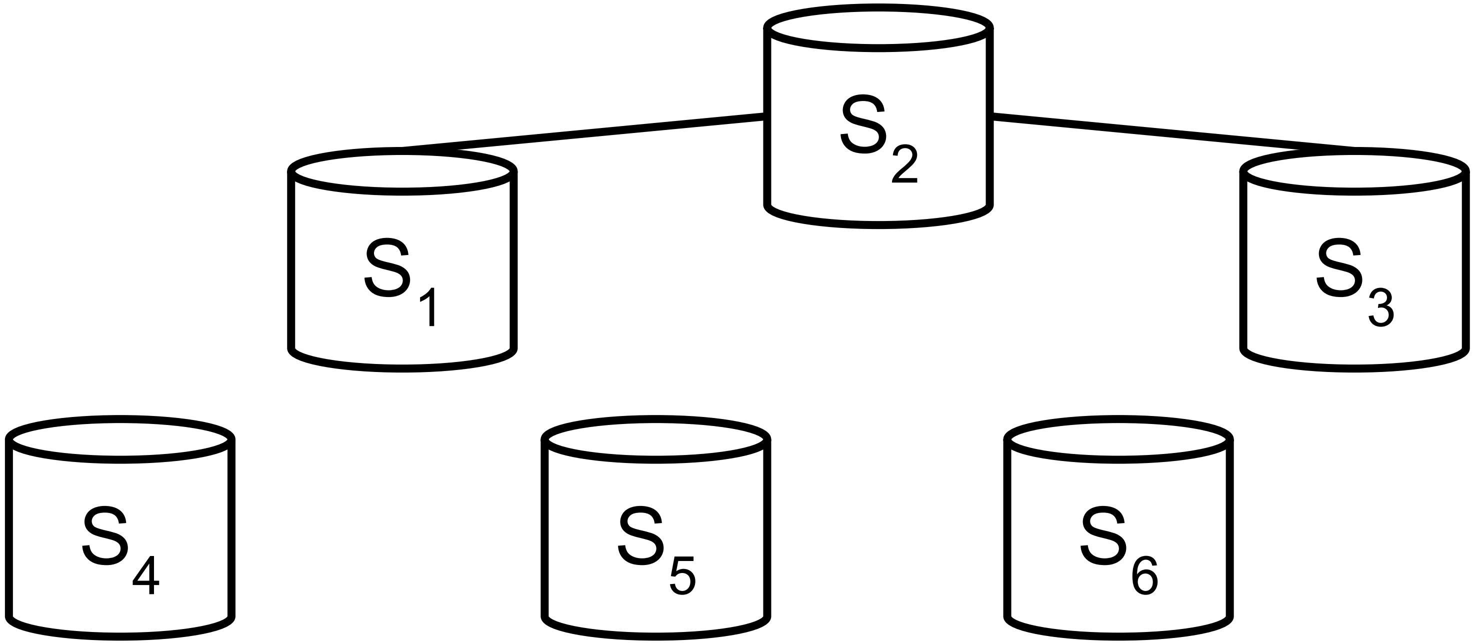
(Wolfson et al., TODS 1997)

Extensions (Network Topology)



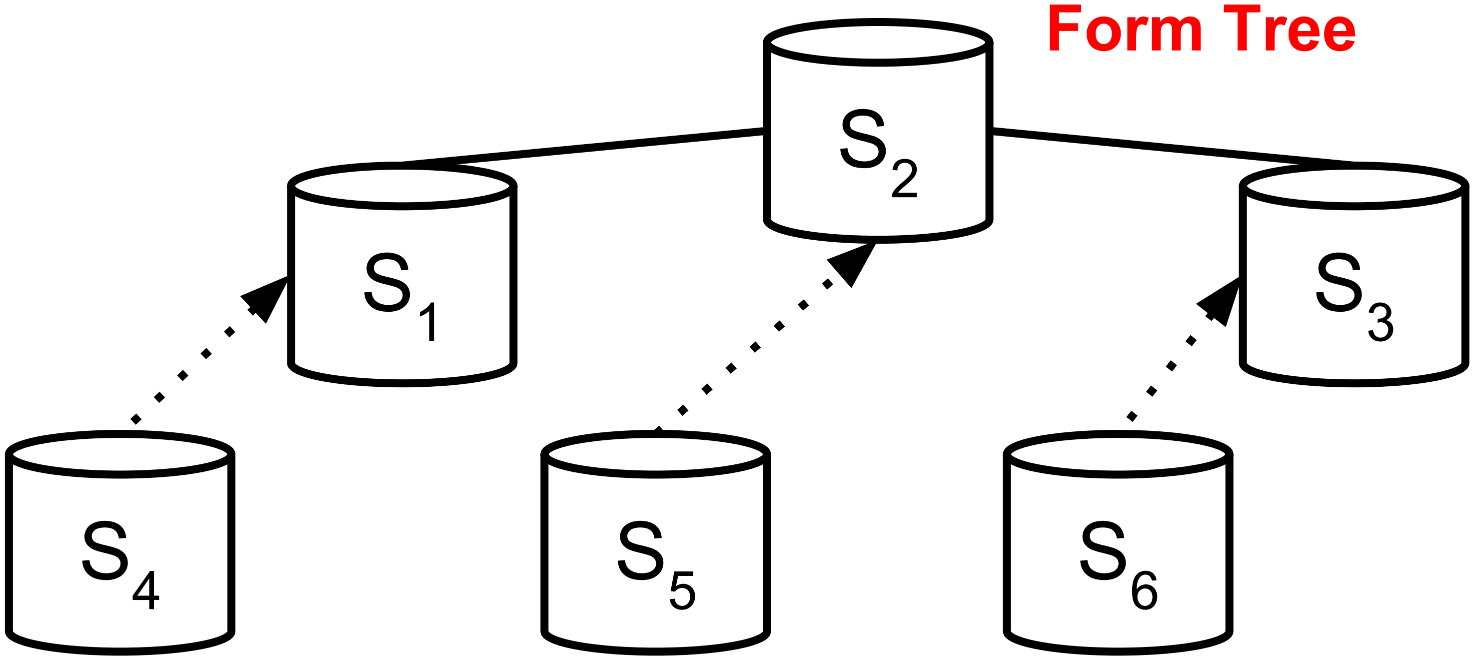
(Wolfson et al., TODS 1997)

Extensions (Network Topology)



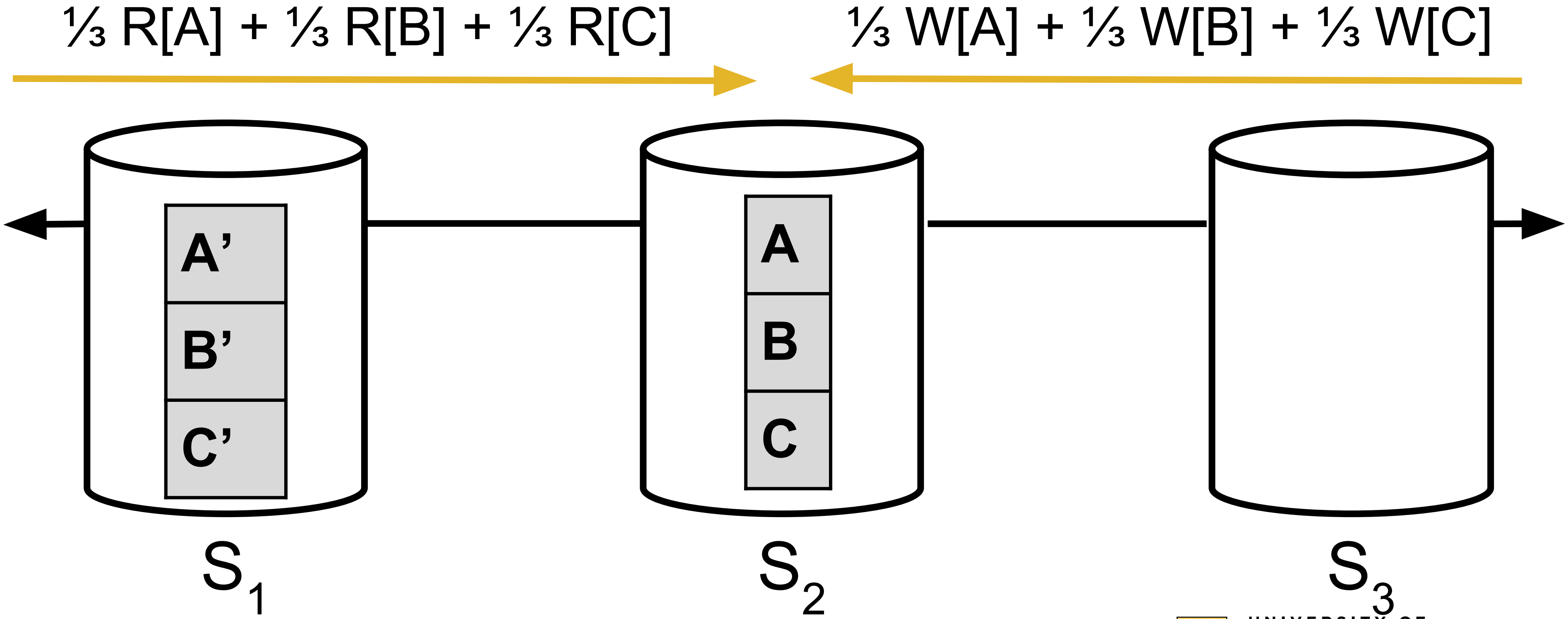
(Wolfson et al., TODS 1997)

Extensions (Network Topology)



(Wolfson et al., TODS 1997)

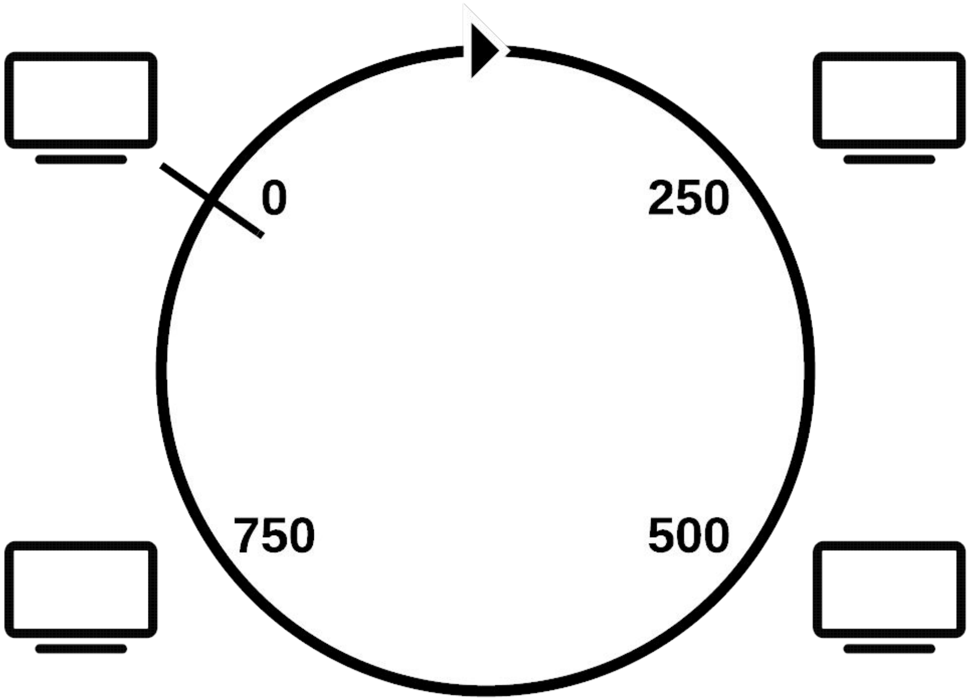
Extensions (Blocks)



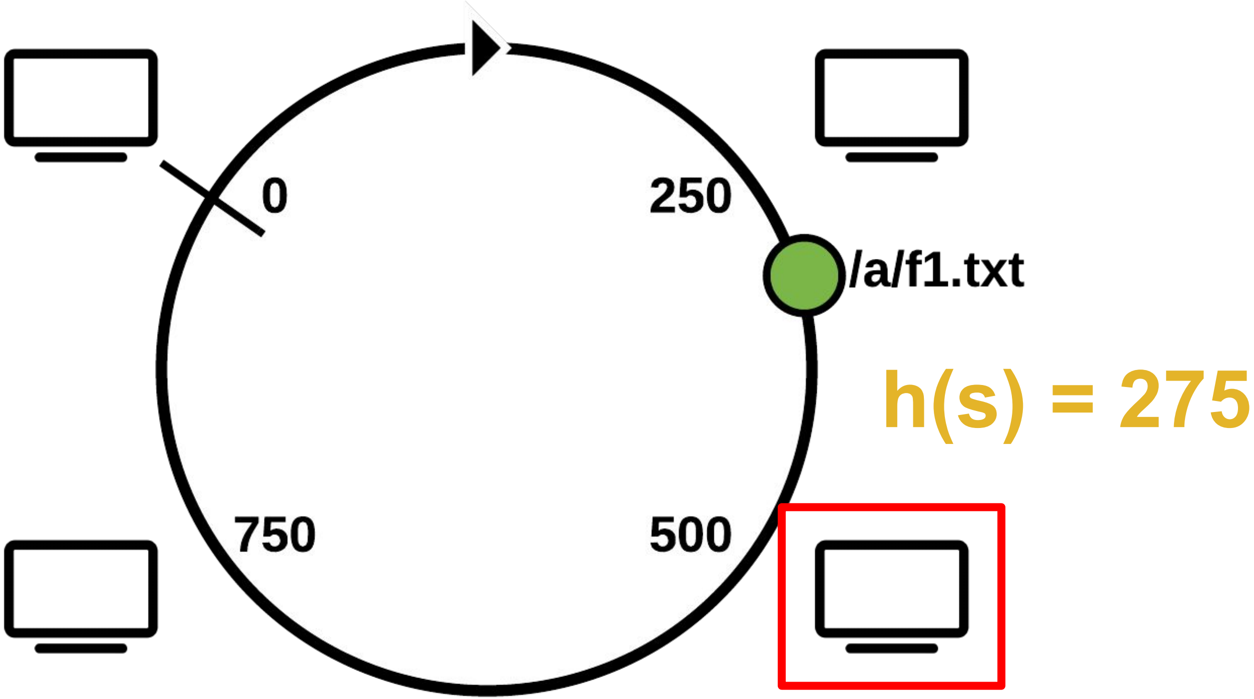
(Wolfson et al., TODS 1997)

UNIVERSITY OF WATERLOO

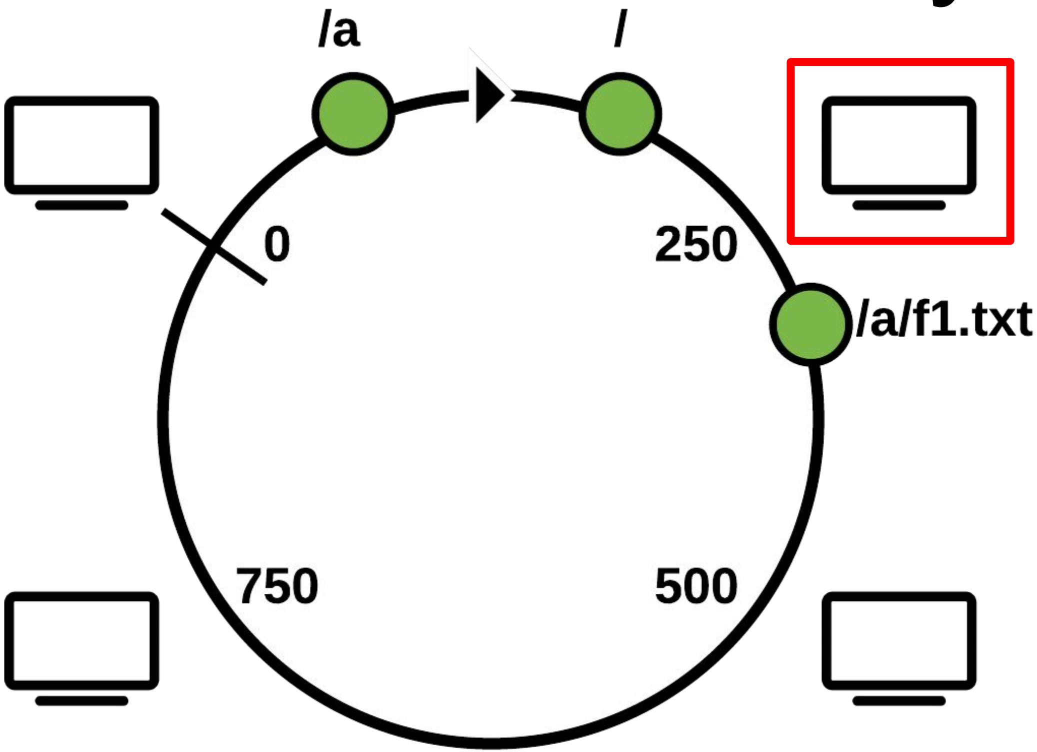
Peer-to-Peer File Systems



Hash-Based P2P File System

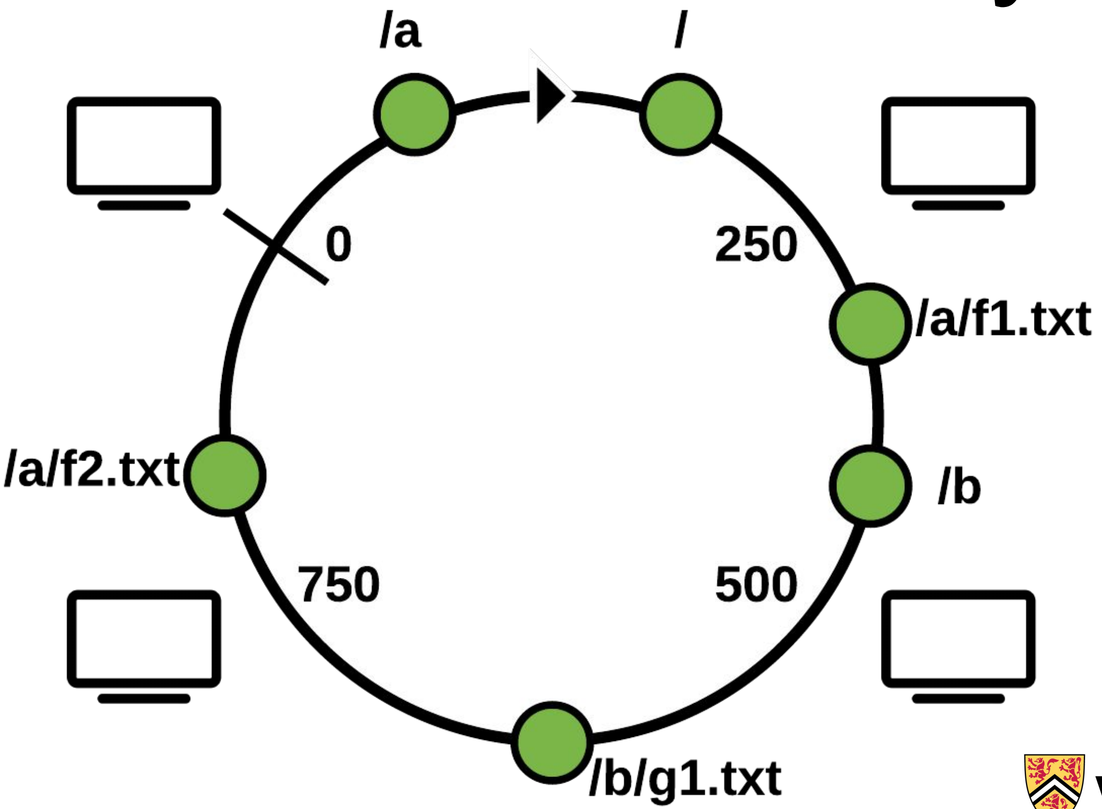


Hash-Based P2P File System



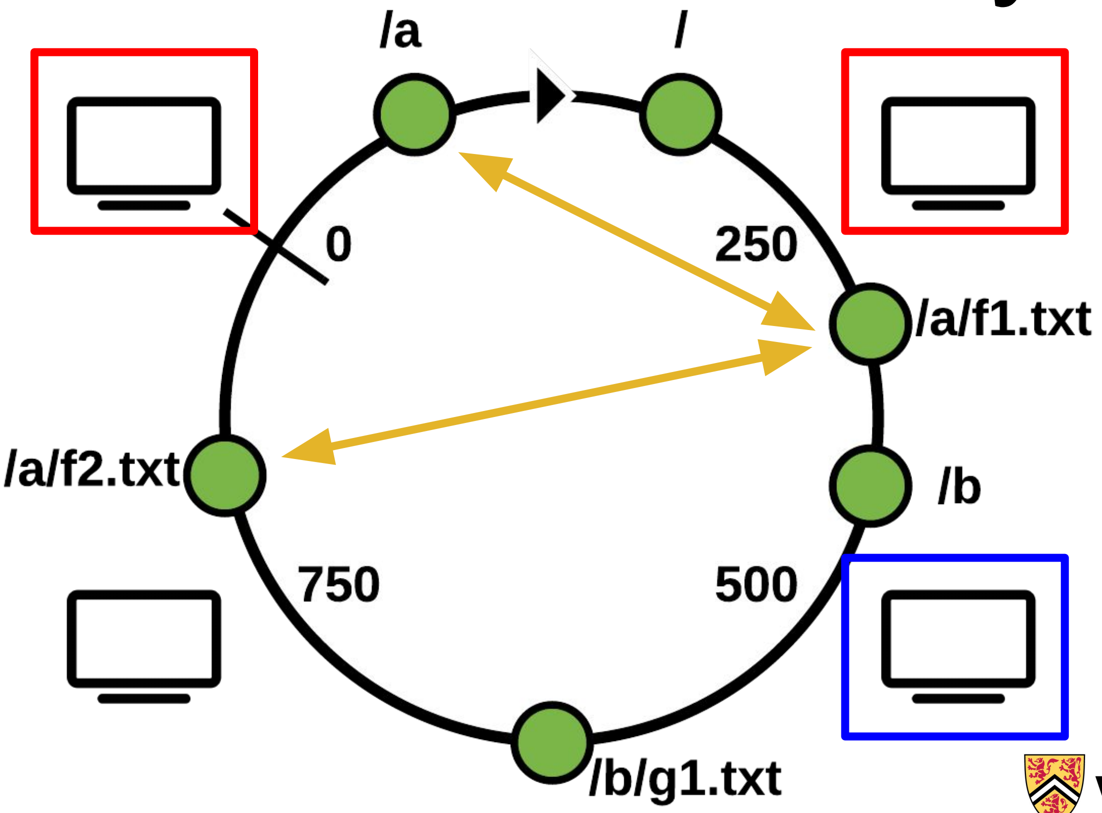
UNIVERSITY OF WATERLOO

Hash-Based P2P File System

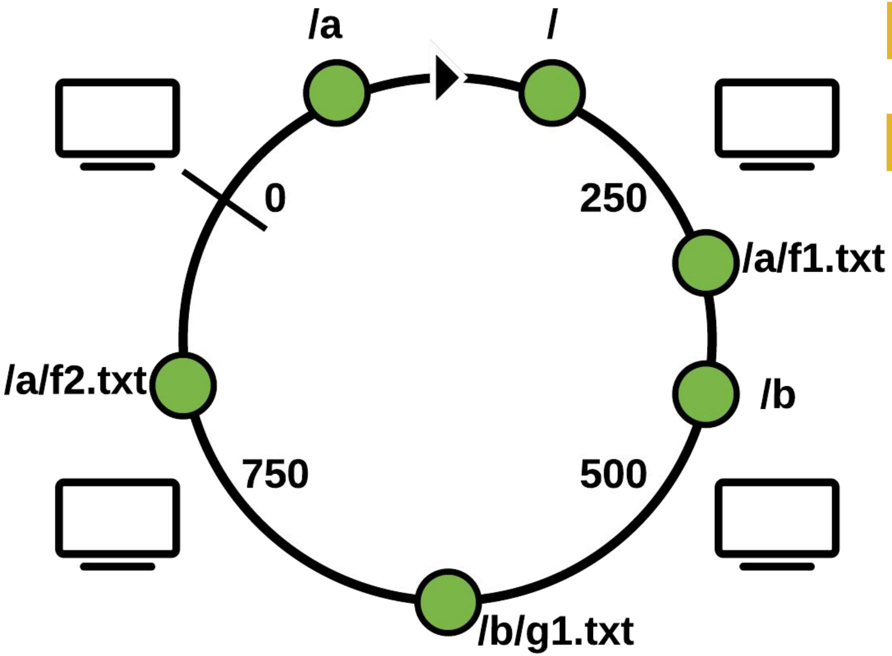


UNIVERSITY OF WATERLOO

Hash-Based P2P File System

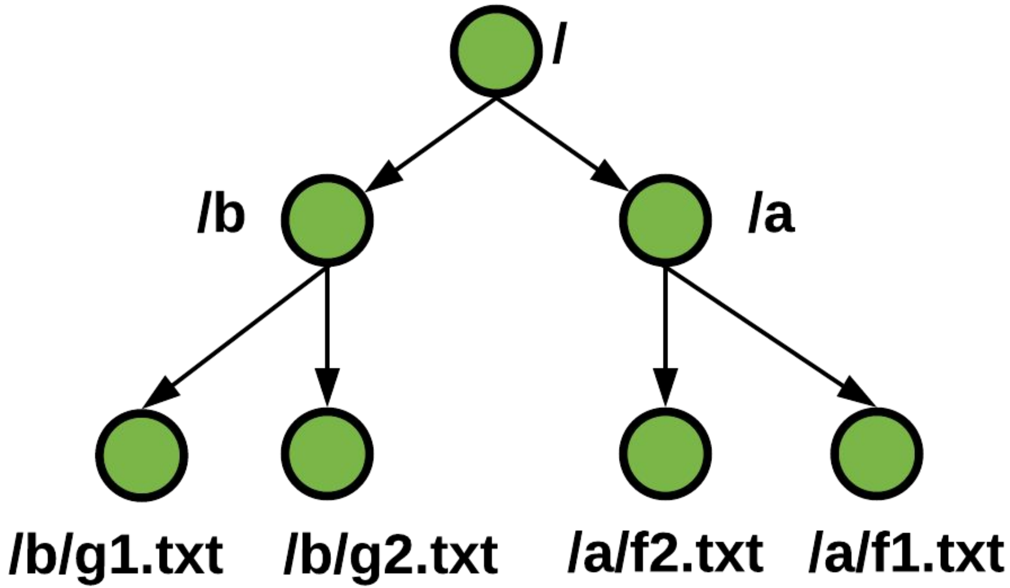


Hash-Based P2P FS

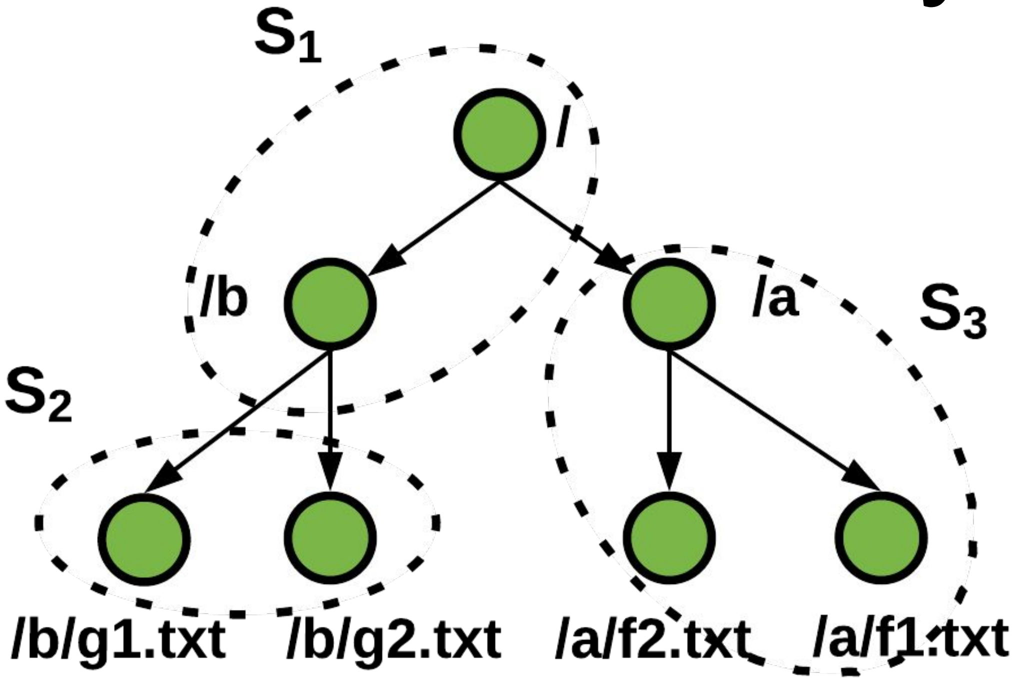


Balanced load, but poor access locality!

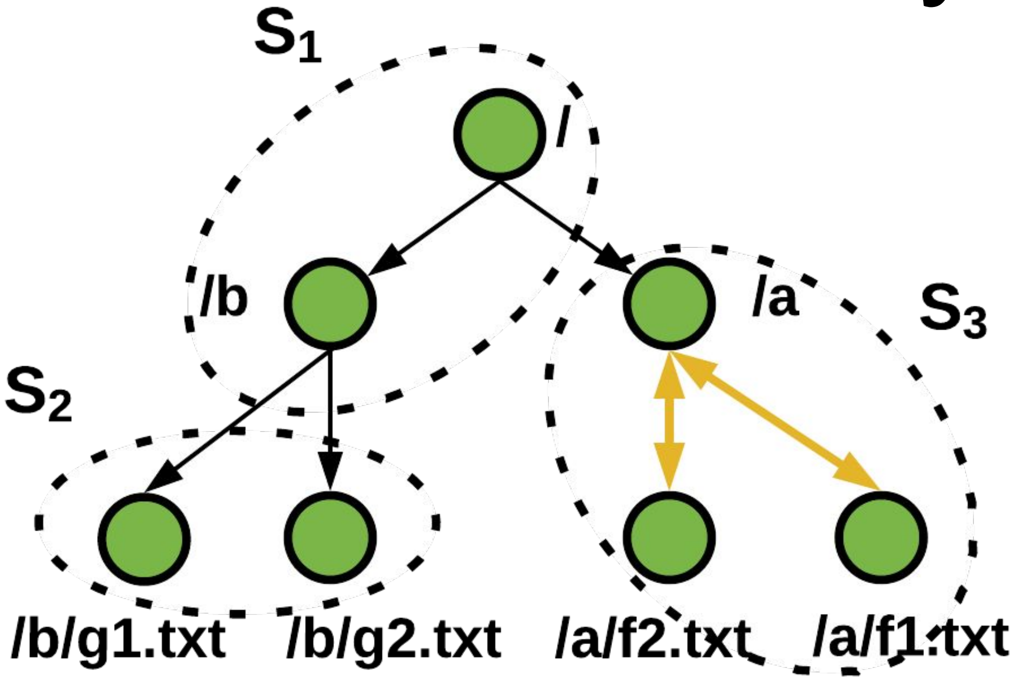
Hierarchical P2P File Systems



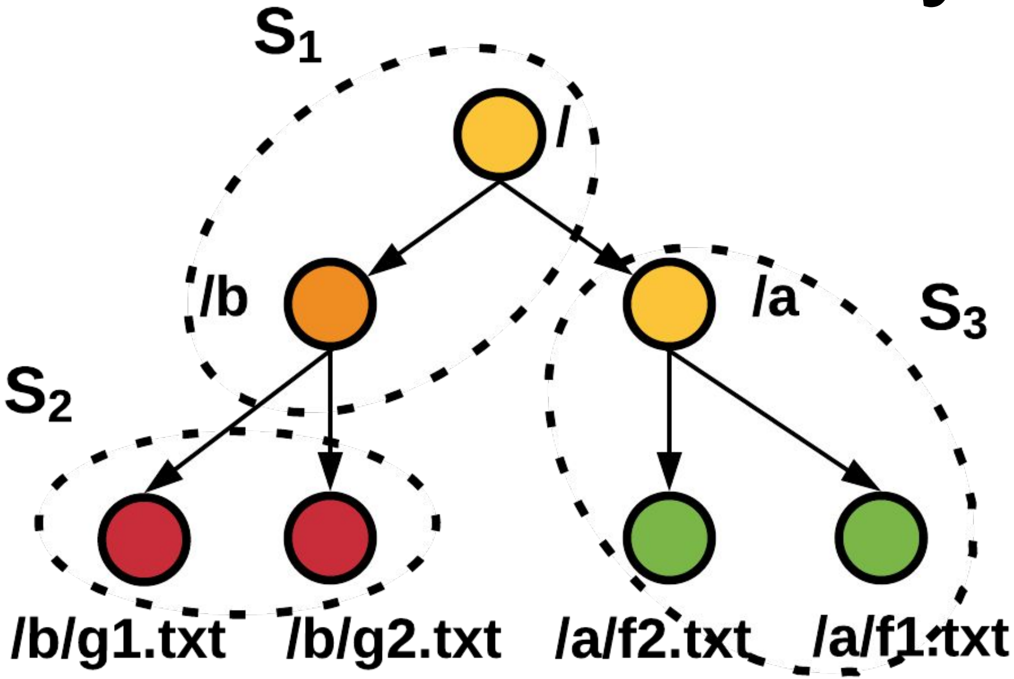
Hierarchical P2P File Systems



Hierarchical P2P File Systems

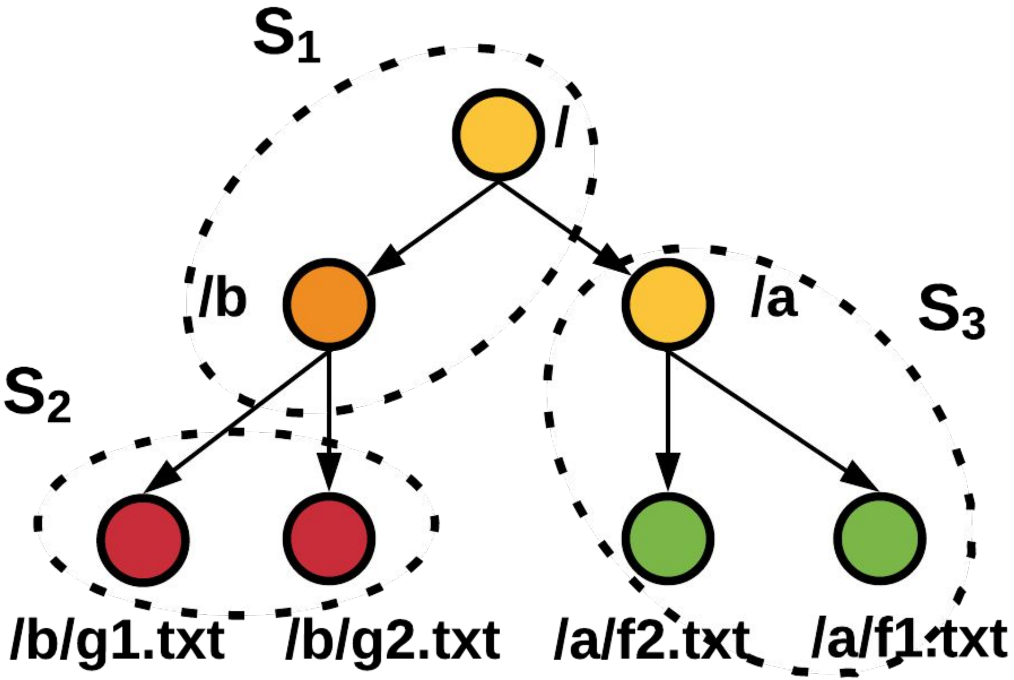


Hierarchical P2P File Systems



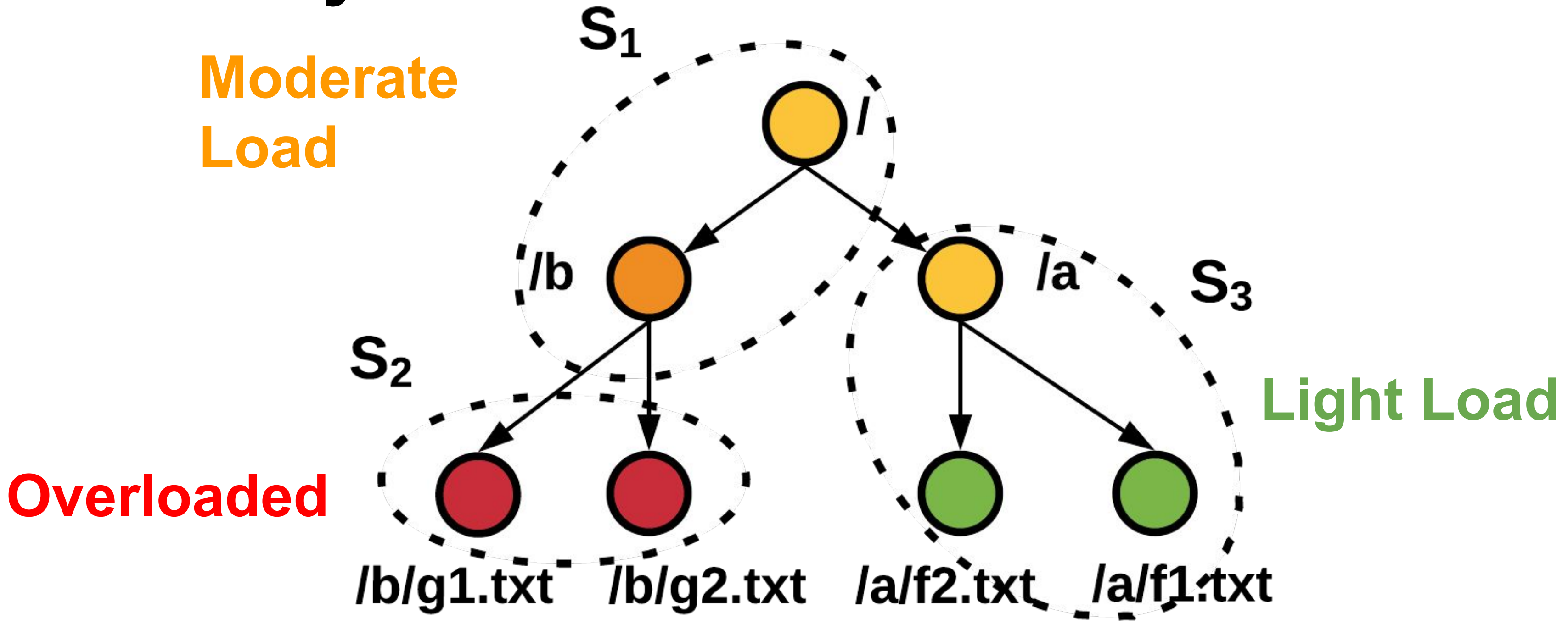
UNIVERSITY OF WATERLOO

Hierarchical P2P File System



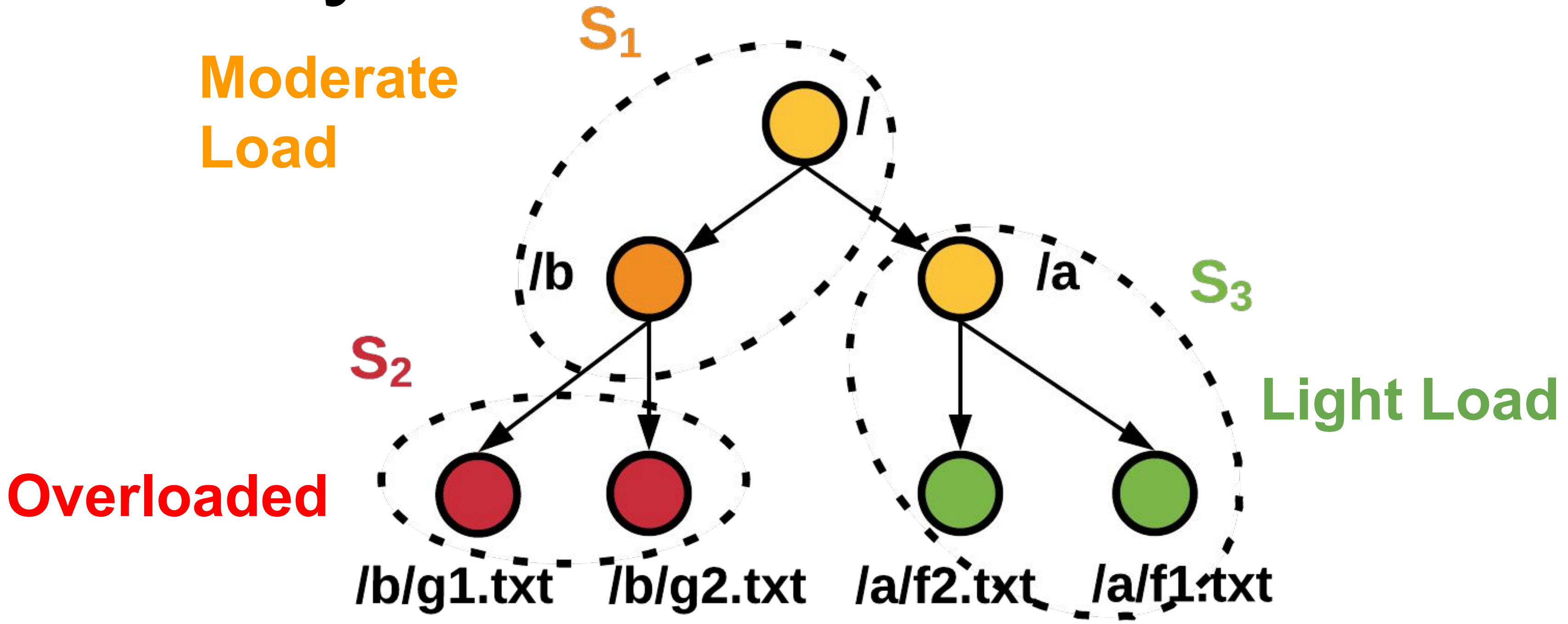
Good locality, but poor balance

Locality **and** Load Balance



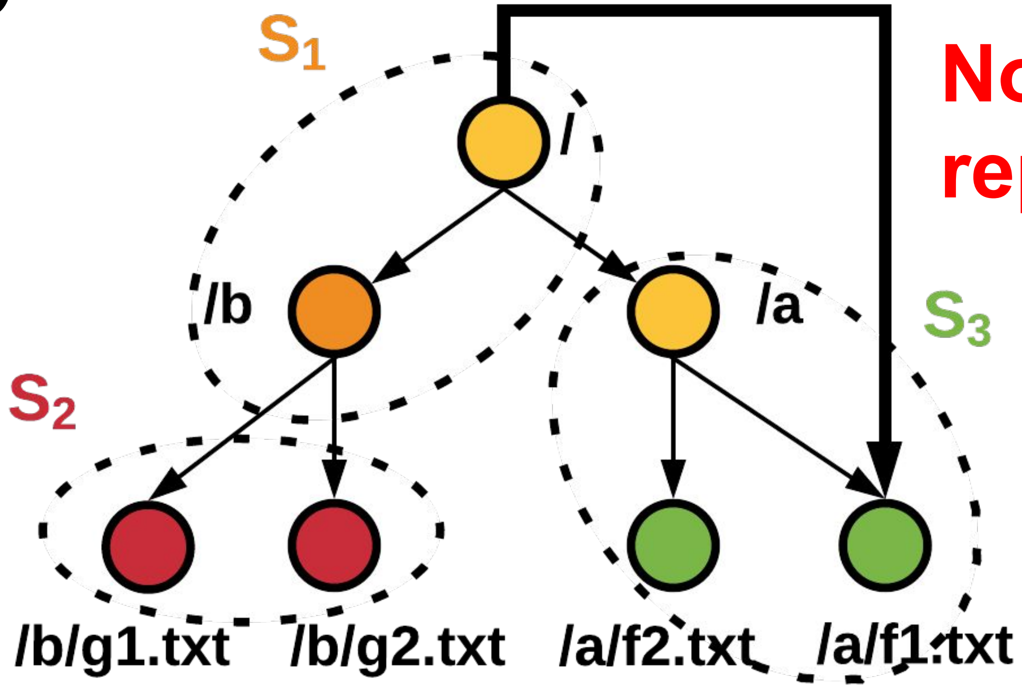
(Gopalakrishnan et al., ICDCS'04)

Locality **and** Load Balance



(Gopalakrishnan et al., ICDCS'04)

Locality **and** Load Balance

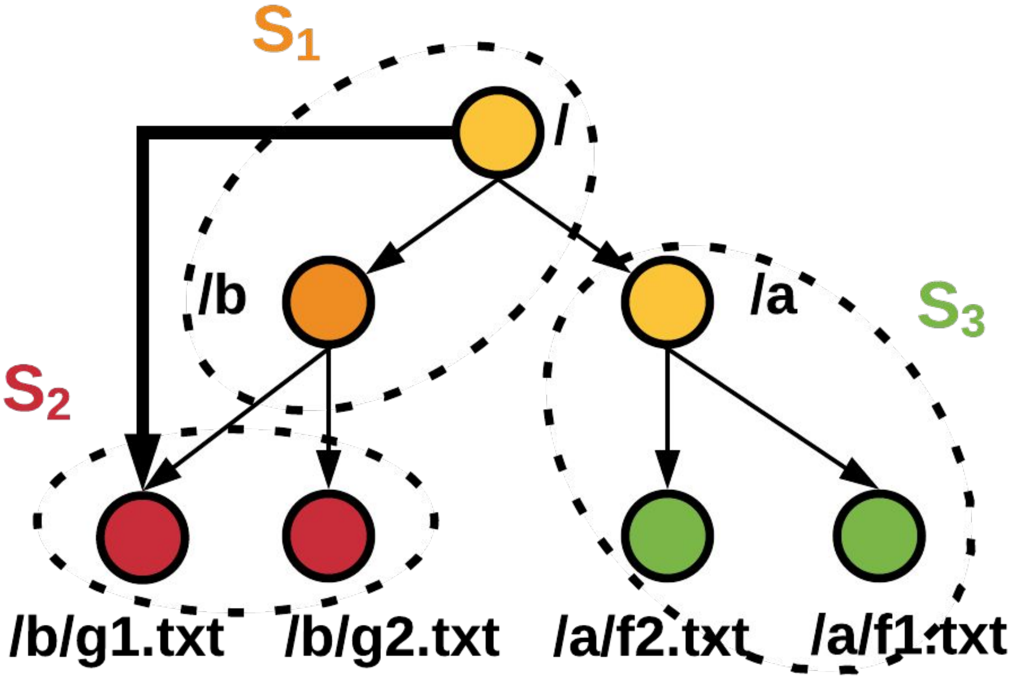


No replication

(Gopalakrishnan et al., ICDCS'04)

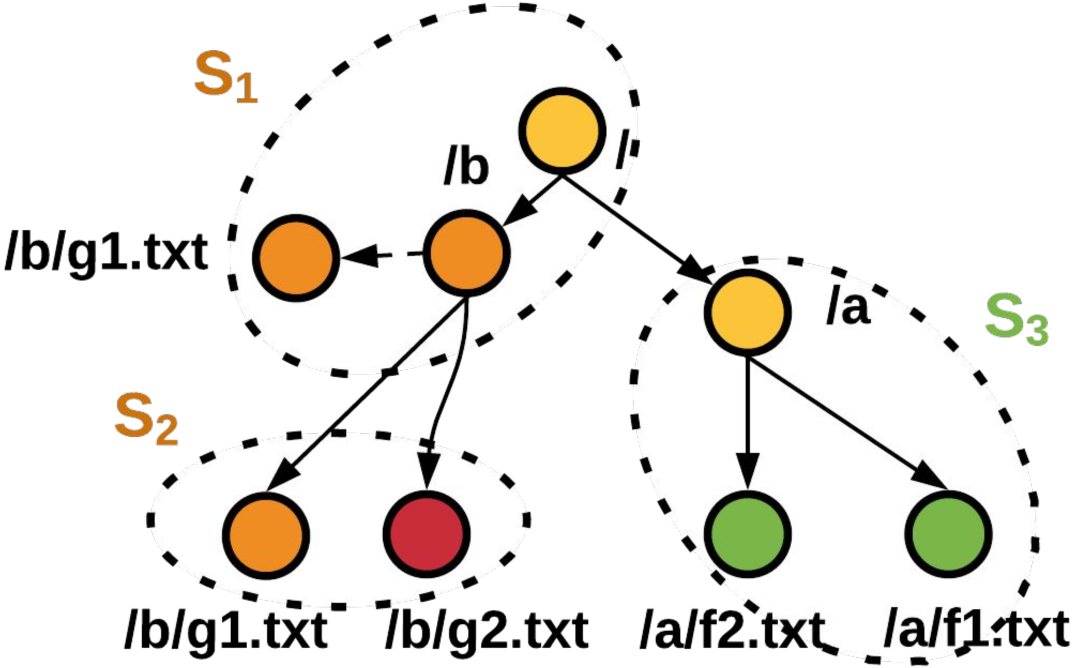
Locality **and** Load Balance

Replicate to balance load!



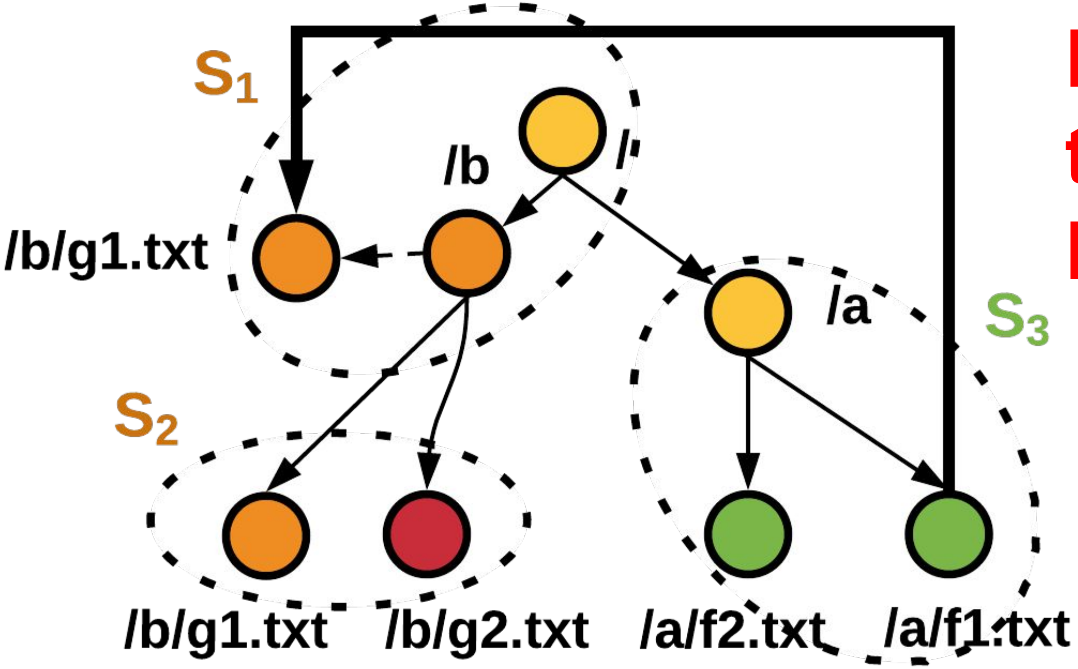
(Gopalakrishnan et al., ICDCS'04)

Locality **and** Load Balance



(Gopalakrishnan et al., ICDCS’04)

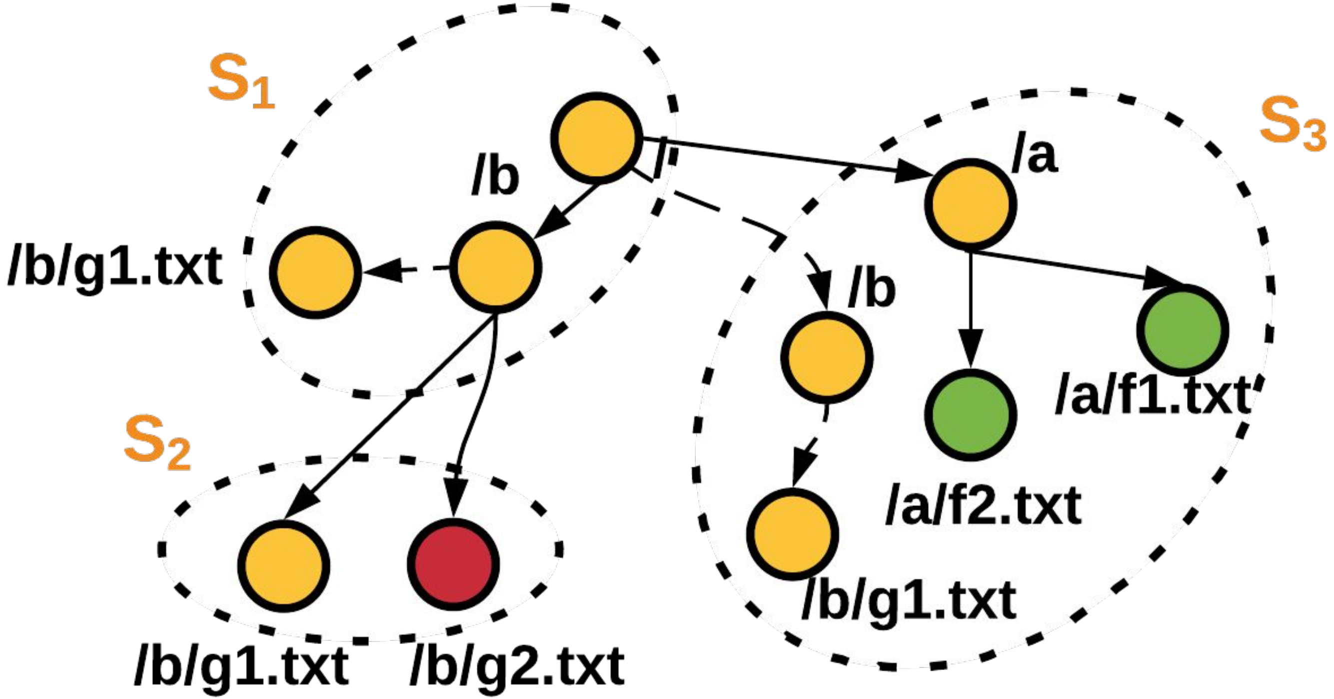
Locality **and** Load Balance



Replicate to balance load!

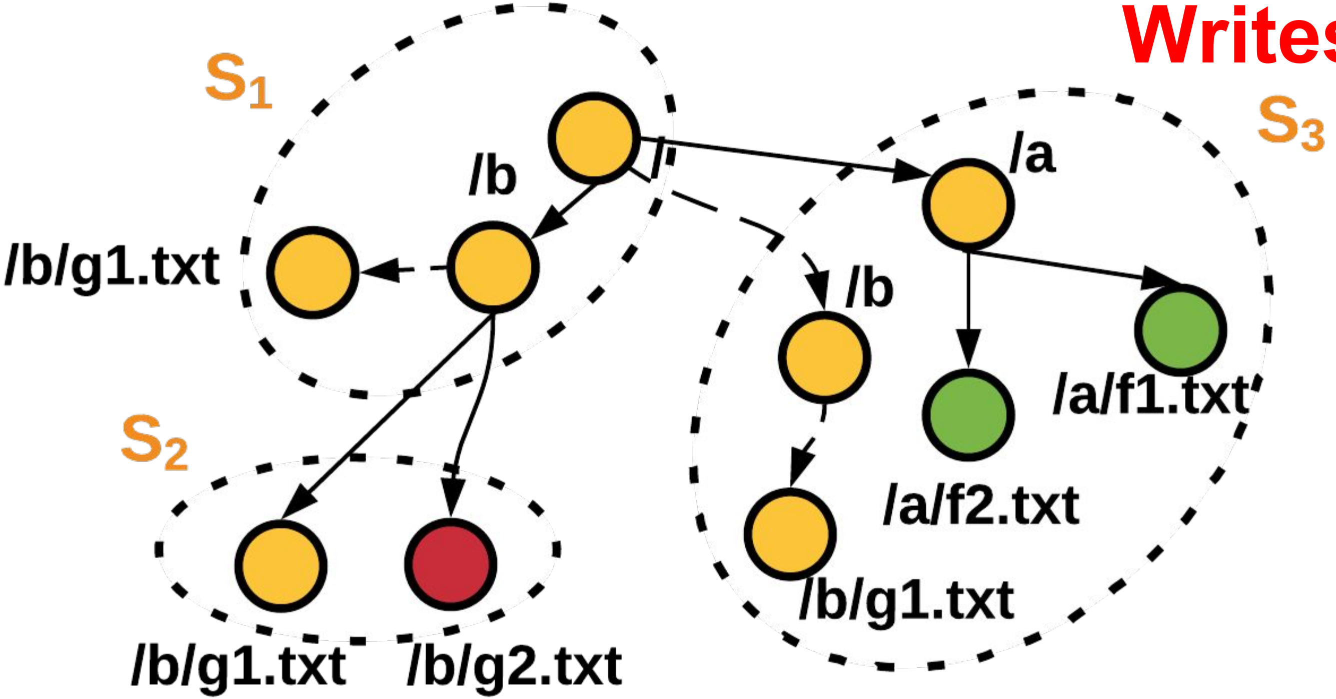
(Gopalakrishnan et al., ICDCS’04)

Locality **and** Load Balance



(Gopalakrishnan et al., ICDCS’04)

Decentralized Replicas



Writes?

(Gopalakrishnan et al., ICDCS'04)

Replication Decisions

- **How many replicas?**

  Decentralized decision

- **Where to place replicas?**

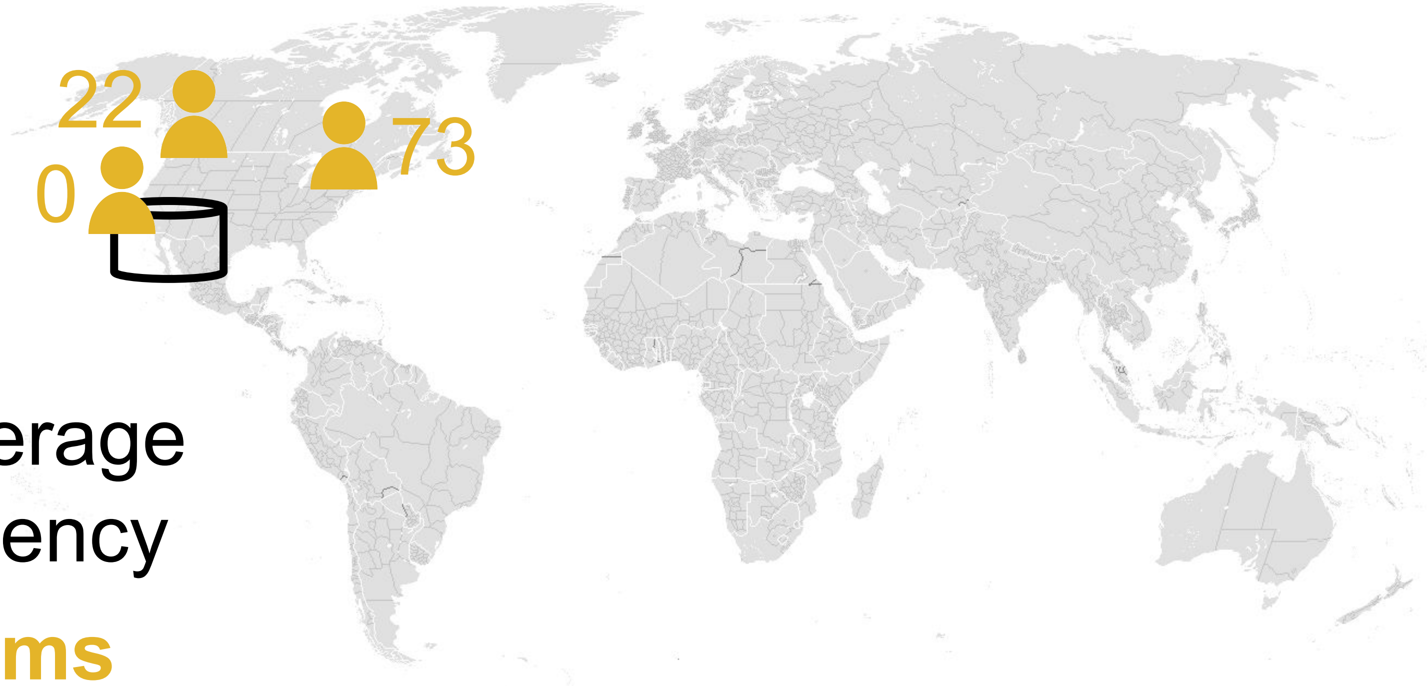
  At the requester

- **How to propagate updates?**

  ADR: Synchronous, P2P: read-only

(Wolfson et al., TODS 1997) (Gopalakrishnan et al., ICDCS'04)

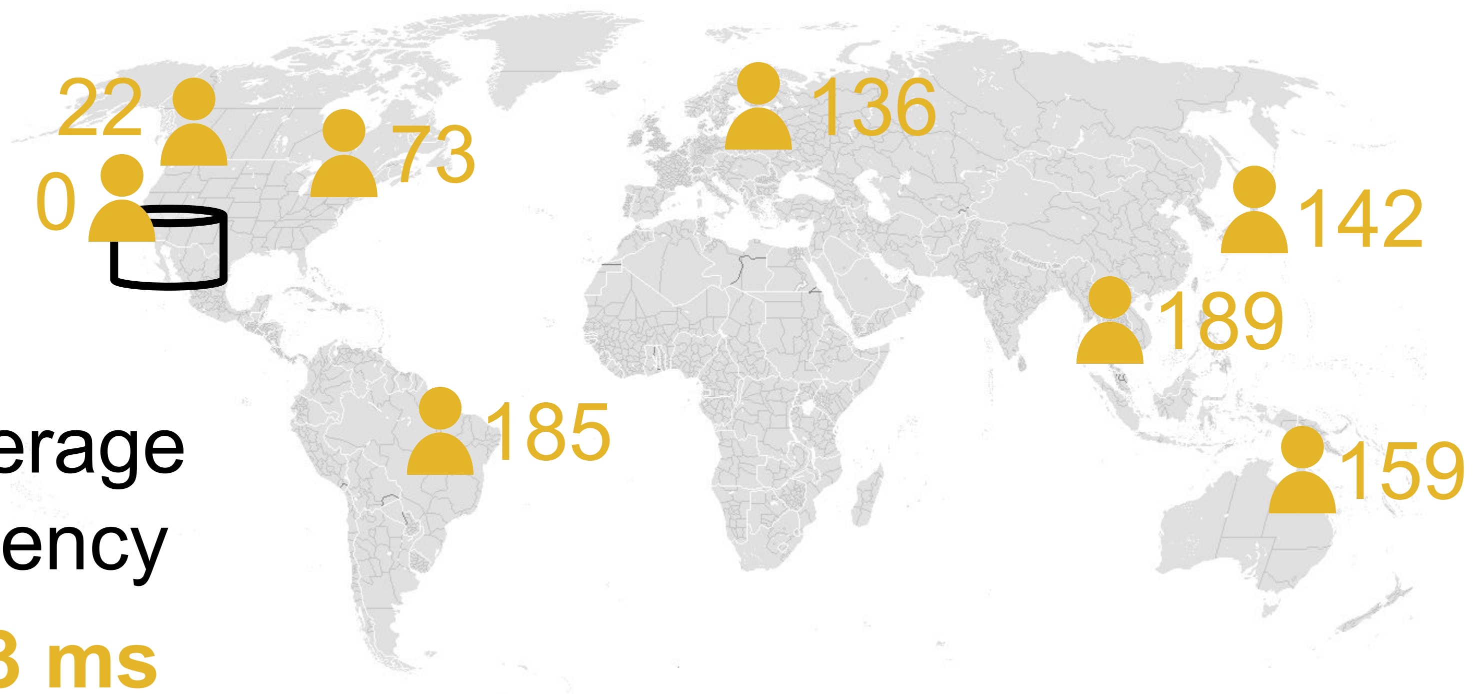
Global Scale Replication



Average Latency

31 ms

Global Scale Replication

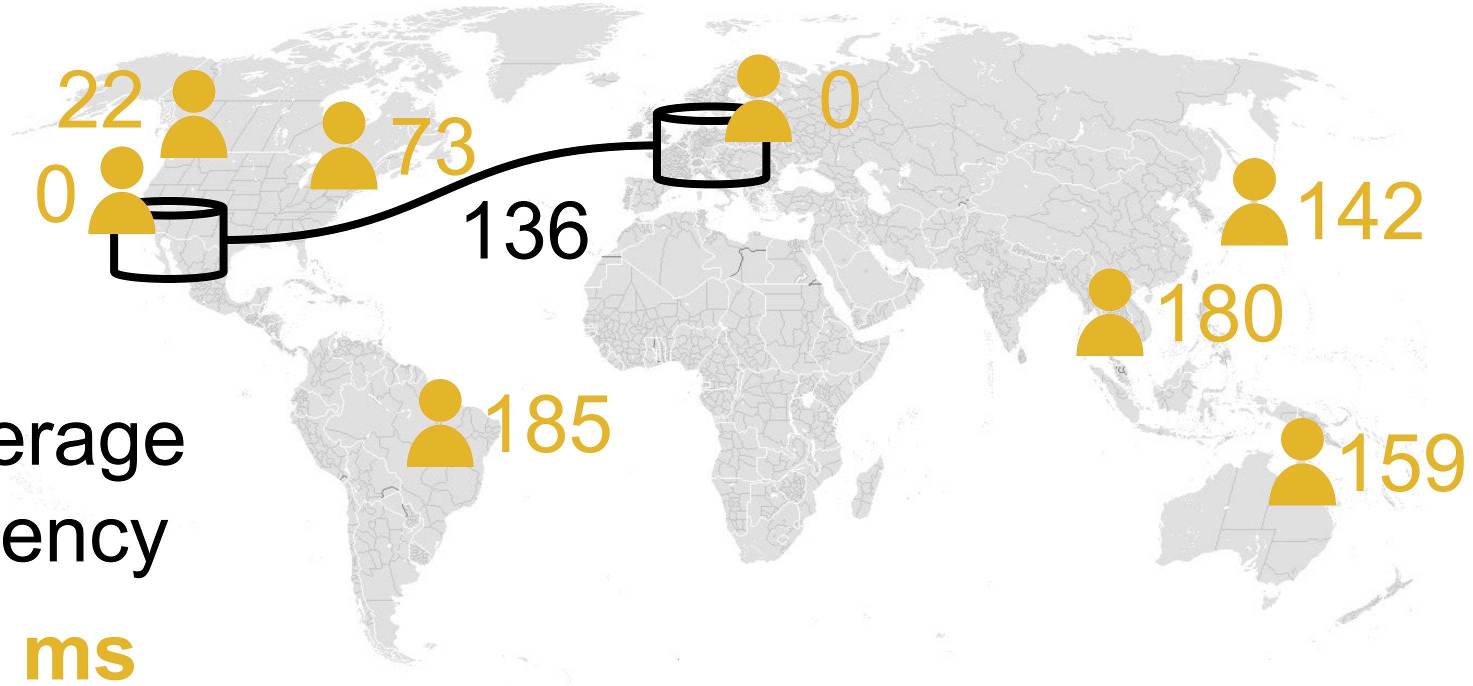


Average Latency

113 ms

UNIVERSITY OF WATERLOO

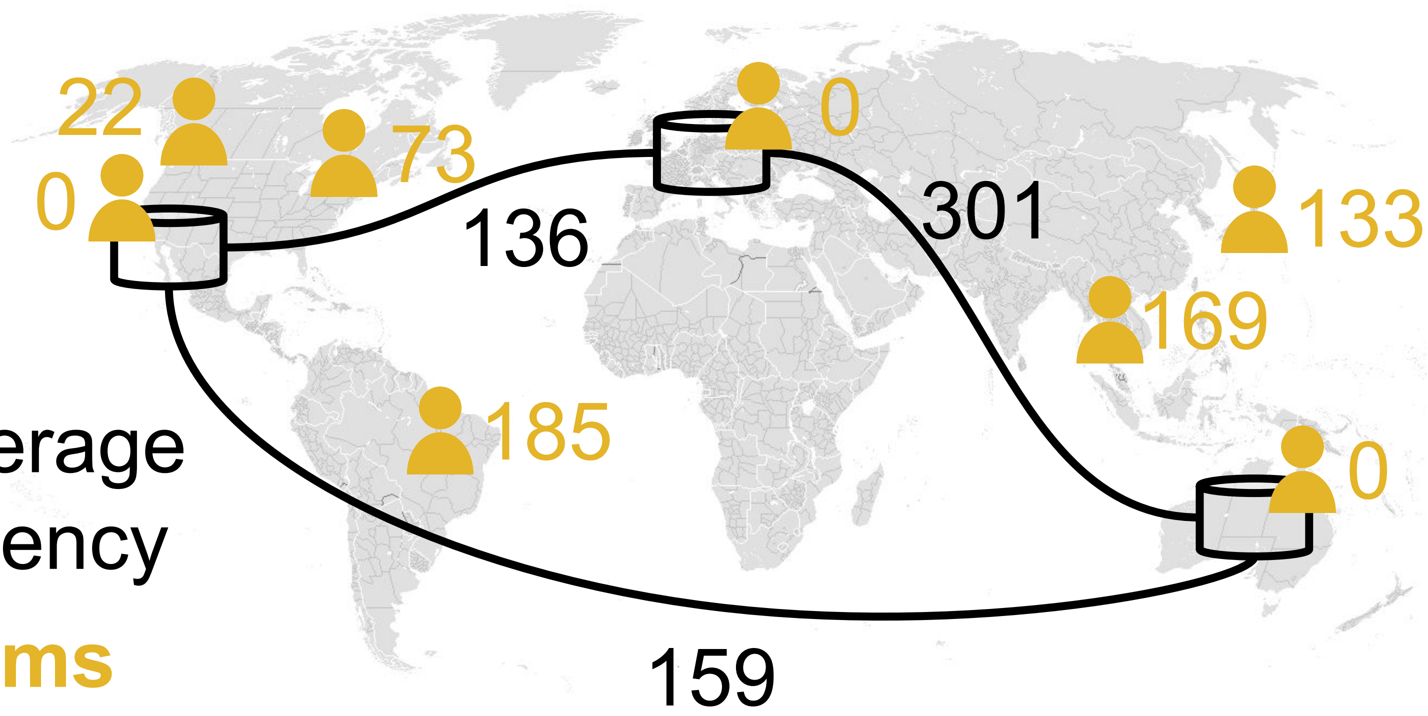
Global Scale Replication



Average Latency

95 ms

Global Scale Replication



Average Latency

64 ms

Global Scale Replication

Place data around the world

Minimize cost of access

Take me to your leader!

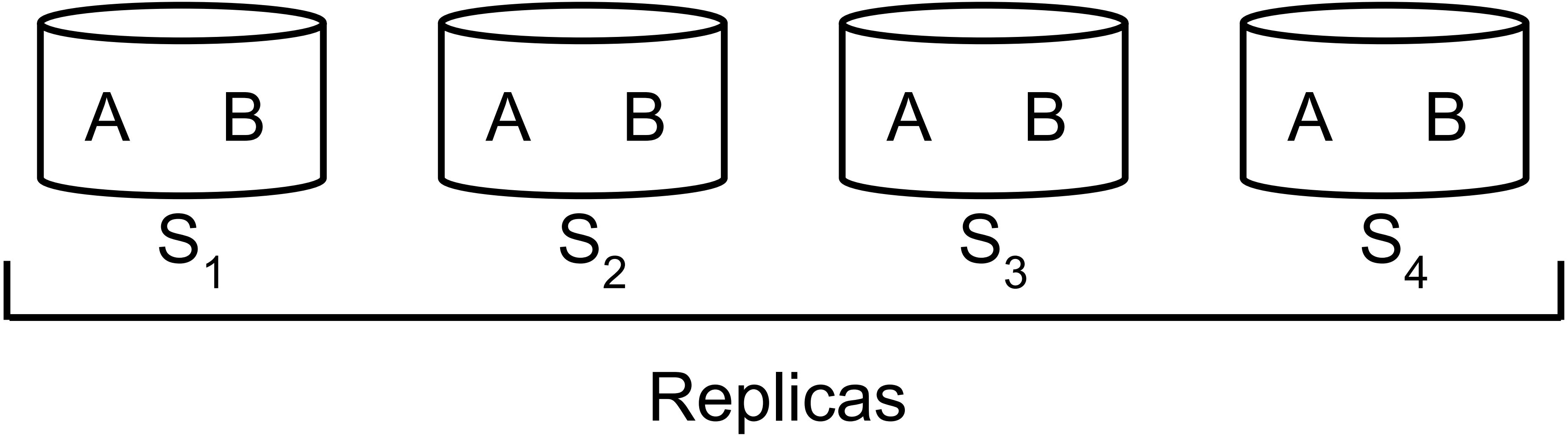
(Sharov et al., VLDB 2015)

GPlacer

(Zakhary et al., EDBT 2018)

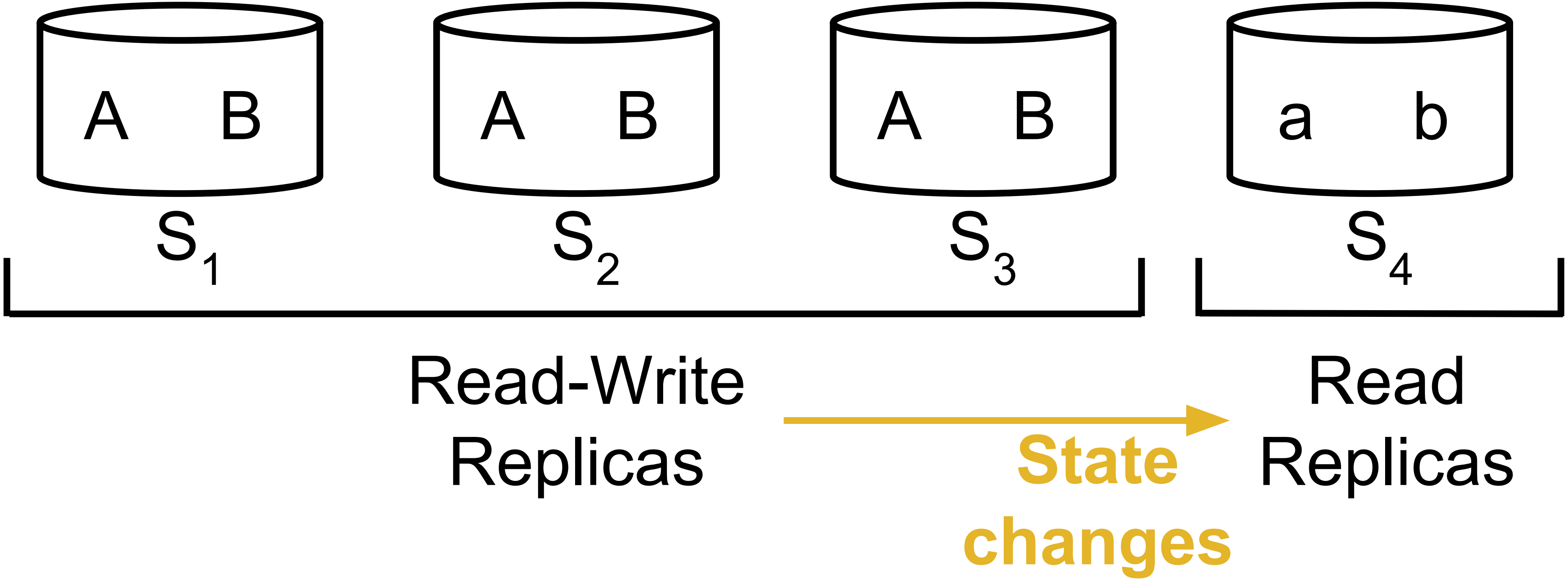
UNIVERSITY OF WATERLOO

Data Distribution Model



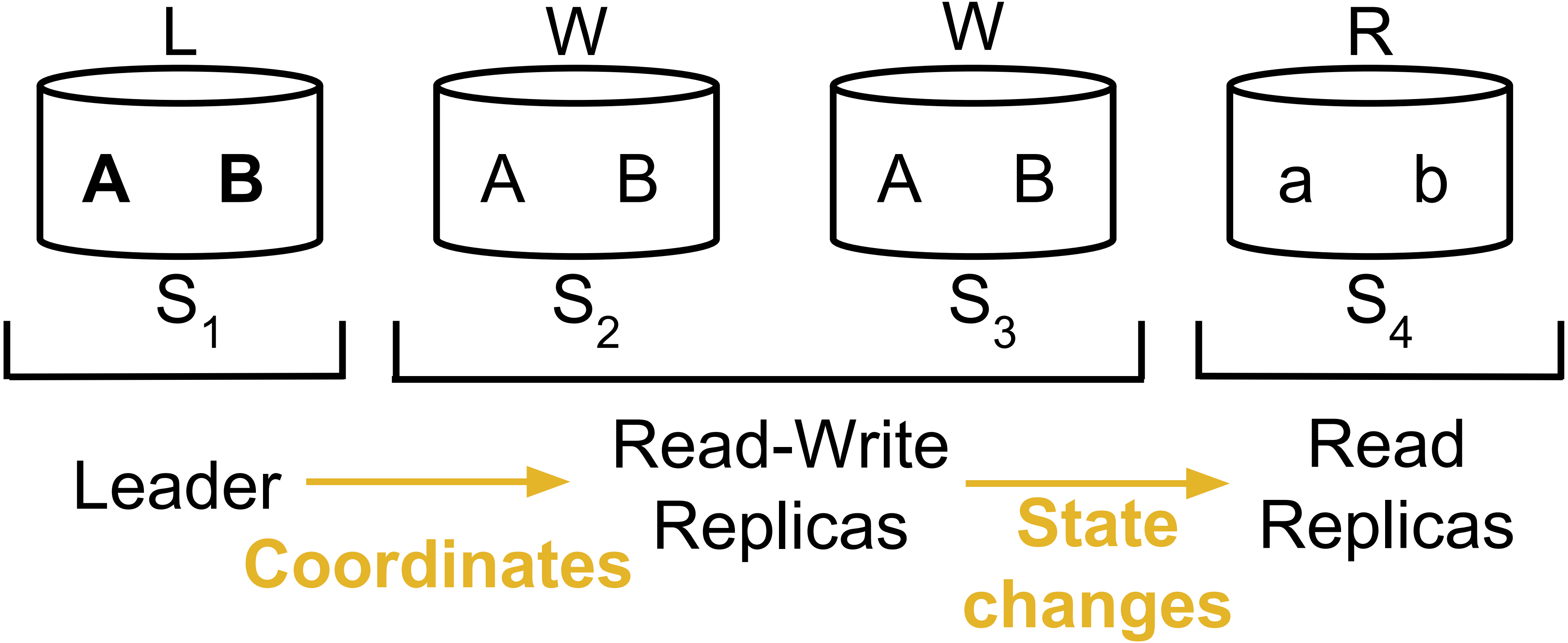
(Sharov et al., VLDB 2015)

Data Distribution Model



(Sharov et al., VLDB 2015)

Data Distribution Model



(Sharov et al., VLDB 2015)

Global Replication Problem

- Select replicas
- Assign replica roles (read or read-write)
- Assign leader

(Sharov et al., VLDB 2015)

Global Replication Problem

- Select replicas
- Assign replica roles (read or read-write)
- Assign leader

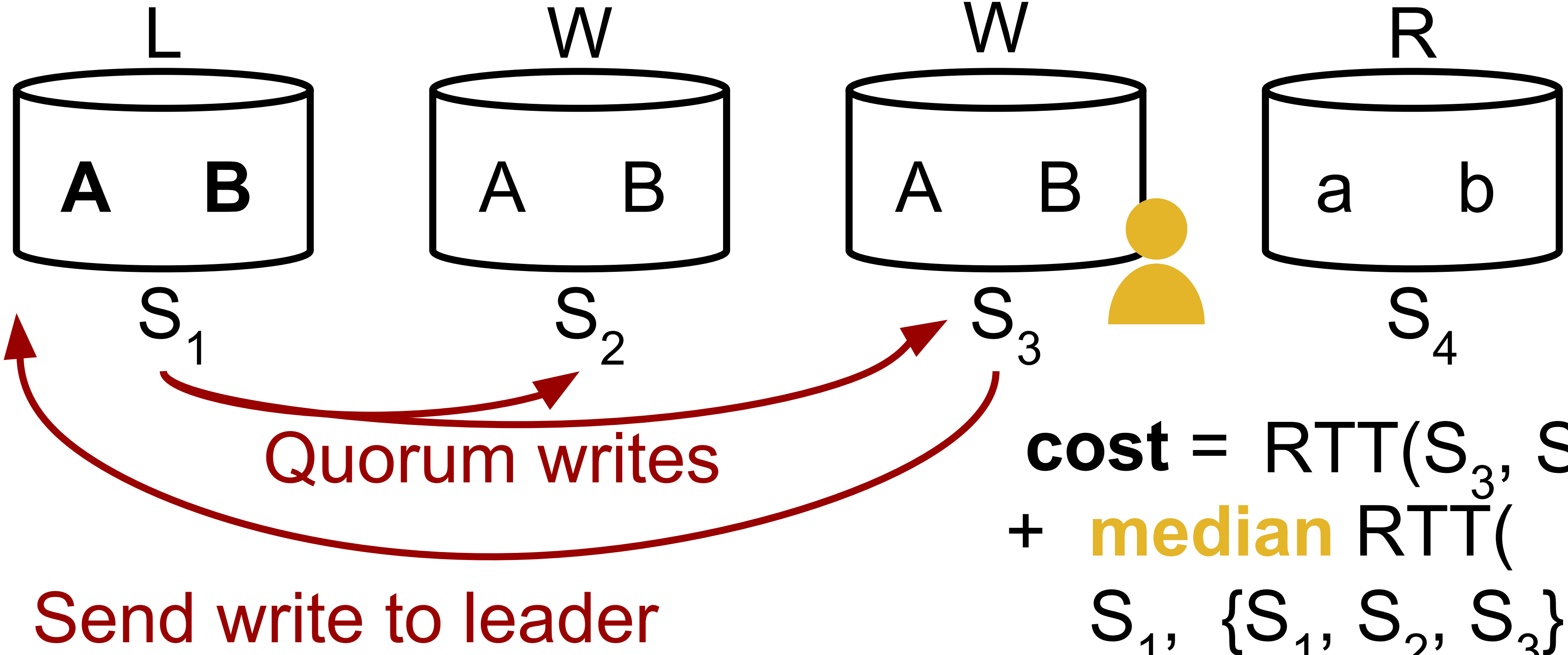
(Sharov et al., VLDB 2015)

Assign Leader

Leader: site that **minimizes access costs**

(Sharov et al., VLDB 2015)

Write transaction cost



$\textbf{cost} = \text{RTT}(S_3, S_1)$
$+$ **median** $\text{RTT}(S_1, \{S_1, S_2, S_3\})$

(Sharov et al., VLDB 2015)

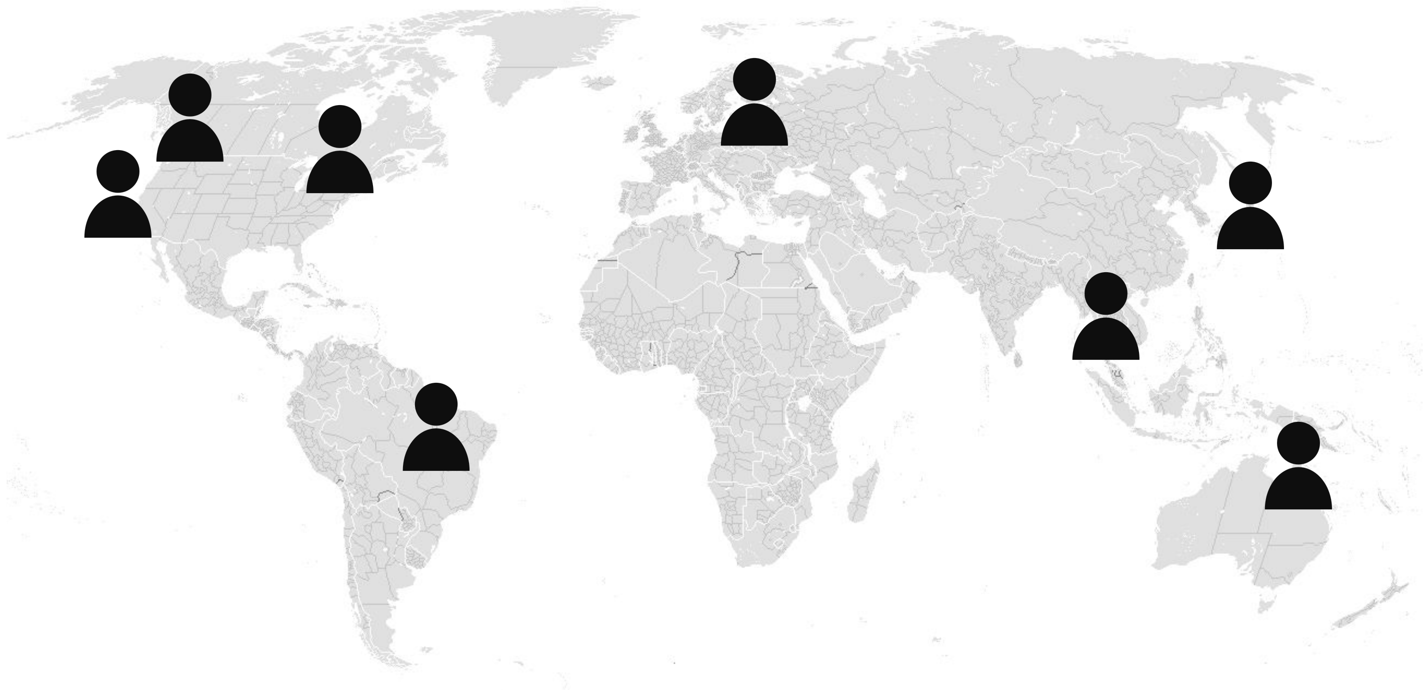
Assign Leader

Leader: site that **minimizes access costs**

Client cost: RTT(client, replica) + cost(transaction)

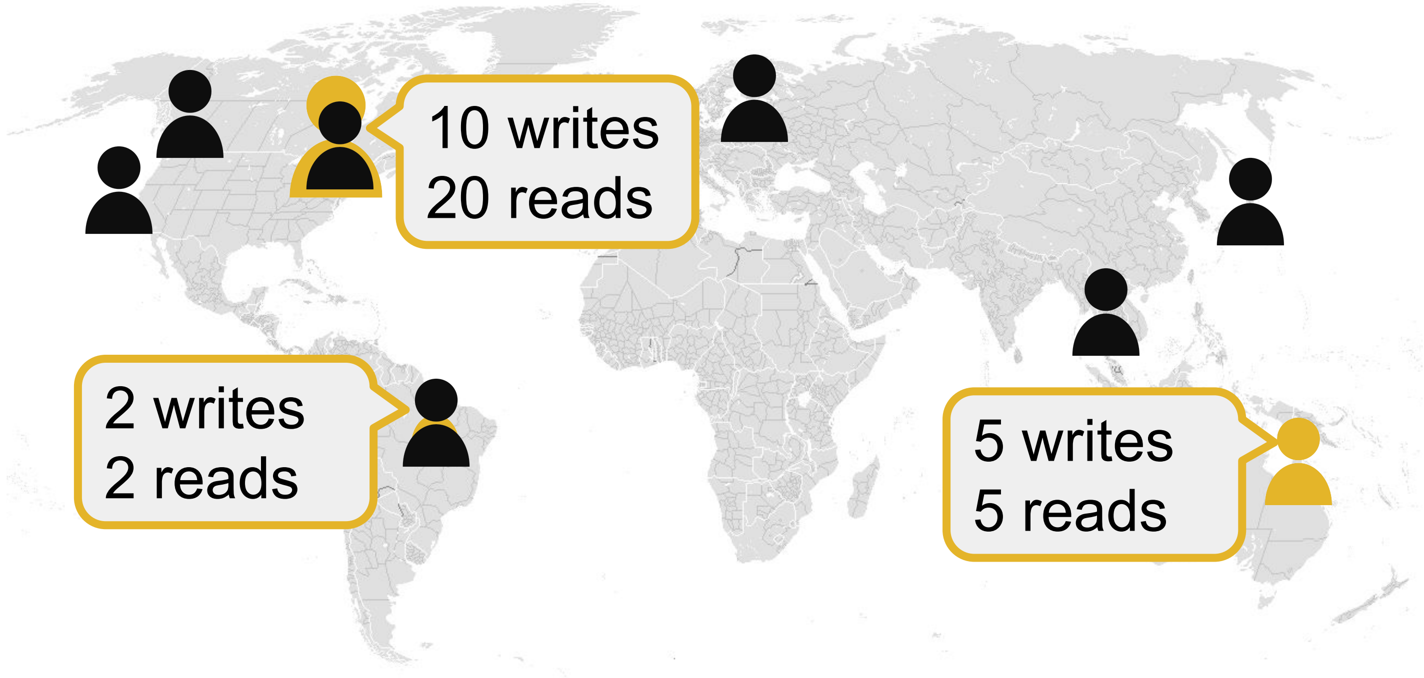
(Sharov et al., VLDB 2015)

Weighting Client Cost



(Sharov et al., VLDB 2015)

Weighting Client Cost



(Sharov et al., VLDB 2015)

Assign Leader

Leader: site that **minimizes access costs**

Client cost: RTT(client, replica) + cost(transaction)

Cost: Weighted average of client costs

(Sharov et al., VLDB 2015)

Global Replication Problem

- Select replicas
- Assign replica roles (read or read-write)
- Assign leader

(Sharov et al., VLDB 2015)

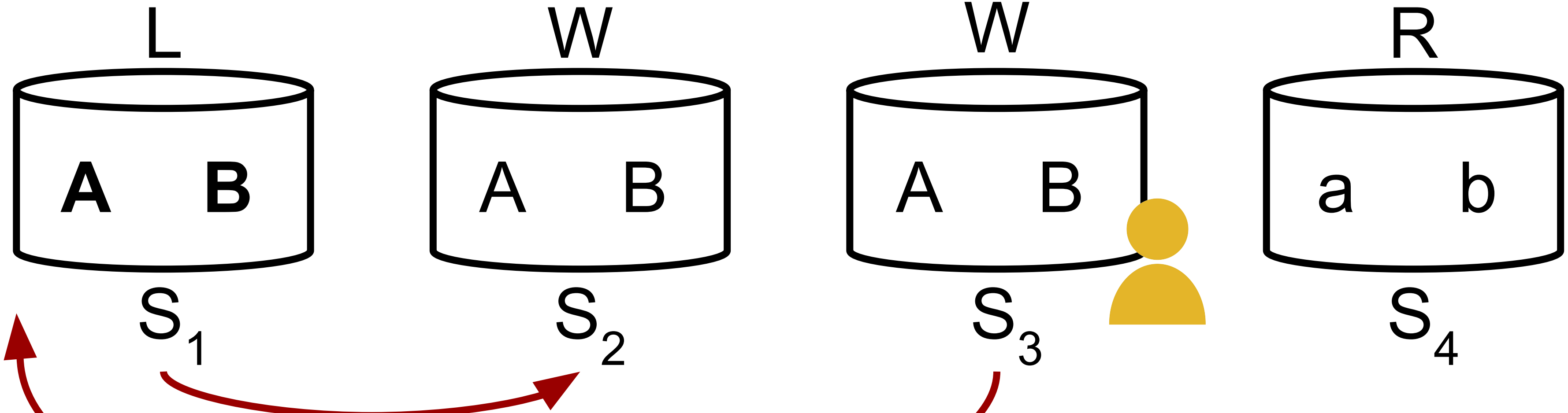
Assign Replica Roles

Leader: **minimizes median RTT** to read-write replicas

Read-write replicas:

(Sharov et al., VLDB 2015)

Write transaction cost



$$\textbf{cost} = \text{RTT}(S_3, S_1) + \text{median RTT}(S_1, \{S_1, S_2, S_3\})$$

(Sharov et al., VLDB 2015)

Assign Replica Roles

Leader: **minimizes median RTT** to read-write replicas

Read-write replicas: Lowest RTT to leader

(Sharov et al., VLDB 2015)

Global Replication Problem

- Select replicas
- Assign replica roles (read or read-write)
- Assign leader

(Sharov et al., VLDB 2015)

UNIVERSITY OF WATERLOO

Replica selection

Leader: **minimizes median RTT** to read-write replicas

Read-write replicas: Lowest RTT to leader

Read replicas:

(Sharov et al., VLDB 2015)

Replica selection

Client cost: RTT(client, replica) + cost(transaction)

(Sharov et al., VLDB 2015)

Replica selection

Client cost: **RTT(client, replica)** +
cost(transaction)

Read replicas: Lowest RTT to **clients**

(Sharov et al., VLDB 2015)

Replica selection

Leader: **minimizes median RTT** to read-write replicas

Read-write replicas: Lowest RTT to leader

Read replicas: Lowest RTT to **clients**

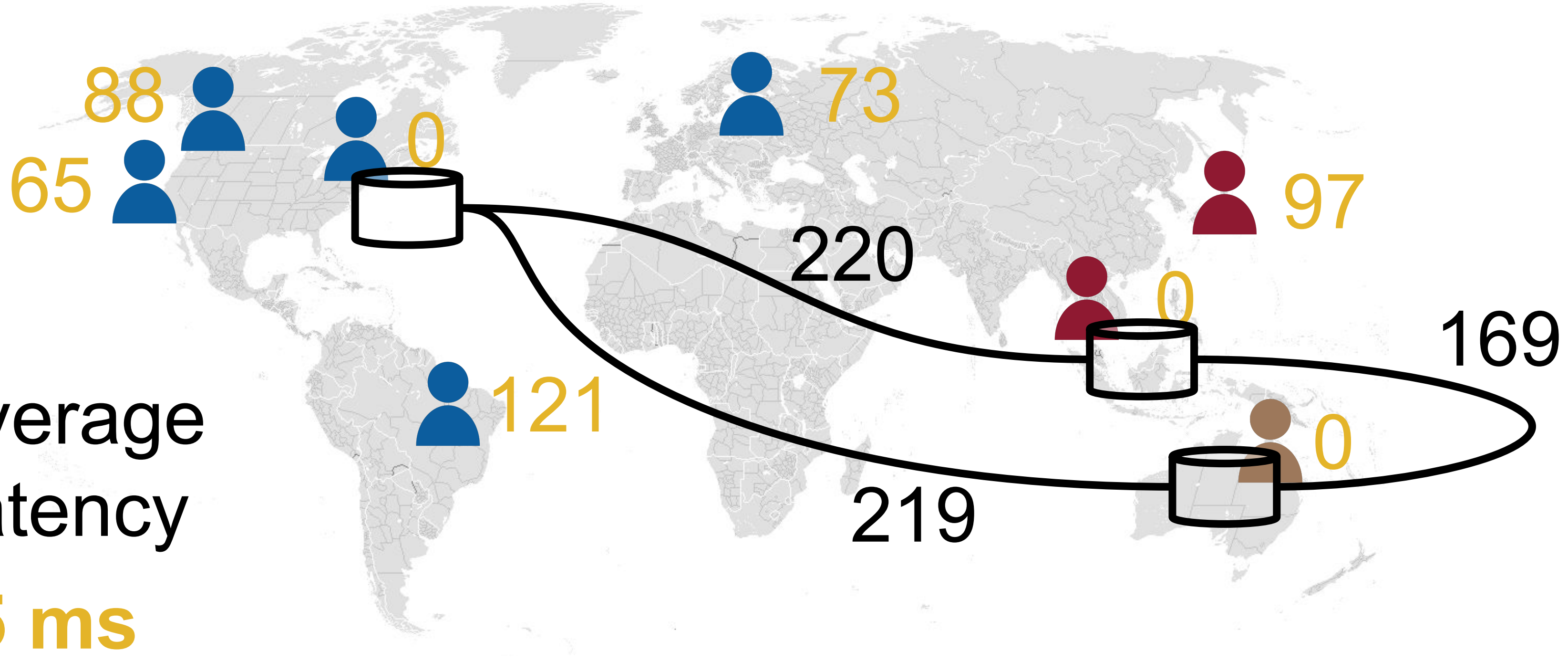
(Sharov et al., VLDB 2015)

K-Means Replica selection

(Sharov et al., VLDB 2015)

UNIVERSITY OF WATERLOO

K-Means Replica selection



Average Latency

55 ms

(Sharov et al., VLDB 2015)

UNIVERSITY OF WATERLOO

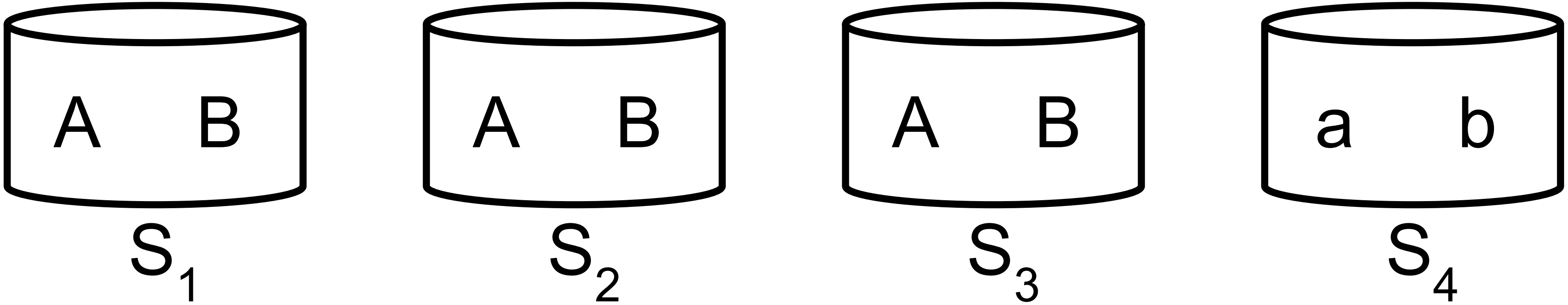
K-Means Replica selection

Select replicas

Assign leader and read-write replicas

(Sharov et al., VLDB 2015)

Leaderless Protocols

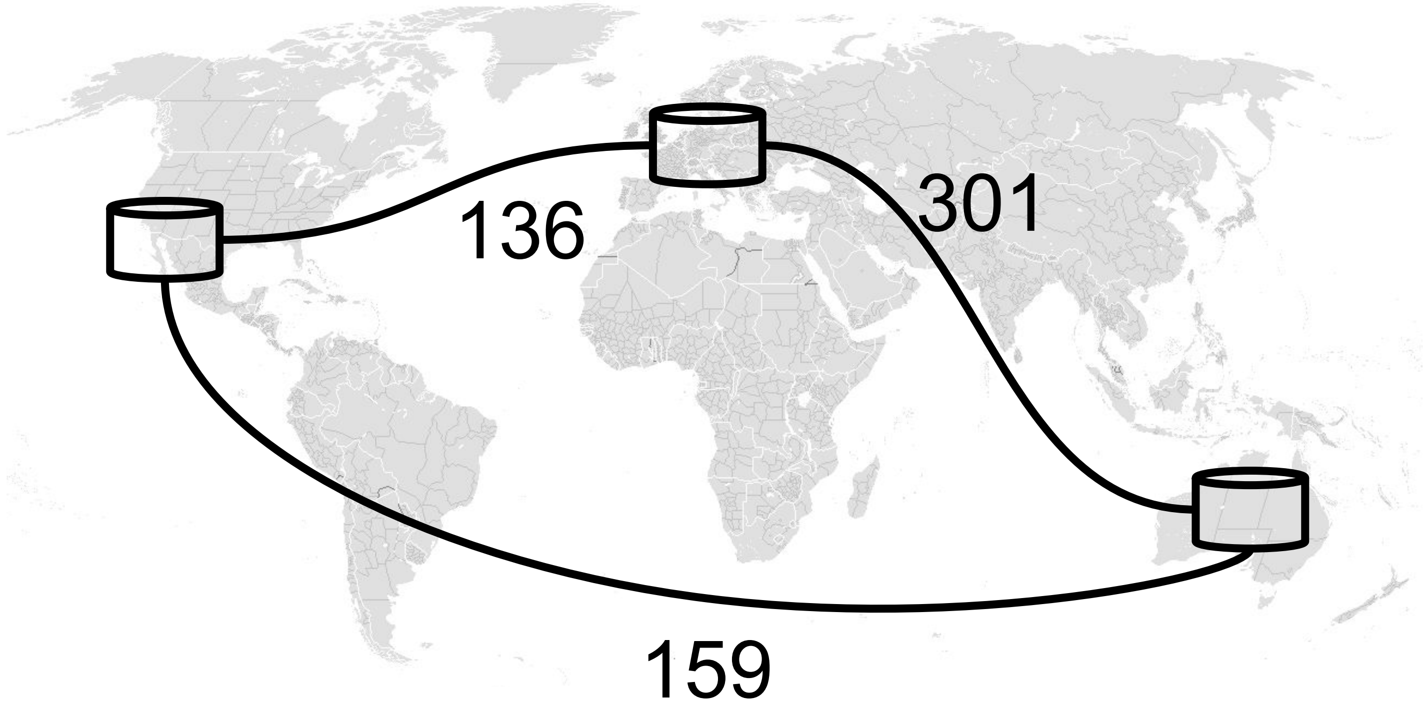


Any quorum member can coordinate

(Zakhary et al., EDBT 2018)

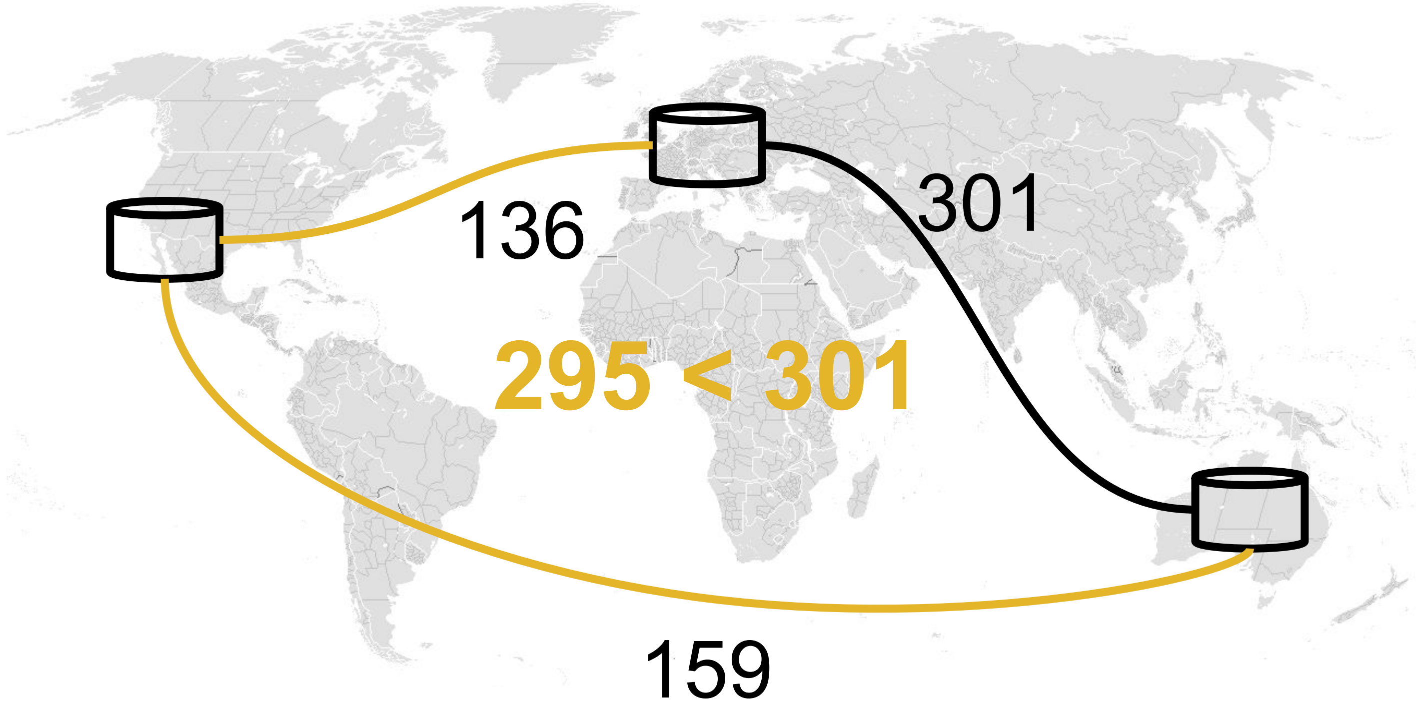
UNIVERSITY OF WATERLOO

Hinted Hand off



(Zakhary et al., EDBT 2018)

Hinted Hand off



(Zakhary et al., EDBT 2018)

Hinted Hand off

cost(S) = cost of executing request **at S**

Hand off request from S_1 to S_2 if:

$$\text{cost}(S_1) > \mathbf{RTT}(S_1, S_2) + \text{cost}(S_2)$$

(Zakhary et al., EDBT 2018)

Replication Decisions

- **How many replicas?**

  Centralized, given client workload

- **Where to place replicas?**

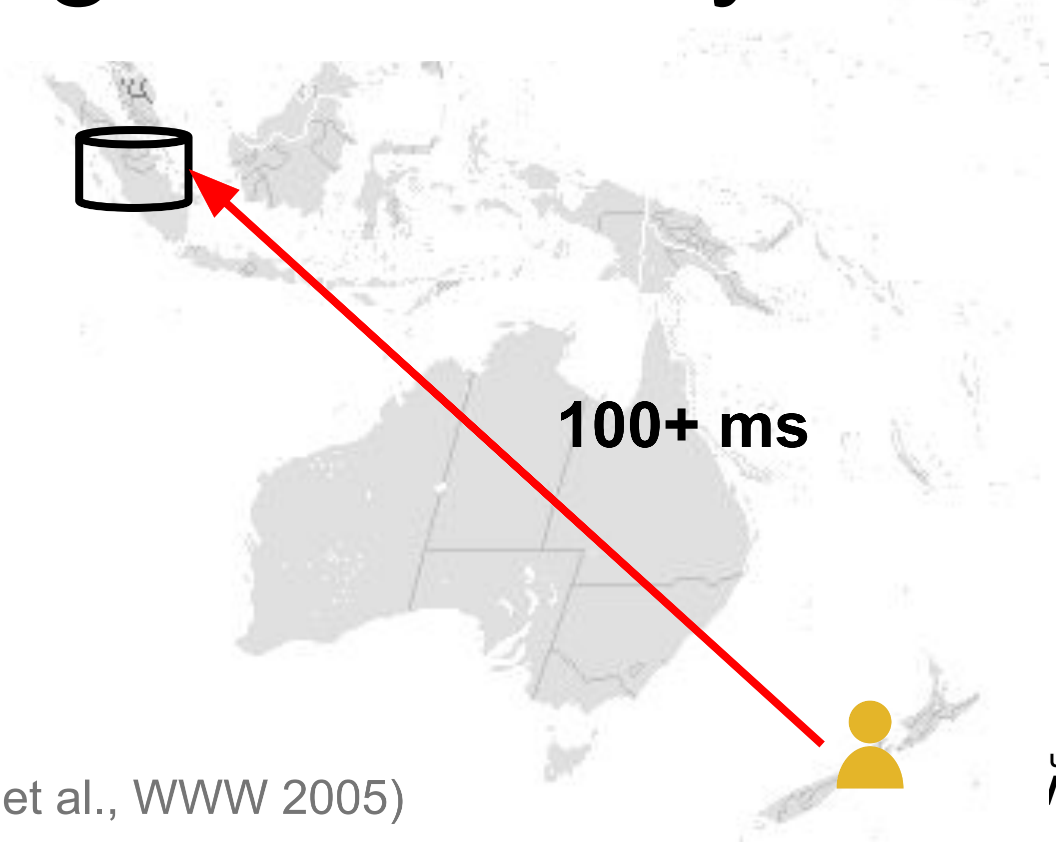
  Heuristic (clustering)

- **How to propagate updates?**

  Quorums / Leader-based (Sharov)

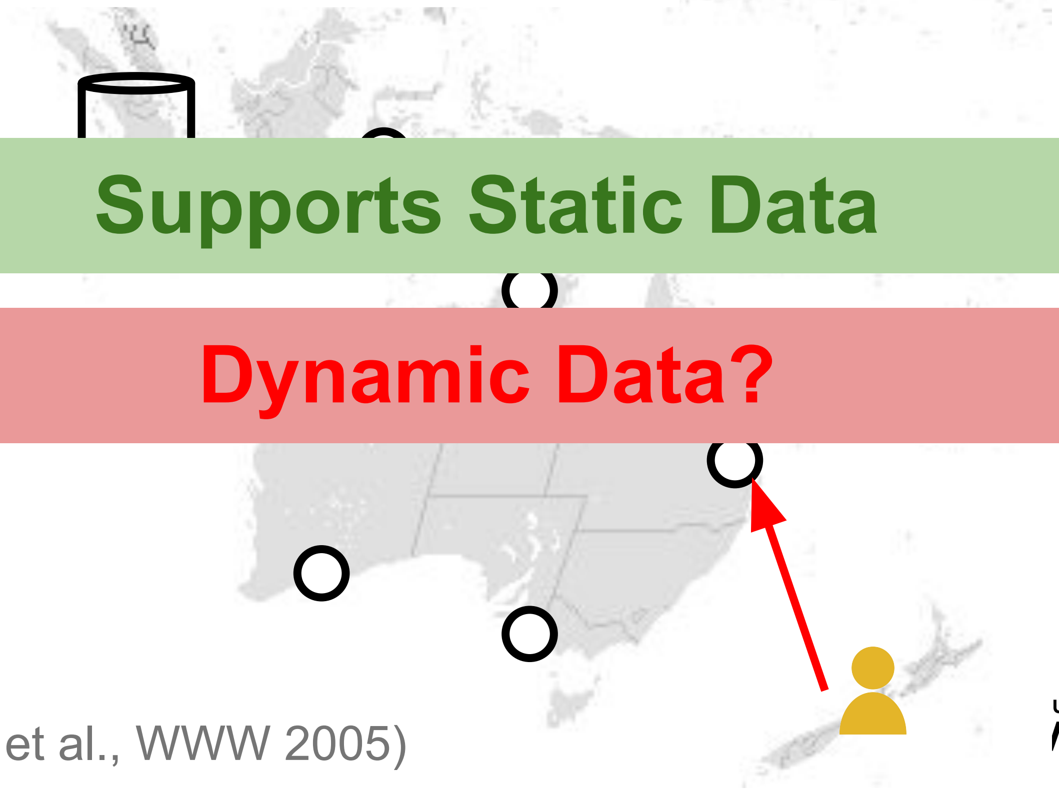
(Sharov et al., VLDB 2015) (Zakhary et al., EDBT 2018)

Intra-Region Latency



(Sivasubramanian et al., WWW 2005)

Edge Nodes

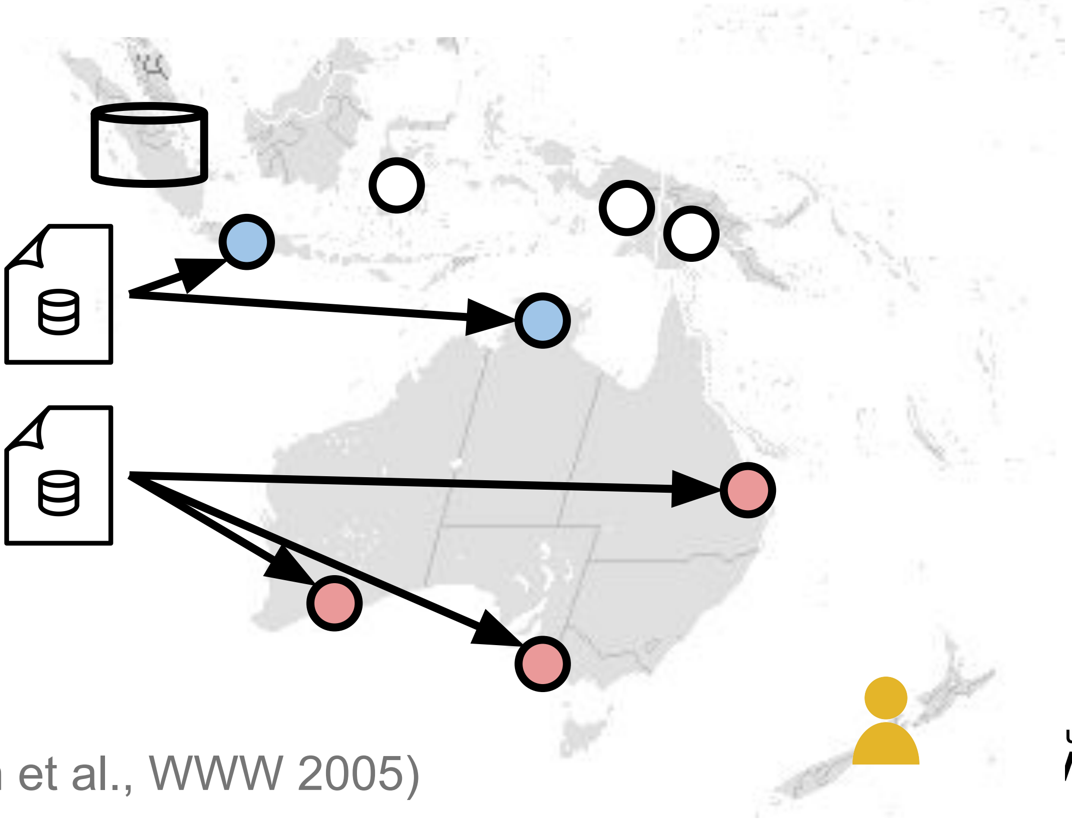


Supports Static Data

Dynamic Data?

(Sivasubramanian et al., WWW 2005)

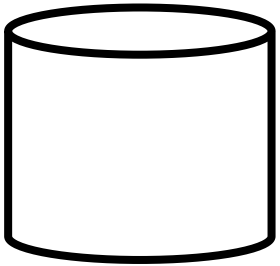
GlobeDB



(Sivasubramanian et al., WWW 2005)

Replication Granularity

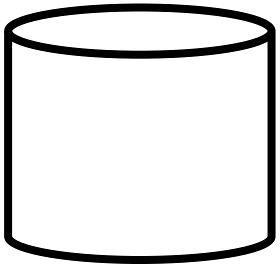
ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne



(Sivasubramanian et al., WWW 2005)

Replication Granularity

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne



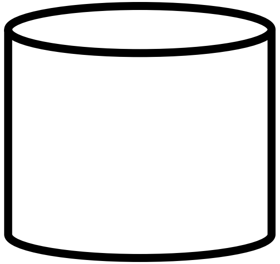
Per-Record?

High Overhead

(Sivasubramanian et al., WWW 2005)

Replication Granularity

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne



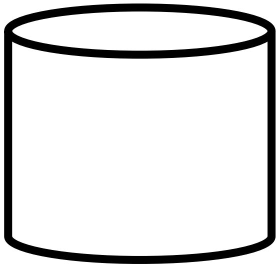
Per-Table?

Inflexible

(Sivasubramanian et al., WWW 2005)

Access-Driven Replicas

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne



When would these be replicated together?

(Sivasubramanian et al., WWW 2005)

Access-Driven Replicas

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne

$$A_{bryan} = \langle r_1, \ldots r_n, w_1, \ldots w_n \rangle$$

(Sivasubramanian et al., WWW 2005)

Access-Driven Replicas

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne

$A_{justin} = \langle r_1,\ldots r_n, w_1,\ldots w_n \rangle$ **$Sim(A_{bryan}, A_{justin}) \geq \tau$?**

(Sivasubramanian et al., WWW 2005)

UNIVERSITY OF WATERLOO

Access-Driven Replicas

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne

Shared Replication Scheme

(Sivasubramanian et al., WWW 2005)

Access-Driven Replicas

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne

$A_{p1} = <r_1,...r_n,w_1,...w_n>$

(Sivasubramanian et al., WWW 2005)

Access-Driven Replicas

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne

$A_{avril} = \langle r_1, \ldots r_n, w_1, \ldots w_n \rangle$

$\mathbf{Sim(A_{p1}, A_{avril}) \geq \tau?}$

(Sivasubramanian et al., WWW 2005)

Access-Driven Replicas

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne

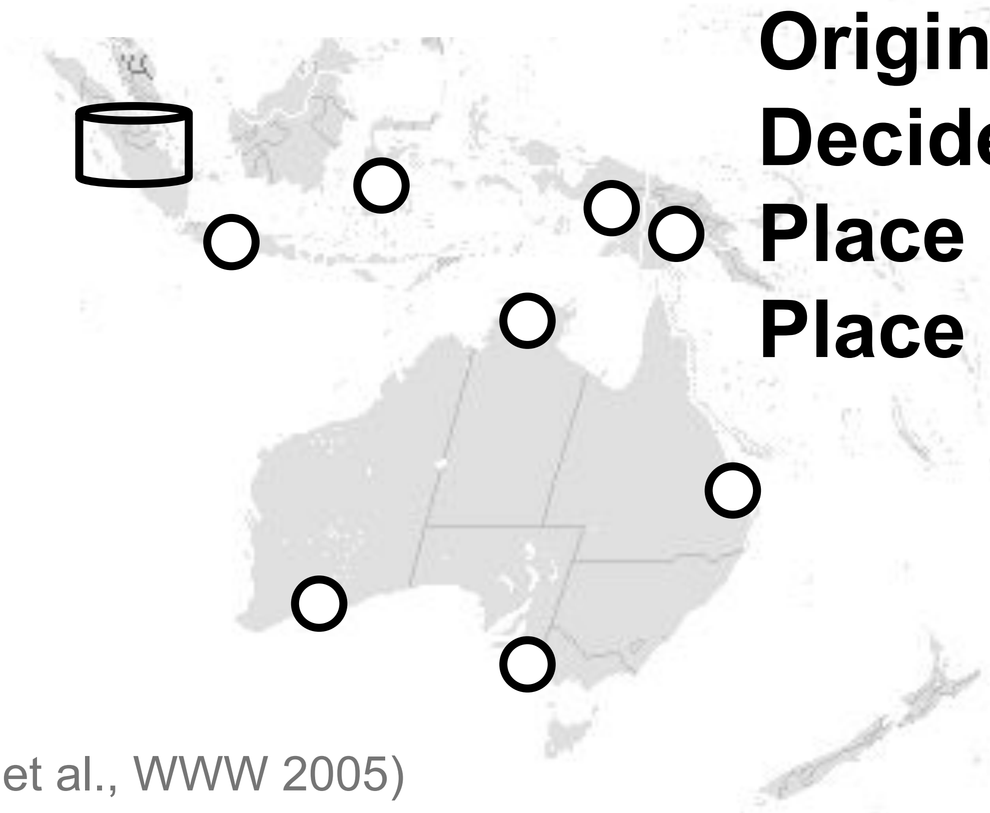
(Sivasubramanian et al., WWW 2005)

Access-Driven Replicas

ID	ARTIST
1	Bryan Adams
2	Justin Bieber
3	Avril Lavigne
4	Kanye West
5	Drake
6	David Guetta
7	Ed Sheeran

(Sivasubramanian et al., WWW 2005)

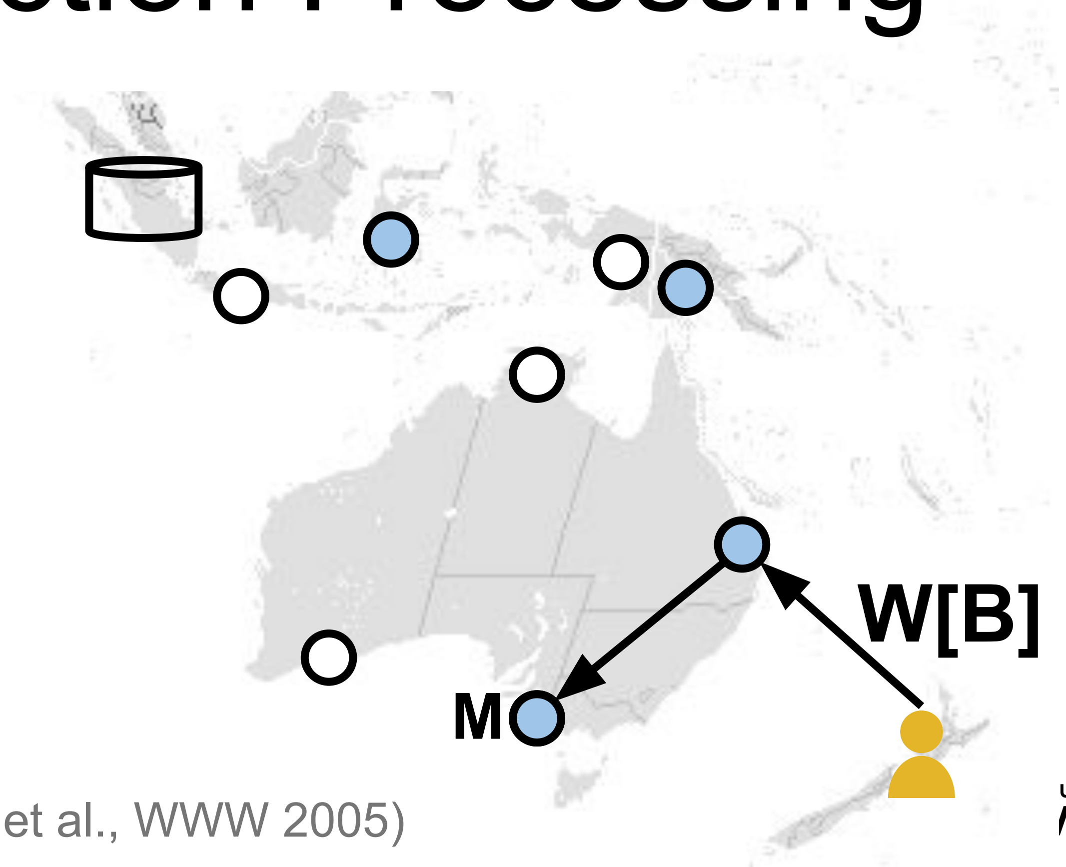
Transaction Processing



Origin Server: Decide Partitions, Place Replicas, Place Master

(Sivasubramanian et al., WWW 2005)

Transaction Processing



(Sivasubramanian et al., WWW 2005)

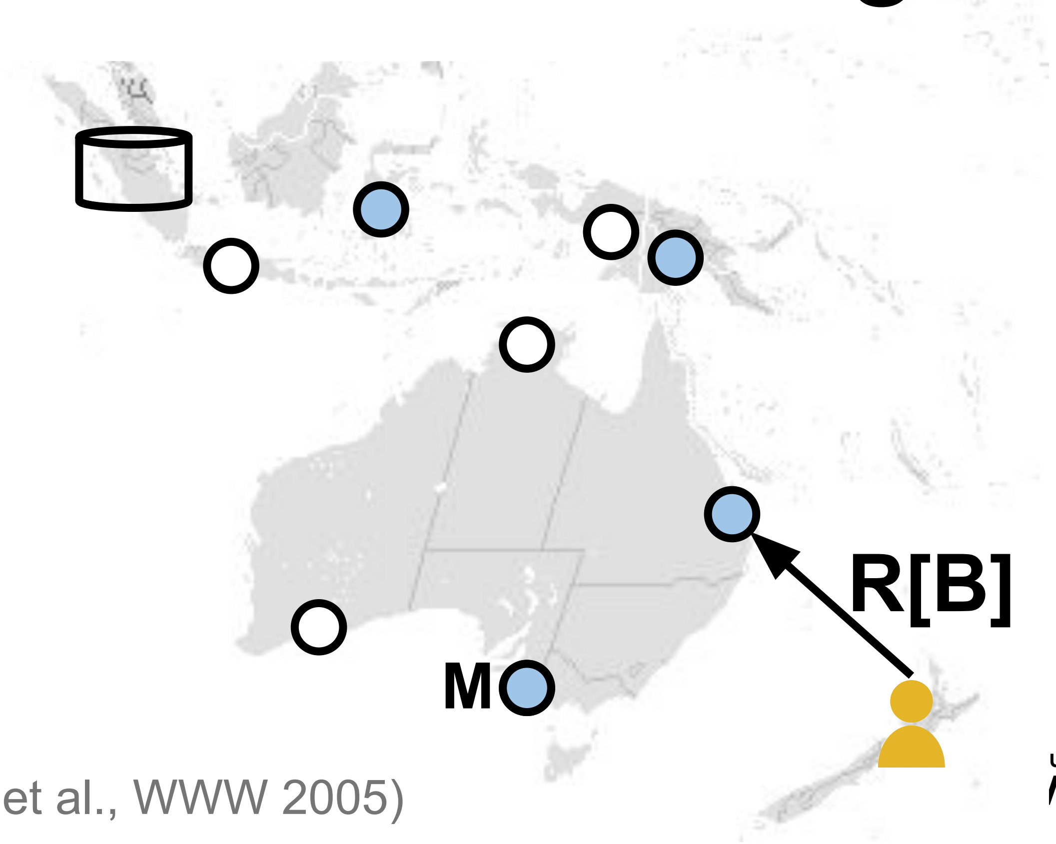
UNIVERSITY OF WATERLOO

Transaction Processing

(Sivasubramanian et al., WWW 2005)

UNIVERSITY OF WATERLOO

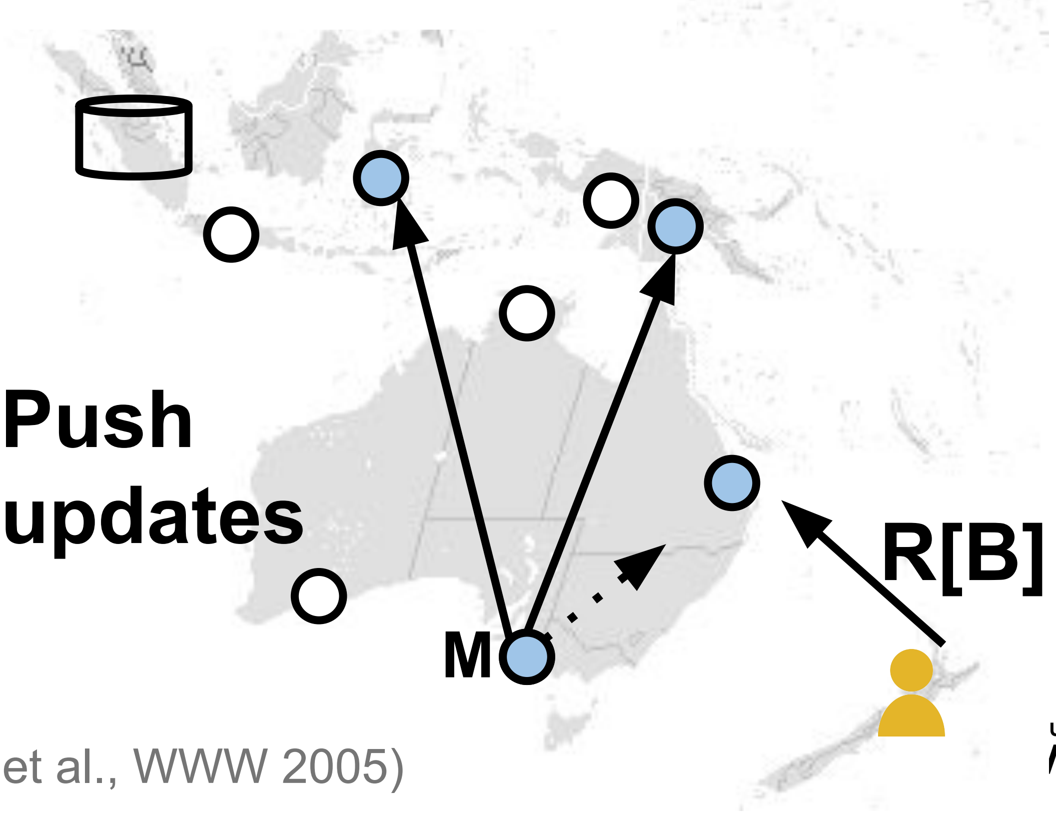
Transaction Processing



(Sivasubramanian et al., WWW 2005)

UNIVERSITY OF WATERLOO

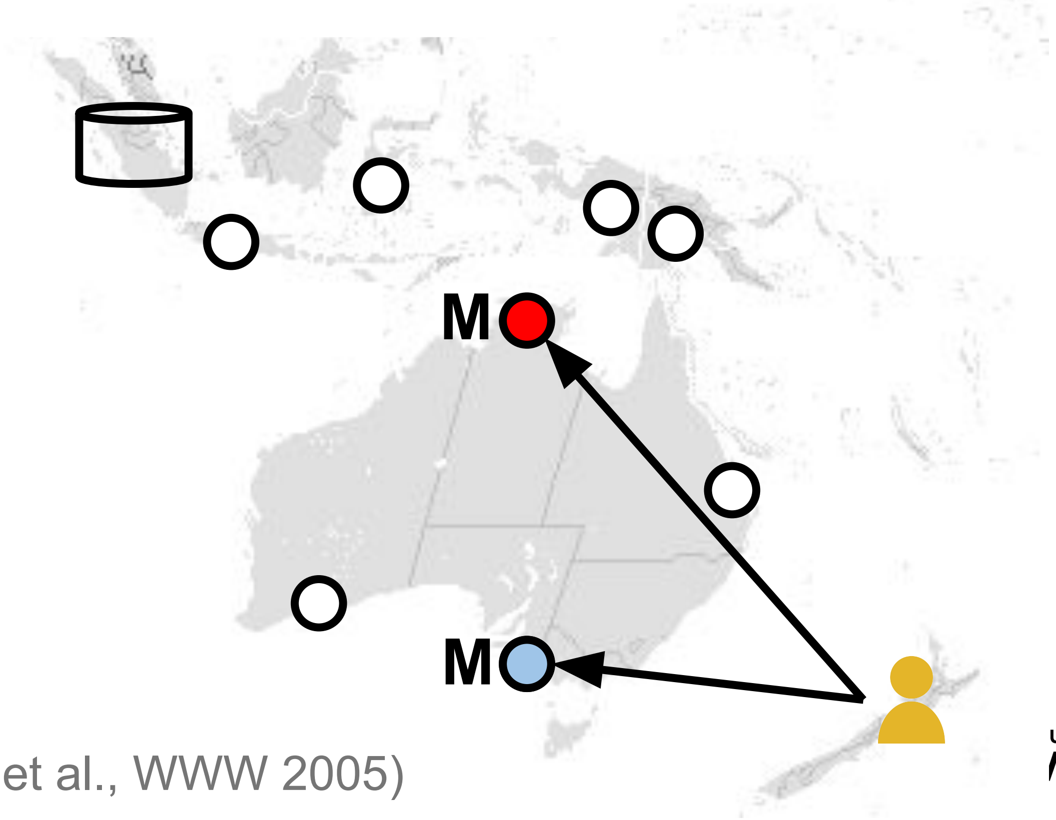
Transaction Processing



(Sivasubramanian et al., WWW 2005)

UNIVERSITY OF WATERLOO

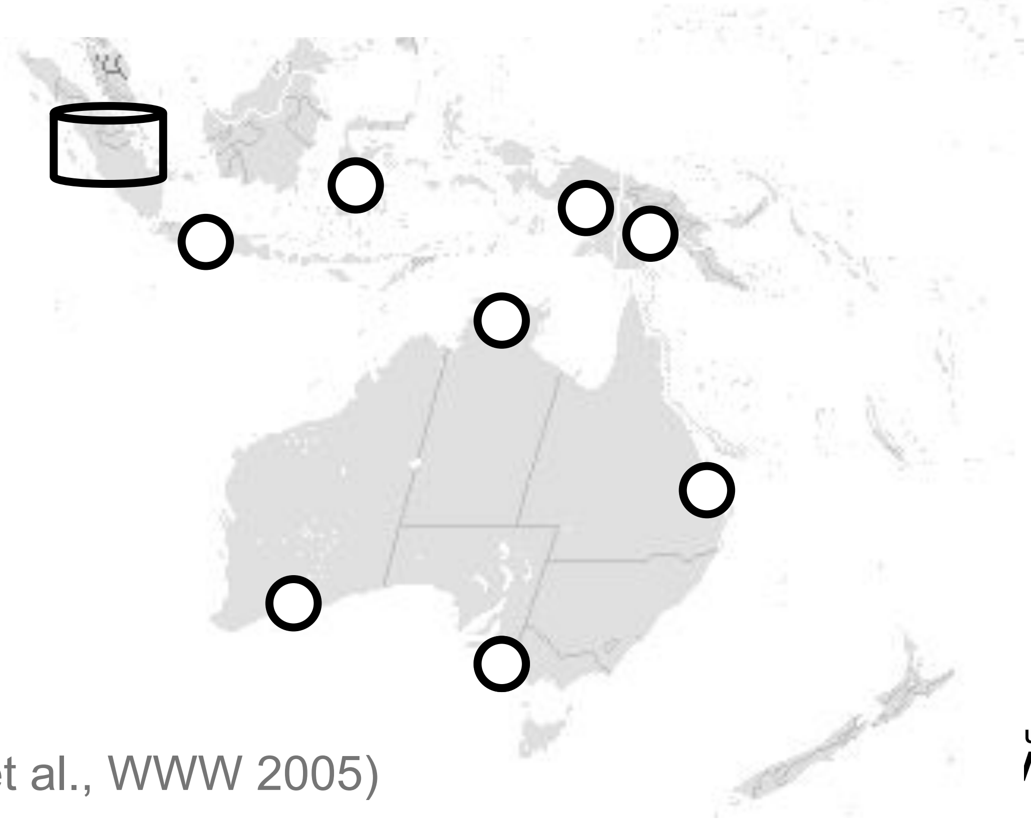
Transaction Processing



(Sivasubramanian et al., WWW 2005)

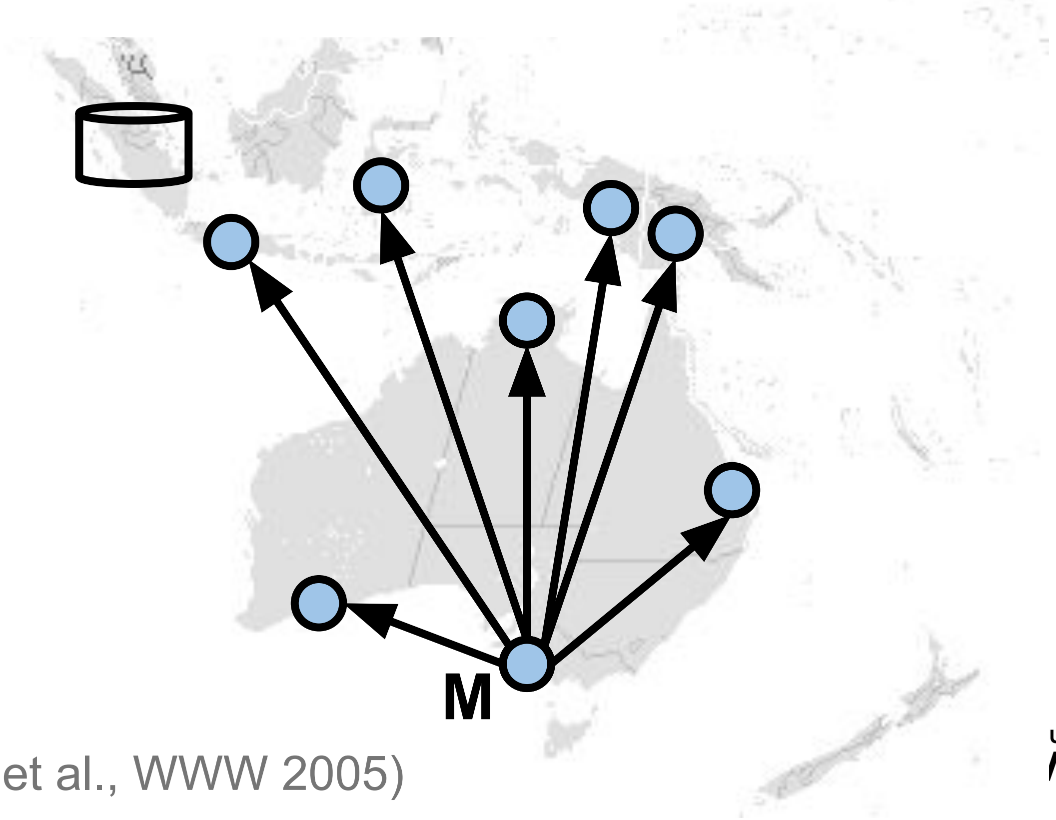
UNIVERSITY OF WATERLOO

Replica and Master Placement



(Sivasubramanian et al., WWW 2005)

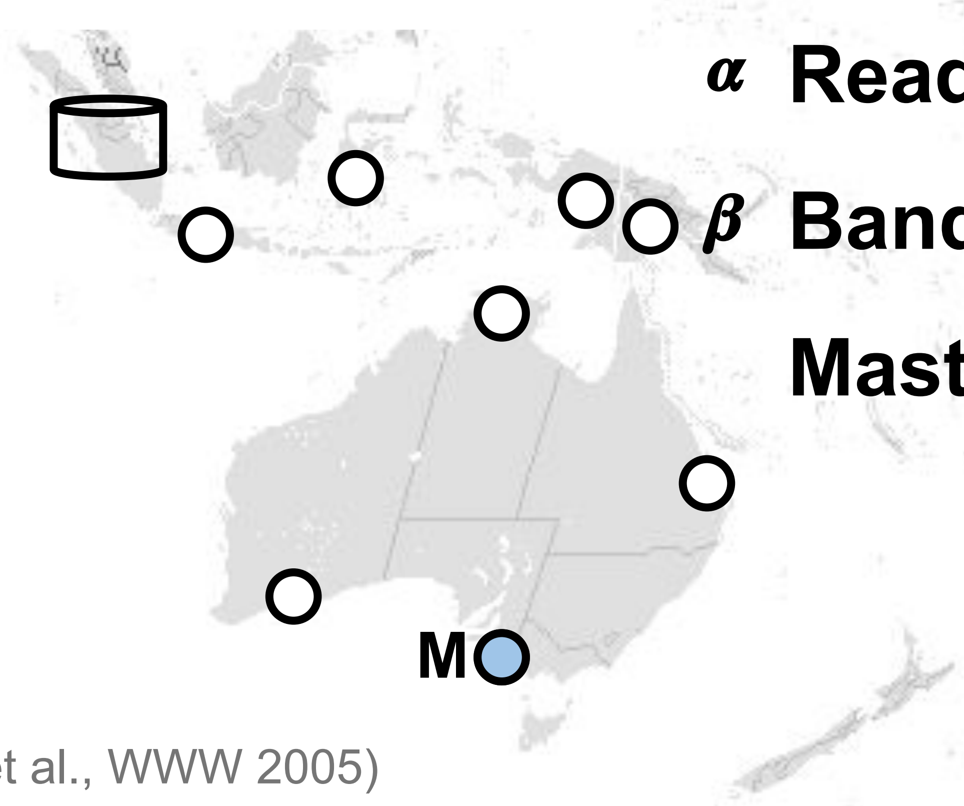
Read Latency vs. Bandwidth



(Sivasubramanian et al., WWW 2005)

UNIVERSITY OF WATERLOO

Read Latency vs. Bandwidth



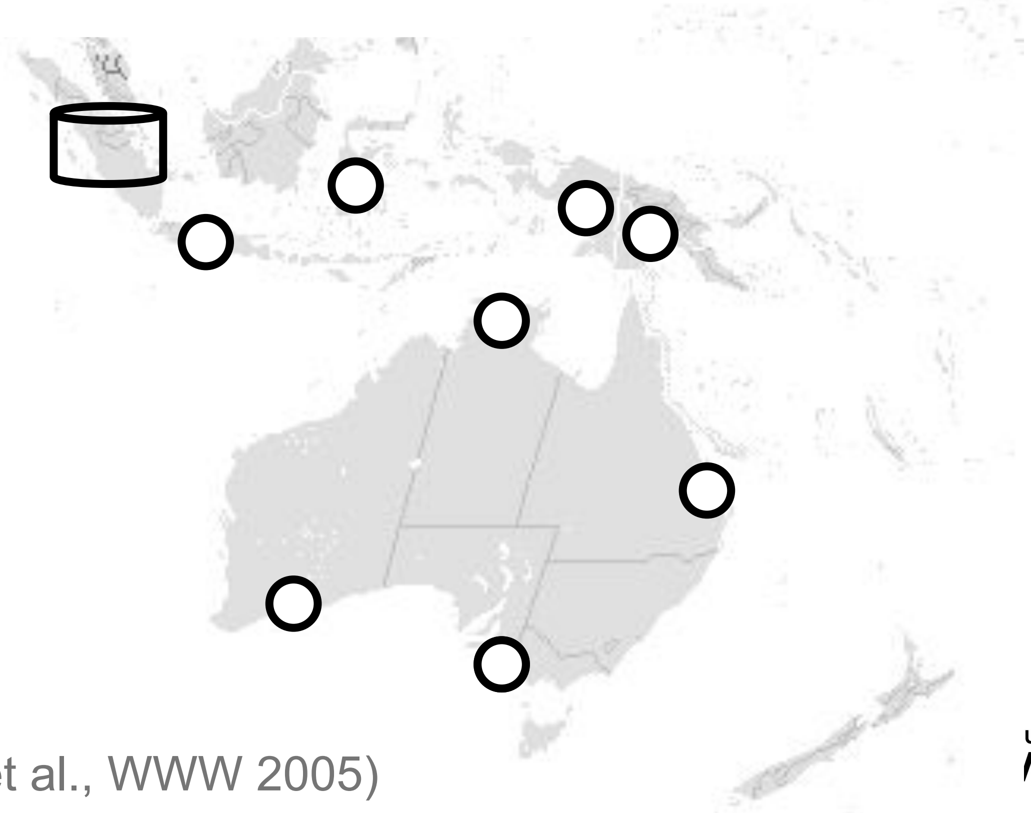
α **Read Latency**

β **Bandwidth**

Master?

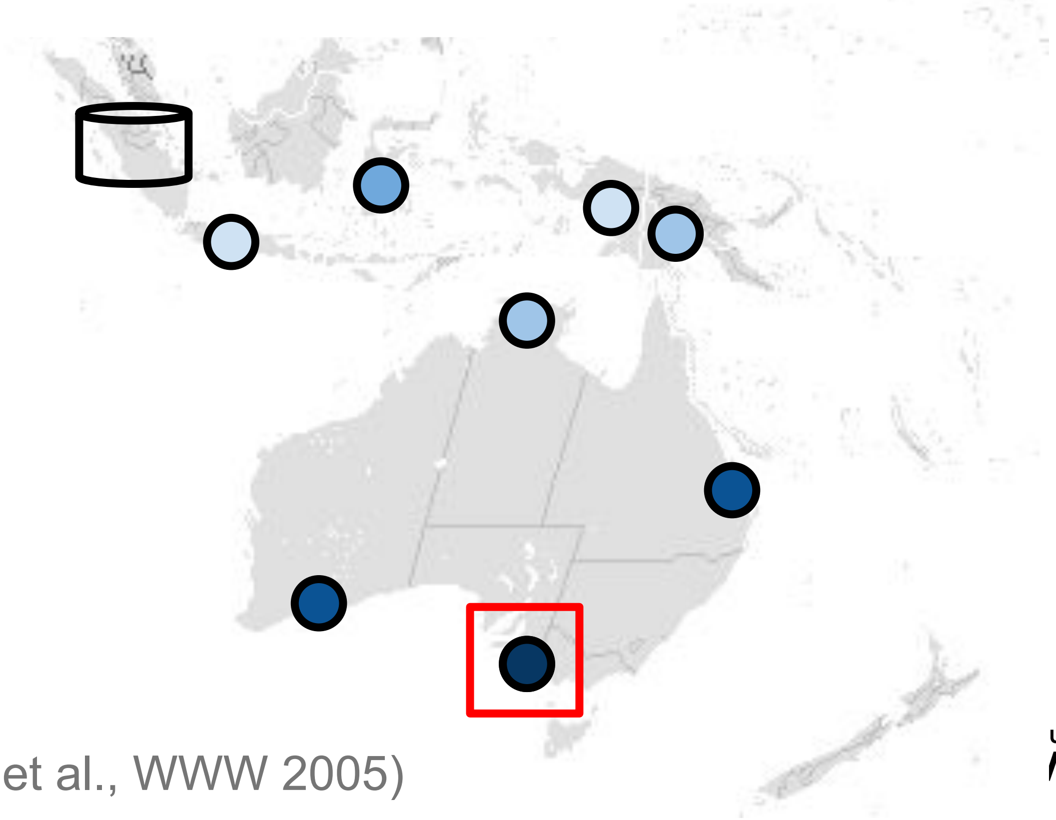
(Sivasubramanian et al., WWW 2005)

Master Partition Placement



(Sivasubramanian et al., WWW 2005)

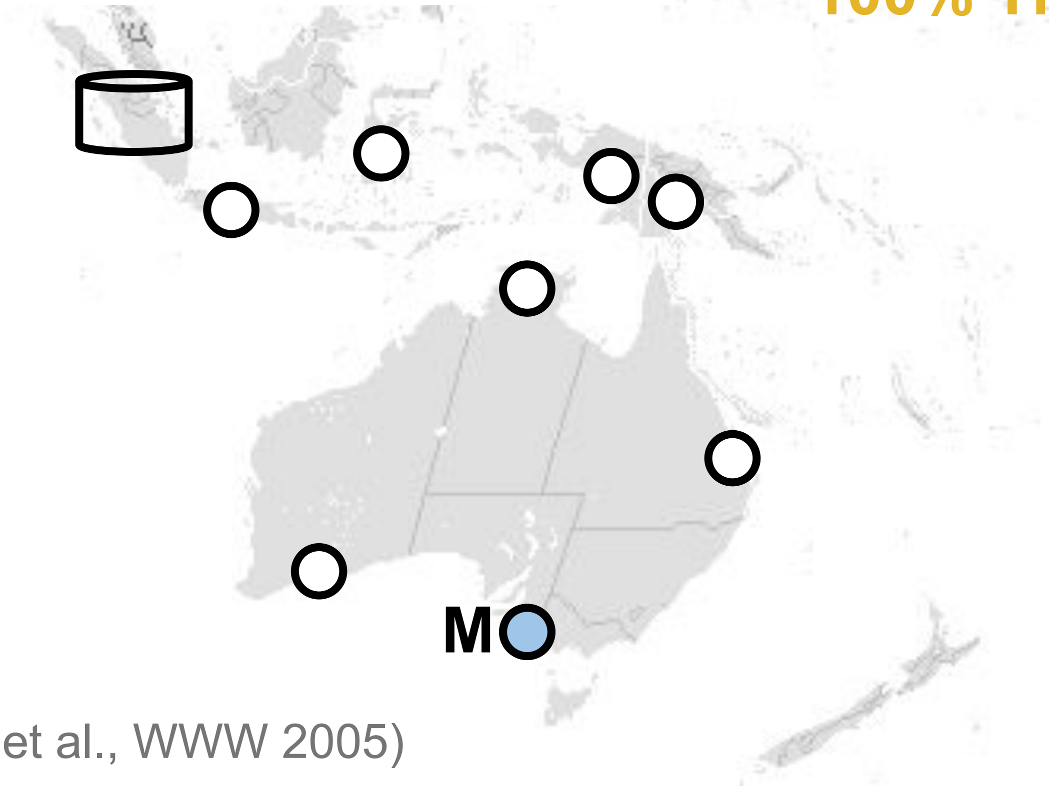
Master Partition Placement



(Sivasubramanian et al., WWW 2005)

Placement Heuristic

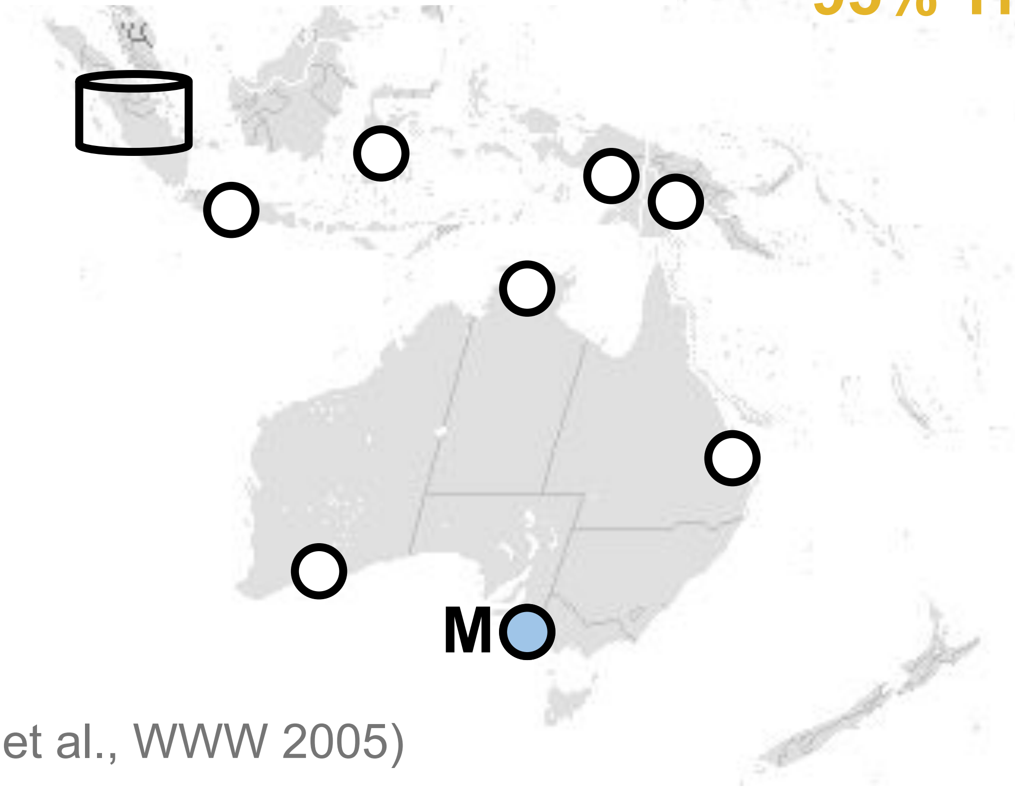
100% Threshold



(Sivasubramanian et al., WWW 2005)

Placement Heuristic

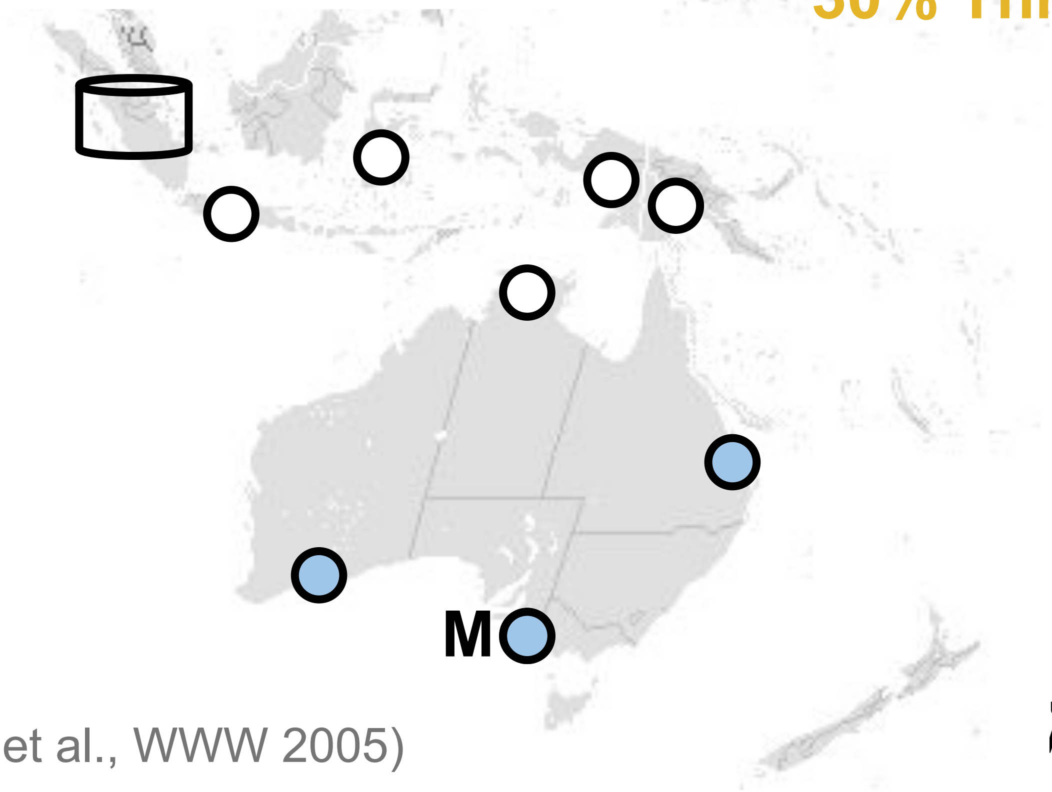
95% Threshold



(Sivasubramanian et al., WWW 2005)

Placement Heuristic

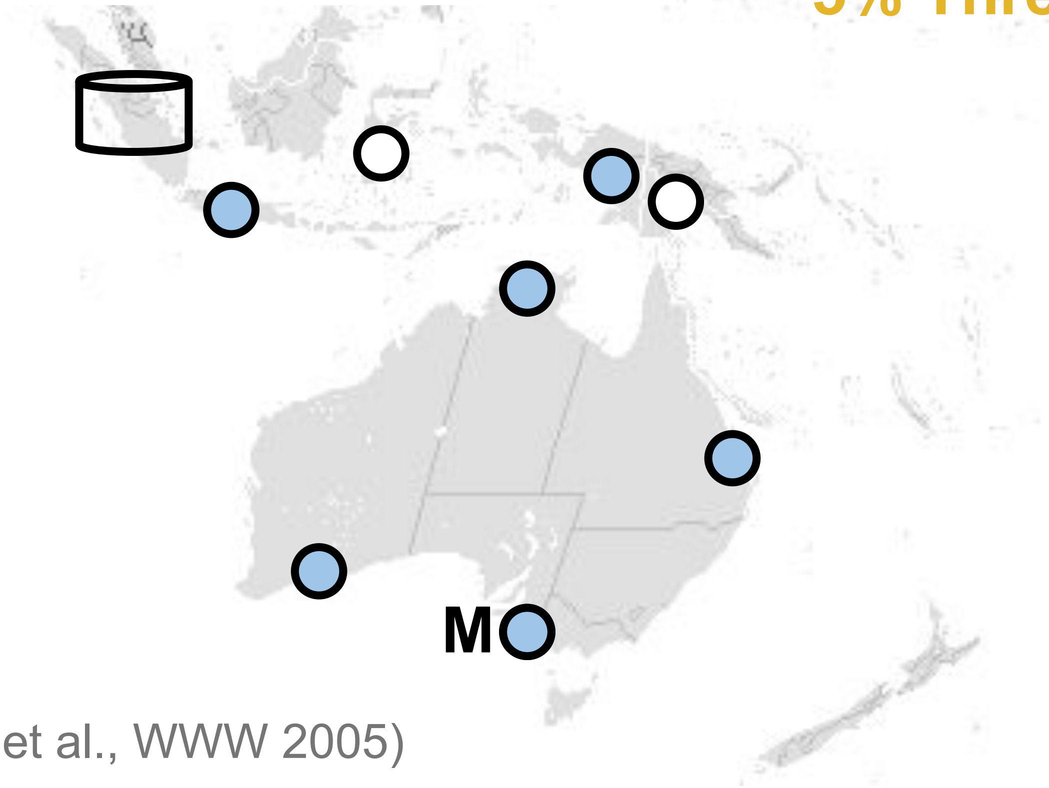
30% Threshold



(Sivasubramanian et al., WWW 2005)

Placement Heuristic

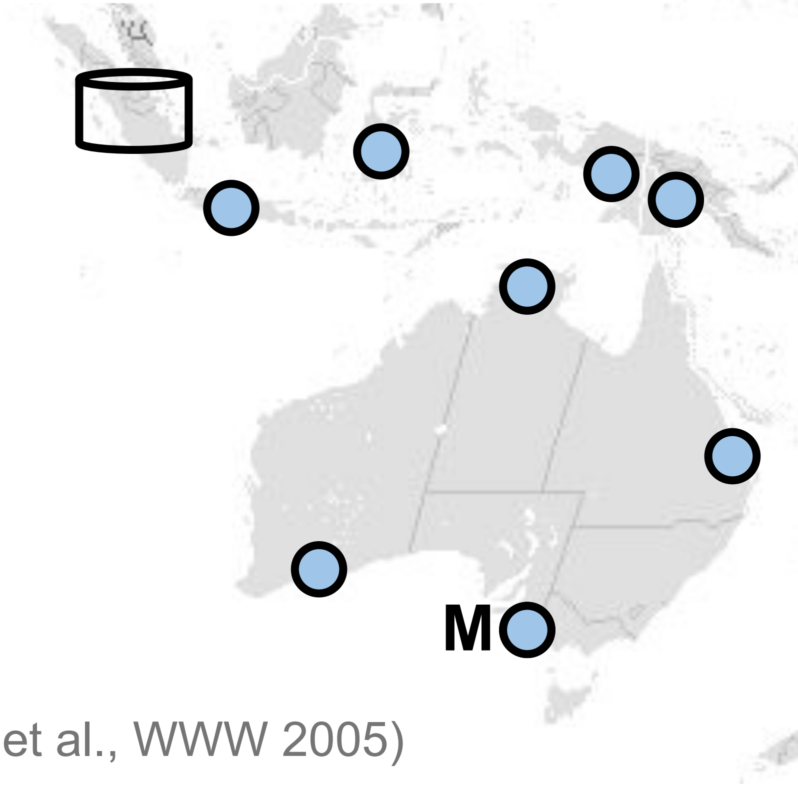
5% Threshold



(Sivasubramanian et al., WWW 2005)

Placement Heuristic

0% Threshold



Minimize:

α **r +** β **b**

(Sivasubramanian et al., WWW 2005)

Replication Decisions

- **How many replicas?**

  Cost-based given requests

- **Where to place replicas?**

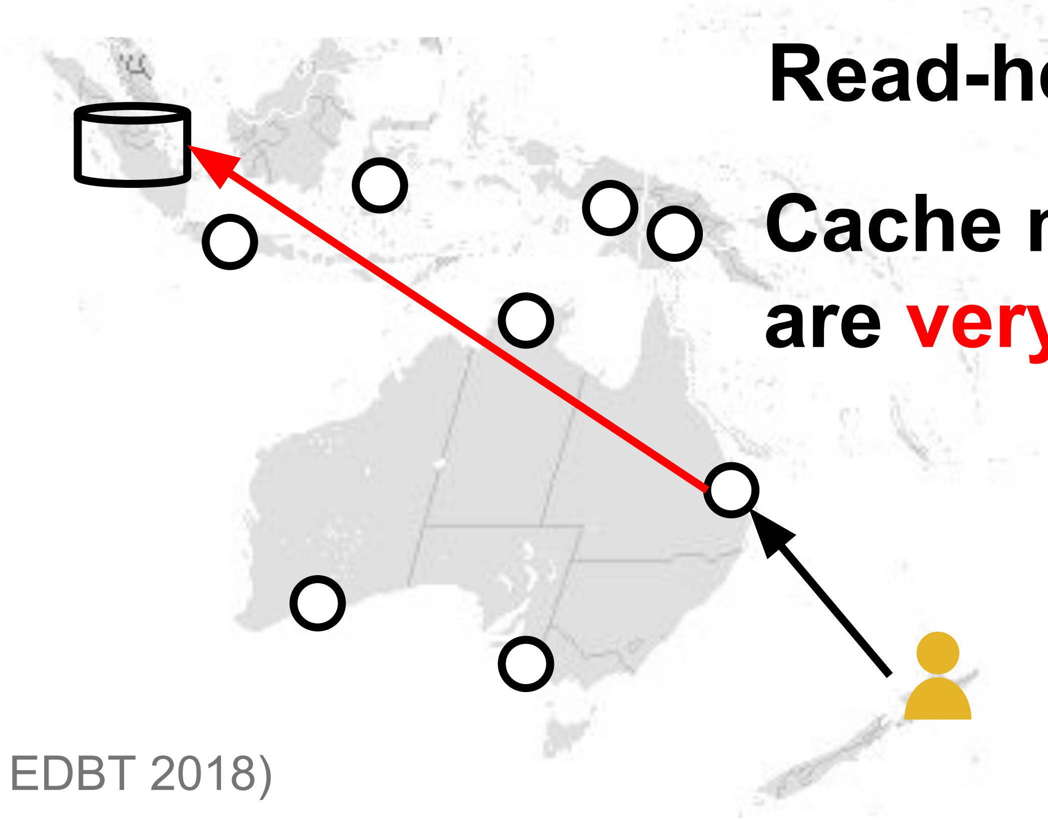
  Cost-based given requests

- **How to propagate updates?**

  Single-master, eventual consistency

(Sivasubramanian et al., WWW 2005)

Web Workload Characteristics

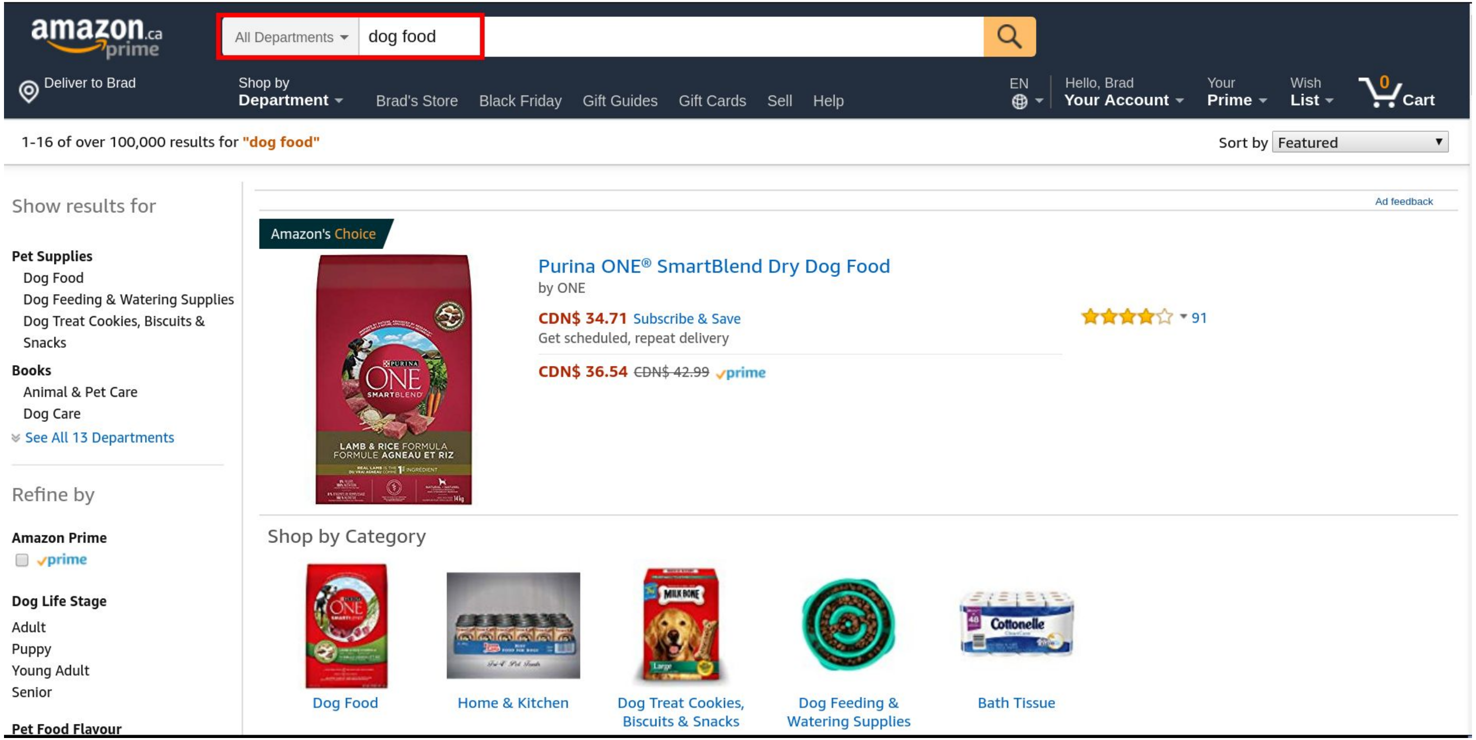


Read-heavy

Cache misses are very painful

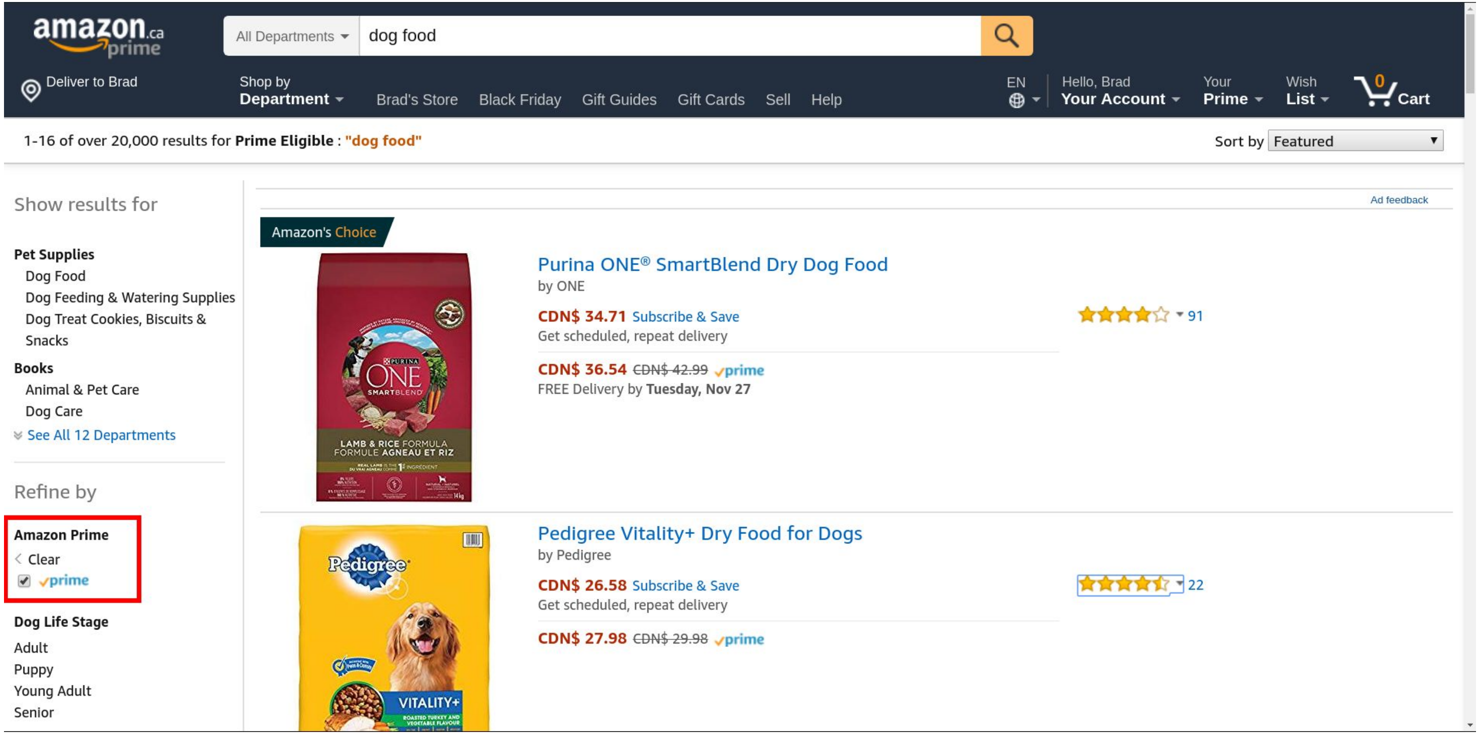
(Glasbergen et al., EDBT 2018)

Web Workload Characteristics



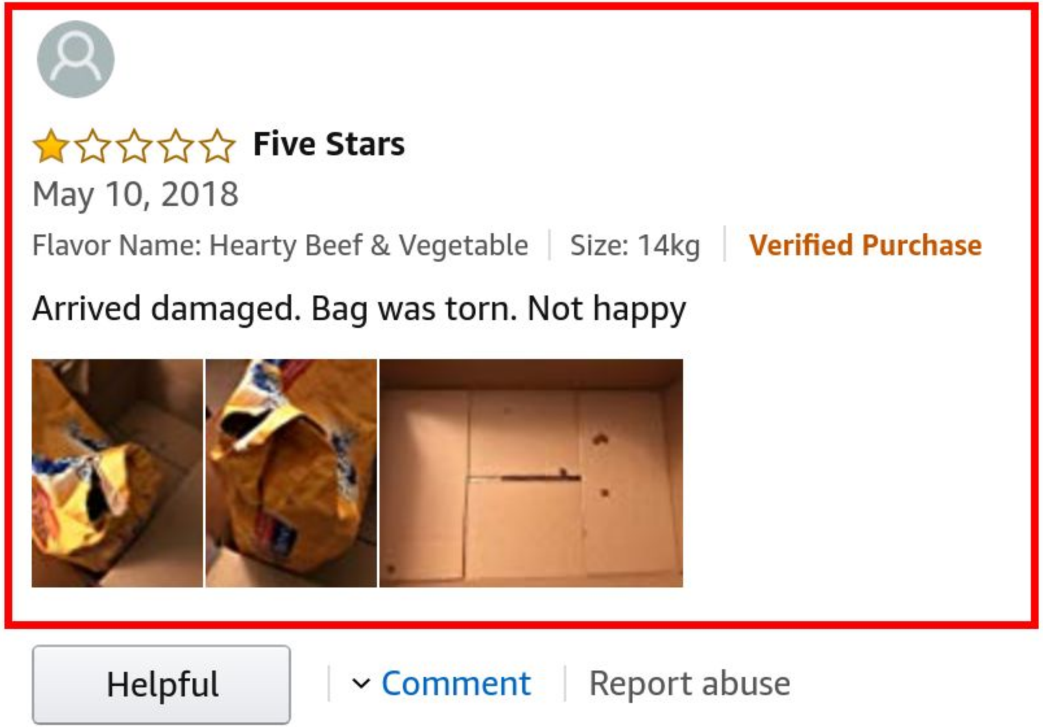
(Glasbergen et al., EDBT 2018)

Web Workload Characteristics



(Glasbergen et al., EDBT 2018)

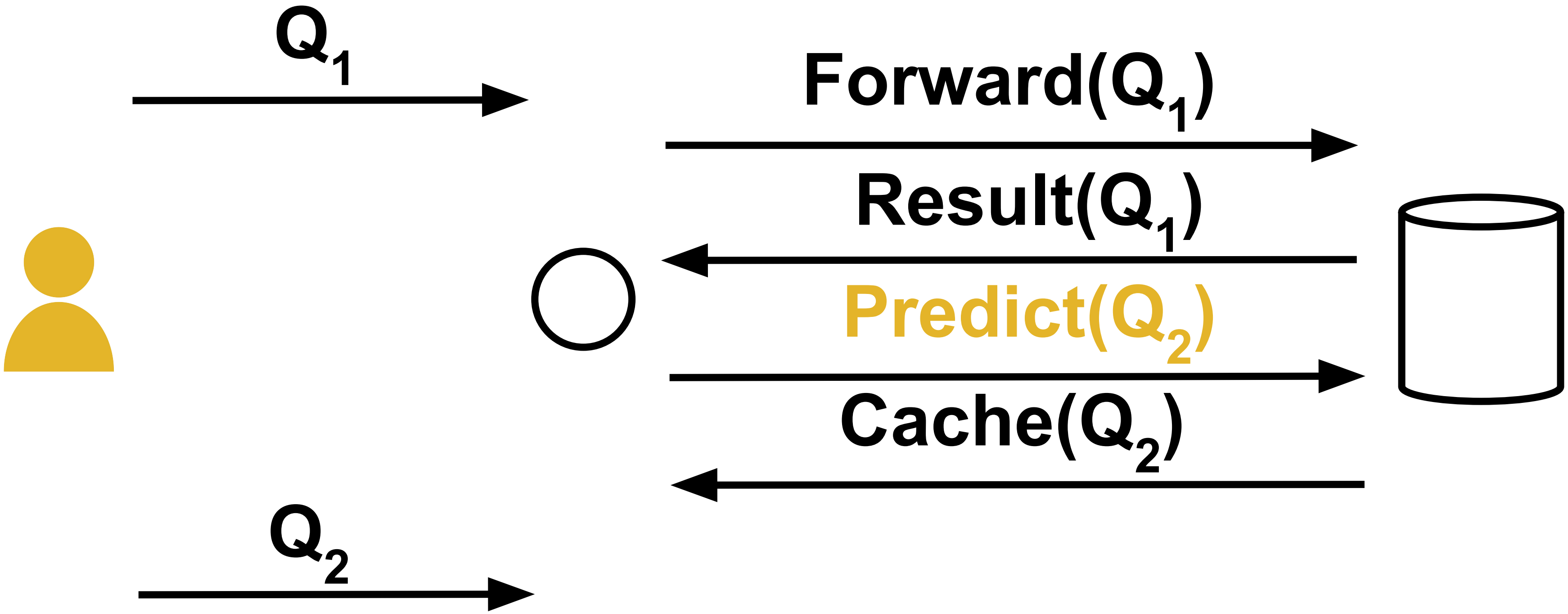
Web Workload Characteristics



(Glasbergen et al., EDBT 2018)

UNIVERSITY OF WATERLOO

Predictive Caching



(Glasbergen et al., EDBT 2018)

Query Patterns (TPC-W)

SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

(Glasbergen et al., EDBT 2018)

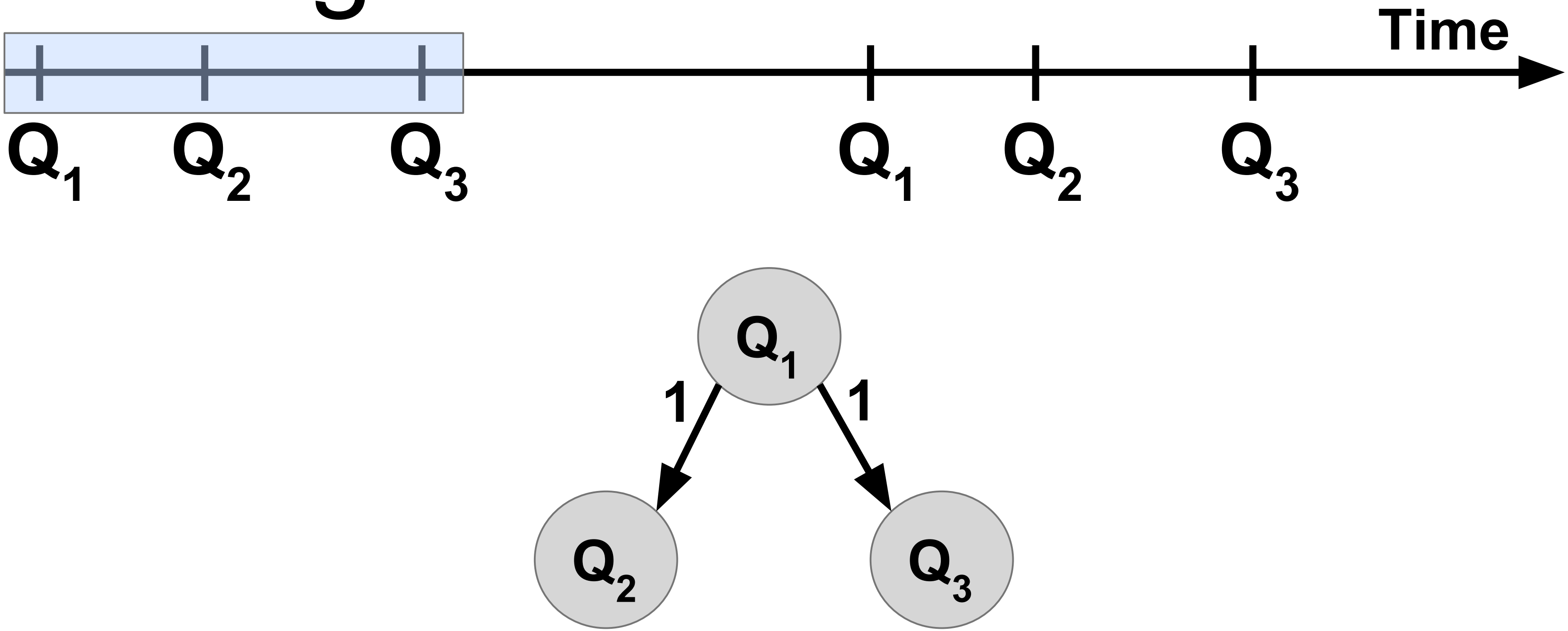
SELECT C_ID FROM CUSTOMER WHERE C_UNAME = 'Alice' and C_PASSWD = 'pass'

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = 3

SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

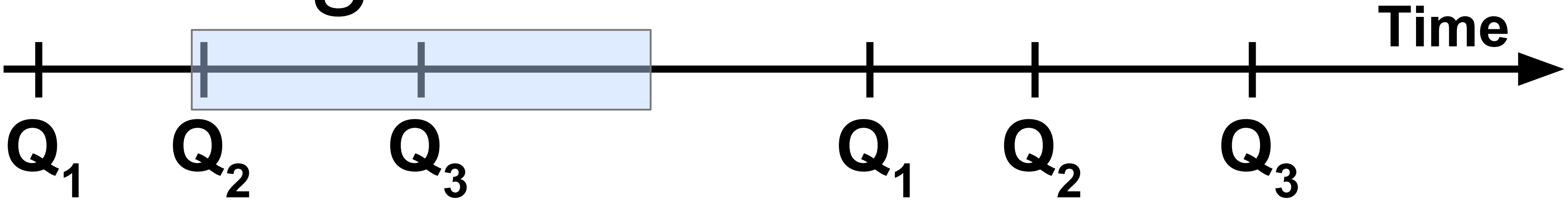
Building a Predictive Model

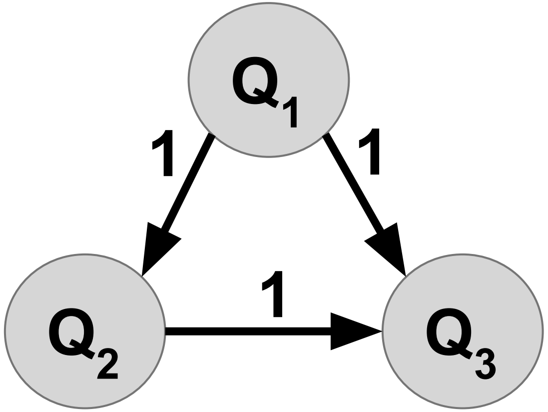


(Glasbergen et al., EDBT 2018)

UNIVERSITY OF WATERLOO

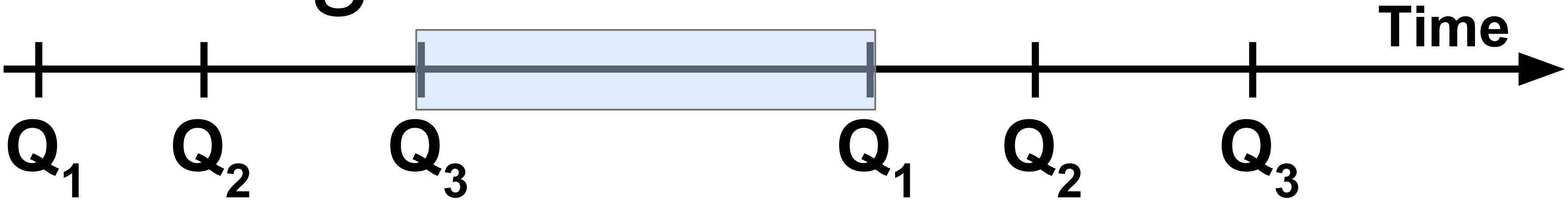
Building a Predictive Model

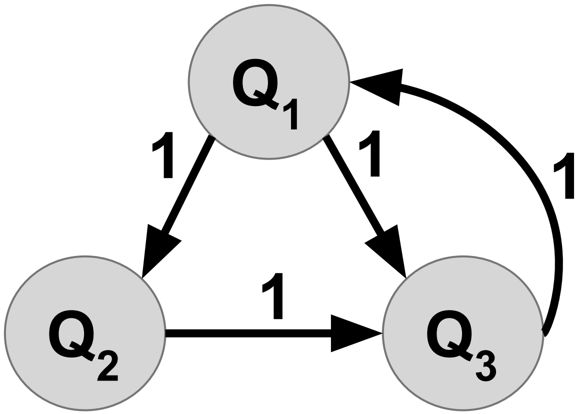




(Glasbergen et al., EDBT 2018)

Building a Predictive Model

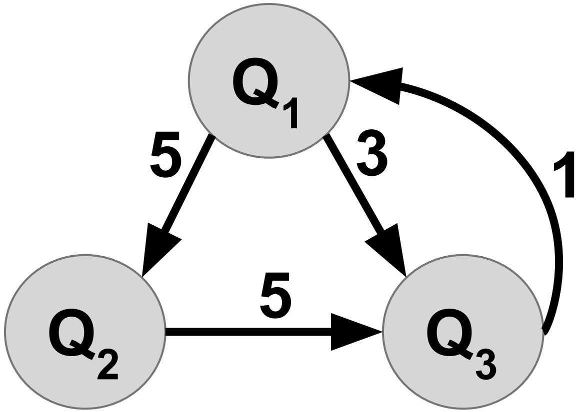




(Glasbergen et al., EDBT 2018)

Building a Predictive Model

All executed 5 times



100% probability Q_2 follows Q_1

20% probability Q_1 follows Q_3

(Glasbergen et al., EDBT 2018)

Finding Parameter Mappings

SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

(Glasbergen et al., EDBT 2018)

UNIVERSITY OF WATERLOO

Finding Parameter Mappings

3

SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

(Glasbergen et al., EDBT 2018)

UNIVERSITY OF WATERLOO

Finding Parameter Mappings

3

SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

3

(Glasbergen et al., EDBT 2018)

UNIVERSITY OF WATERLOO

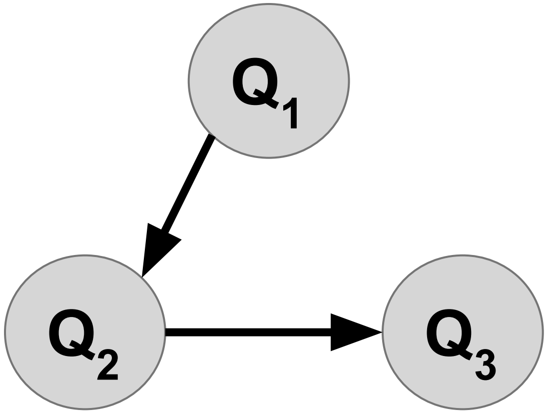
Finding Parameter Mappings

SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? and C_PASSWD = ?

SELECT MAX(O_ID) FROM ORDERS WHERE O_C_ID = ?

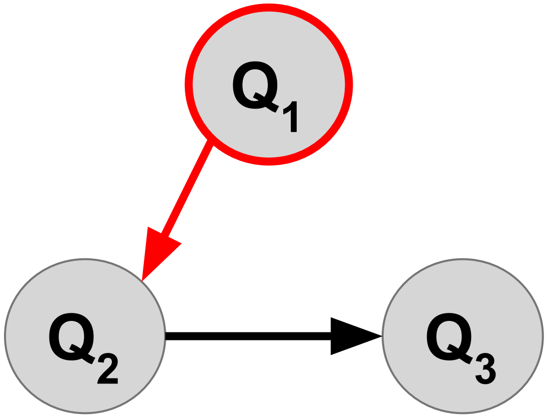
(Glasbergen et al., EDBT 2018)

Predictive Caching



(Glasbergen et al., EDBT 2018)

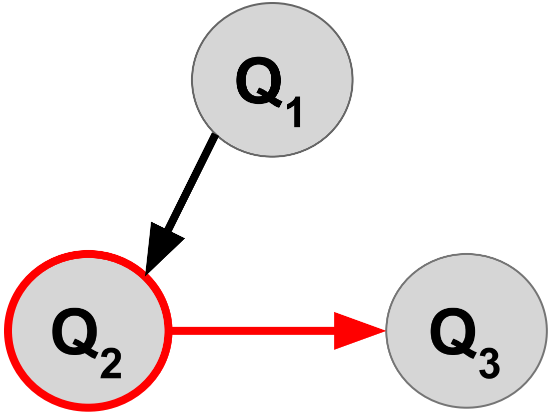
Predictive Caching



Predictively Cache Q_2

(Glasbergen et al., EDBT 2018)

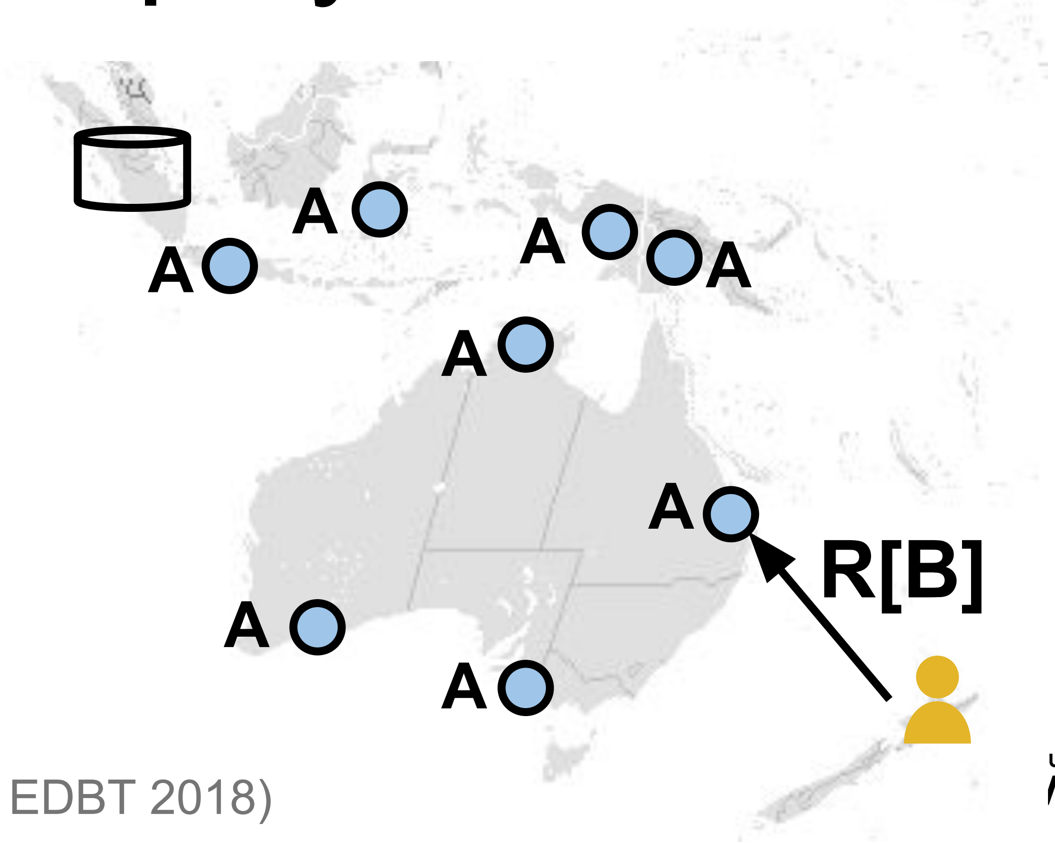
Predictive Caching



Predictively Cache Q_3

(Glasbergen et al., EDBT 2018)

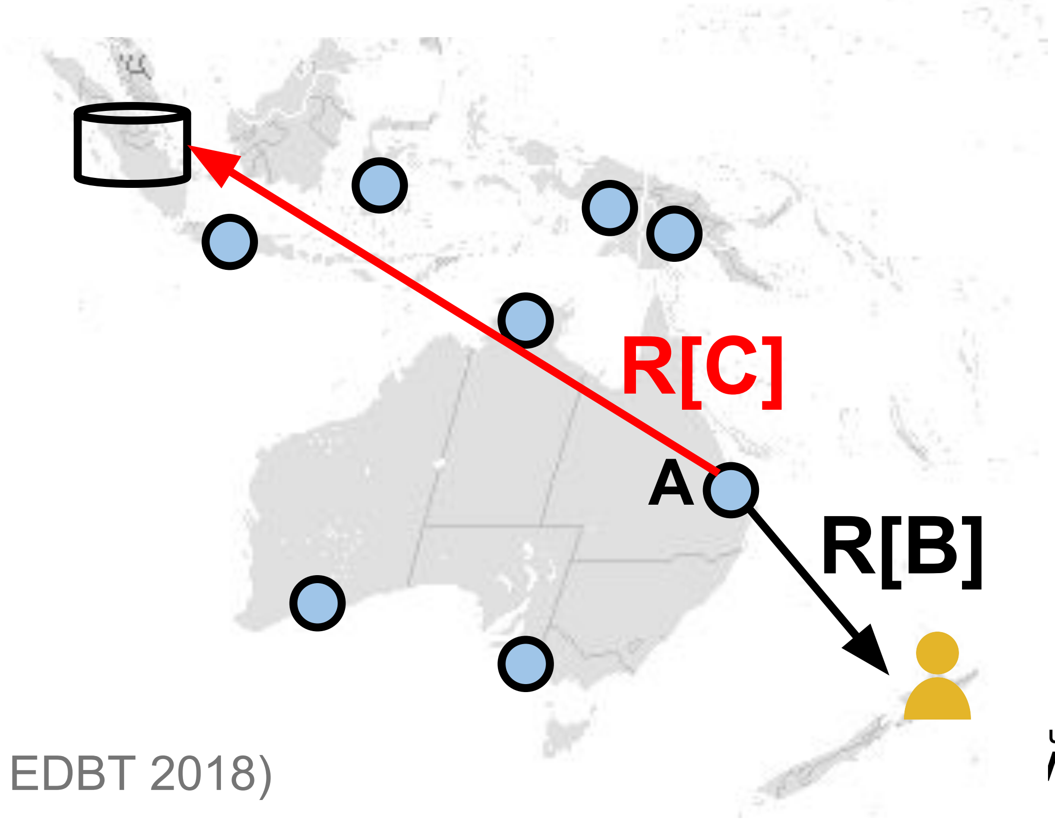
Apollo Deployment



(Glasbergen et al., EDBT 2018)

UNIVERSITY OF WATERLOO

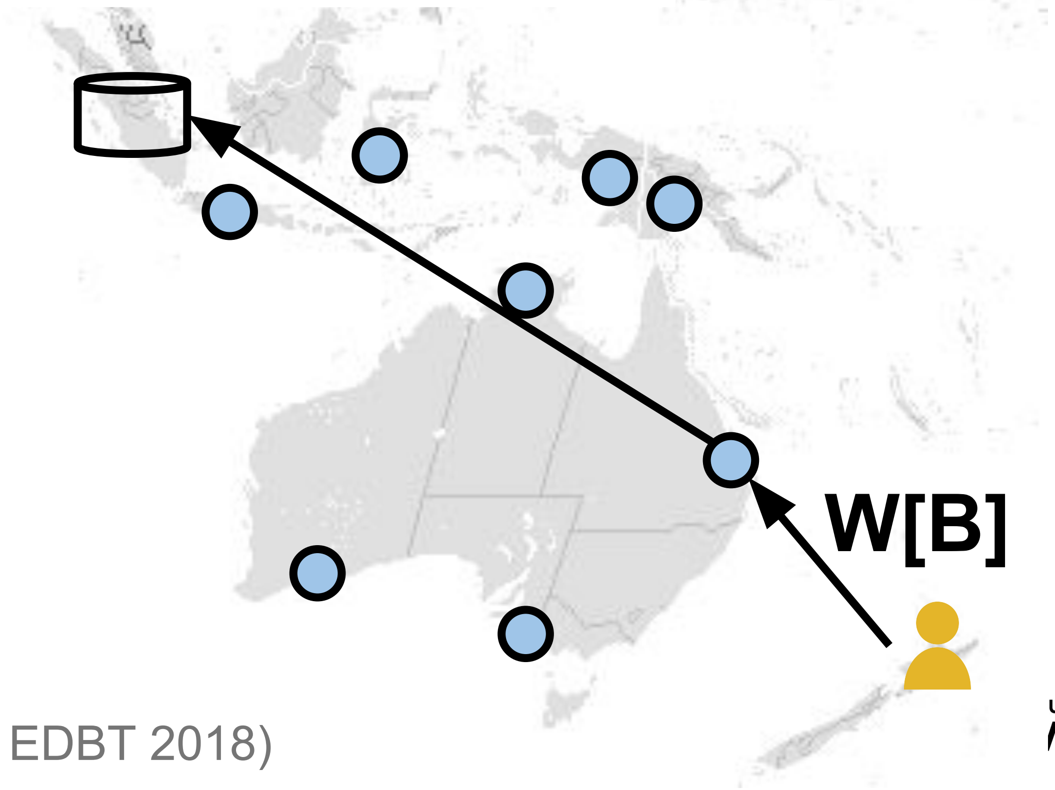
Apollo Deployment



(Glasbergen et al., EDBT 2018)

UNIVERSITY OF WATERLOO

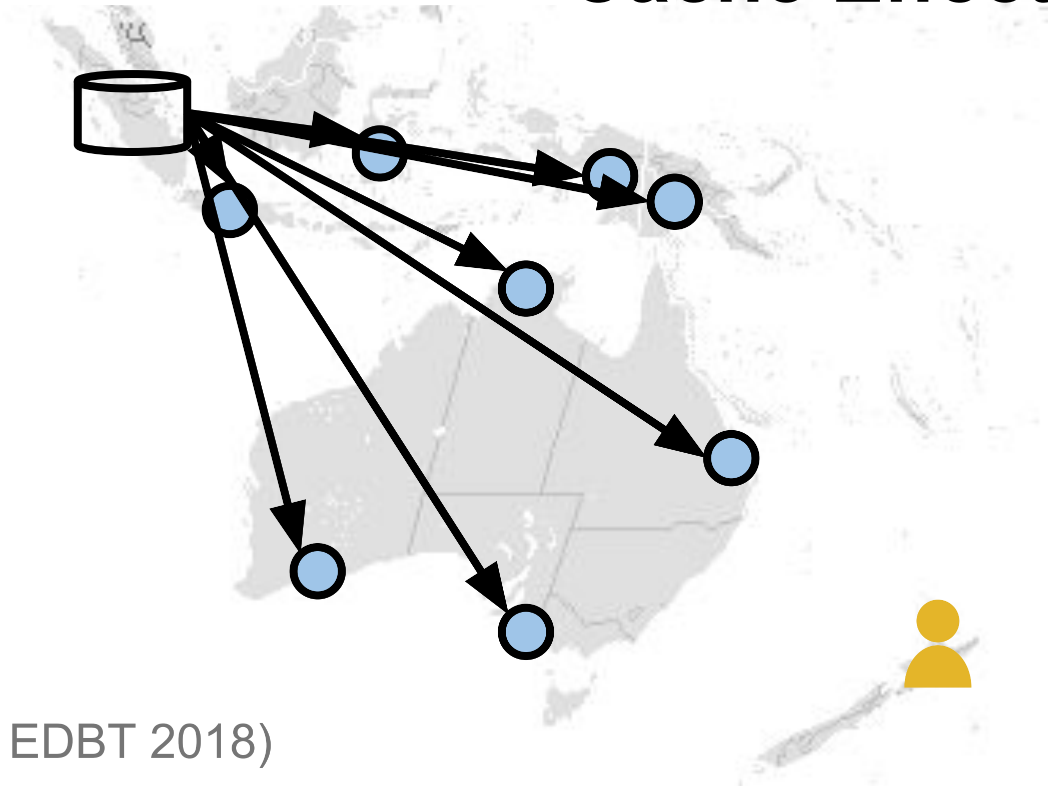
Invalidations



(Glasbergen et al., EDBT 2018)

Invalidations

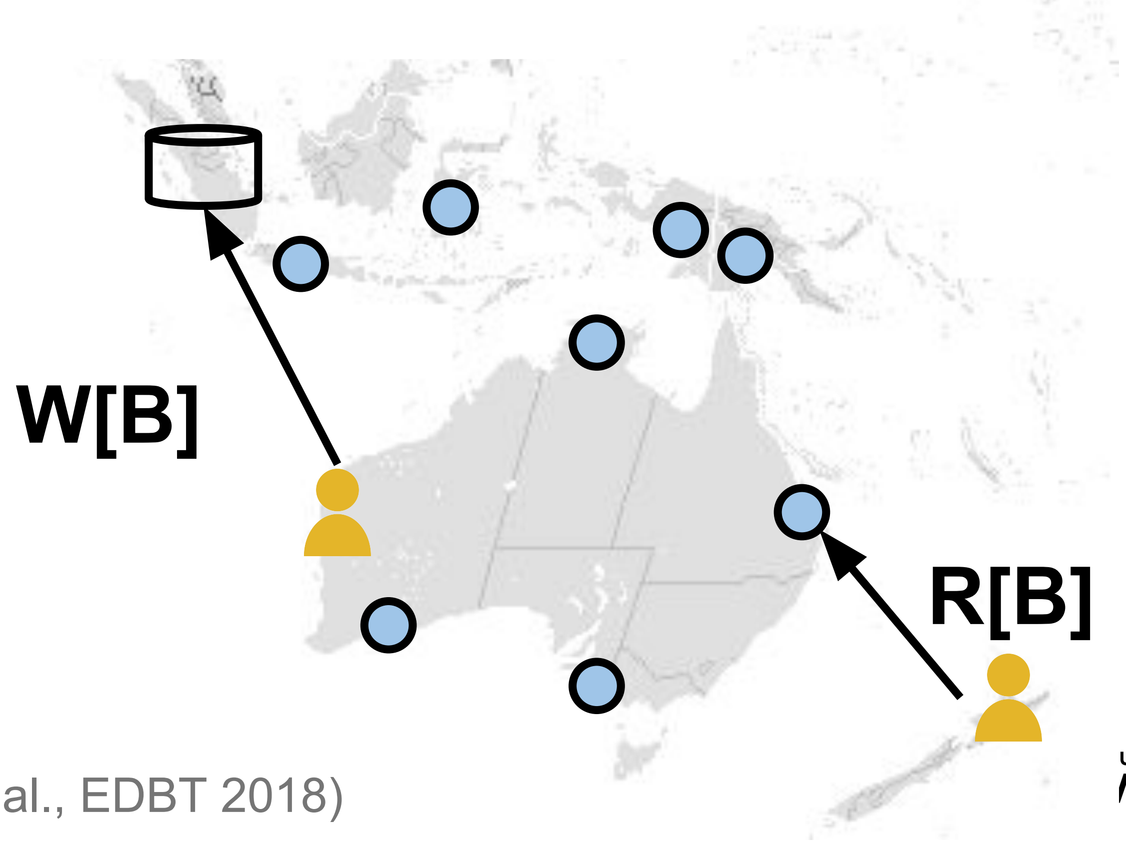
Invalidations Limit Cache Effectiveness



(Glasbergen et al., EDBT 2018)

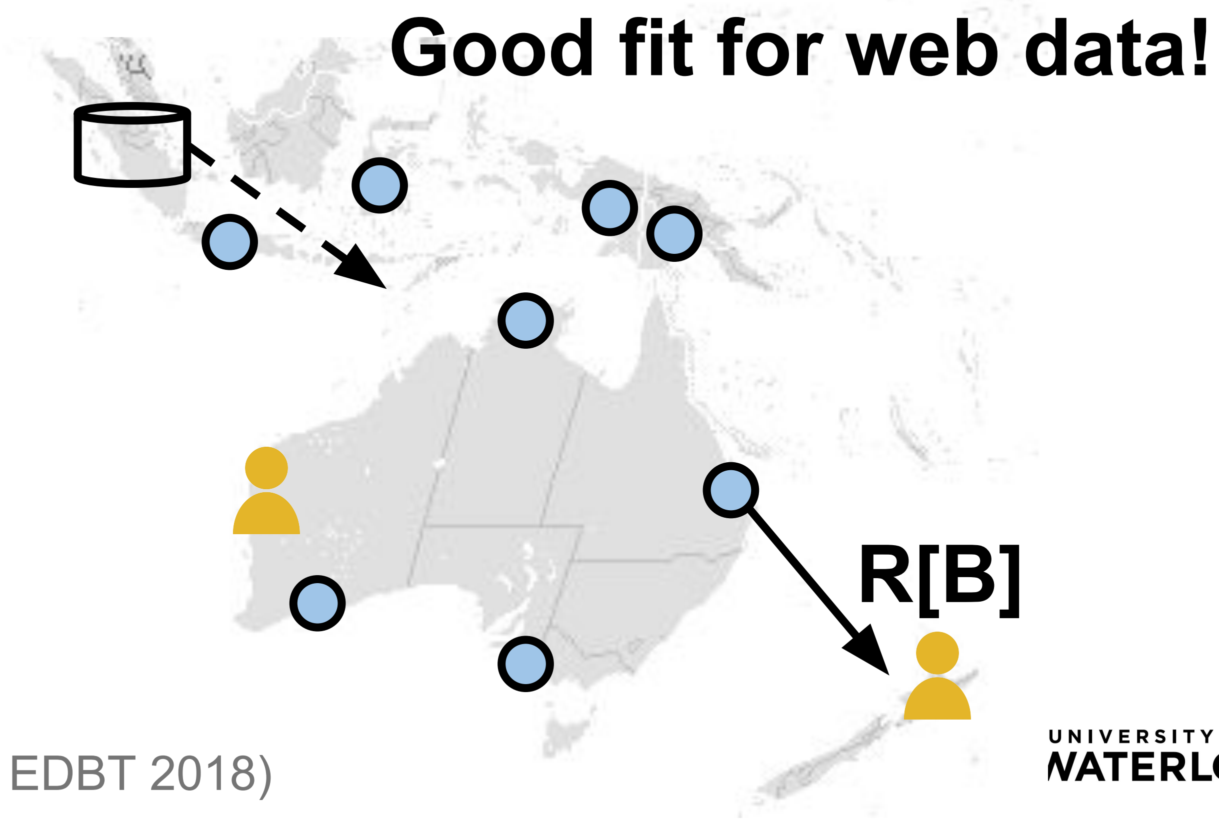
UNIVERSITY OF WATERLOO

Session Semantics



(Glasbergen et al., EDBT 2018)

Session Semantics



(Glasbergen et al., EDBT 2018)

Replication Decisions

- **How many replicas?**

  Predictively based on requests

- **Where to place replicas?**

  Client edge cache, predictively

- **How to propagate updates?**

  Cache updates with sessions

(Glasbergen et al., EDBT 2018)

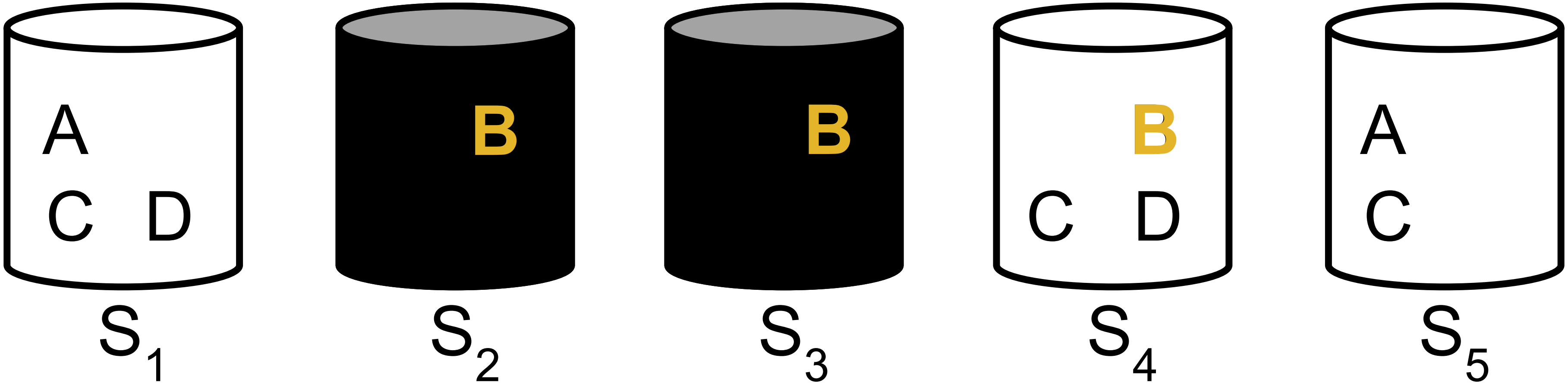
Replication for Availability

Failures are common

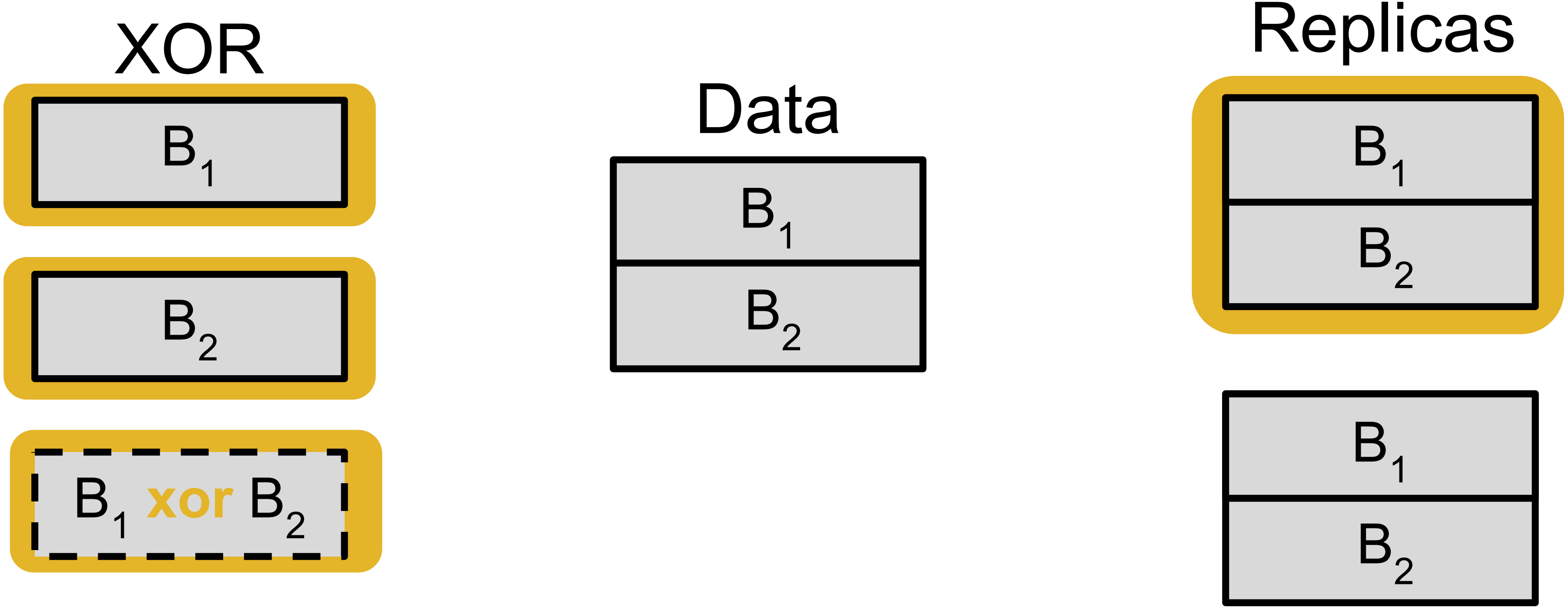
Data systems must remain **available**

Replication for Availability

Tolerating **r faults** requires **r + 1 replicas**



Lower Overhead



$$B_2 \text{ xor } (B_1 \text{ xor } B_2) = B_1$$

Erasure Coding

Data

1
2
...
k

k partitions

1
2
...
k

r parity partitions

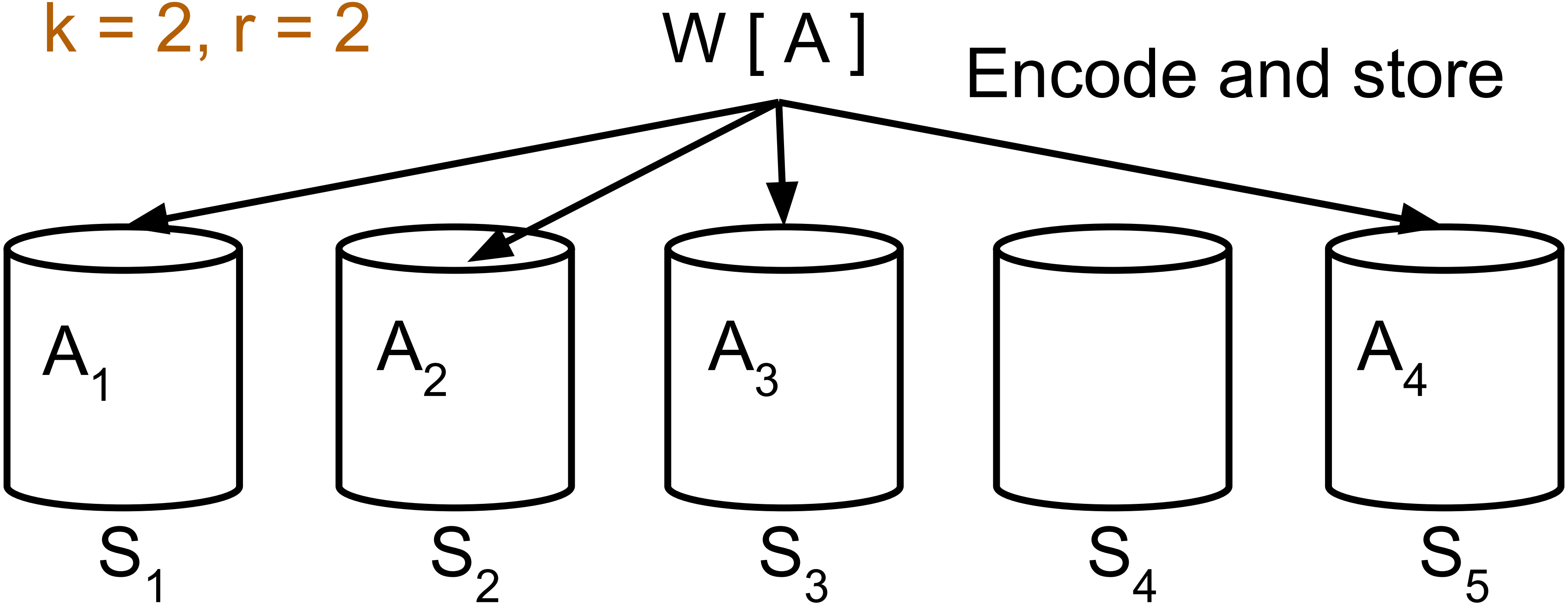
k + 1
...
k + r

Tolerating **r faults** requires **(k+r)/k** space

UNIVERSITY OF WATERLOO

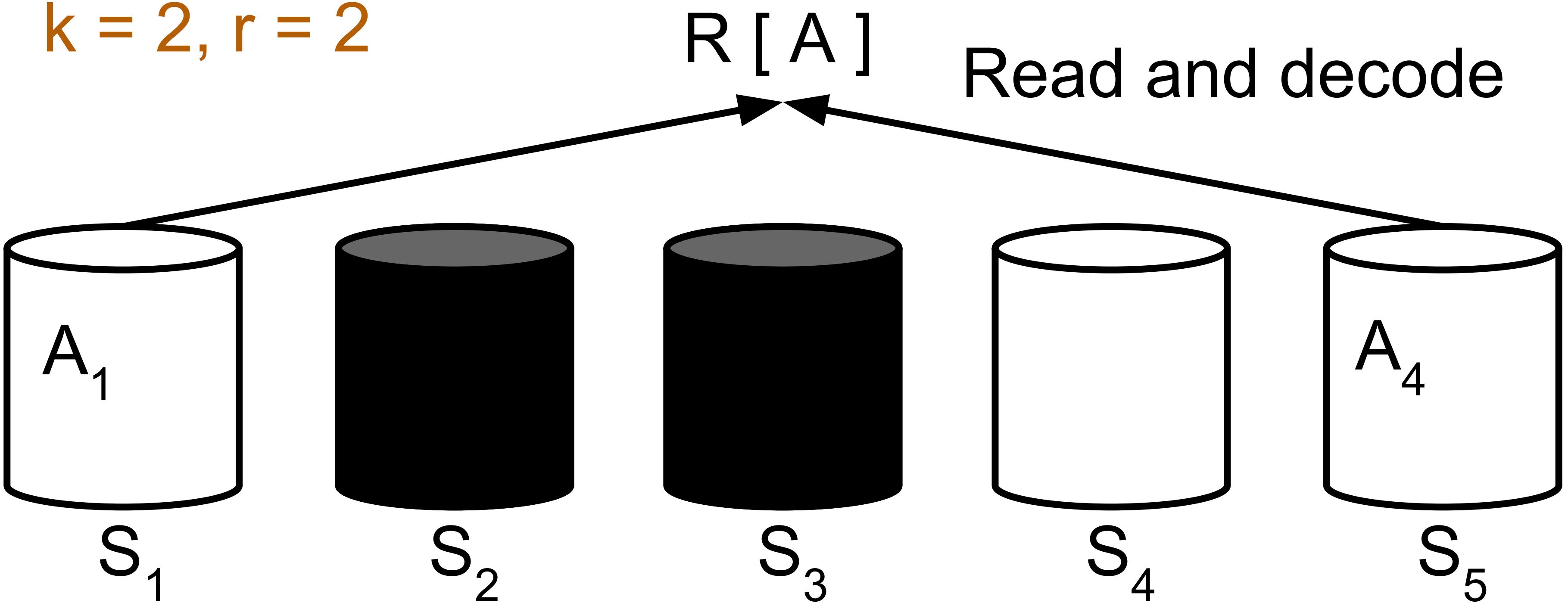
Erasure Coding

k = 2, r = 2



Erasure Coding

k = 2, r = 2



Erasure Coding

Reduces storage overhead

Requires **parallel** retrieval

Erasure Coded Storage

Where to **place** data

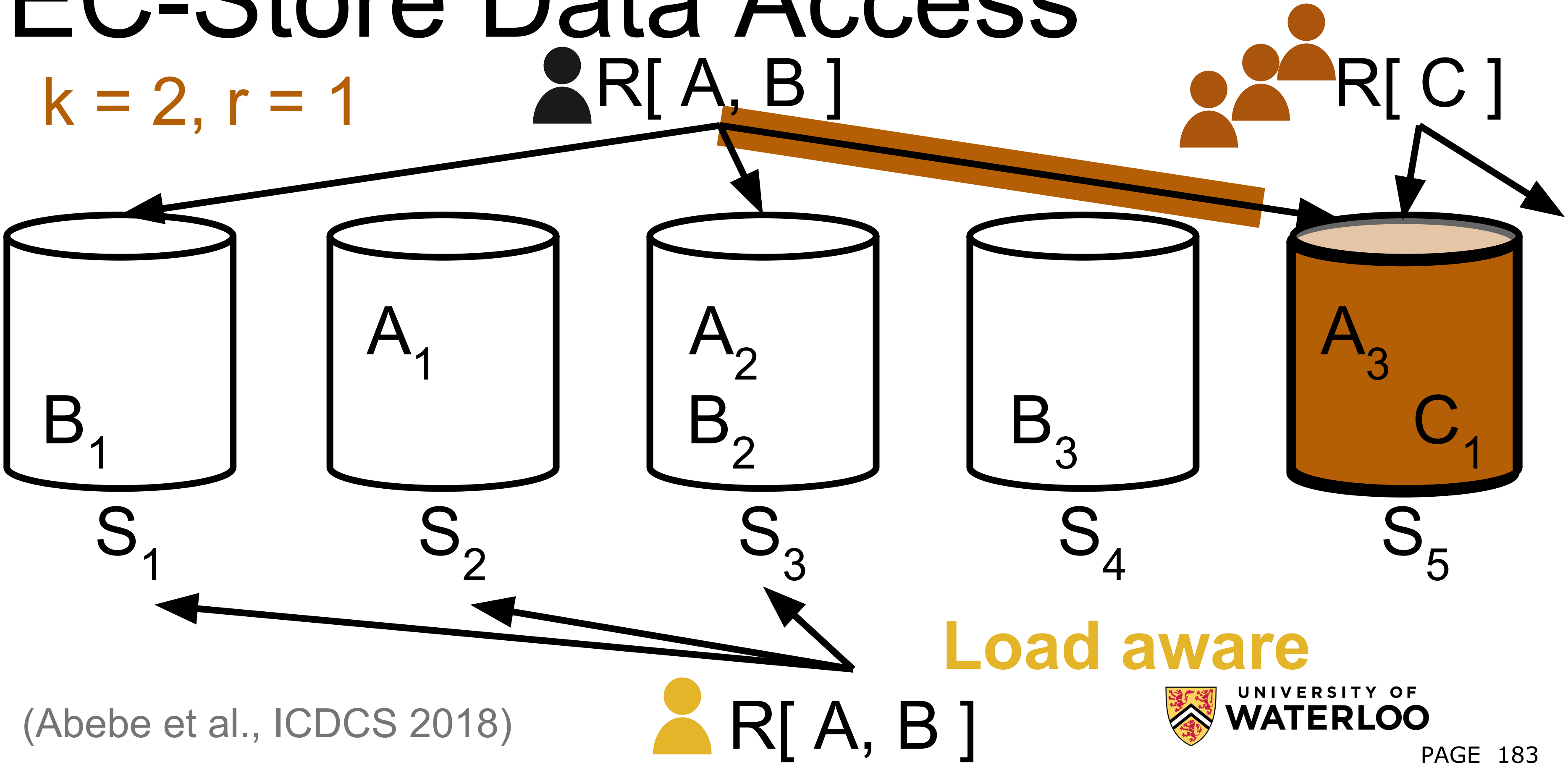
How to **access** data

EC-Store

(Abebe, ICDCS 2018)

EC-Store Data Access

k = 2, r = 1



Load aware

(Abebe et al., ICDCS 2018)

UNIVERSITY OF WATERLOO

EC-Store Data Access

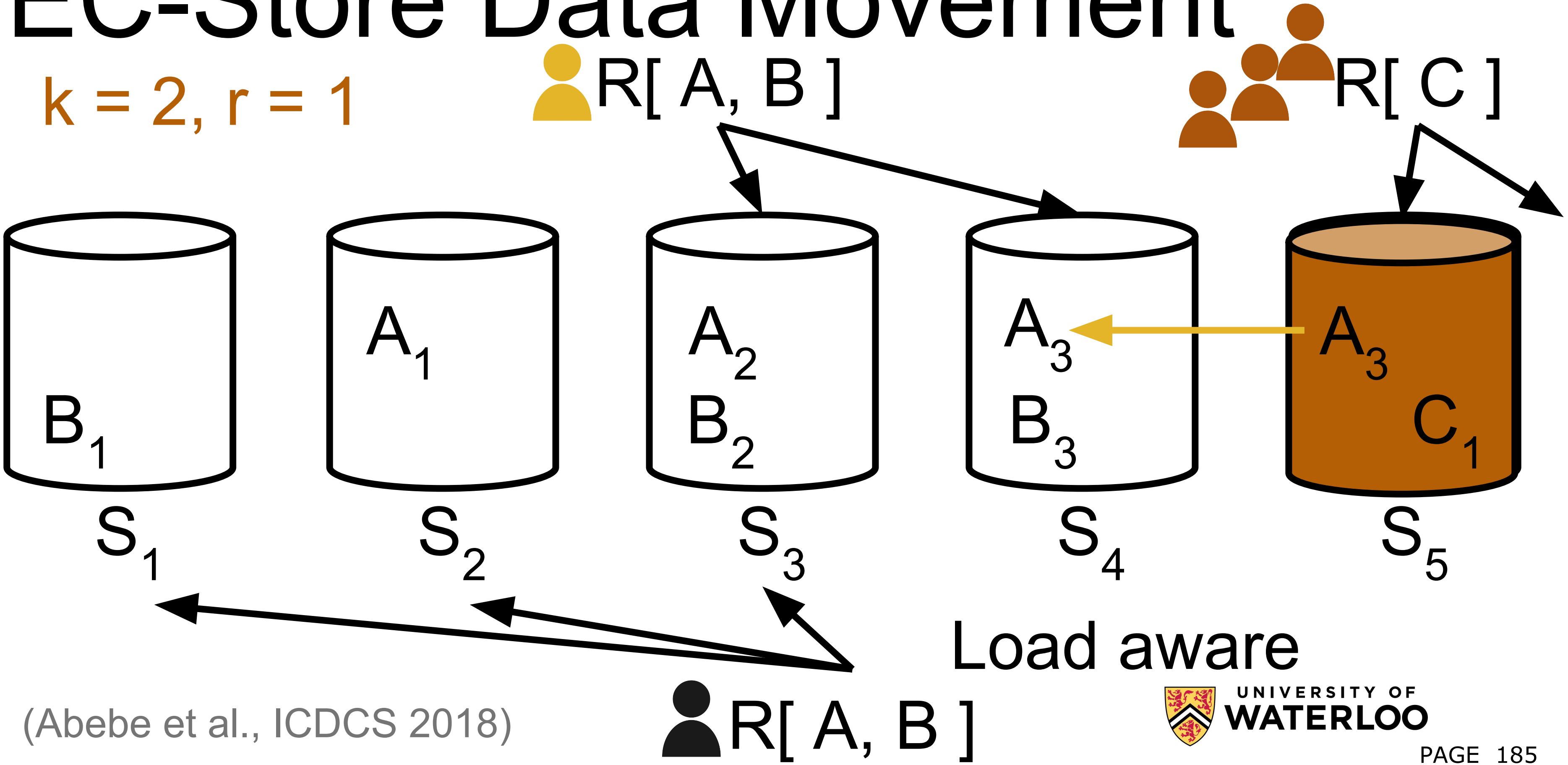
Access Strategy: Minimize **cost of access**

Cost of site access: **load** at site + **I/O** at site

(Abebe et al., ICDCS 2018)

EC-Store Data Movement

k = 2, r = 1



Load aware

(Abebe et al., ICDCS 2018)

EC-Store Data Movement

Move data to **minimize** cost of **future accesses** and **balance** **system load**

Model access patterns to **predict** **future accesses**

(Abebe et al., ICDCS 2018)

Replication Decisions

- **How many replicas?**

  Fault tolerance requirements

- **Where to place replicas?**

  Dynamic movement, using access costs

- **How to propagate updates?**

  Synchronous updates

(Abebe et al., ICDCS 2018)

Road Map

- Adaptive Replication
- Adaptive Partitioning
- Outlook

Adaptive Partitioning

Partitioning Decisions

- **How to form partitions?**
- **Where to place partitions?**
- **How to execute multi-partition operations?**

Adaptive Partitioning

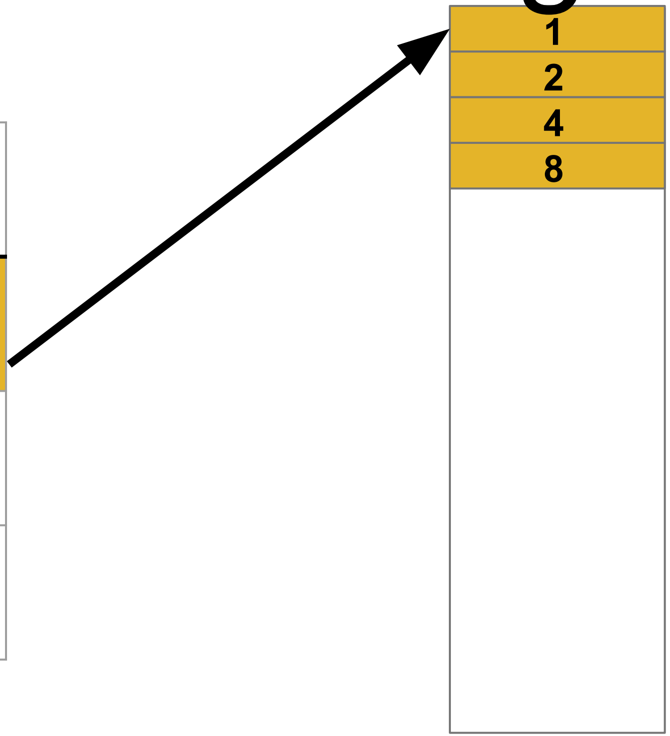
- Iterative improvements
- Partitioning per request
- Considering the overall workload
  - Heuristics

Physical Database Design

A	B	C	D
1	2	4	2
2	4	6	8
3	6	7	5

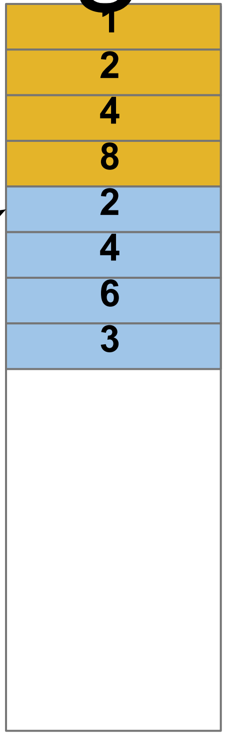
Physical Database Design

A	B	C	D
1	2	4	8
2	4	6	3
3	6	7	10



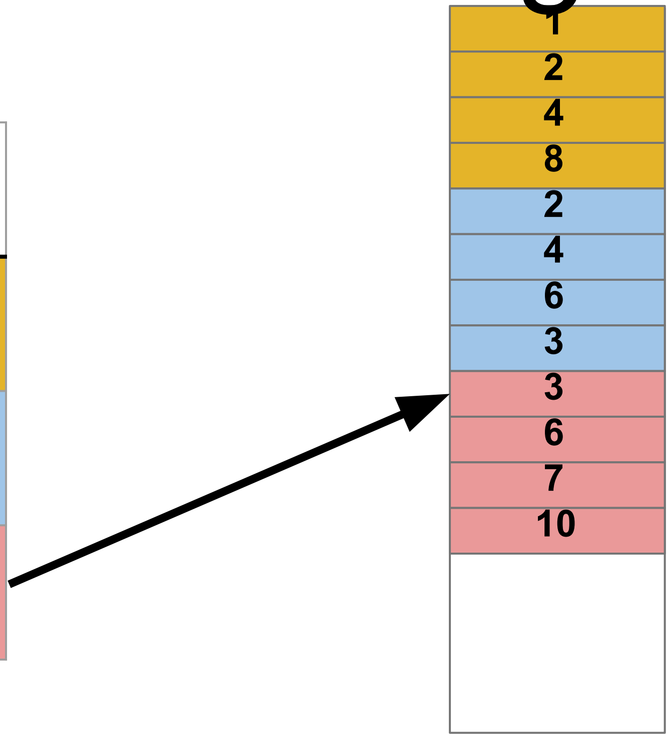
Physical Database Design

A	B	C	D
1	2	4	8
2	4	6	3
3	6	7	10



Physical Database Design

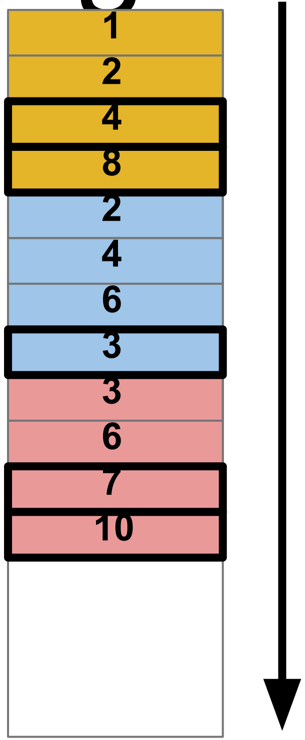
A	B	C	D
1	2	4	8
2	4	6	3
3	6	7	10



Physical Database Design

SELECT AVERAGE(C)
FROM R WHERE R.D > 5;

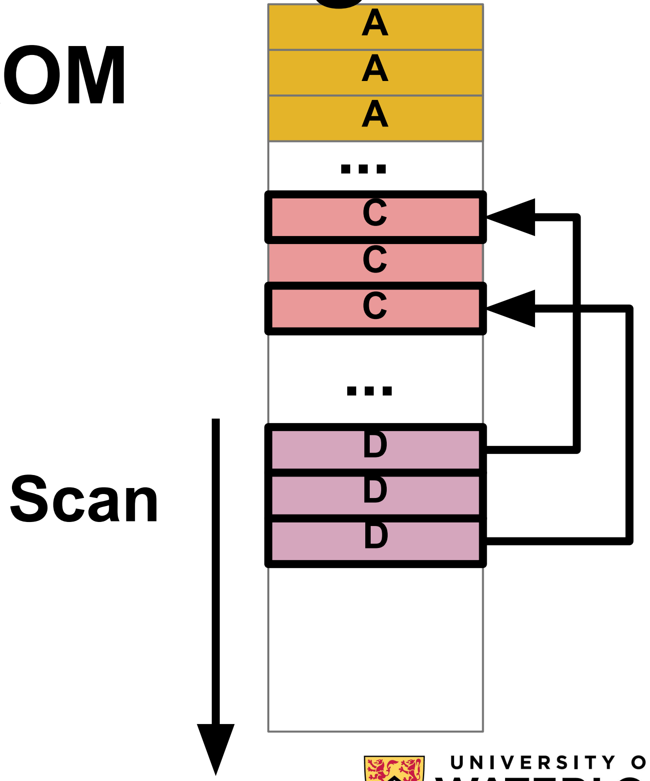
A	B	C	D
1	2	**4**	**8**
2	4	6	3
3	6	**7**	**10**



Analytic Database Design

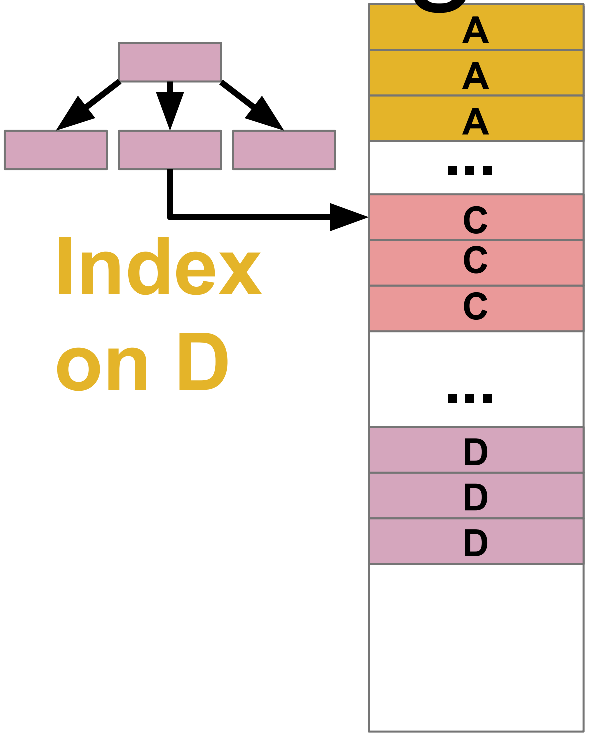
SELECT AVERAGE(C) FROM R WHERE R.D > 5;

A	B	C	D
1	2	**4**	**8**
2	4	6	3
3	6	**7**	**10**



Analytic Database Design

SELECT AVERAGE(C) FROM R WHERE R.D > 5;



Need to know what to index upfront

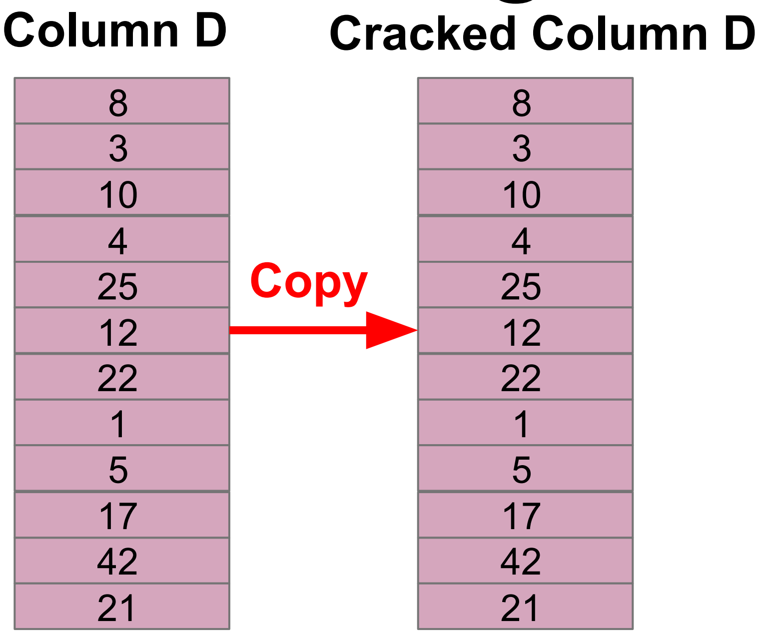
Adaptive Range Indexing

SELECT ... FROM R WHERE
R.D > 5;

SELECT … FROM R WHERE
R.D > 5 AND R.D < 10;

SELECT … FROM R WHERE
R.D > 10 AND R.D < 20;

Database Cracking



(Idreos et al., CIDR 2007)

UNIVERSITY OF WATERLOO

Indexes via Partitioning

Cracked Column D

8
3
10
4
25
12
22
1
5
17
42
21

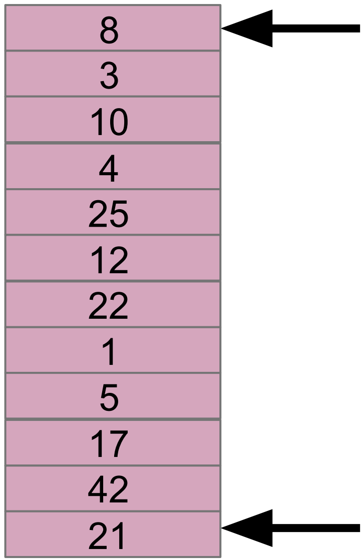
SELECT … WHERE R.D > 5

(Idreos et al., CIDR 2007)

UNIVERSITY OF WATERLOO

Indexes via Partitioning

Cracked Column D

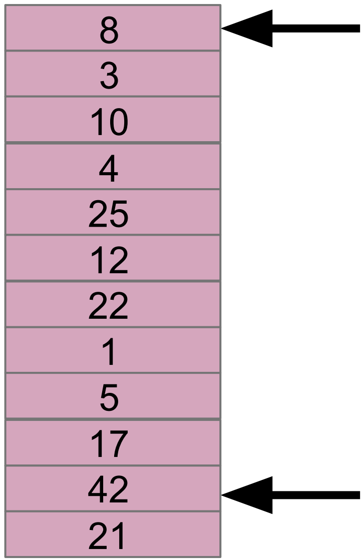


SELECT … WHERE
R.D > 5

(Idreos et al., CIDR 2007)

Indexes via Partitioning

Cracked Column D

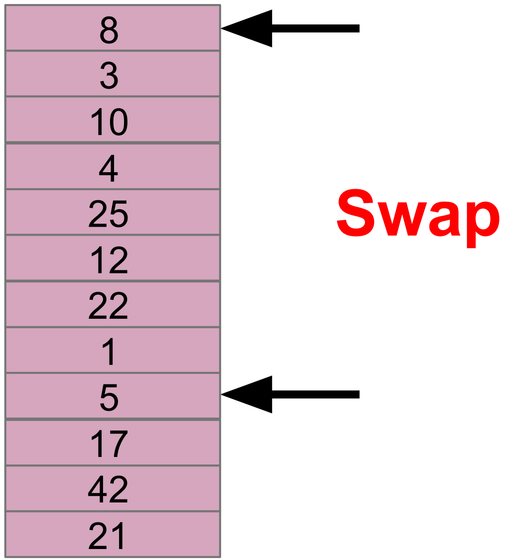


SELECT … WHERE
R.D > 5

(Idreos et al., CIDR 2007)

Indexes via Partitioning

Cracked Column D

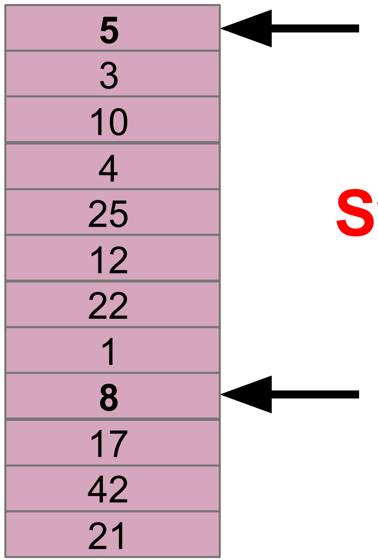


SELECT … WHERE
R.D > 5

(Idreos et al., CIDR 2007)

Indexes via Partitioning

Cracked Column D



SELECT … WHERE R.D > 5

Swap

(Idreos et al., CIDR 2007)

Indexes via Partitioning

SELECT … WHERE
R.D > 5

Cracked Column D

5
3
10 ←
4
25
12
22
1 ←
8
17
42
21

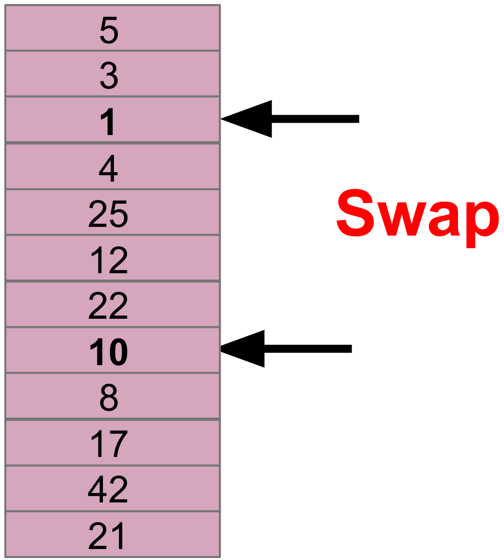
Swap

(Idreos et al., CIDR 2007)

Indexes via Partitioning

SELECT … WHERE R.D > 5

Cracked Column D



(Idreos et al., CIDR 2007)

UNIVERSITY OF WATERLOO

Indexes via Partitioning

SELECT … WHERE R.D > 5

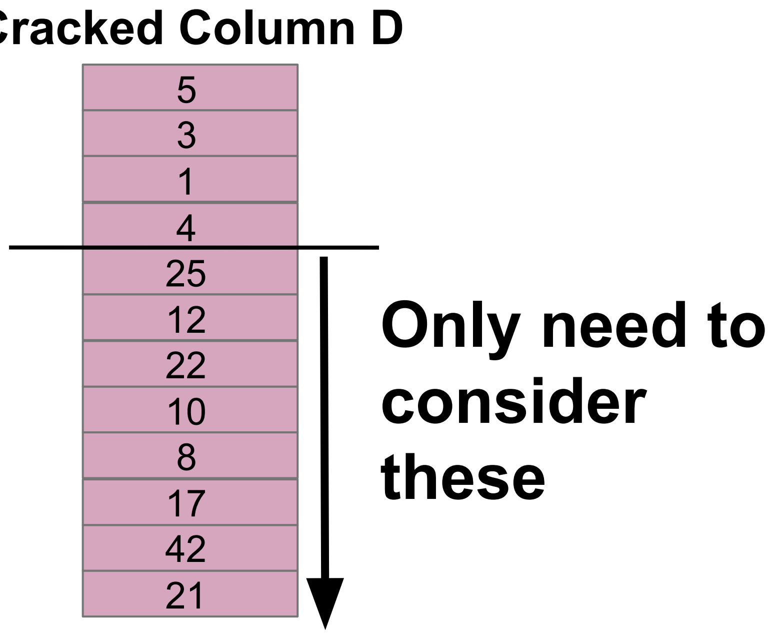
Cracked Column D
5
3
1
4
25
12
22
10
8
17
42
21

(Idreos et al., CIDR 2007)

UNIVERSITY OF WATERLOO

Indexes via Partitioning

SELECT … WHERE R.D > 5 AND R.D < 10

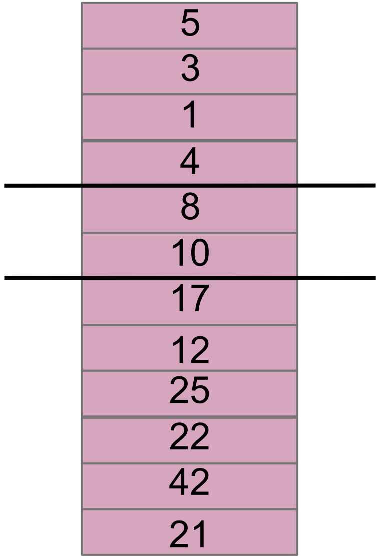


(Idreos et al., CIDR 2007)

Indexes via Partitioning

SELECT … WHERE R.D > 5 AND R.D < 10

Cracked Column D

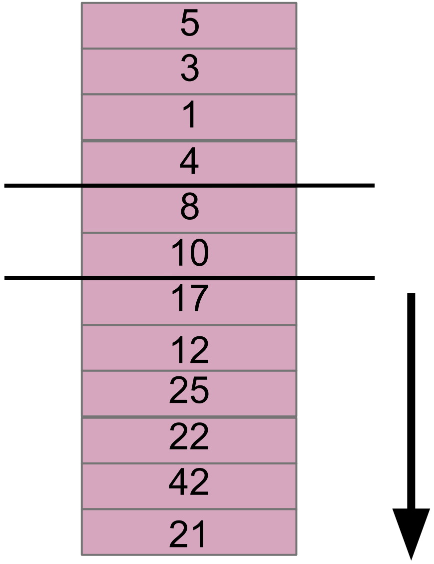


(Idreos et al., CIDR 2007)

Indexes via Partitioning

SELECT … WHERE R.D > 10 AND R.D < 20

Cracked Column D



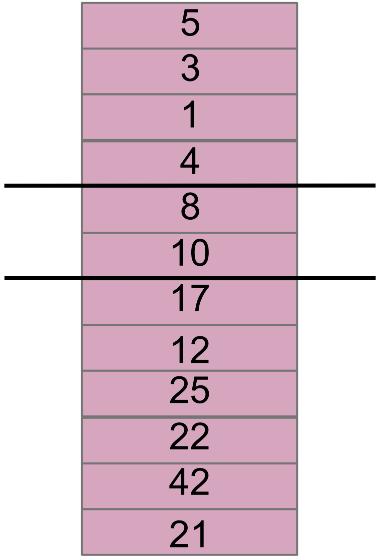
Only need to consider these

(Idreos et al., CIDR 2007)

Indexes via Partitioning

SELECT … WHERE R.D > 10 AND R.D < 20

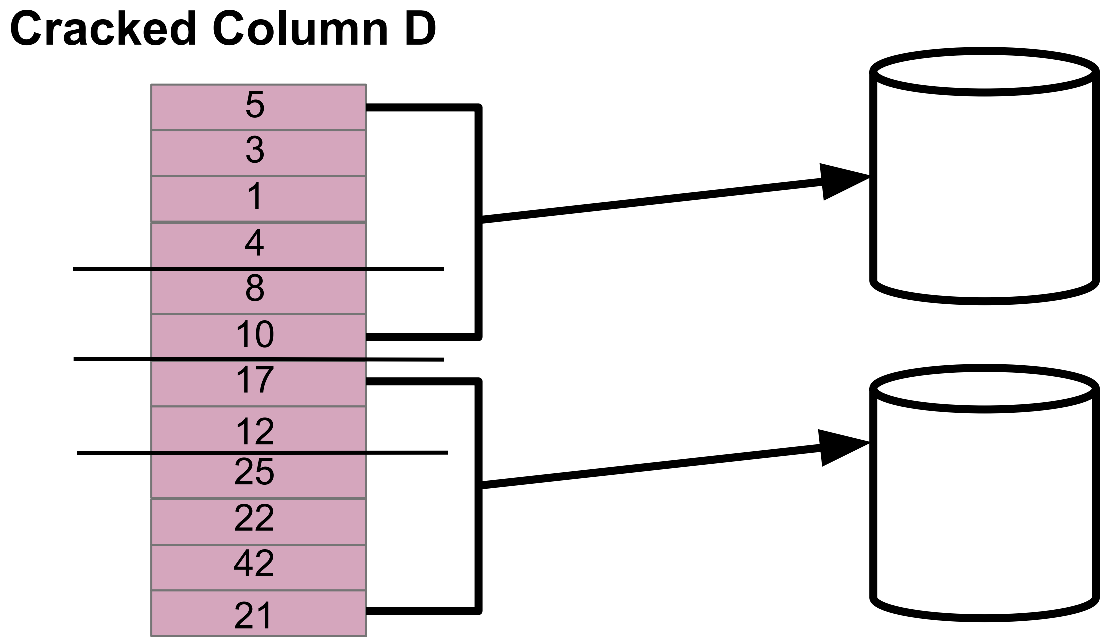
Cracked Column D



Iterative Partitioning for Indexing

(Idreos et al., CIDR 2007)

Cracking: Extensions



Advanced Cracking Methods

Distribution

(Idreos et al., CIDR 2007)

Partitioning Decisions

- **How to form partitions?**

  Iteratively, based on queries

- **Where to place partitions?**

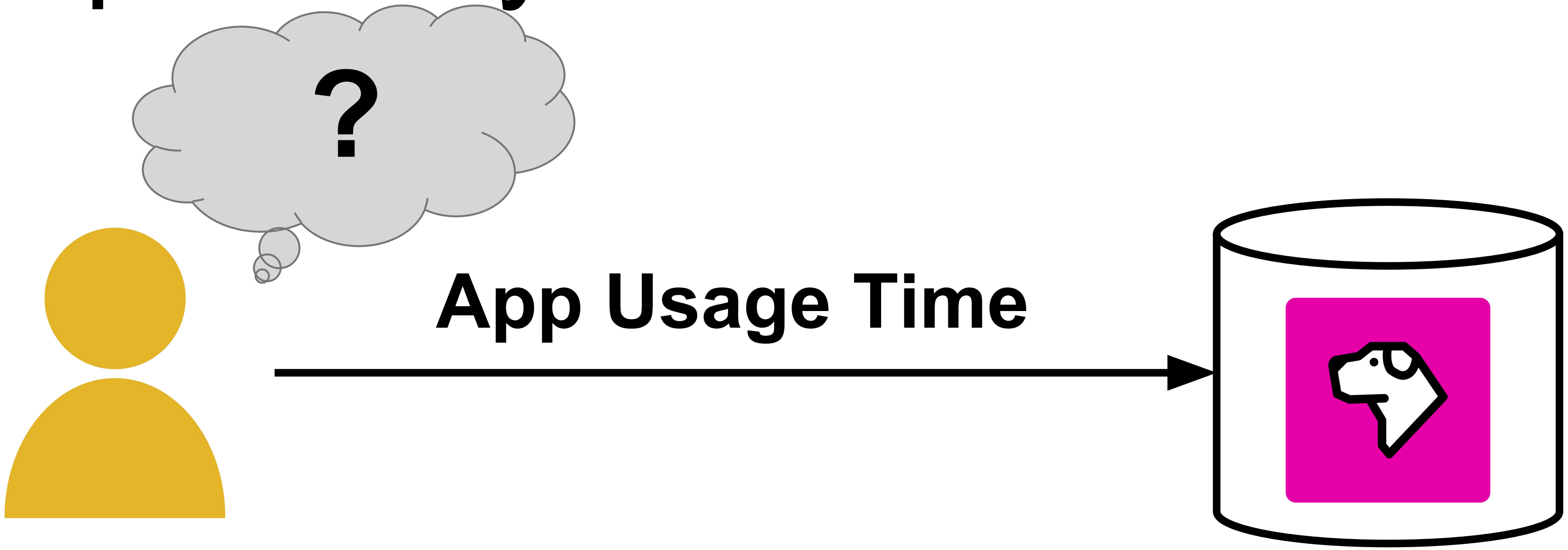
  Sorted in memory

- **How to execute multi-partition operations?**

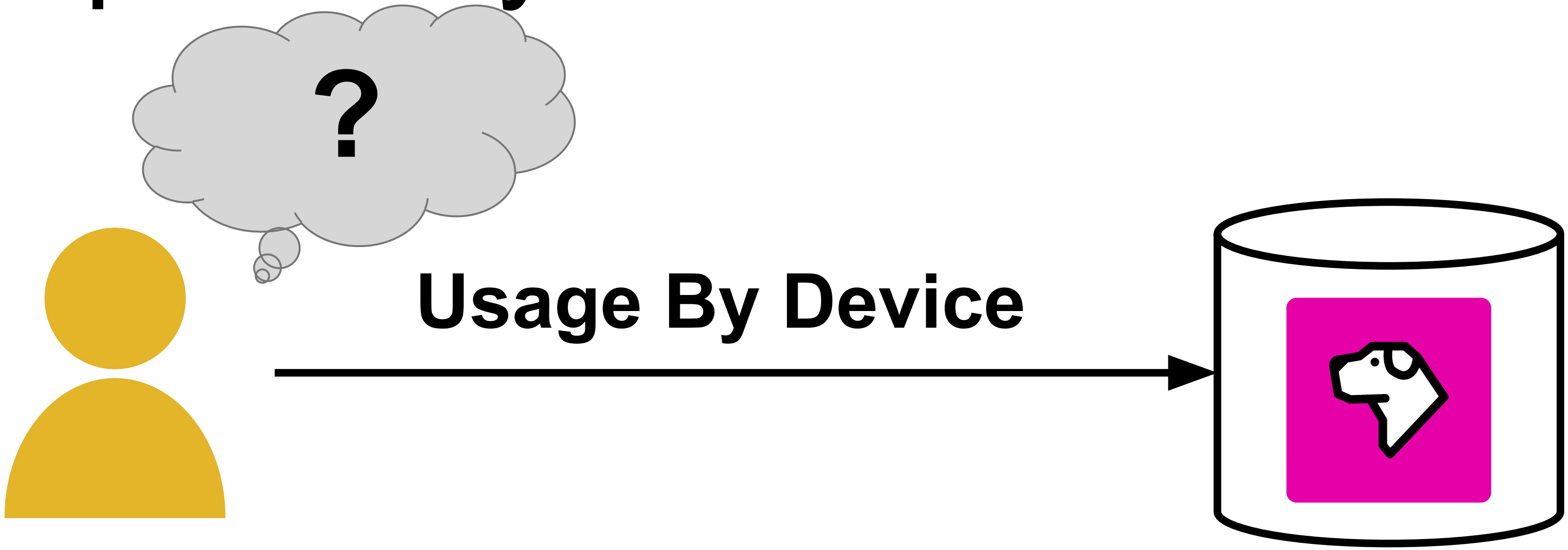
  N/A

(Idreos et al., CIDR 2007)

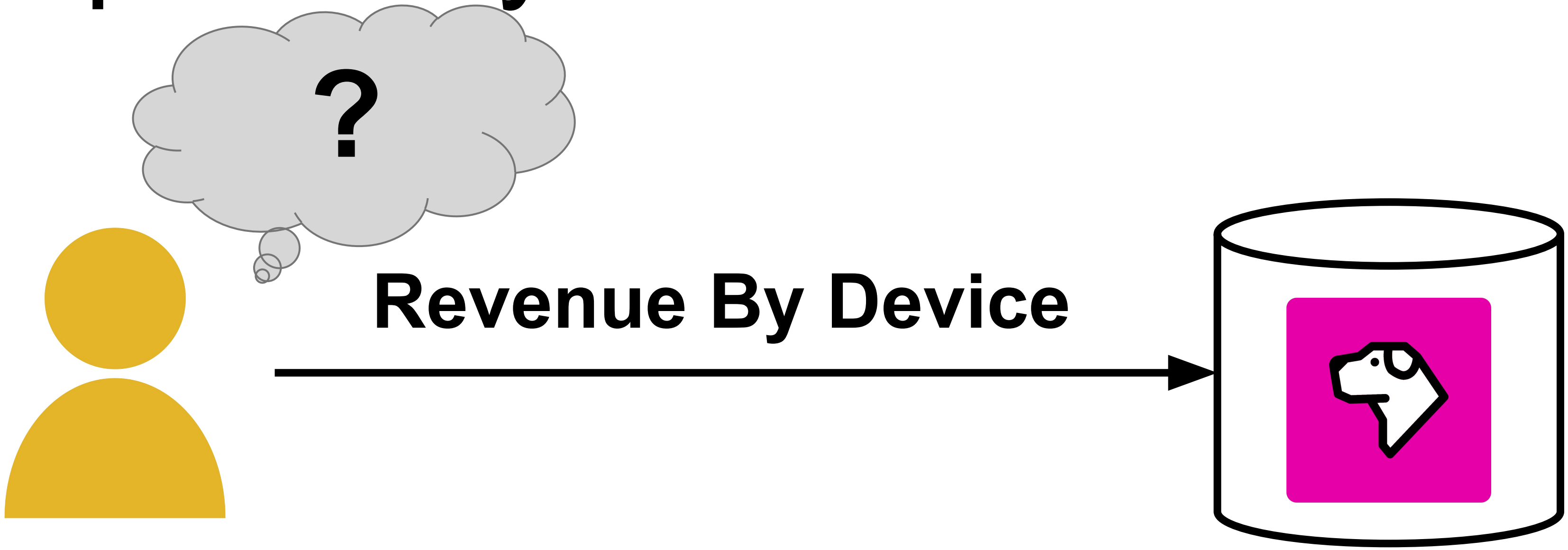
Exploratory Workloads



Exploratory Workloads



Exploratory Workloads



Exploratory Workloads

No upfront information, need generic partitioning!

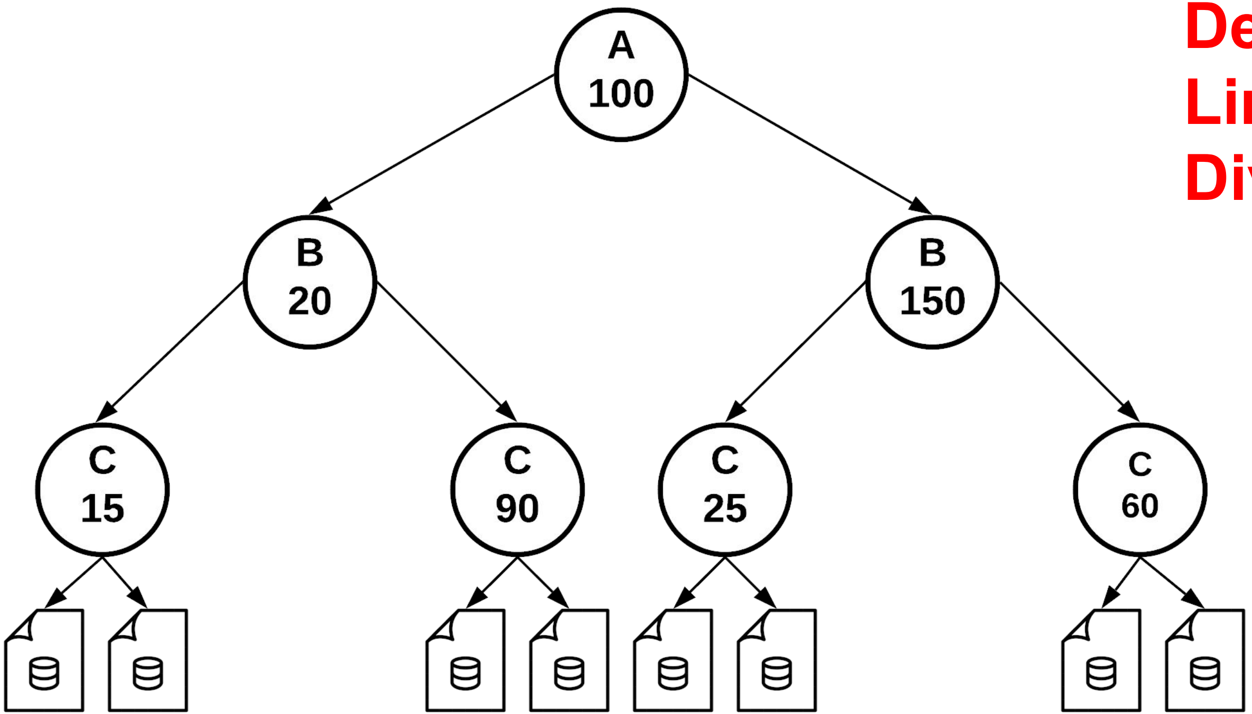
Initial Partitioning (KD-Tree)

512 MB

256 MB

128 MB

64 MB



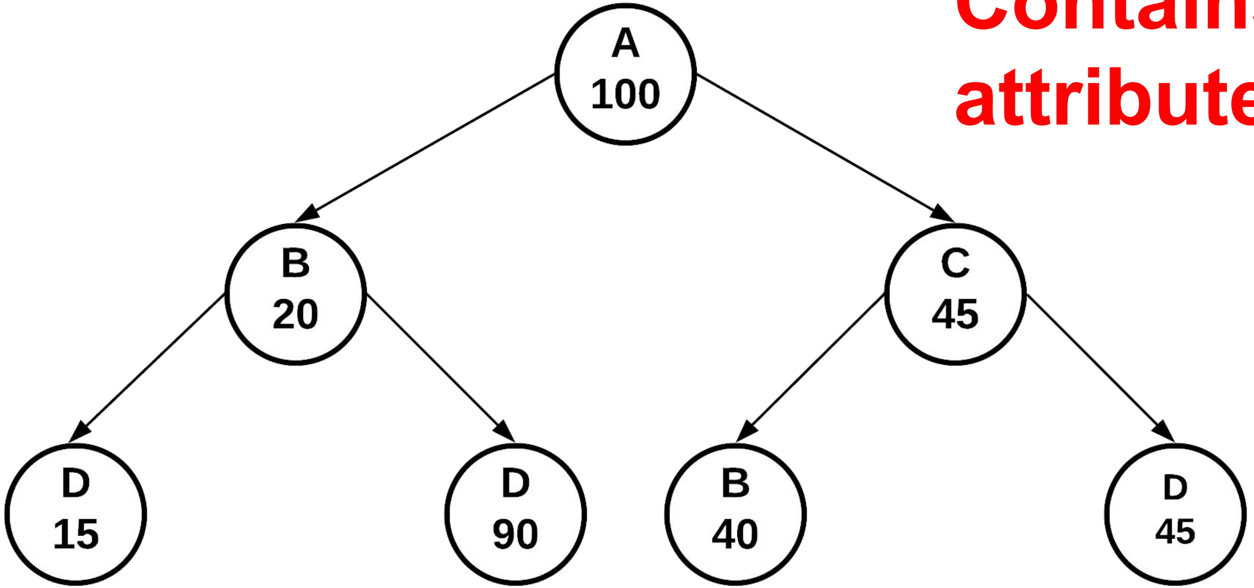
Depth Limits Division

(Shanbhag et al., SoCC 2017)

UNIVERSITY OF WATERLOO

Heterogeneous Tree

Contains more attributes!



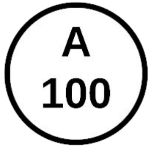
(Shanbhag et al., SoCC 2017)

Building the Partitioning

A: 0.0
B: 0.0
C: 0.0
D: 0.0

(Shanbhag et al., SoCC 2017)

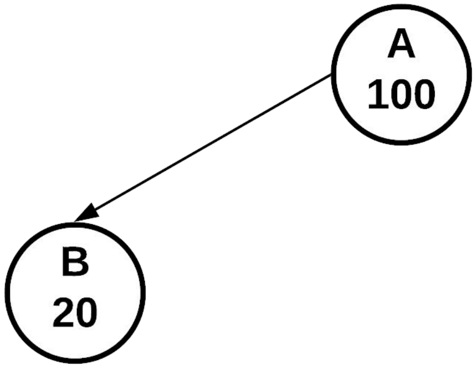
Building the Partitioning



A: 1.0
B: 0.0
C: 0.0
D: 0.0

(Shanbhag et al., SoCC 2017)

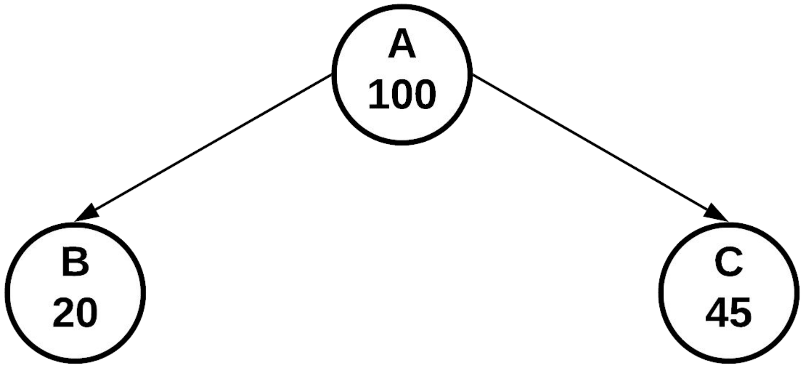
Building the Partitioning



A: 1.0
B: 0.5
C: 0.0
D: 0.0

(Shanbhag et al., SoCC 2017)

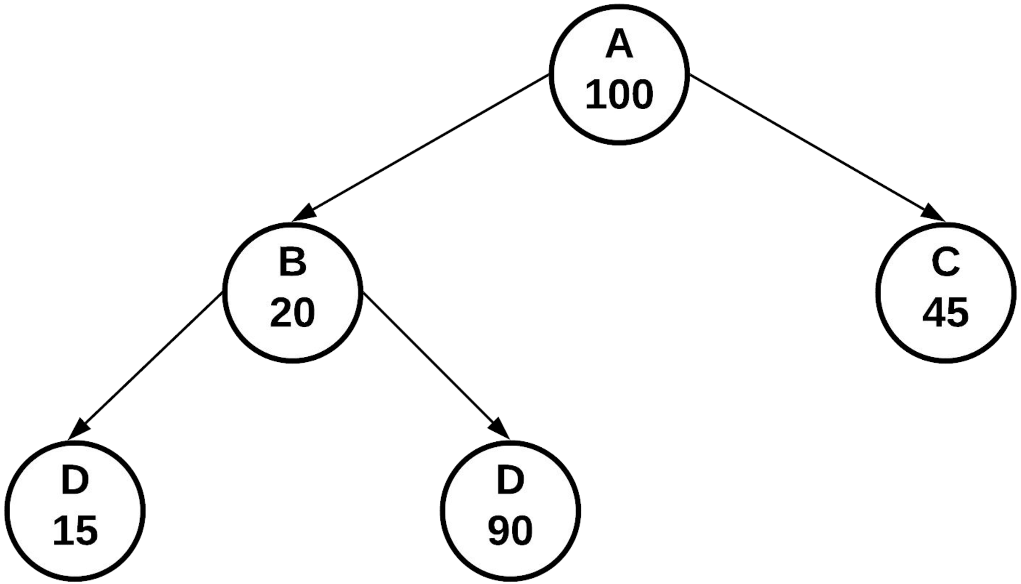
Building the Partitioning



A: 1.0
B: 0.5
C: 0.5
D: 0.0

(Shanbhag et al., SoCC 2017)

Building the Partitioning

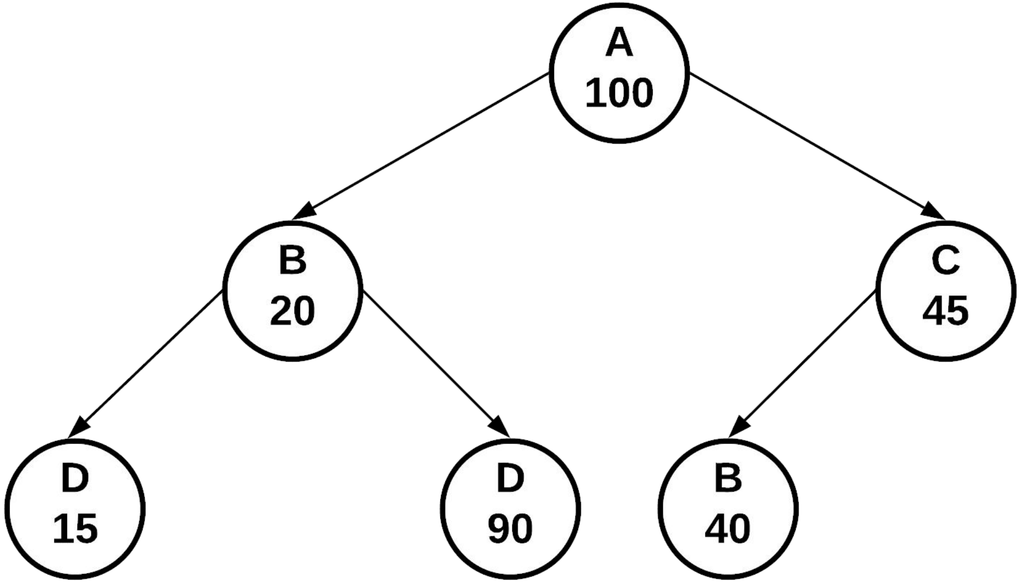


A: 1.0
B: 0.5
C: 0.5
D: 0.5

(Shanbhag et al., SoCC 2017)

UNIVERSITY OF WATERLOO

Building the Partitioning



A: 1.0

B: 0.75

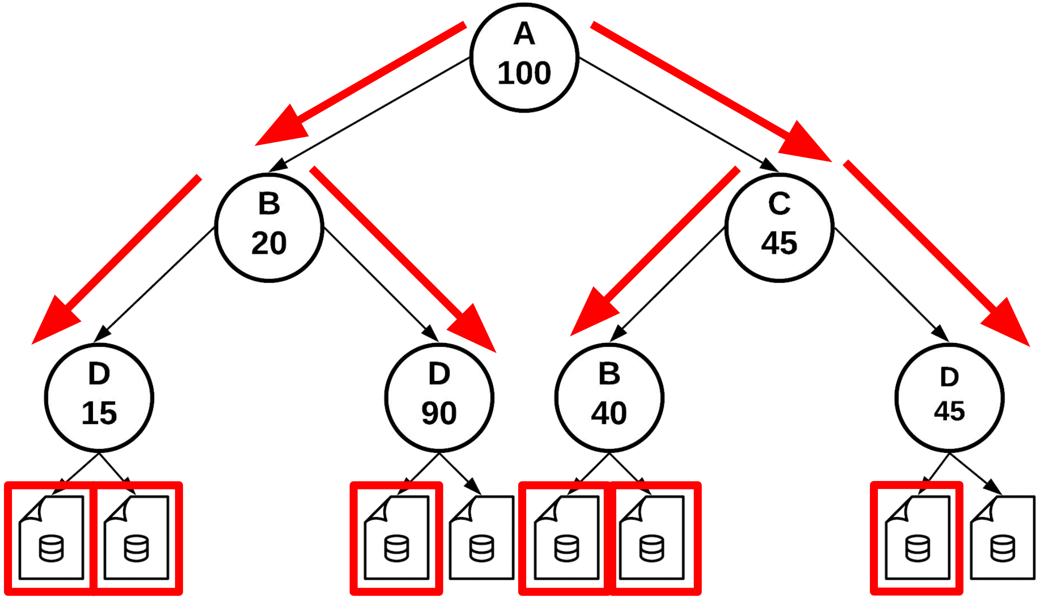
C: 0.5

D: 0.5

(Shanbhag et al., SoCC 2017)

Adaptive Partitioning

$$Q_1 = \sigma_{D \leq 45}$$

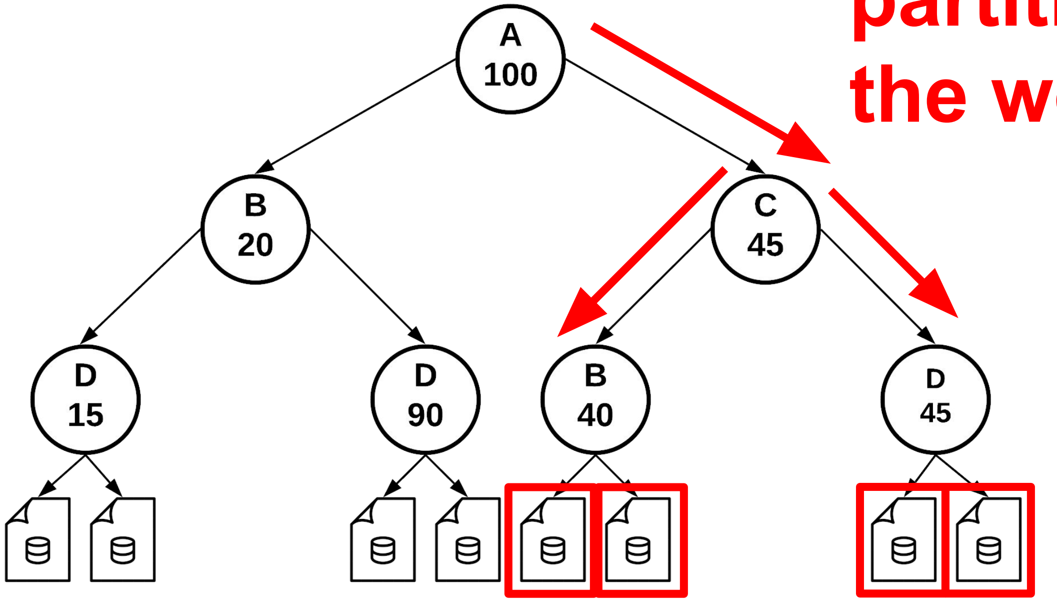


(Shanbhag et al., SoCC 2017)

UNIVERSITY OF WATERLOO

Adaptive Partitioning

$Q_2 = \sigma_{A \geqq 125}$



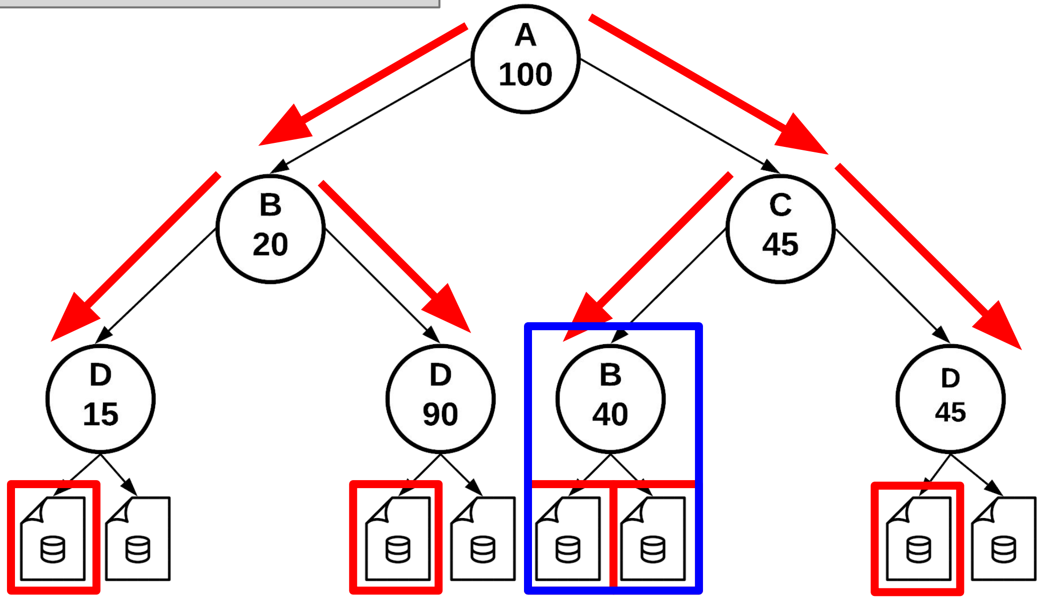
Refine partitioning per the workload!

(Shanbhag et al., SoCC 2017)

Adaptive Partitioning: When?

$Q_1, Q_2, Q_3, Q_1, \ldots$

$Q_1 = \sigma_{D \leq 45}$



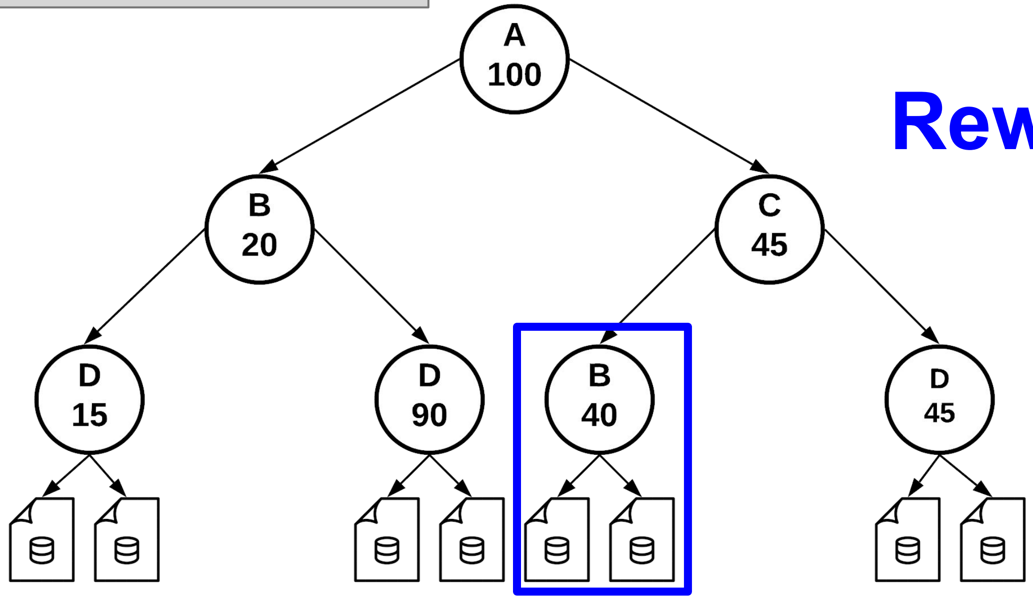
(Shanbhag et al., SoCC 2017)

Swap Operation

$Q_1, Q_2, Q_3, Q_1, \ldots$

$Q_1 = \sigma_{D \leq 45}$

Rewrite Tree

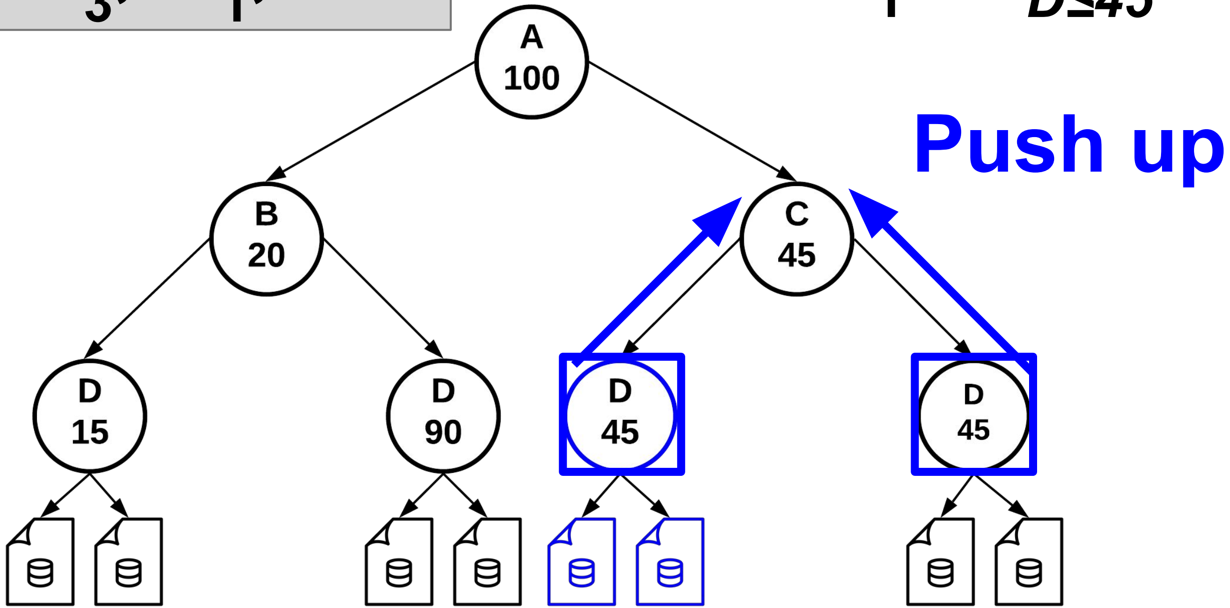


(Shanbhag et al., SoCC 2017)

Push Up Operation

$Q_1, Q_2, Q_3, Q_1, \ldots$

$Q_1 = \sigma_{D \leq 45}$

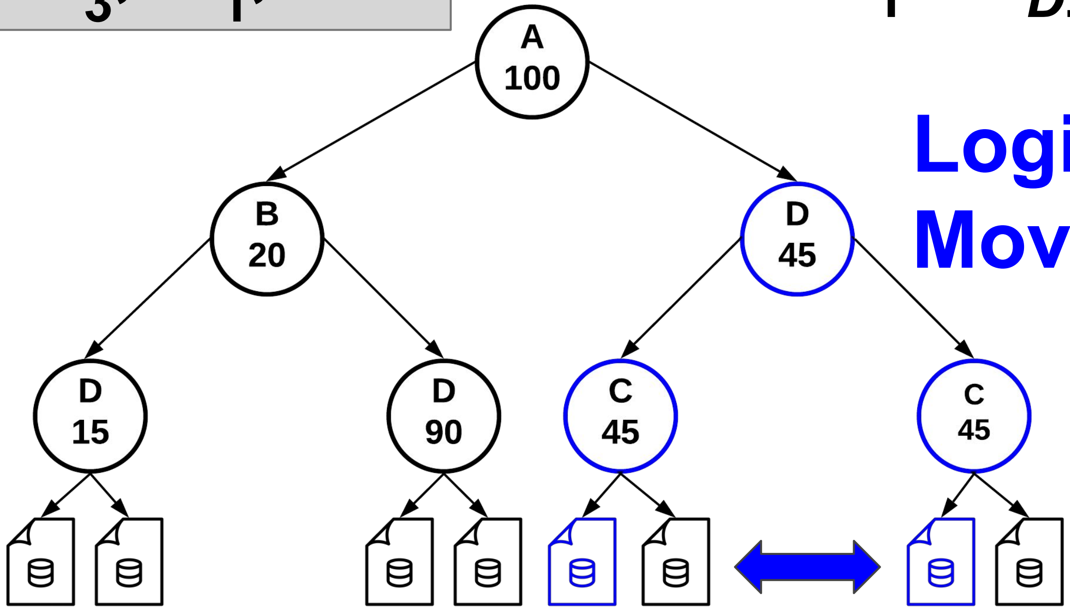


(Shanbhag et al., SoCC 2017)

Push Up Operation

$Q_1, Q_2, Q_3, Q_1, \ldots$

$Q_1 = \sigma_{D \le 45}$

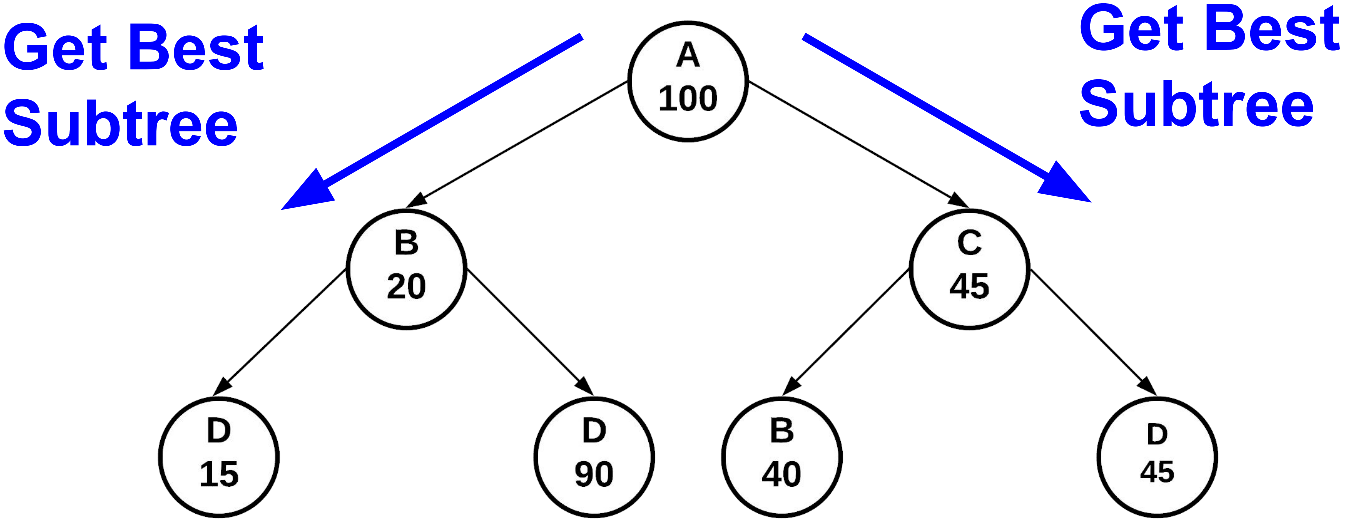


Logical Movement

(Shanbhag et al., SoCC 2017)

Divide and Conquer

$Q_1 = \sigma_{D \leq 45}$



(Shanbhag et al., SoCC 2017)

Partitioning Decisions

- **How to form partitions?**

  Upfront then iteratively, based on queries

- **Where to place partitions?**

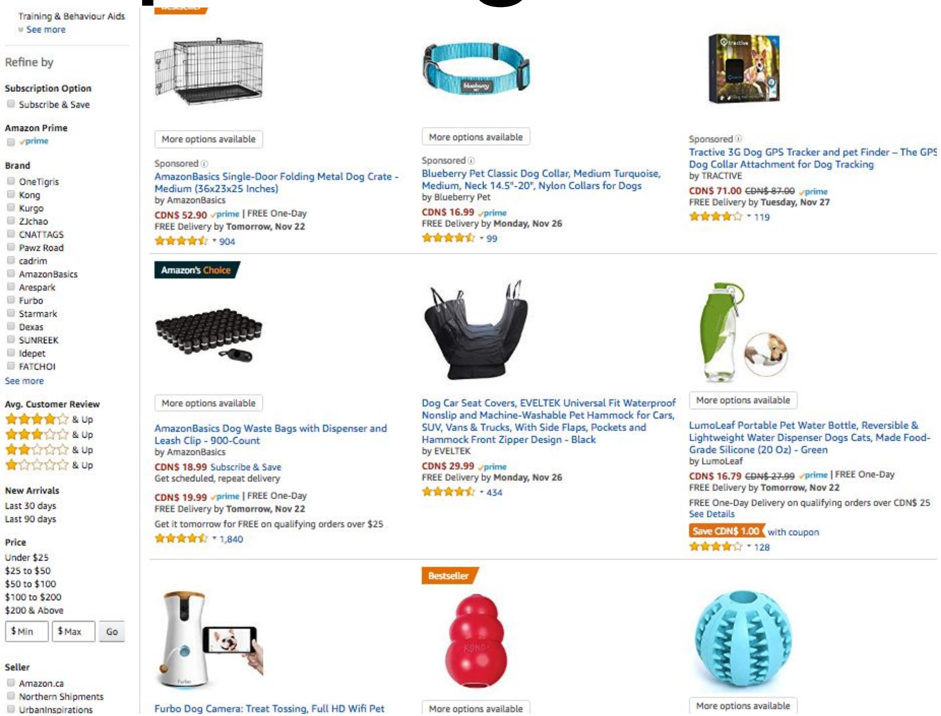
  Rely on HDFS

- **How to execute multi-partition operations?**

  Rely on HDFS

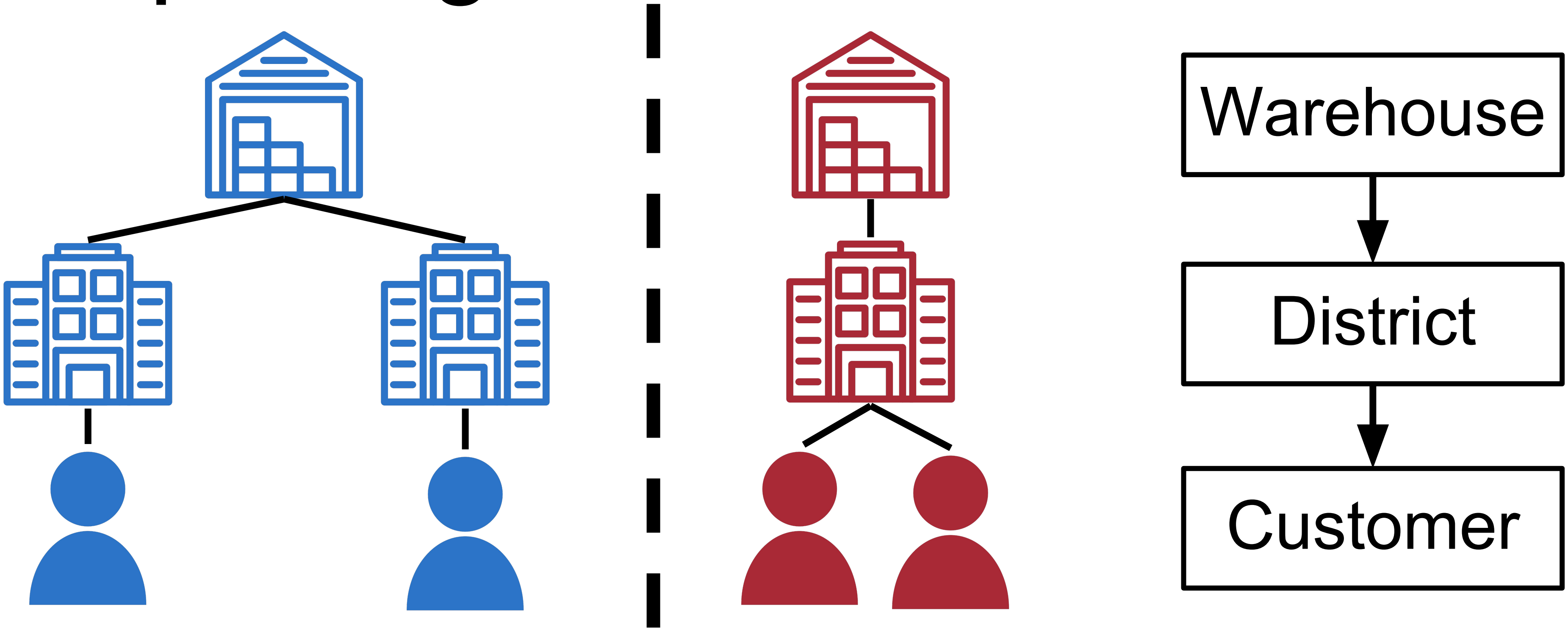
(Shanbhag et al., SoCC 2017)

Exploiting Workloads

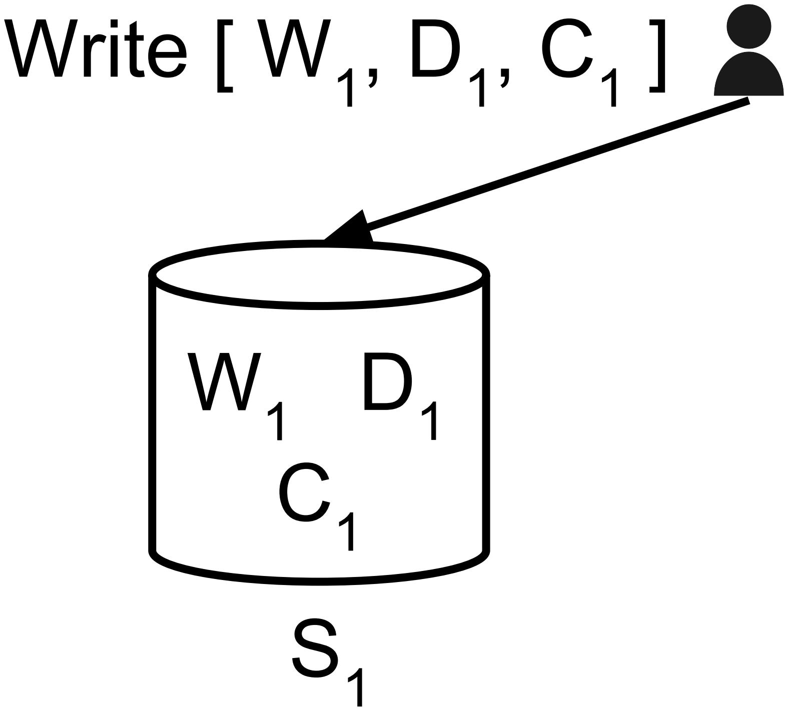


- Known ahead of time
- Parameterized
- Repetitive

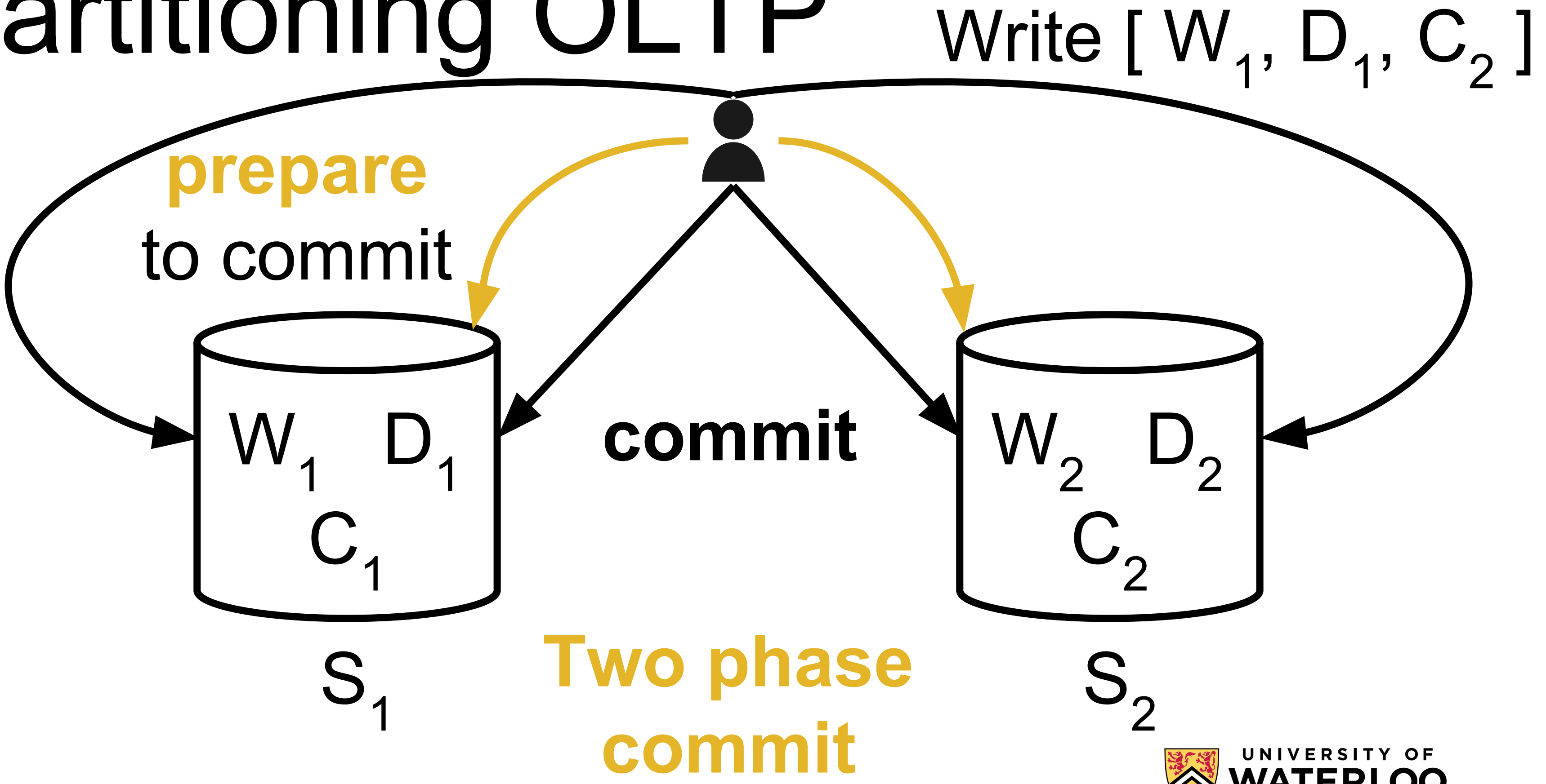
Exploiting Workloads - OLTP



Partitioning OLTP



Partitioning OLTP



Partitioning OLTP

Workload based repartitioning

Per transaction partitioning

Later in the tutorial

G-Store

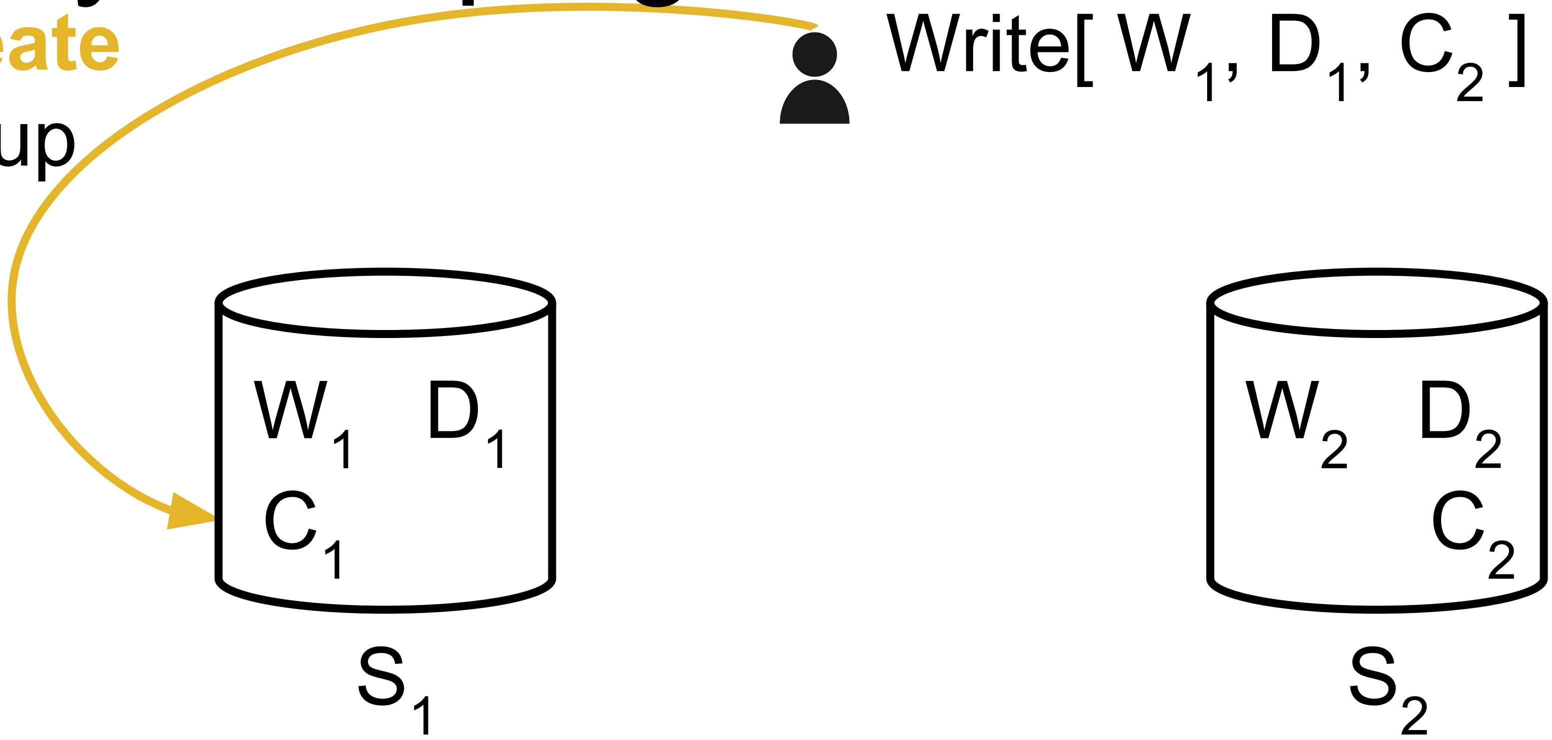
(Das et al., SoCC 2010)

L-Store

(Lin et al., SIGMOD 2016)

Key Grouping

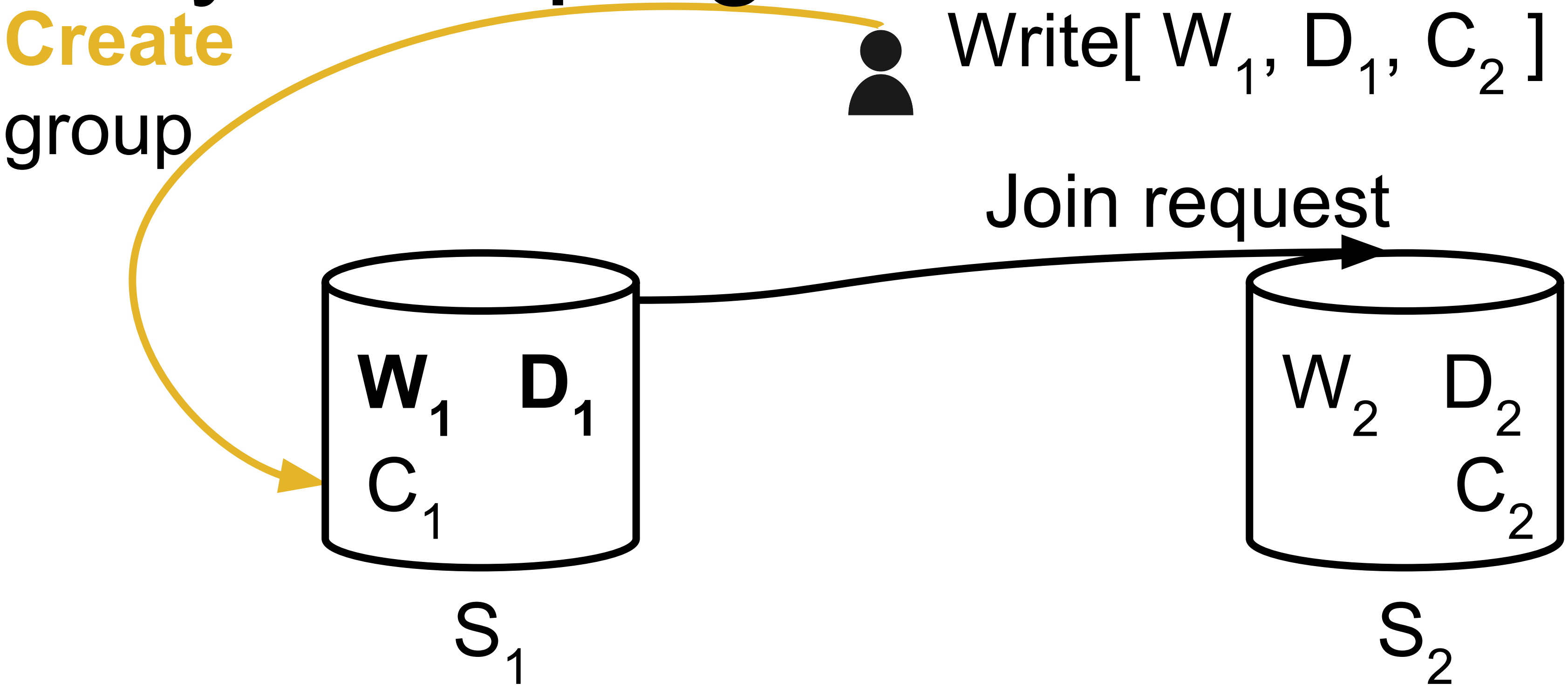
Create group



(Das et al., SoCC 2010)

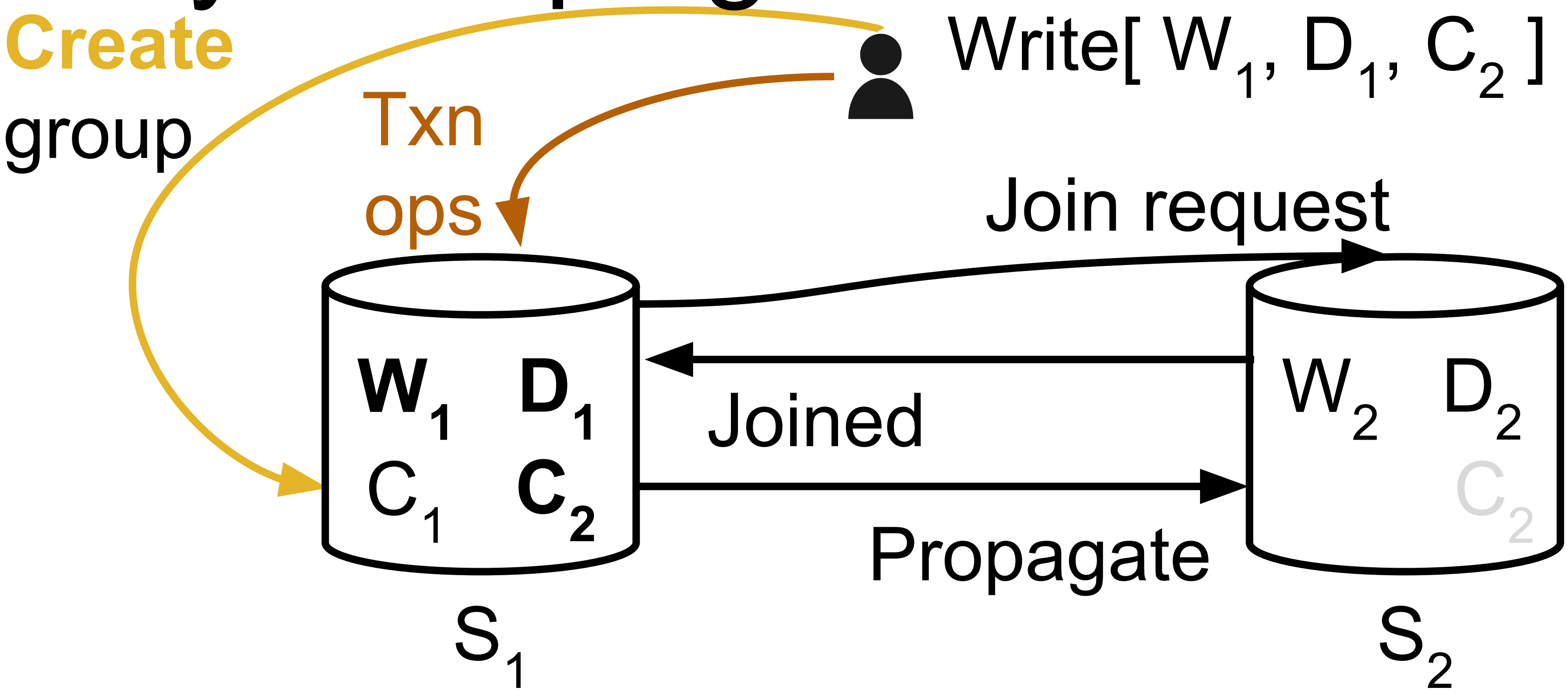
UNIVERSITY OF WATERLOO

Key Grouping



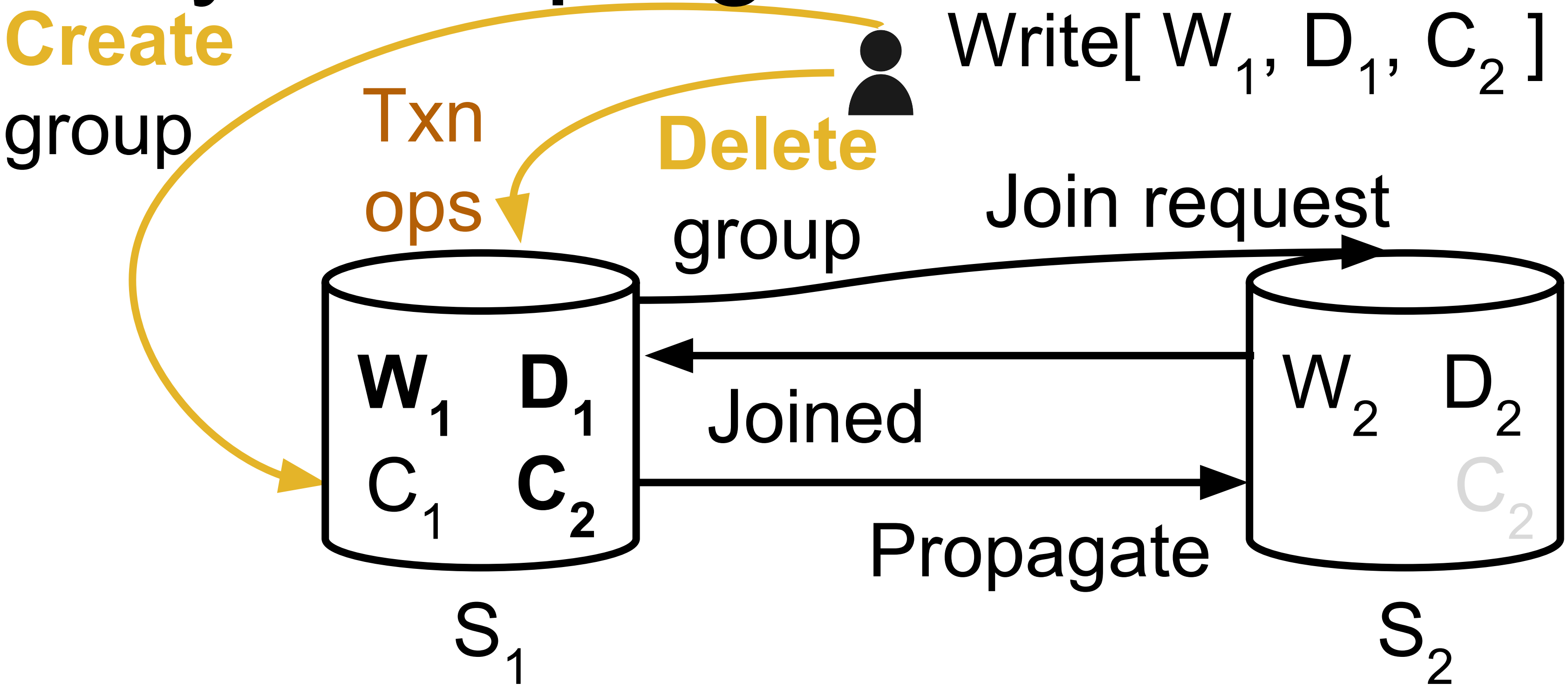
(Das et al., SoCC 2010)

Key Grouping



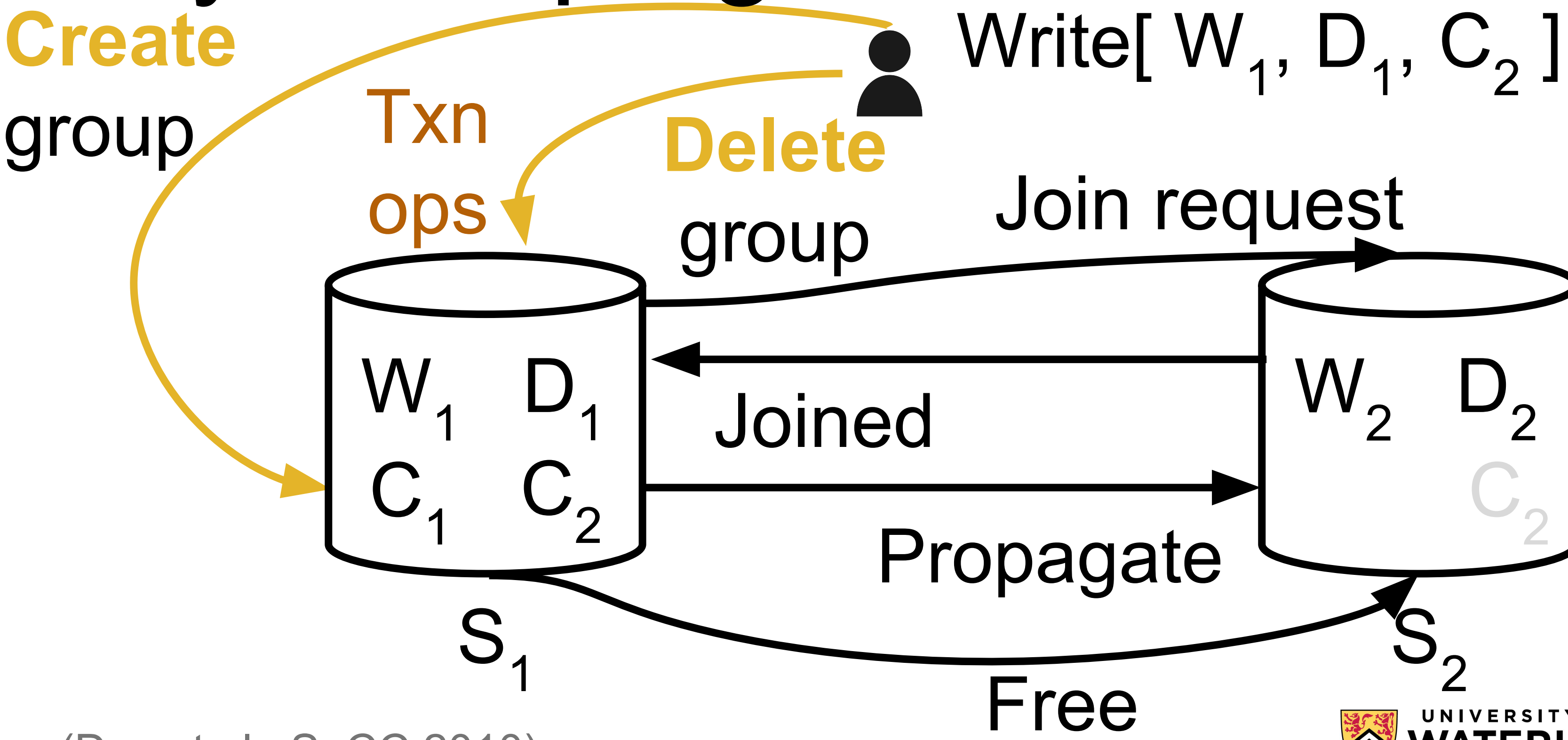
(Das et al., SoCC 2010)

Key Grouping



(Das et al., SoCC 2010)

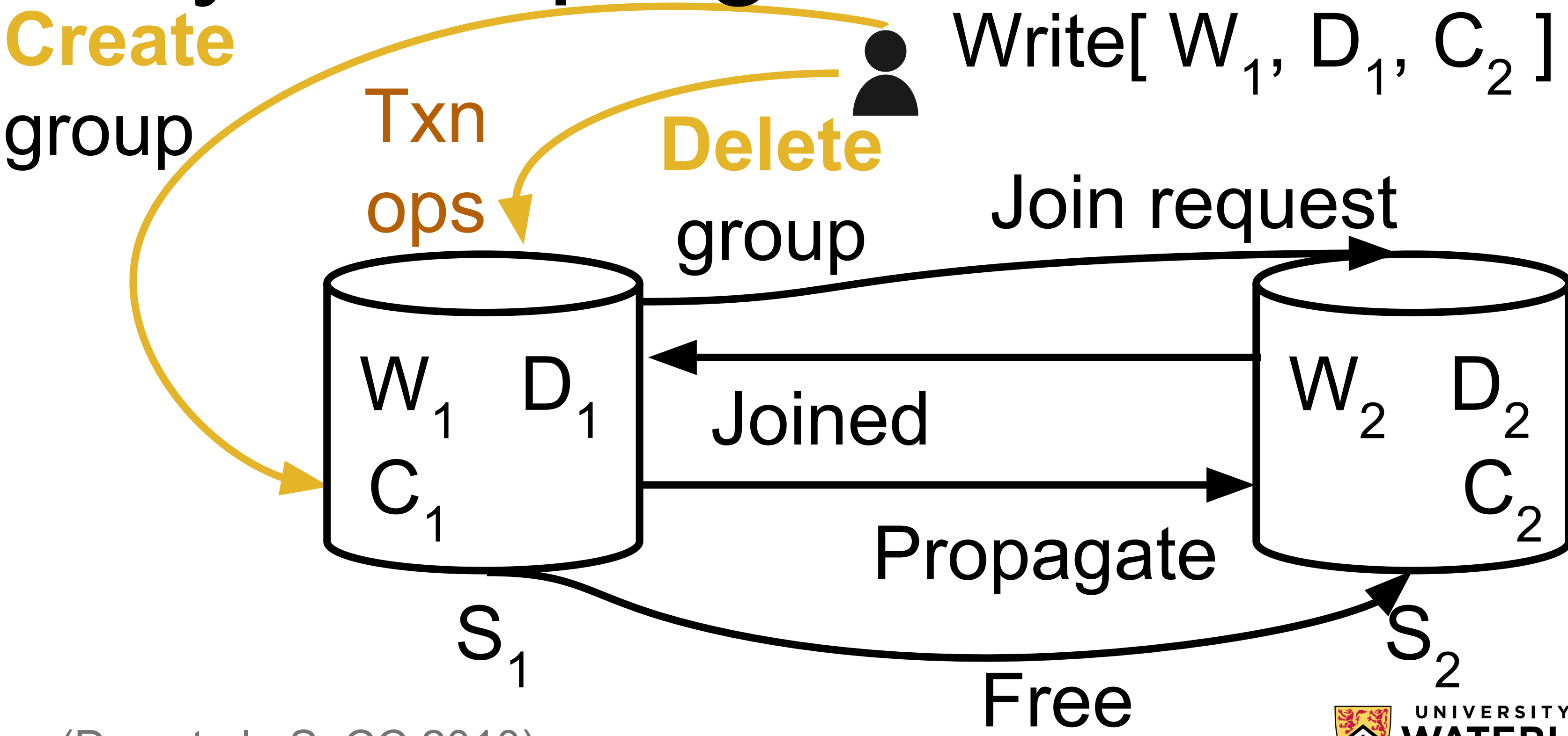
Key Grouping



(Das et al., SoCC 2010)

UNIVERSITY OF WATERLOO

Key Grouping



(Das et al., SoCC 2010)

Key Grouping

On demand transactional partitioning

Works **best** when groups are small and transactions contain **multiple operations**

But groups are **transient**

(Das et al., SoCC 2010)

Localizing Execution

Repartition data via **localization** for **single site** execution

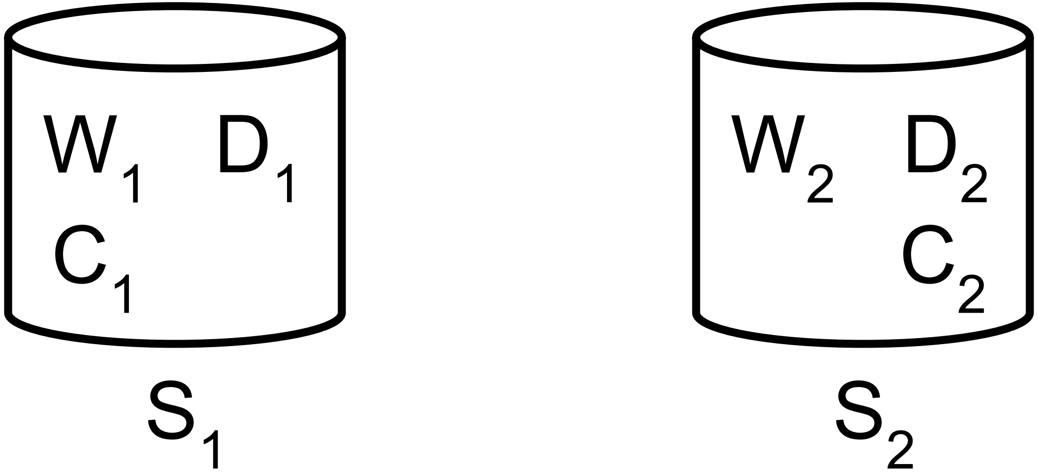
Dynamic partitioning based on **transaction patterns**

(Lin et al., SIGMOD 2016)

Localizing Execution

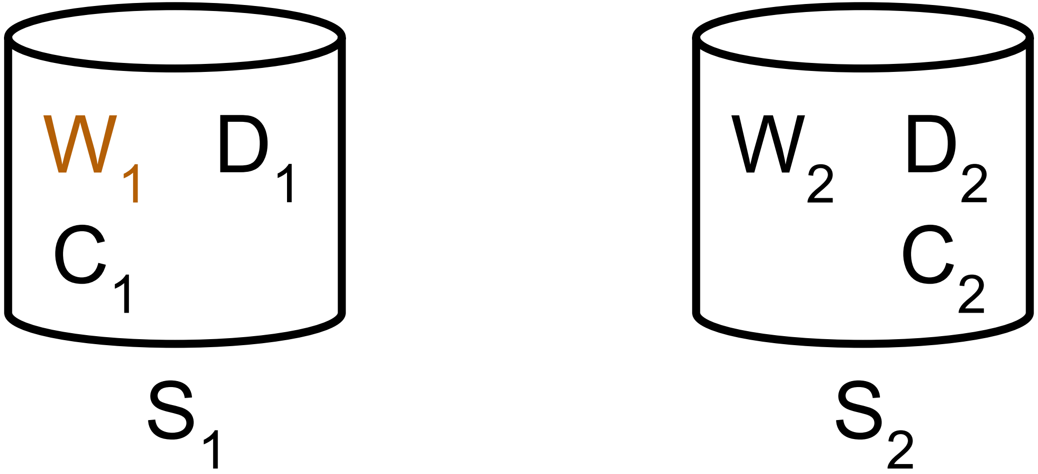
Ownership information

W_1	S_1
W_2	S_2
D_1	S_1
D_2	S_2
C_1	S_1
C_2	S_2



(Lin et al., SIGMOD 2016)

Localizing Execution

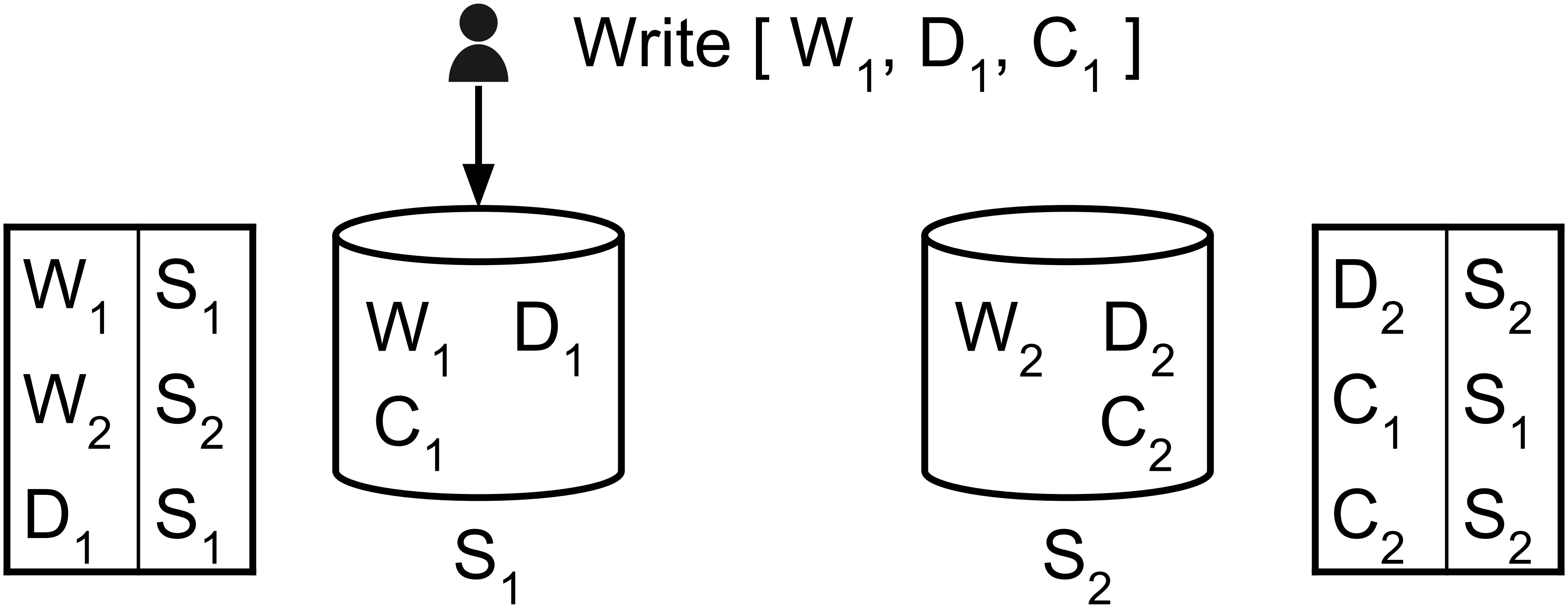


Ownership information

W_1	S_1
W_2	S_2
D_1	S_1
D_2	S_2
C_1	S_1
C_2	S_2

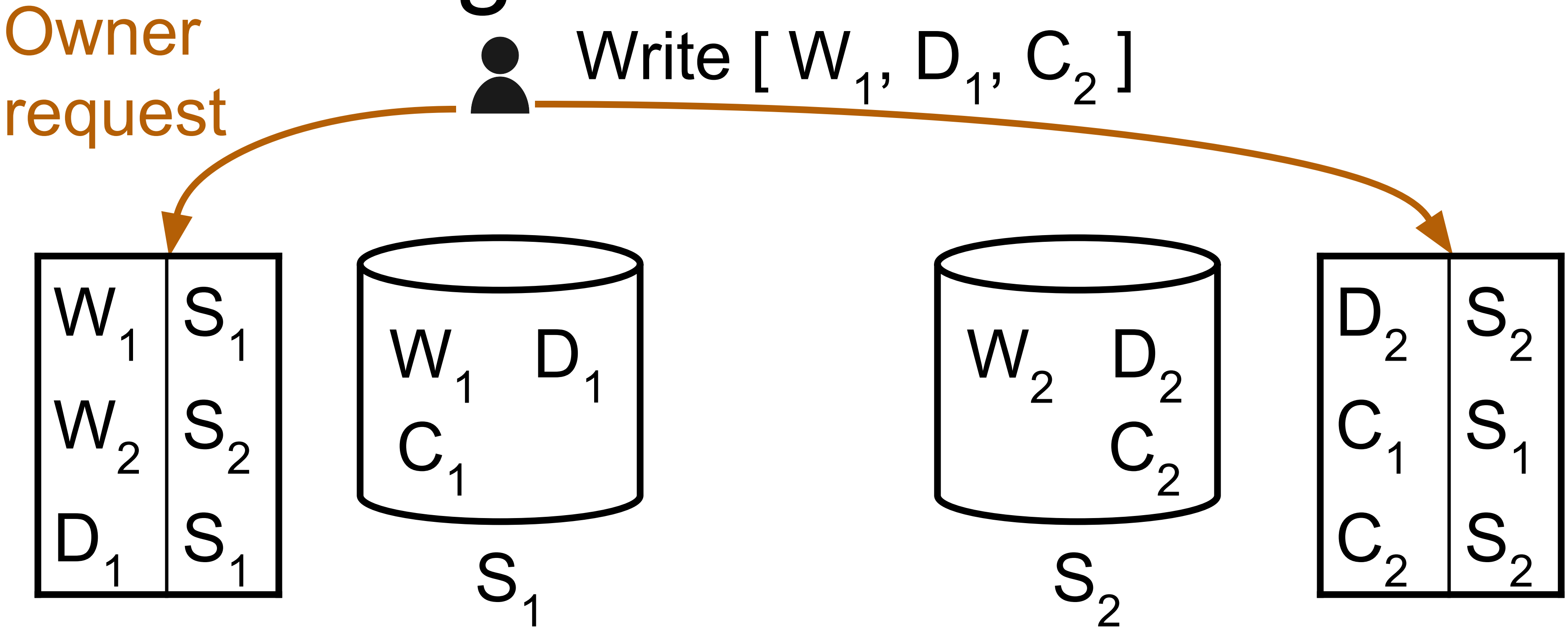
(Lin et al., SIGMOD 2016)

Localizing Execution



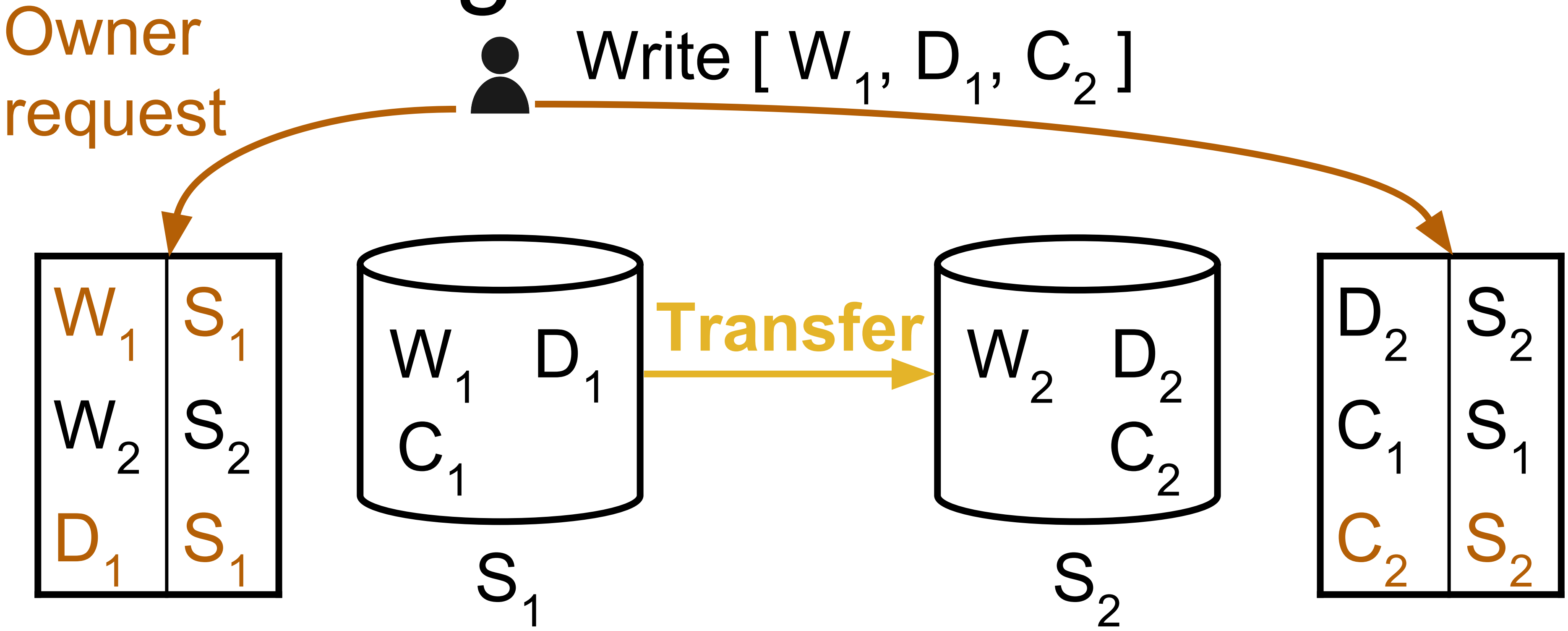
(Lin et al., SIGMOD 2016)

Localizing Execution



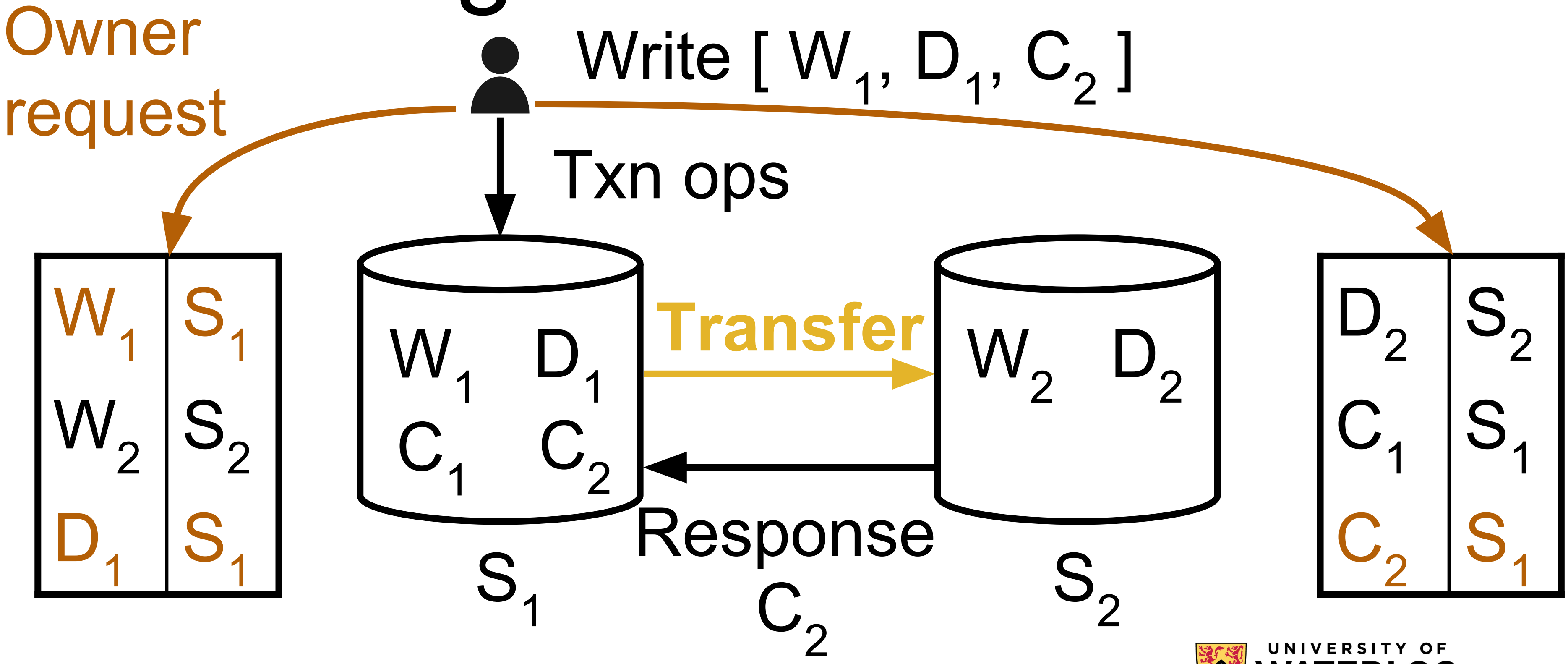
(Lin et al., SIGMOD 2016)

Localizing Execution



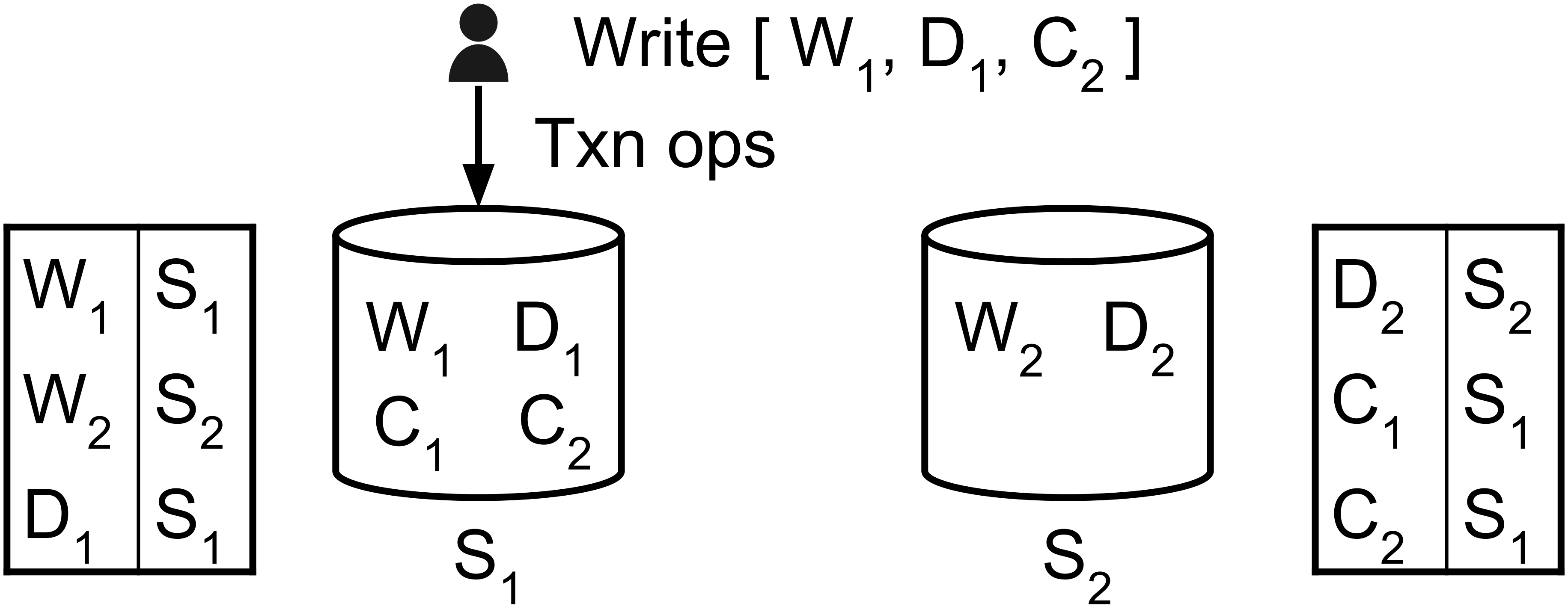
(Lin et al., SIGMOD 2016)

Localizing Execution



(Lin et al., SIGMOD 2016)

Localizing Execution



(Lin et al., SIGMOD 2016)

Localizing Execution

Dynamic partitioning based on **per transaction patterns**

Does **not** consider **workload overall**

(Lin et al., SIGMOD 2016)

Partitioning Decisions

- **How to form partitions?**
  Transaction localization
- **Where to place partitions?**
  At requester
- **How to execute multi-partition operations?**
  L-Store protocol

(Lin et al., SIGMOD 2016)

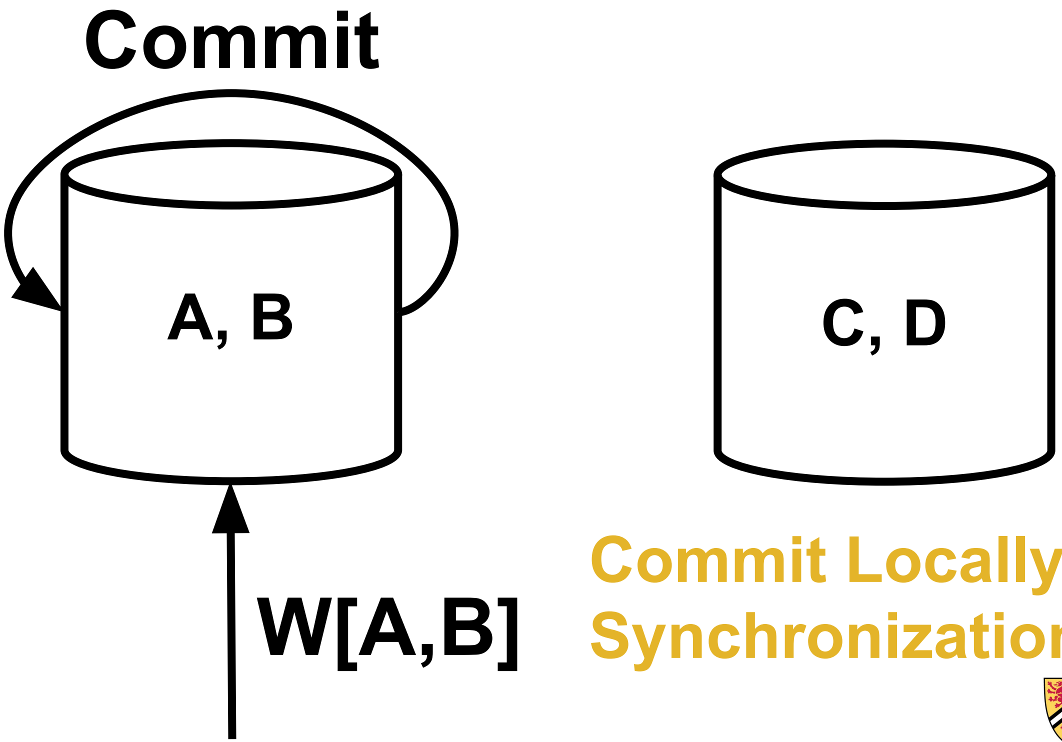
UNIVERSITY OF WATERLOO

Partitioning Decisions

- **How to form partitions?**
  Key groups, temporarily
- **Where to place partitions?**
  Key group leader
- **How to execute multi-partition operations?**
  Key group protocol

(Das et al., SoCC 2010)

Localizing Transactions

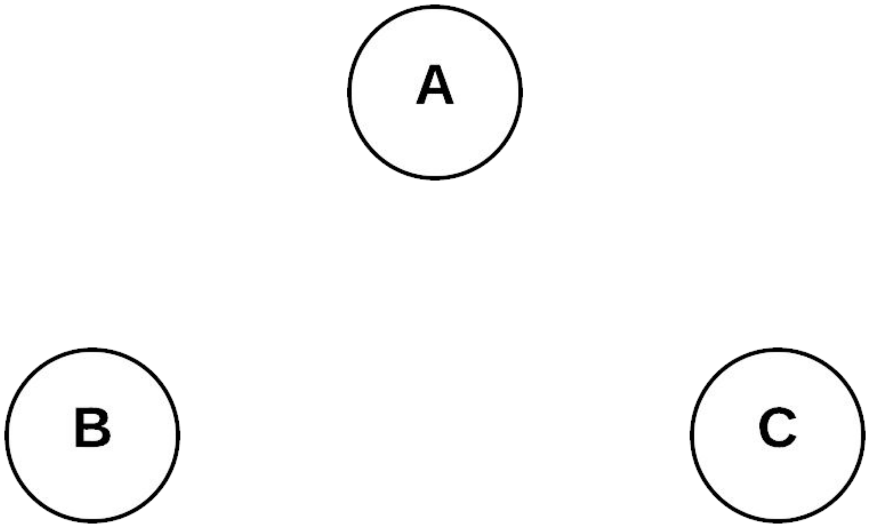


Commit Locally Without Synchronization!

Constructing the Graph

From a workload trace

ID	Name
A	Alice
B	Bob
C	Carol

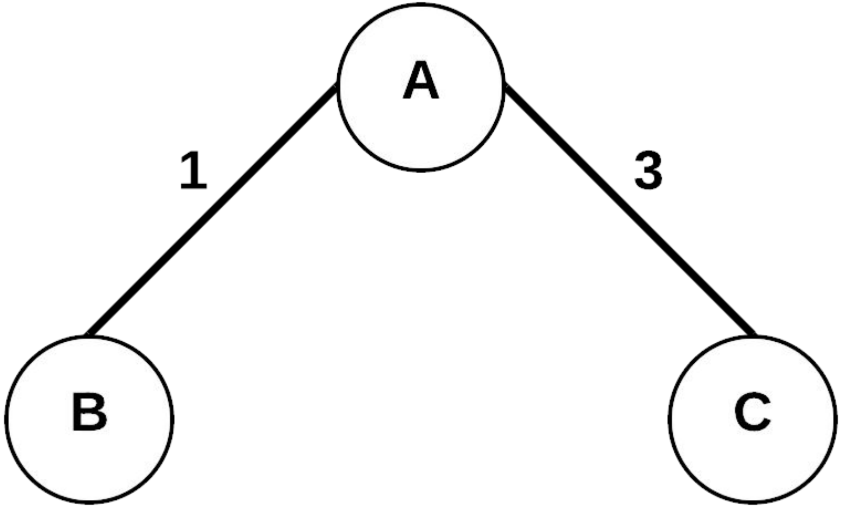


(Curino et al., VLDB 2010)

Constructing the Graph

Add traced transactions: R[A,B], 3x W[A,C]

ID	Name
A	Alice
B	Bob
C	Carol

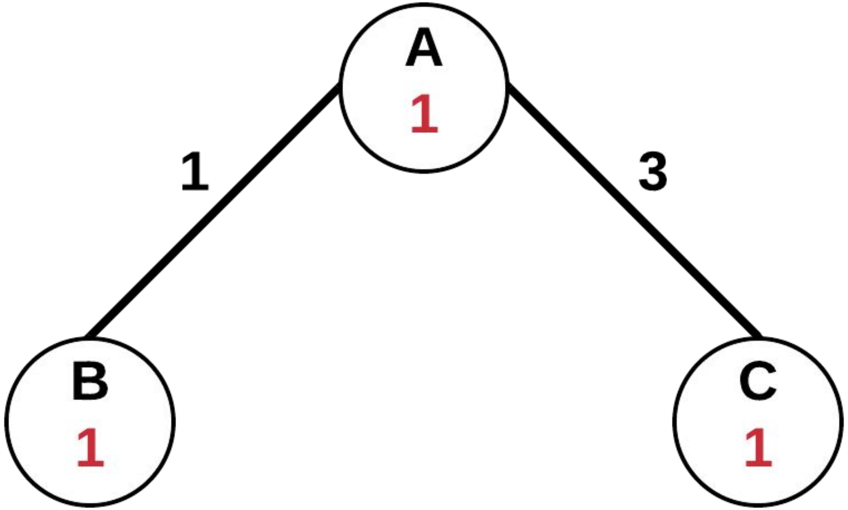


(Curino et al., VLDB 2010)

Constructing the Graph

Add node weights (size, load)

ID	Name
A	Alice
B	Bob
C	Carol

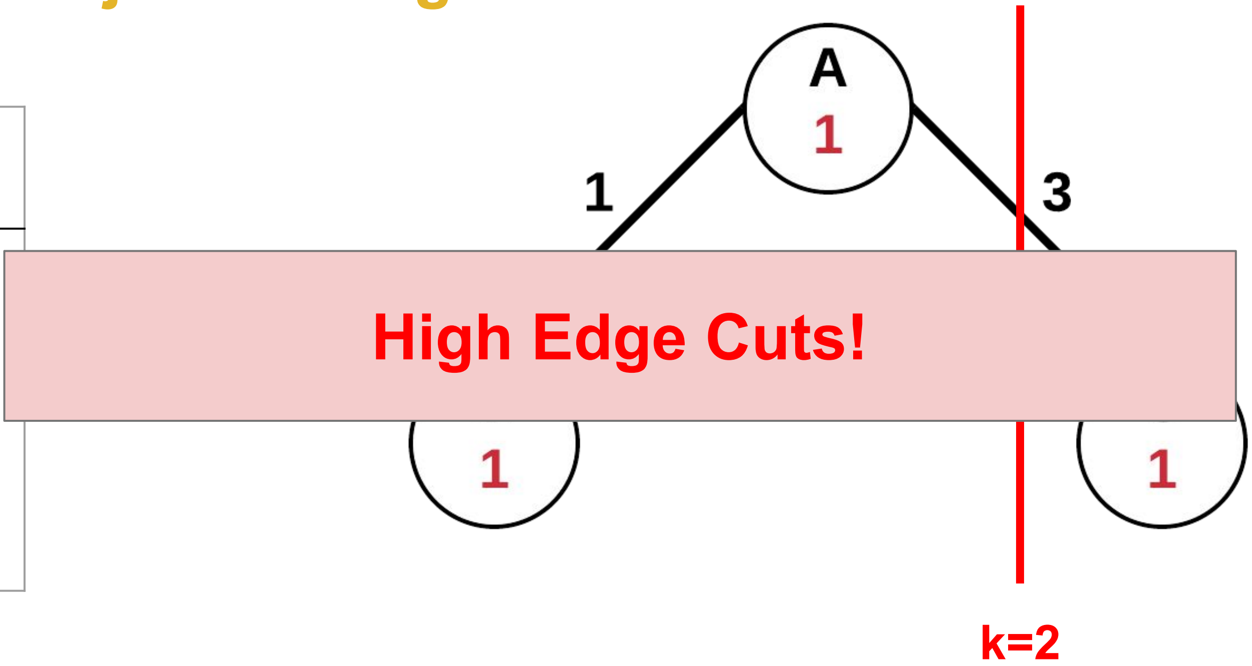


(Curino et al., VLDB 2010)

Constructing the Graph

Min-cut edges subject to weight imbalance

ID	Name
A	Alice
B	Bob
C	Carol

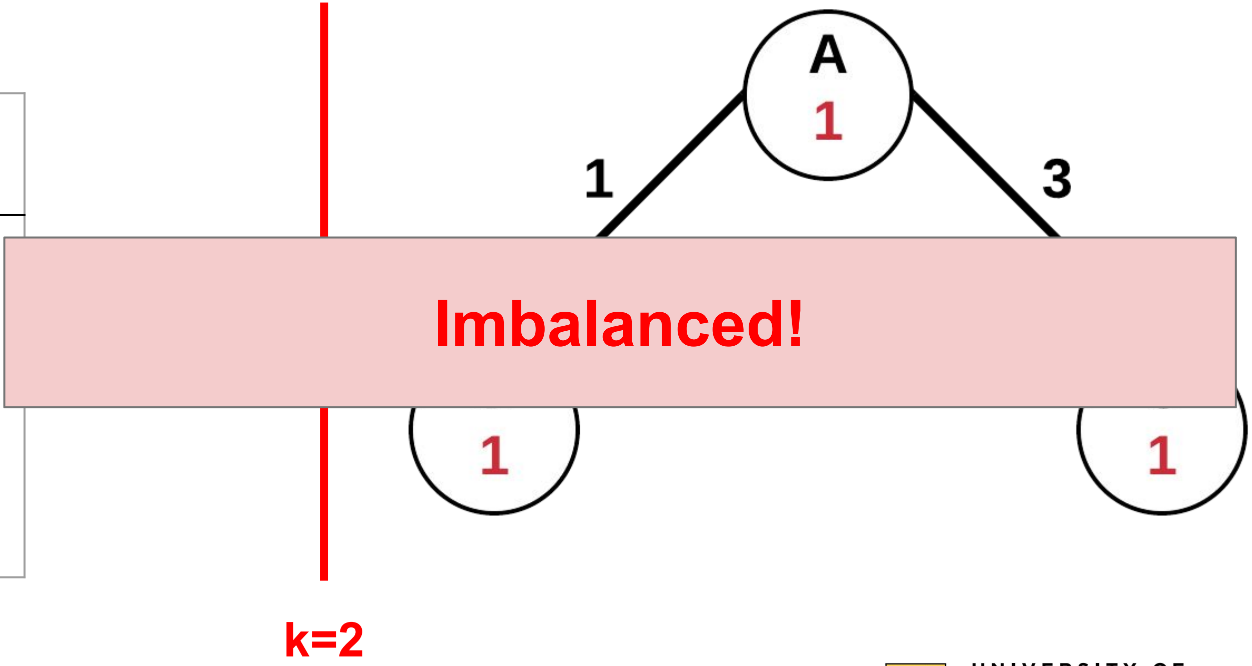


(Curino et al., VLDB 2010)

Constructing the Graph

Min-cut edges subject to weight imbalance

ID	Name
A	Alice
B	Bob
C	Carol

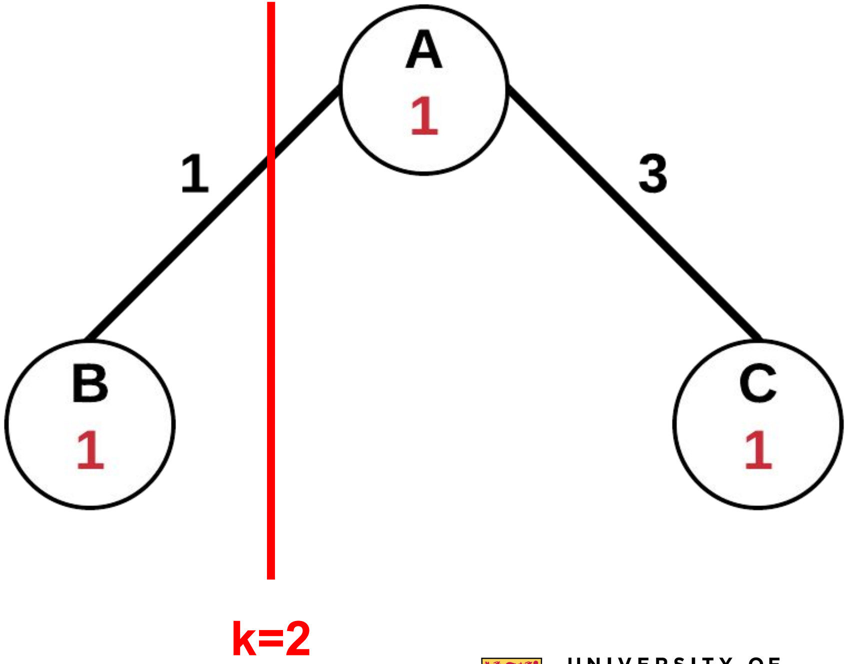


(Curino et al., VLDB 2010)

Constructing the Graph

Min-cut edges subject to weight imbalance

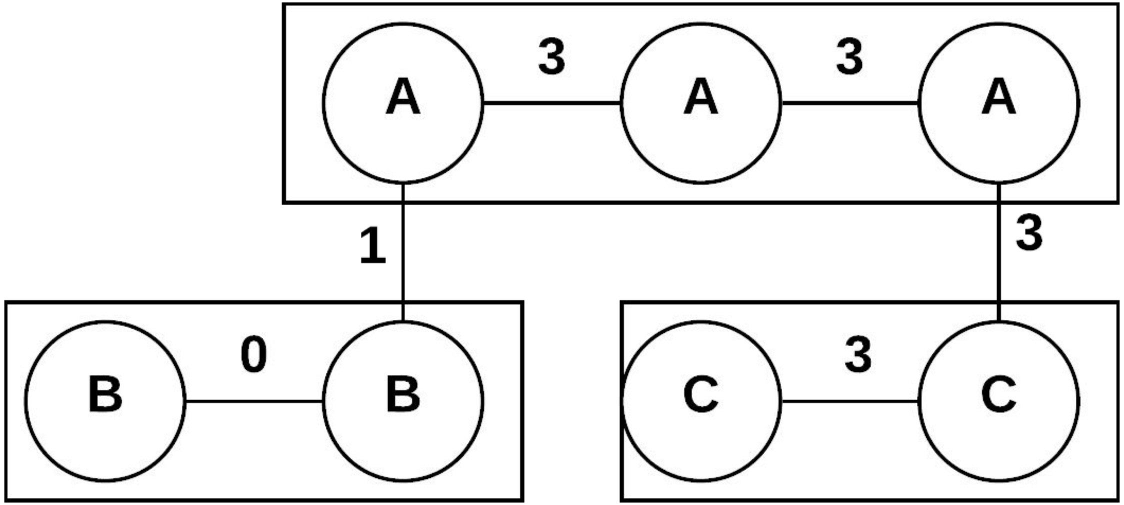
ID	Name
A	Alice
B	Bob
C	Carol



(Curino et al., VLDB 2010)

Adding Replica Support

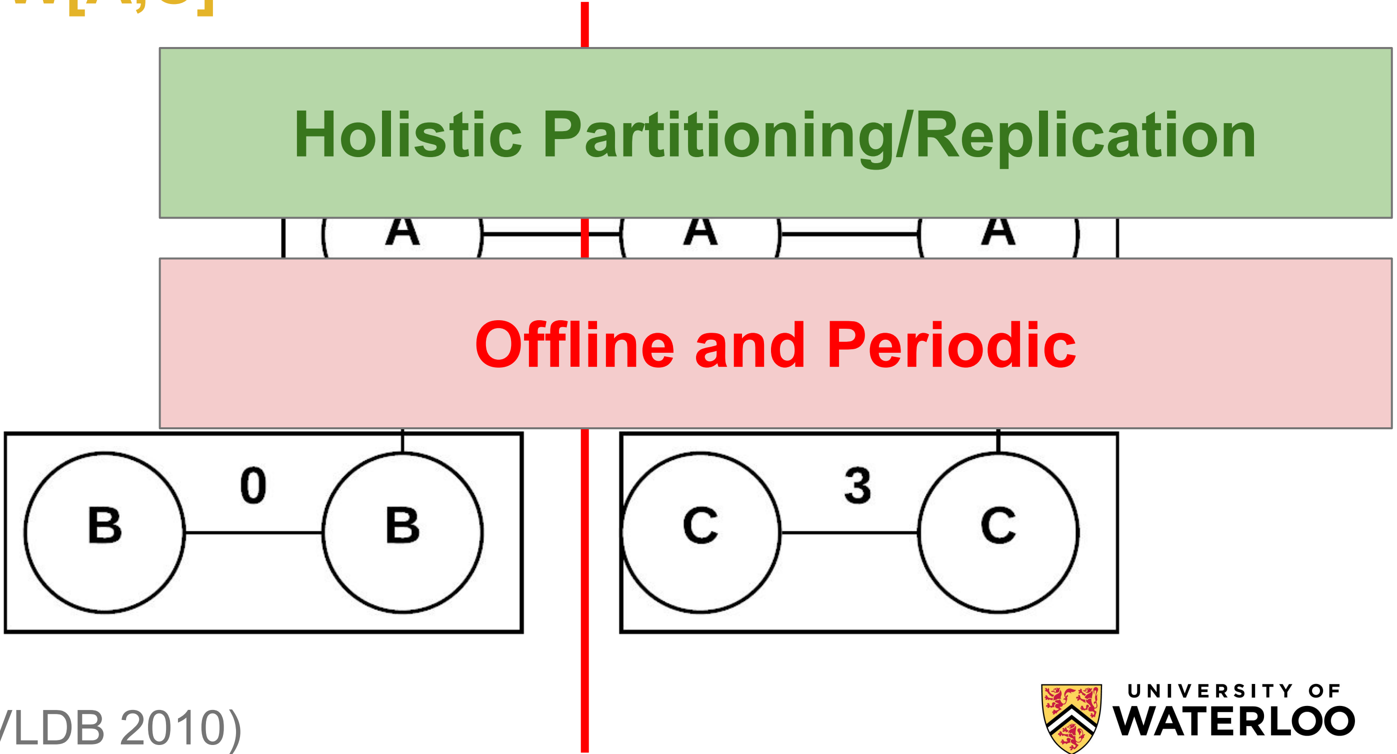
R[A,B] 3x W[A,C]



(Curino et al., VLDB 2010)

Adding Replica Support

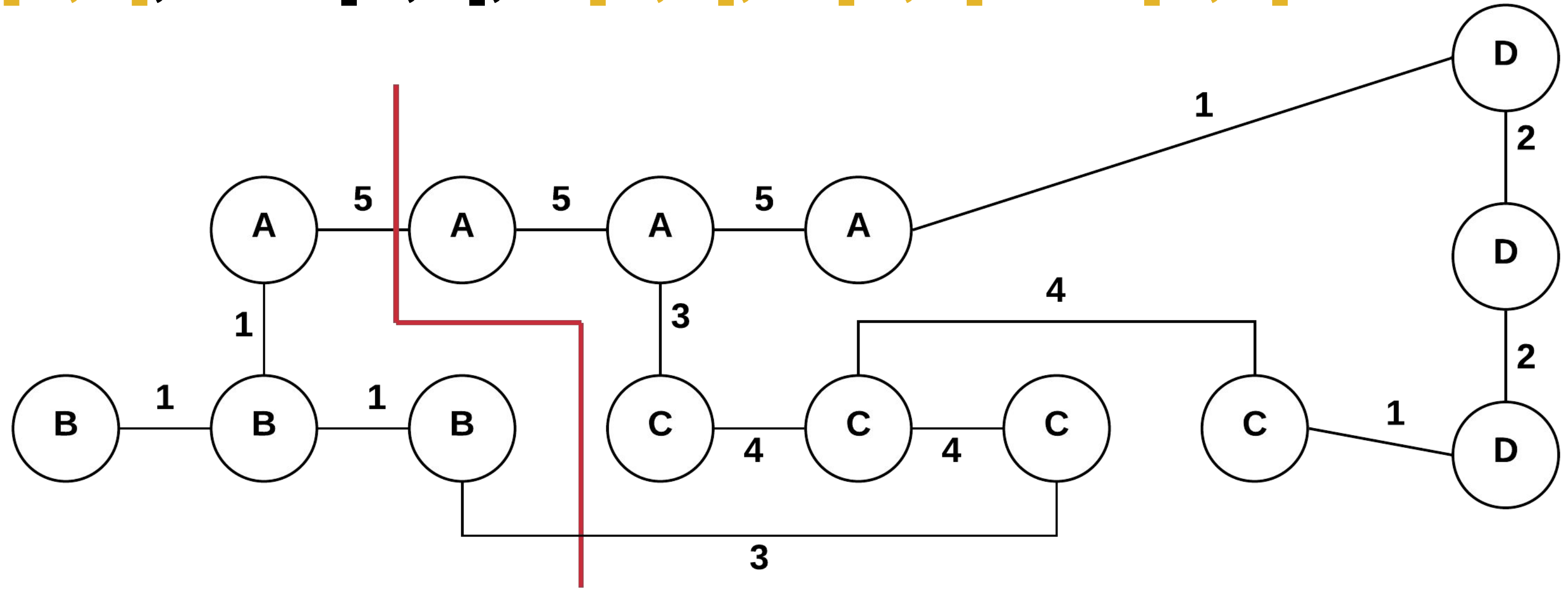
R[A,B] 3x W[A,C]



(Curino et al., VLDB 2010)

Access Patterns Change!

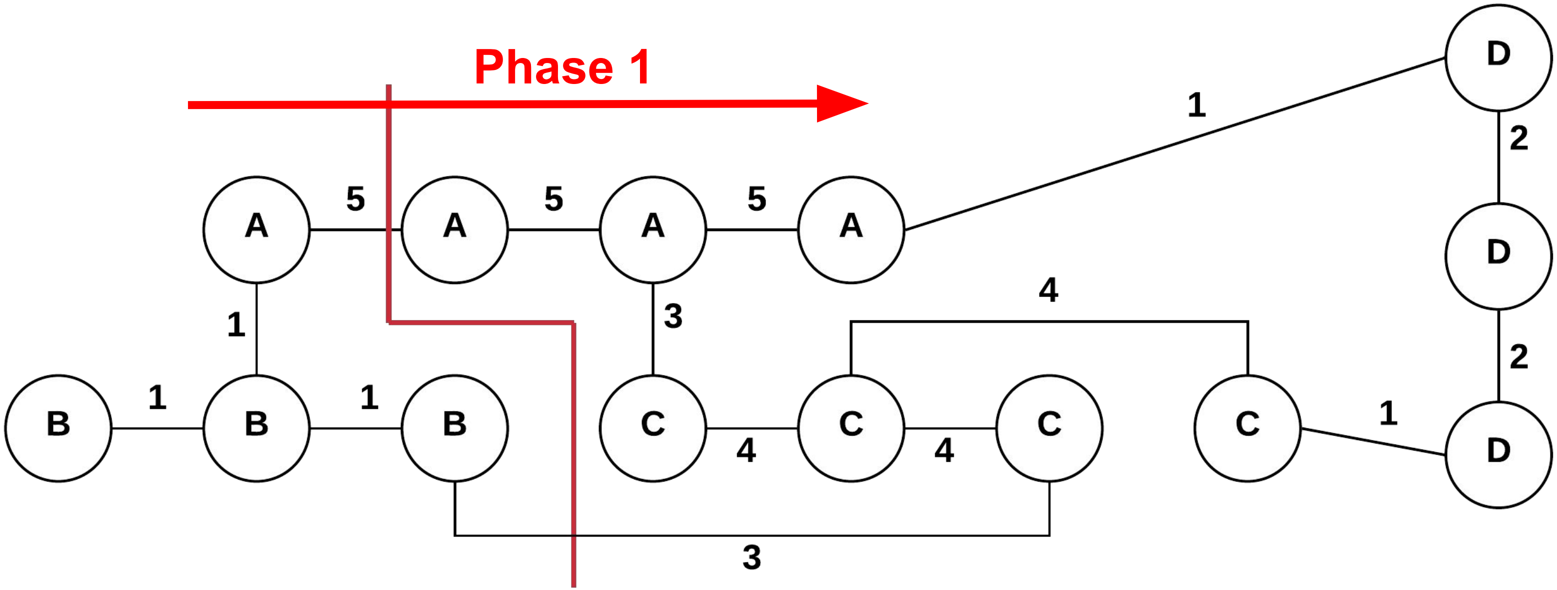
W[A,B], 3x W[A,C], W[A,D], W[C,D] 3x R[B,C]



W(P1)=4, W(P2)=10, EC=8

(Nicoara et al., EDBT 2015)

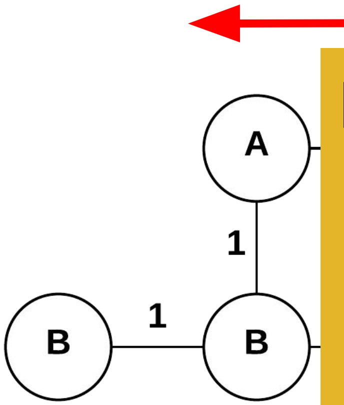
Two Phases



(Nicoara et al., EDBT 2015)

Two Phases

Logical Movement, then Migrate



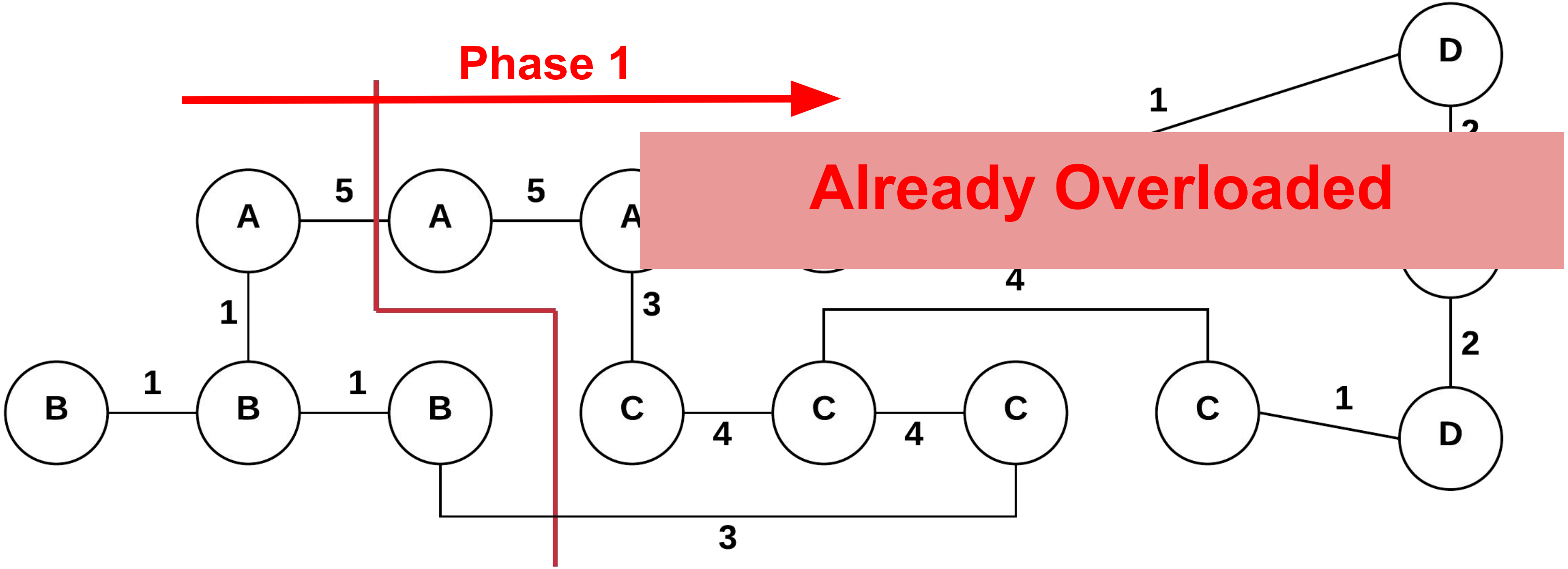
Rule:

- **Movement doesn't overload**
- **Move best-gain candidates**
- **If overloaded, must move!**

(Nicoara et al., EDBT 2015)

UNIVERSITY OF WATERLOO

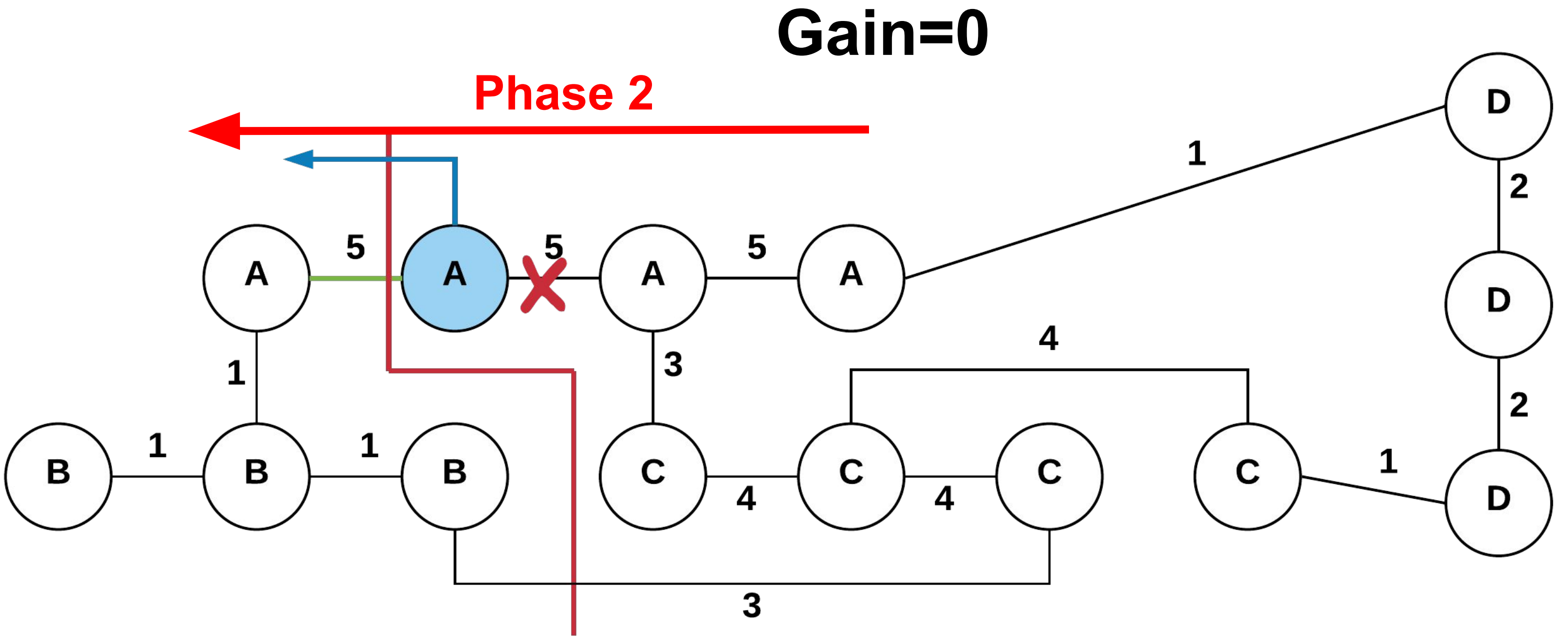
Two Phases



W(P1)=4, W(P2)=10, EC=8, Bounds: (6,8)

(Nicoara et al., EDBT 2015)

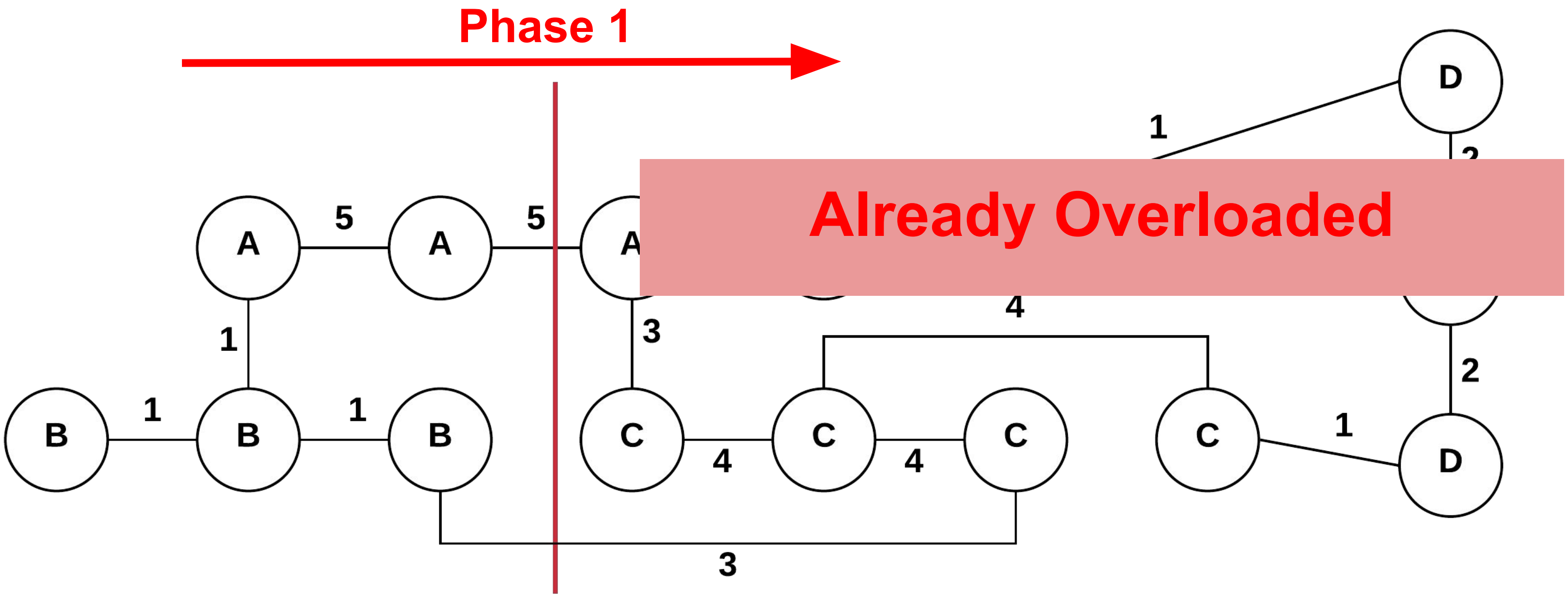
Two Phases



W(P1)=4, W(P2)=10, EC=8, Bounds: (6,8)

(Nicoara et al., EDBT 2015)

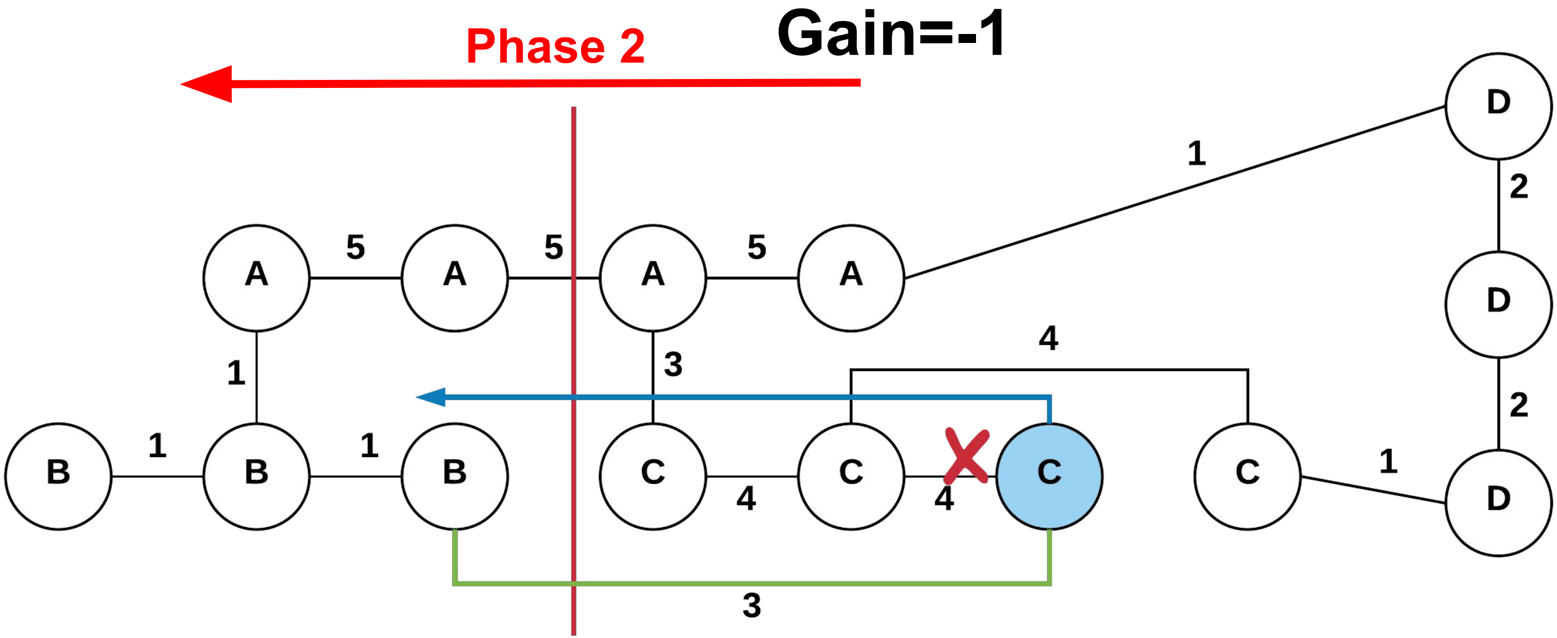
Two Phases



W(P1)=5, W(P2)=9, EC=8, Bounds: (6,8)

(Nicoara et al., EDBT 2015)

Two Phases

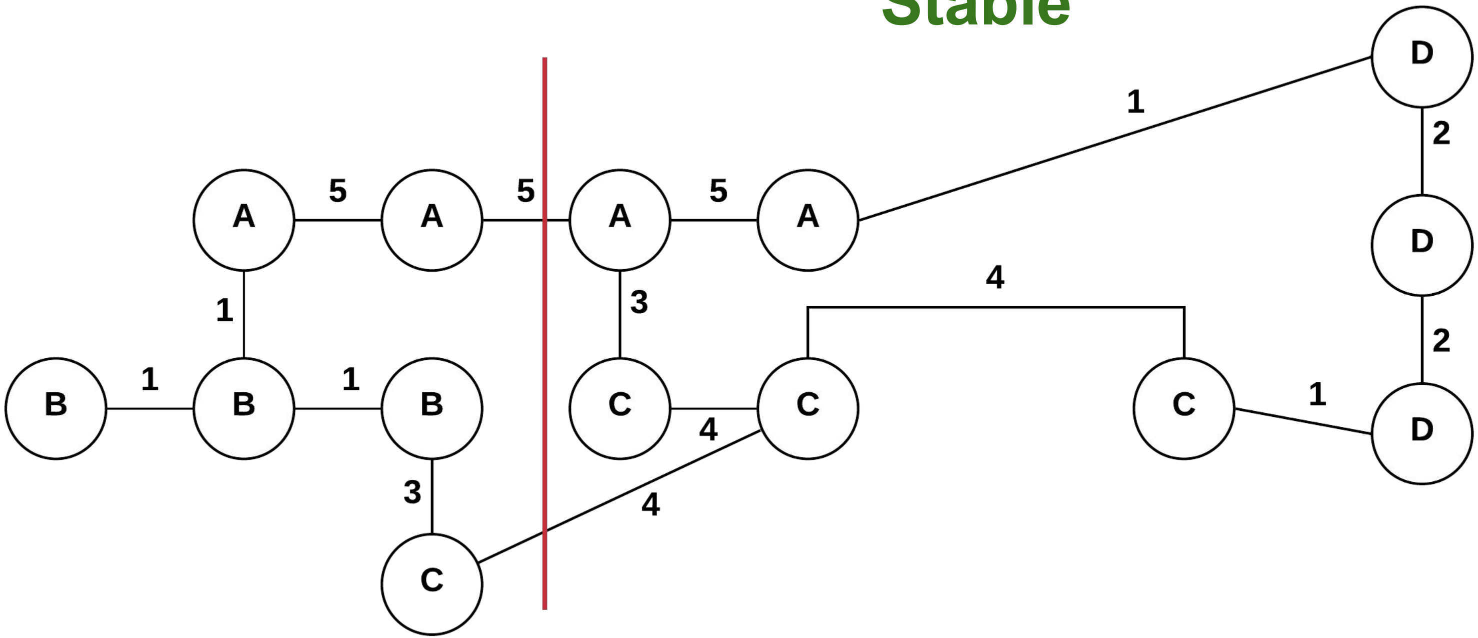


W(P1)=5, W(P2)=9, EC=8, Bounds: (6,8)

(Nicoara et al., EDBT 2015)

Convergence

Stable



W(P1)=6, W(P2)=8, EC=9, Bounds: (6,8)

(Nicoara et al., EDBT 2015)

Partitioning Decisions

- **How to form partitions?**

  Graph partitioning

- **Where to place partitions?**

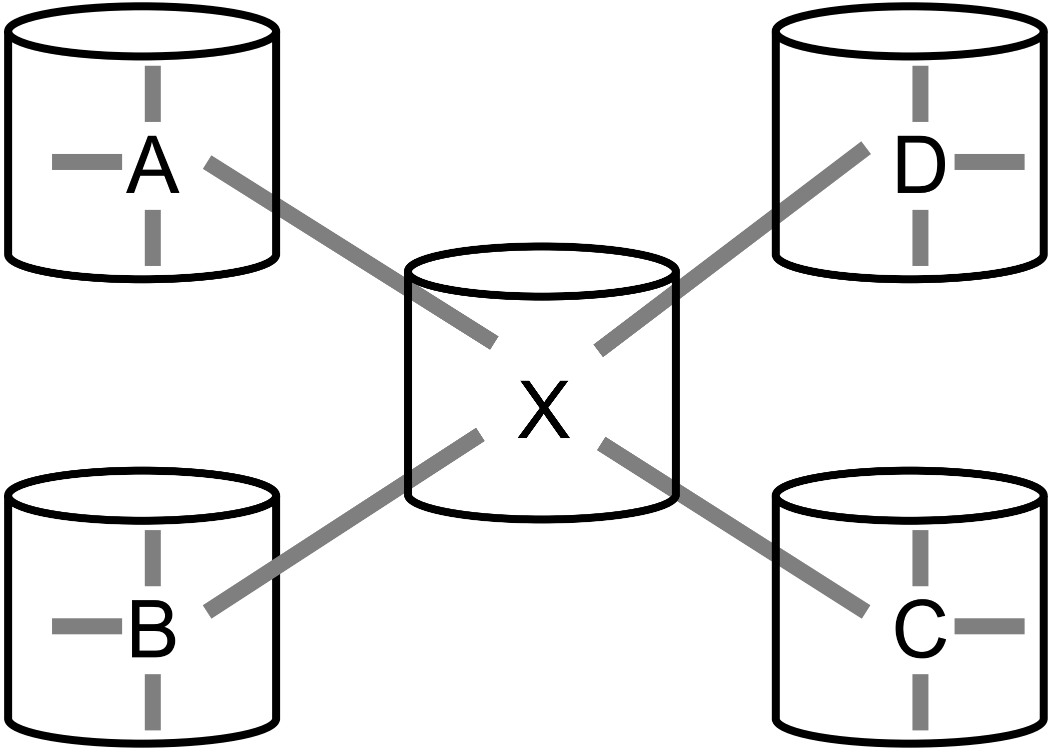
  Based on partitioning

- **How to execute multi-partition operations?**

  2PC

(Curino et al., VLDB 2010) (Nicoara et al., EDBT 2015)

Graph Partitioning



Minimizes **total** number of distributed transactions

Ignores **per node** involvement

Adaptive Database Partitioning

Balance load and minimize distributed transactions

Database elasticity

E-Store

(Taft et al., VLDB 2014)

Clay

(Serafini et al., VLDB 2015)

P-Store

(Taft et al., SIGMOD 2018)

Considering Distributed Cost

General Graph Partitioning: minimize # edge cuts

such that: $load(S_i) < (1 + \varepsilon)$ **avg load(S)**

load balanced

General: $load(S_i) = \sum w(v)$ (v at S_i)

Clay: $load(S_i) = \sum w(v) +$ **$k \sum w(uv)$** (v at S_i) (u not at S_i)

Distributed cost

(Serafini et al., VLDB 2016)

UNIVERSITY OF WATERLOO

Repartitioning Cost

General Graph Partitioning: minimize # edge cuts

Clay: minimize # edge cuts

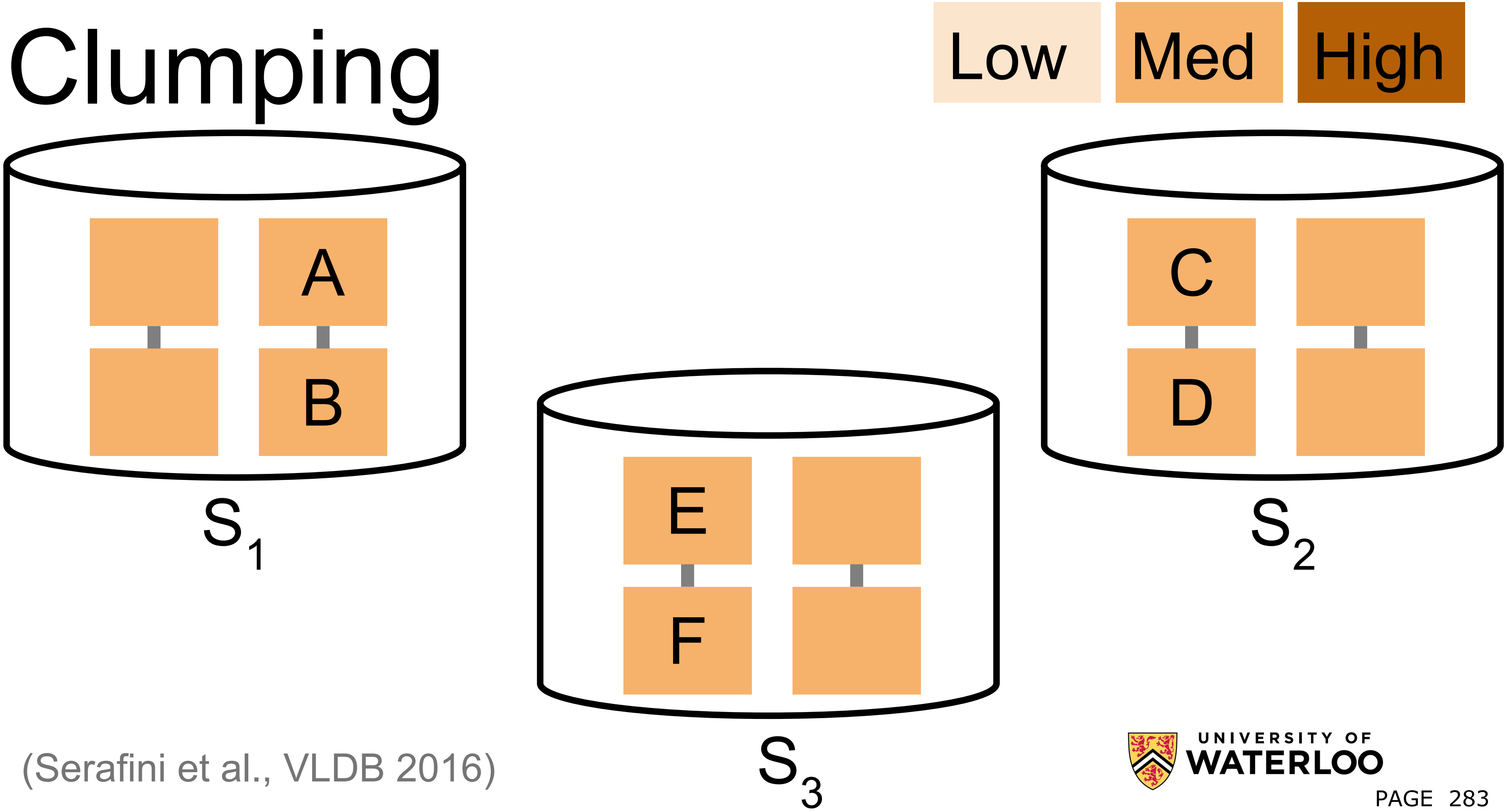
and

# of vertices mapped to new partitions

cost of repartitioning

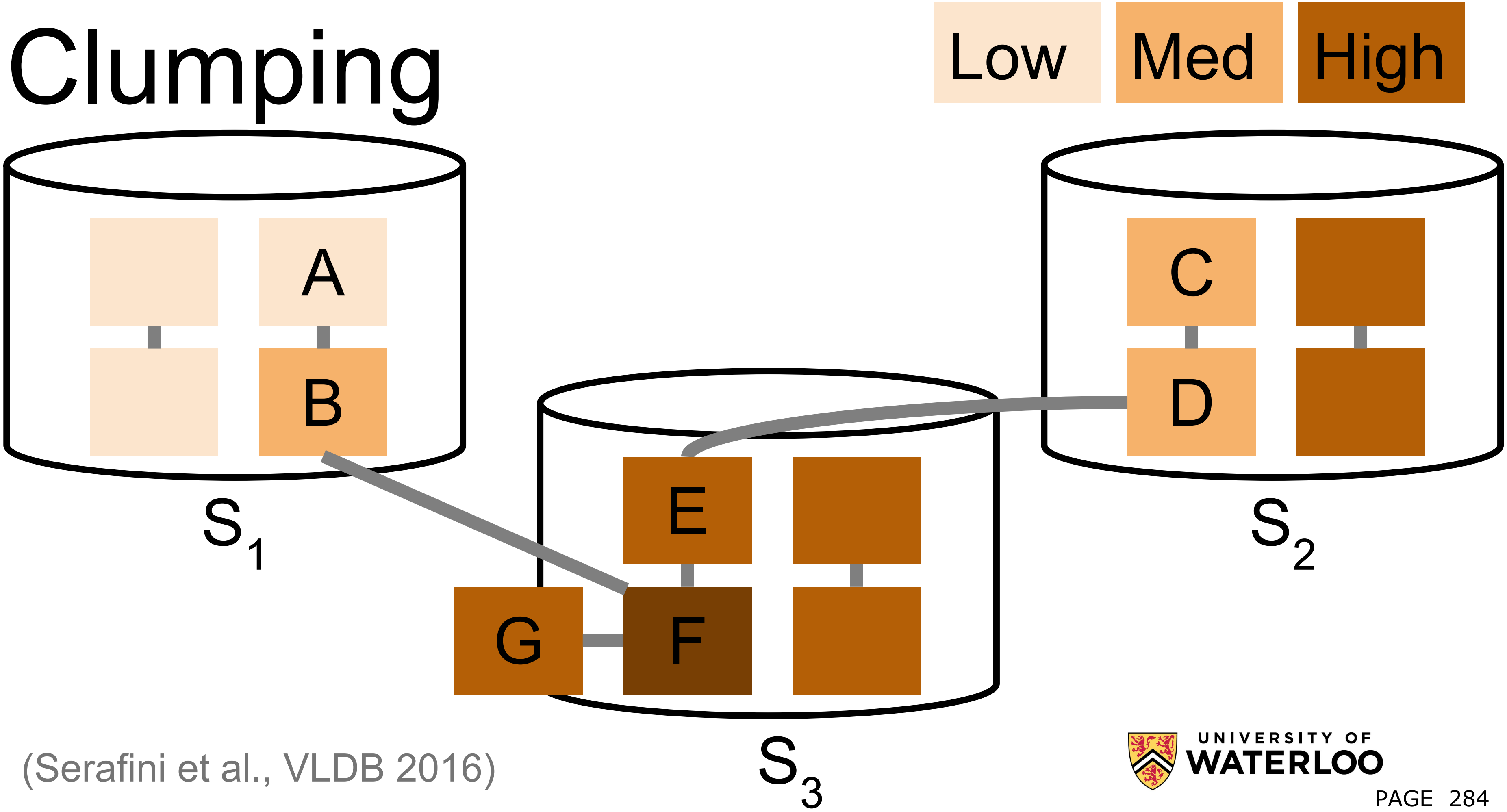
(Serafini et al., VLDB 2016)

Clumping



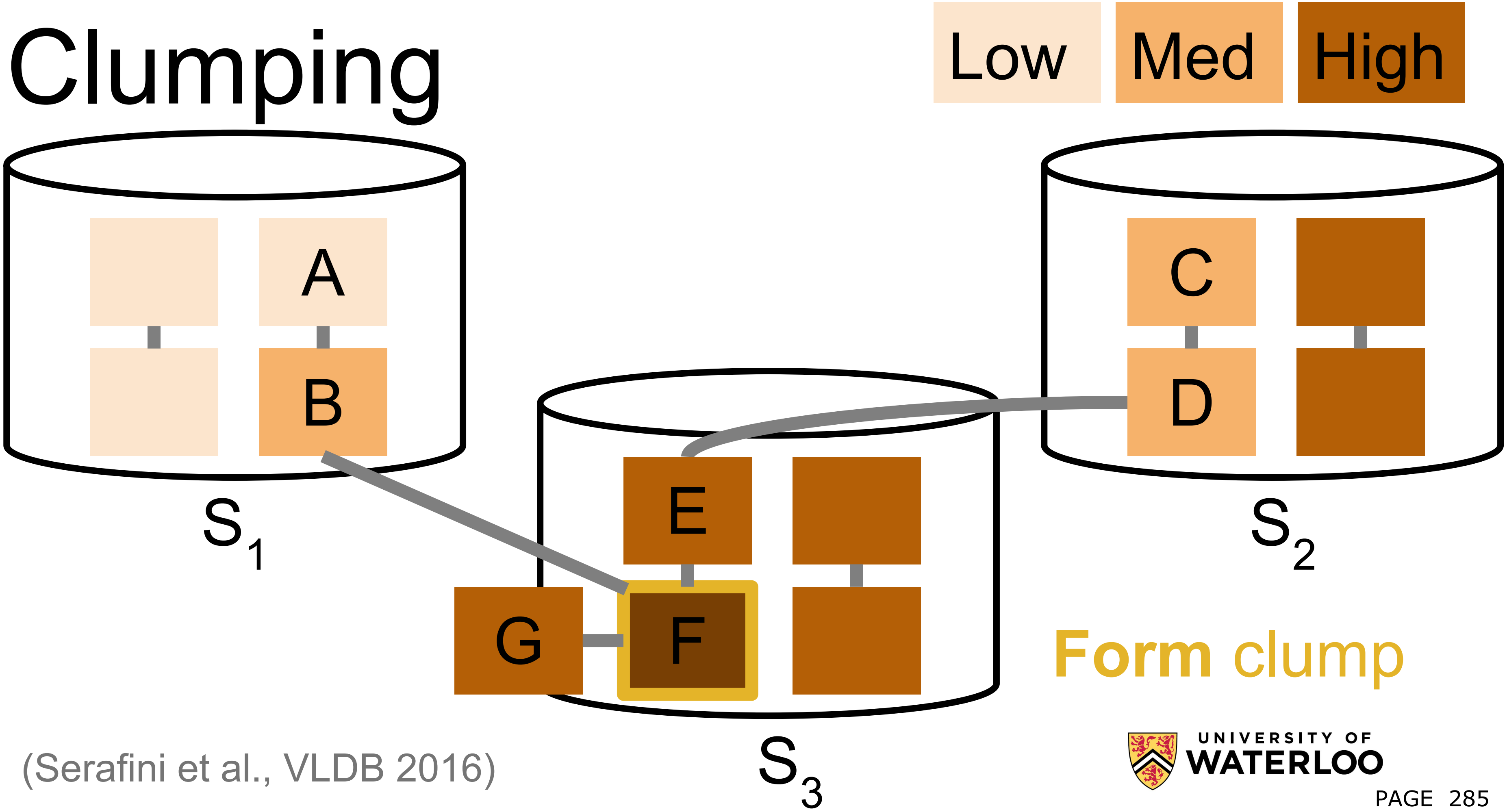
(Serafini et al., VLDB 2016)

Clumping



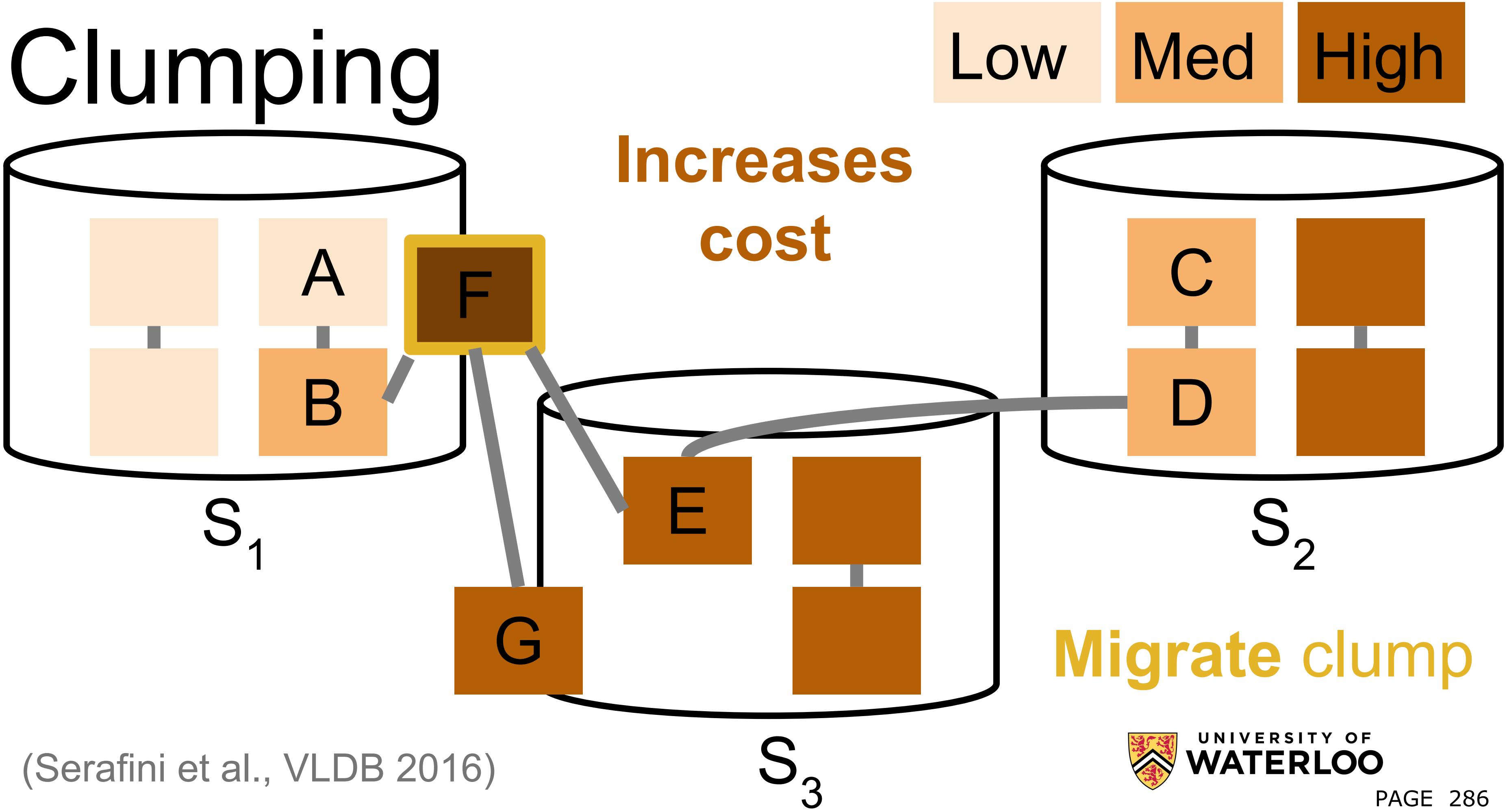
(Serafini et al., VLDB 2016)

Clumping



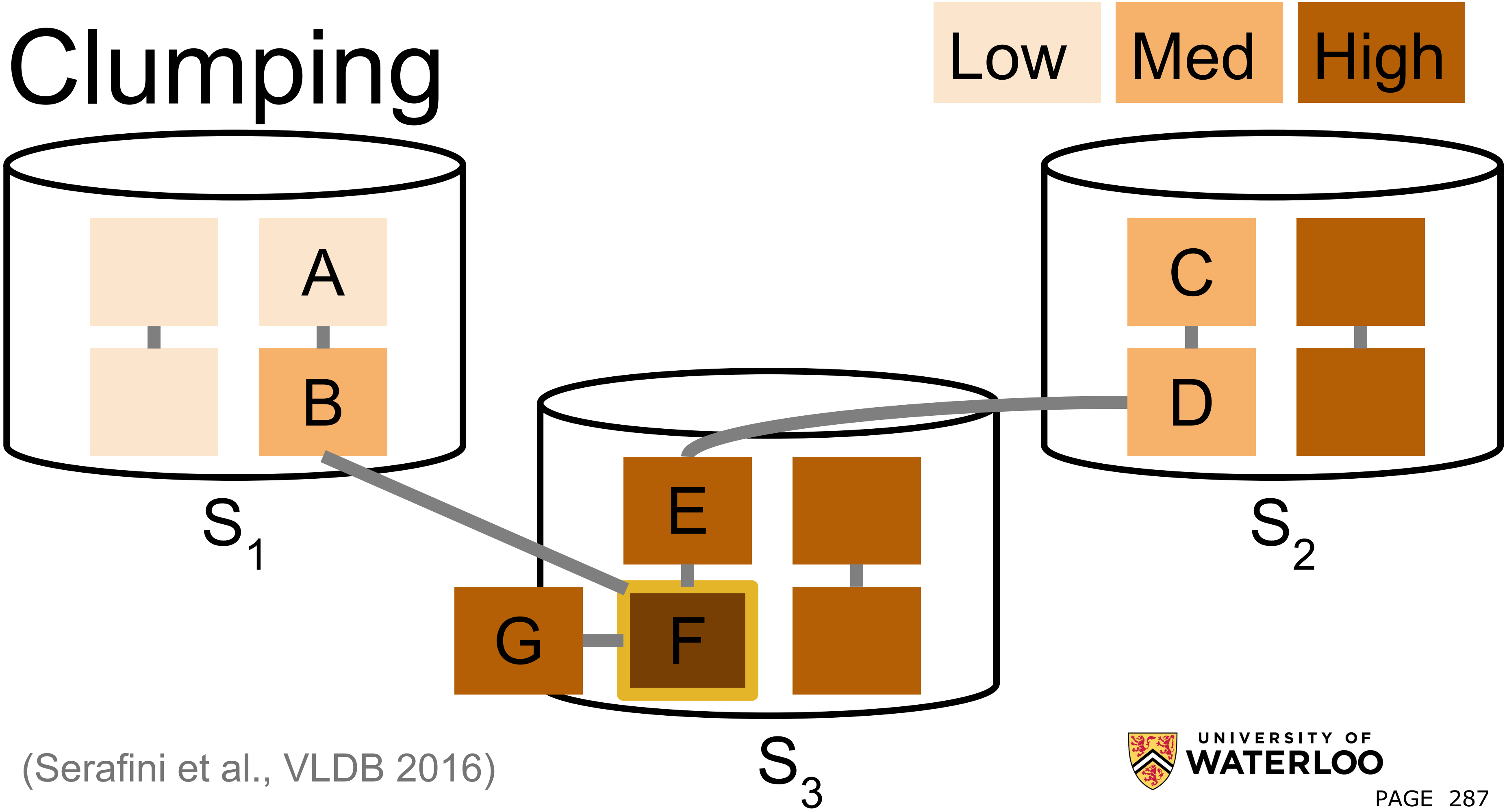
(Serafini et al., VLDB 2016)

Clumping



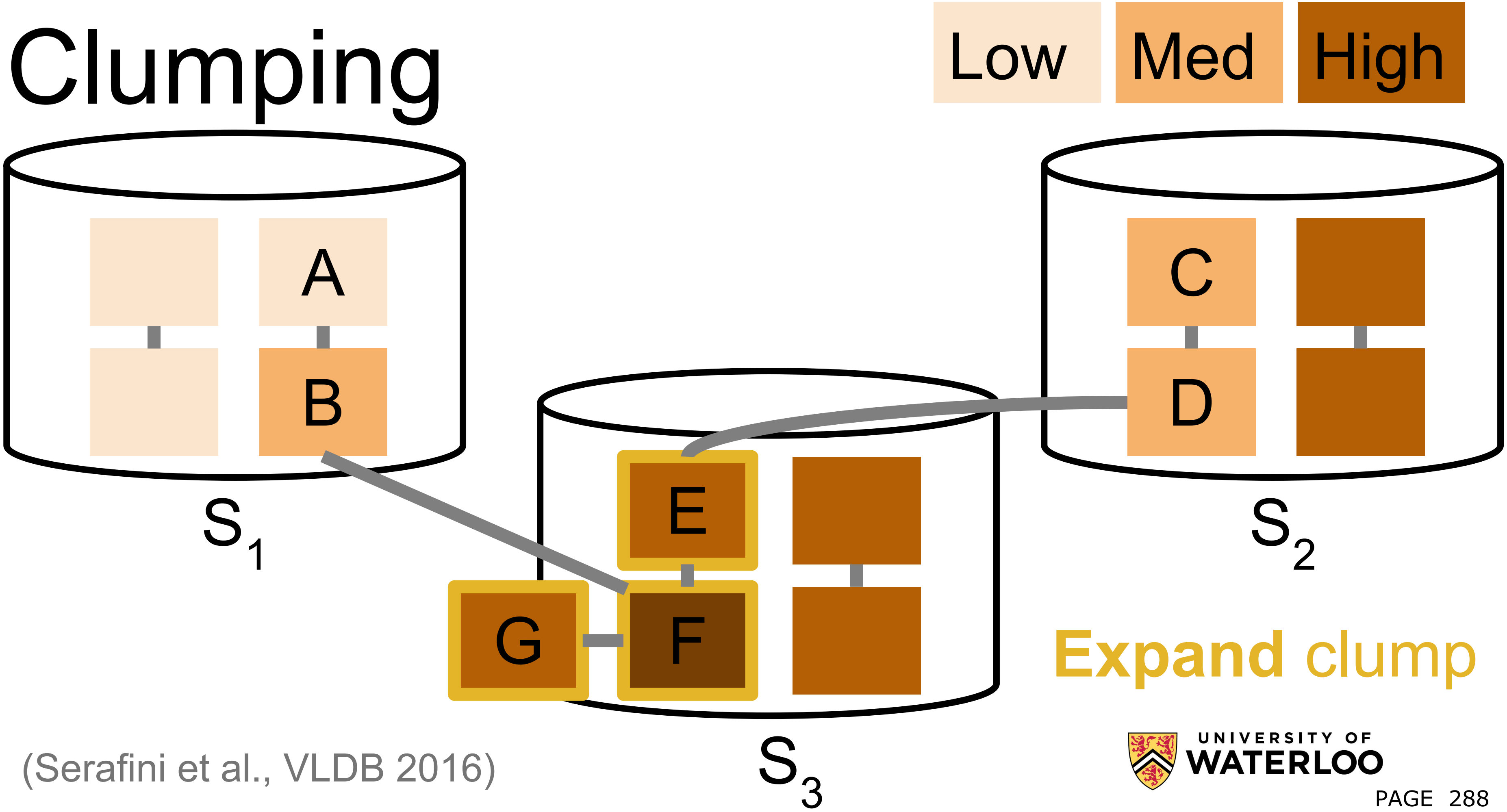
(Serafini et al., VLDB 2016)

Clumping



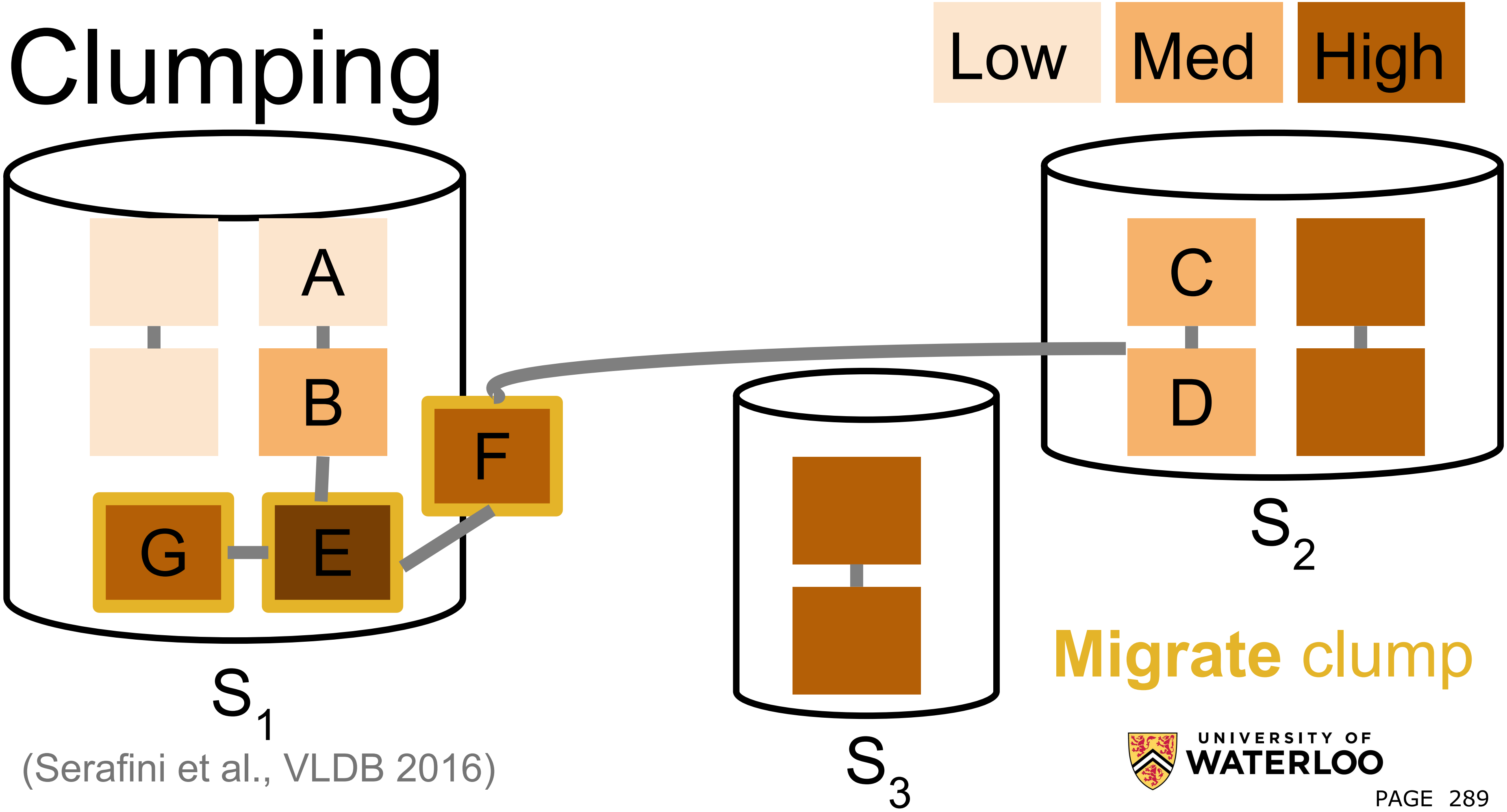
(Serafini et al., VLDB 2016)

Clumping



(Serafini et al., VLDB 2016)

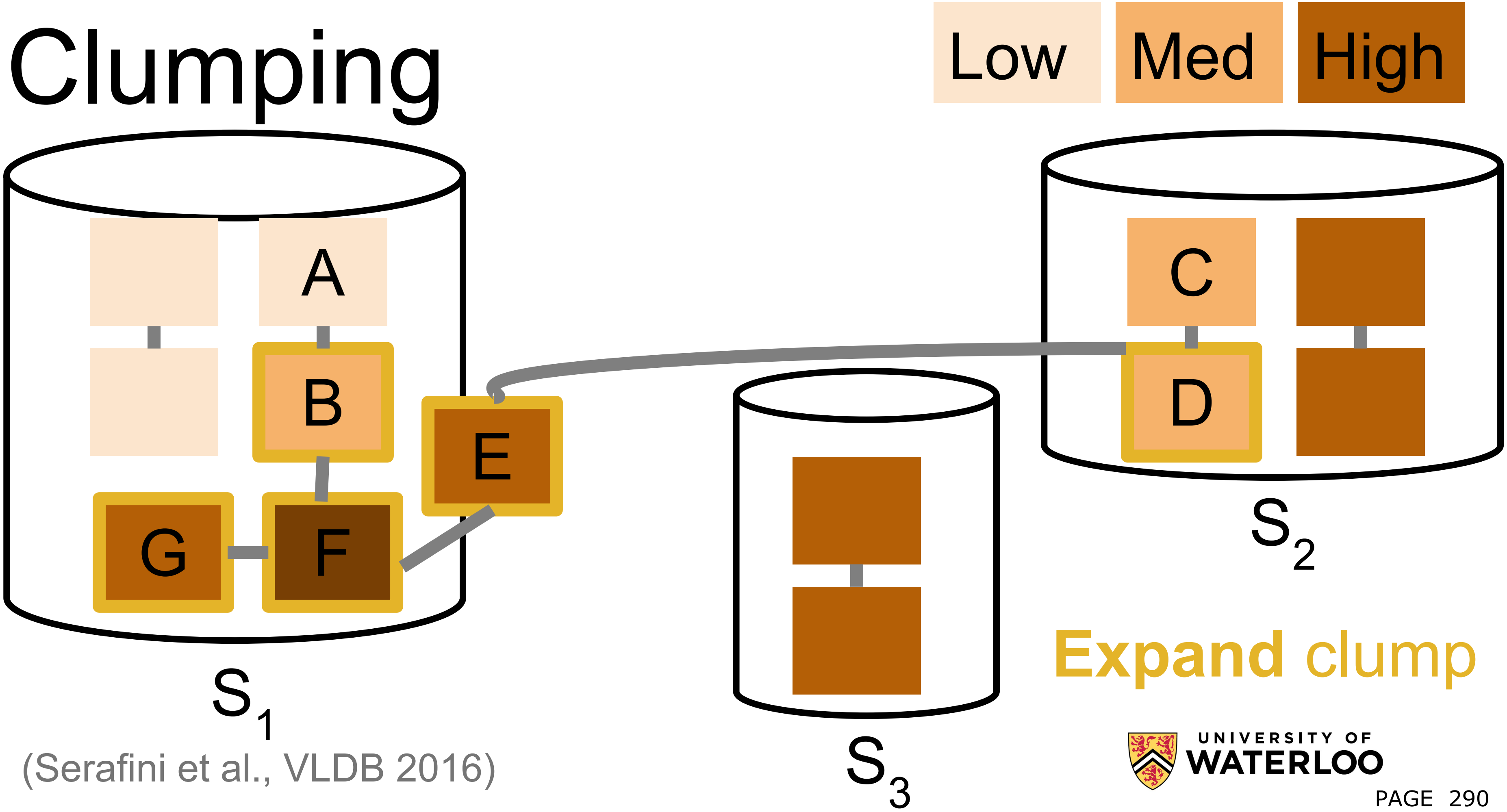
Clumping



Migrate clump

(Serafini et al., VLDB 2016)

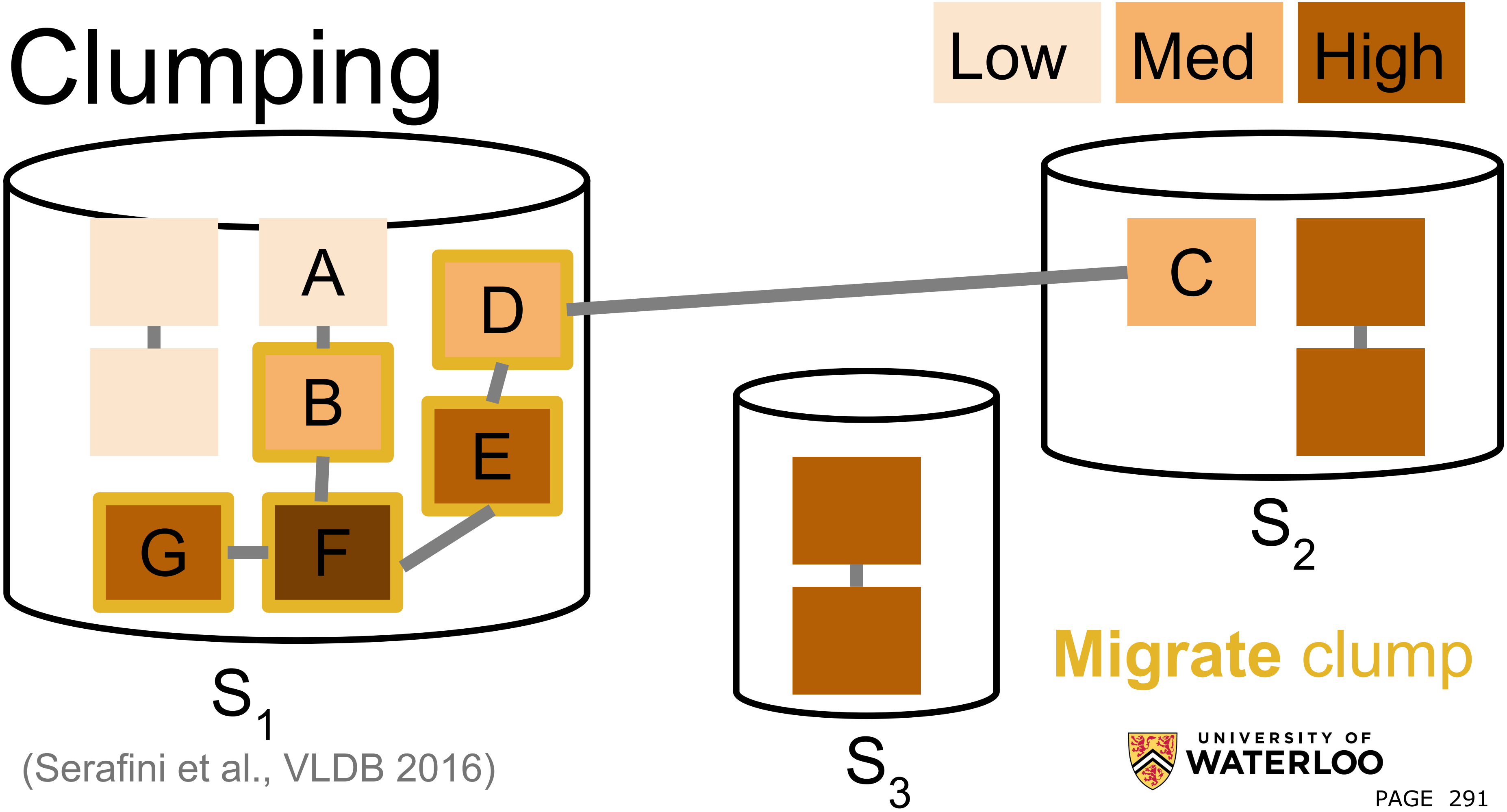
Clumping



Expand clump

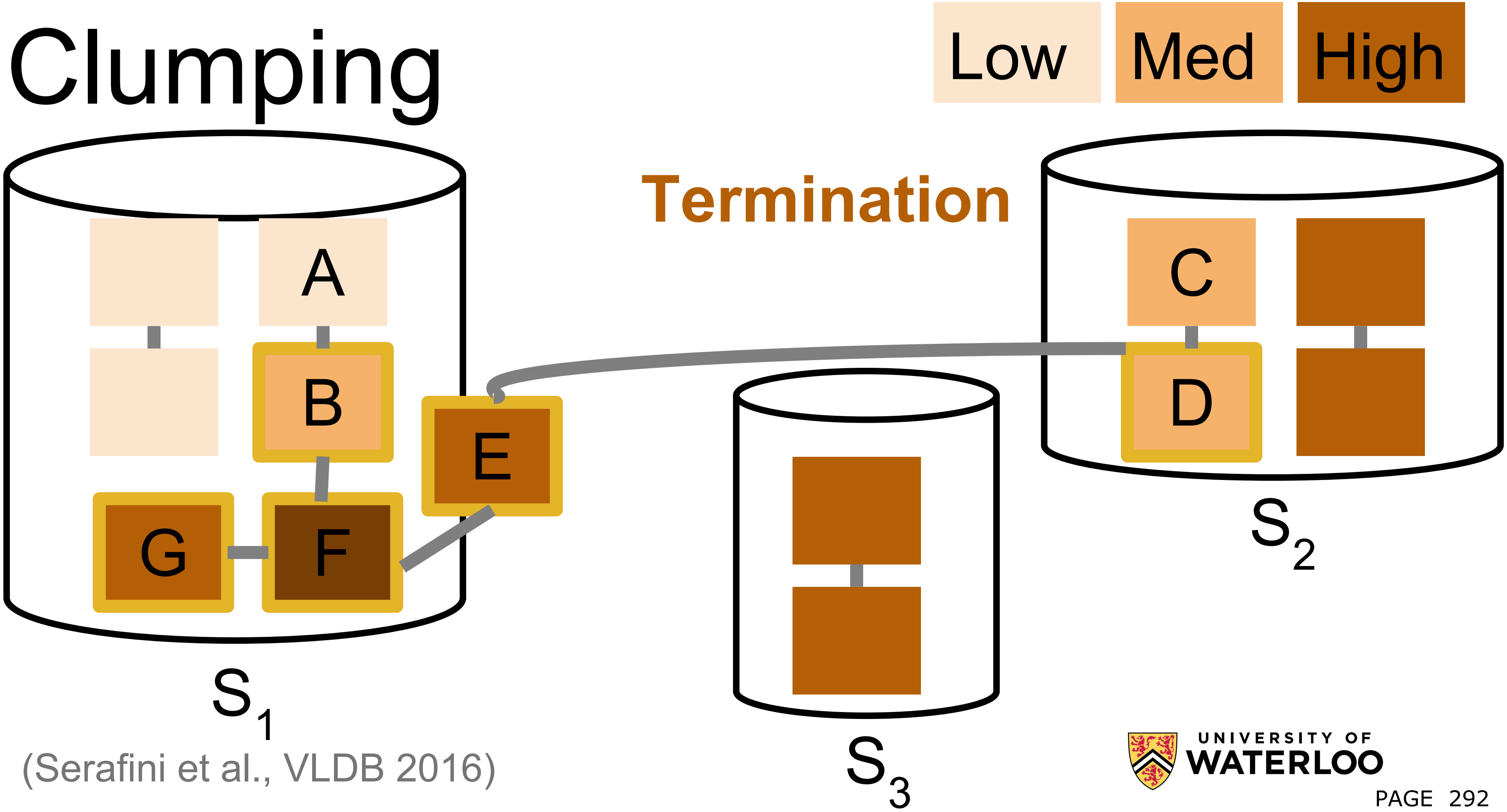
(Serafini et al., VLDB 2016)

Clumping



(Serafini et al., VLDB 2016)

Clumping



(Serafini et al., VLDB 2016)

Clumping

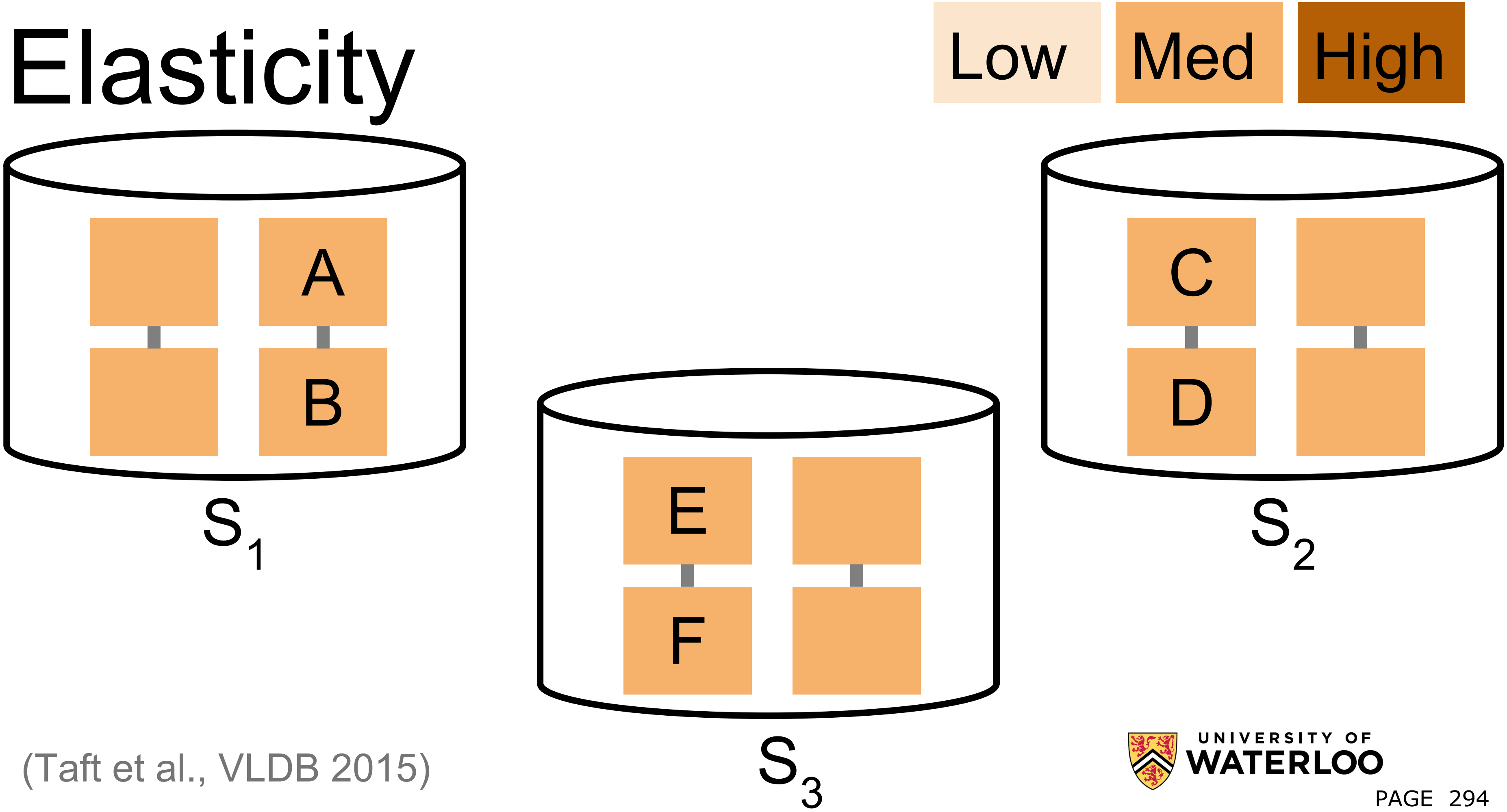
Expands clumps to **frequently accessed neighbours**

Consider moving clump to **lightly loaded sites**

Considers both **re-partitioning and load costs**

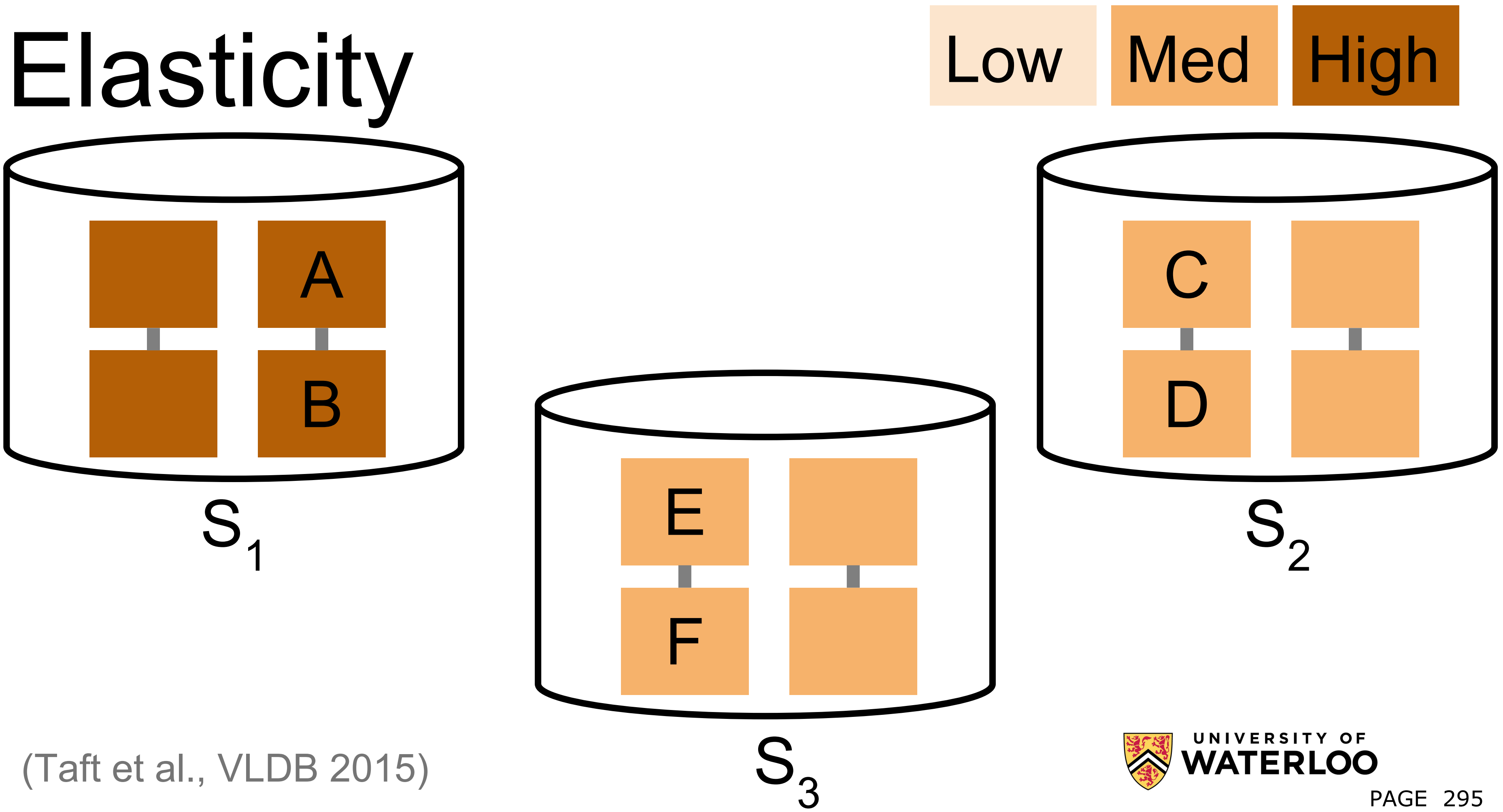
(Serafini et al., VLDB 2016)

Elasticity



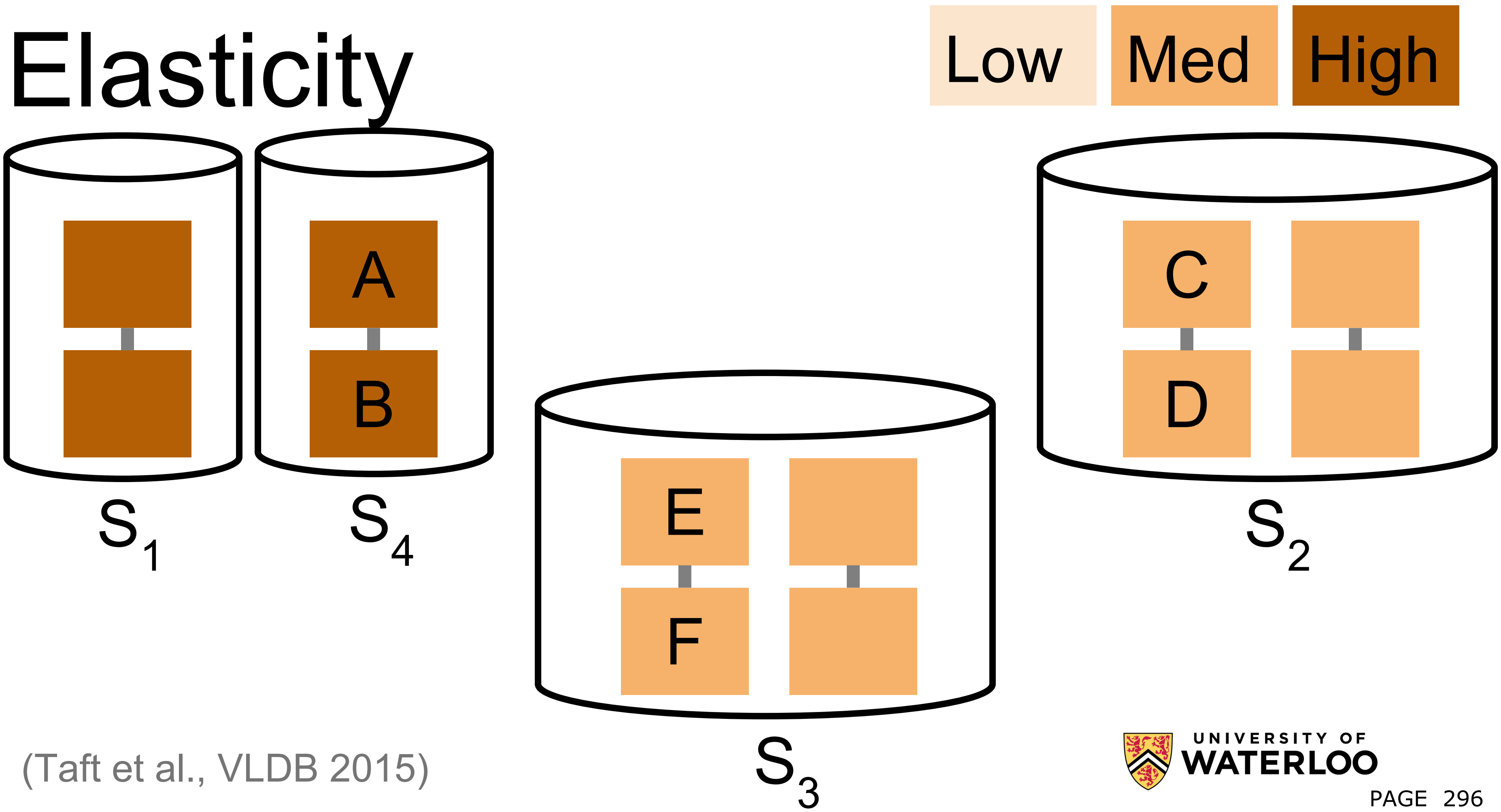
(Taft et al., VLDB 2015)

Elasticity



(Taft et al., VLDB 2015)

Elasticity



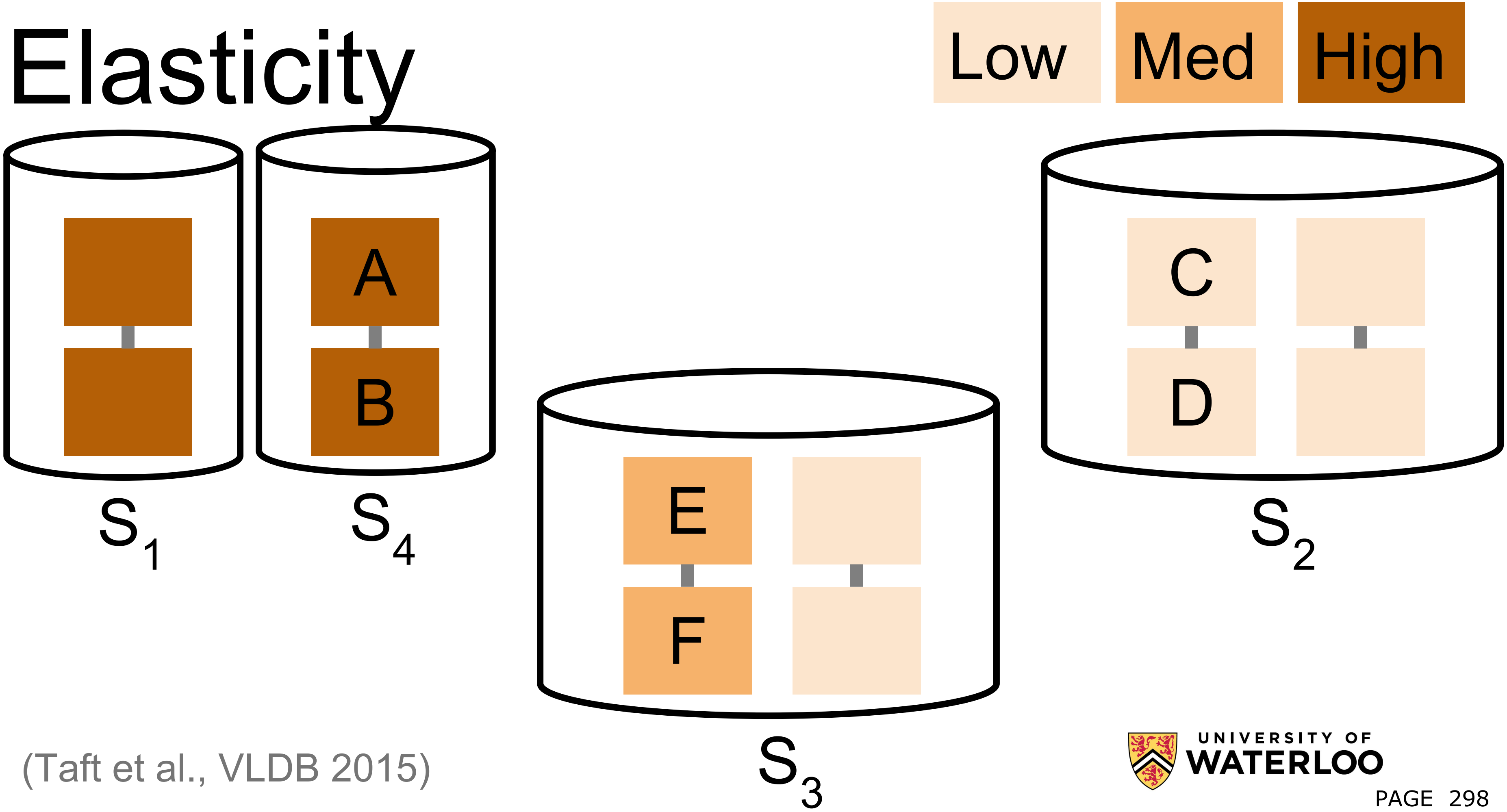
(Taft et al., VLDB 2015)

Elasticity

Repartition to **elastically** add or remove nodes

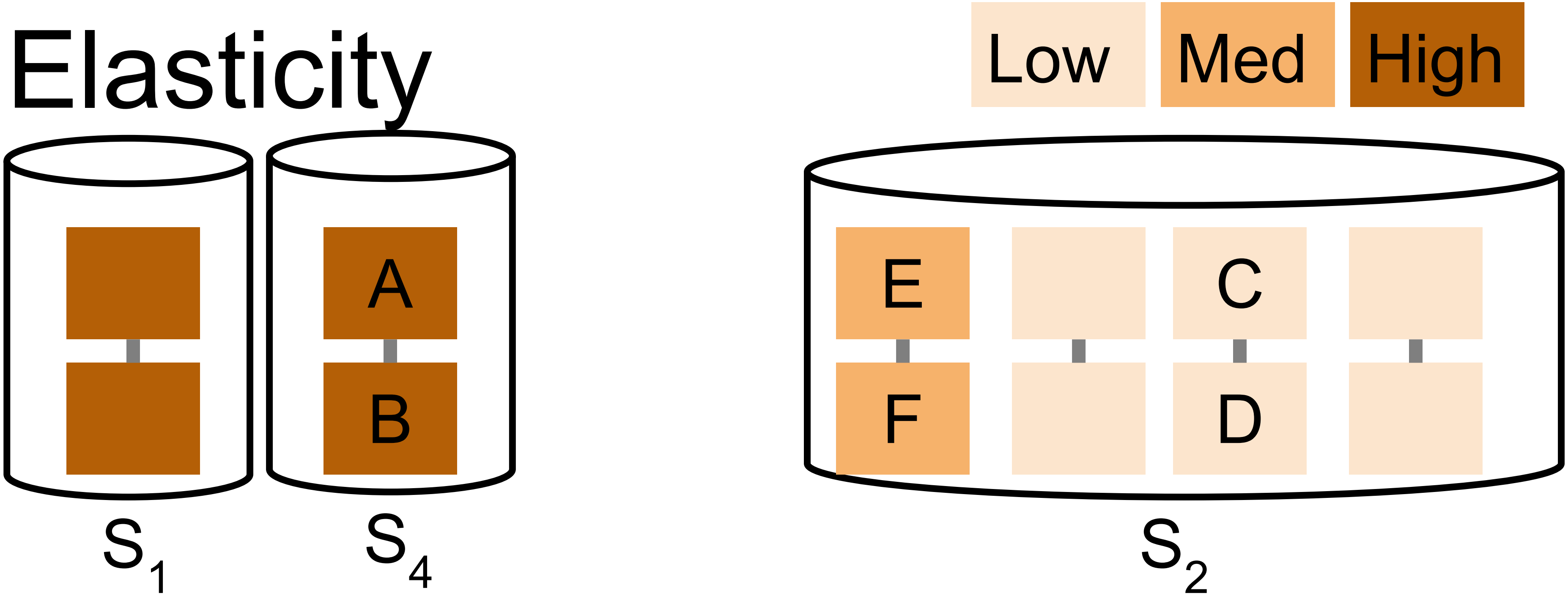
(Taft et al., VLDB 2015)

Elasticity



(Taft et al., VLDB 2015)

Elasticity



(Taft et al., VLDB 2015)

Elasticity Decisions

When the **average load**:

increases: **add** nodes

decreases: **remove** nodes

(Taft et al., VLDB 2015)

Two Tier Data Placement

Identify **hot** data

Evenly distribute hot data

Distribute cold data over remaining capacity

(Taft et al., VLDB 2015)

Identifying Hot Data

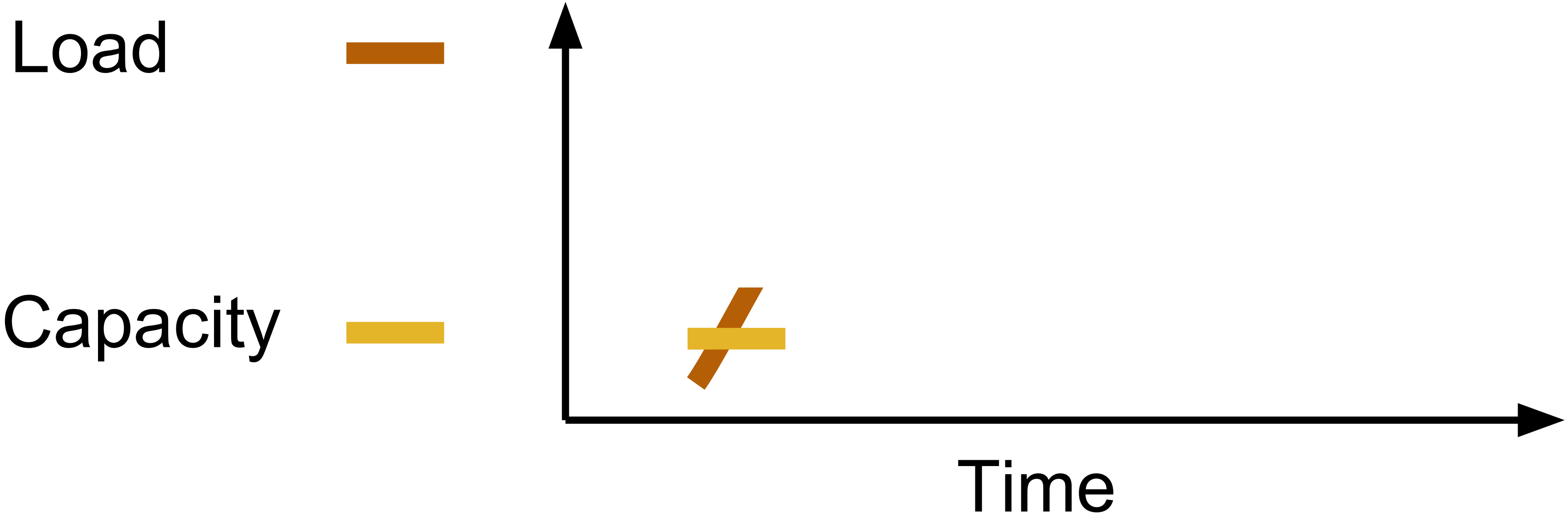
Monitor **partition level** access frequency

If **hot partition** enable **tuple level monitoring**

Reacts to changes in load

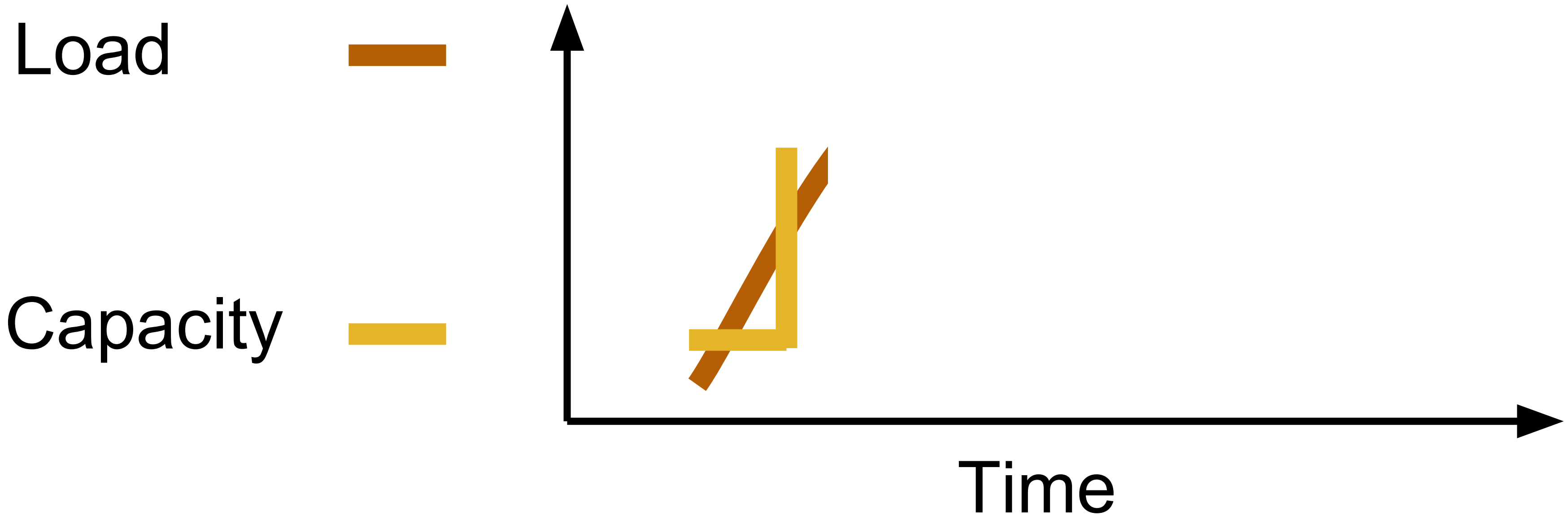
(Taft et al., VLDB 2015)

Reactive Elasticity



(Taft et al., VLDB 2015)

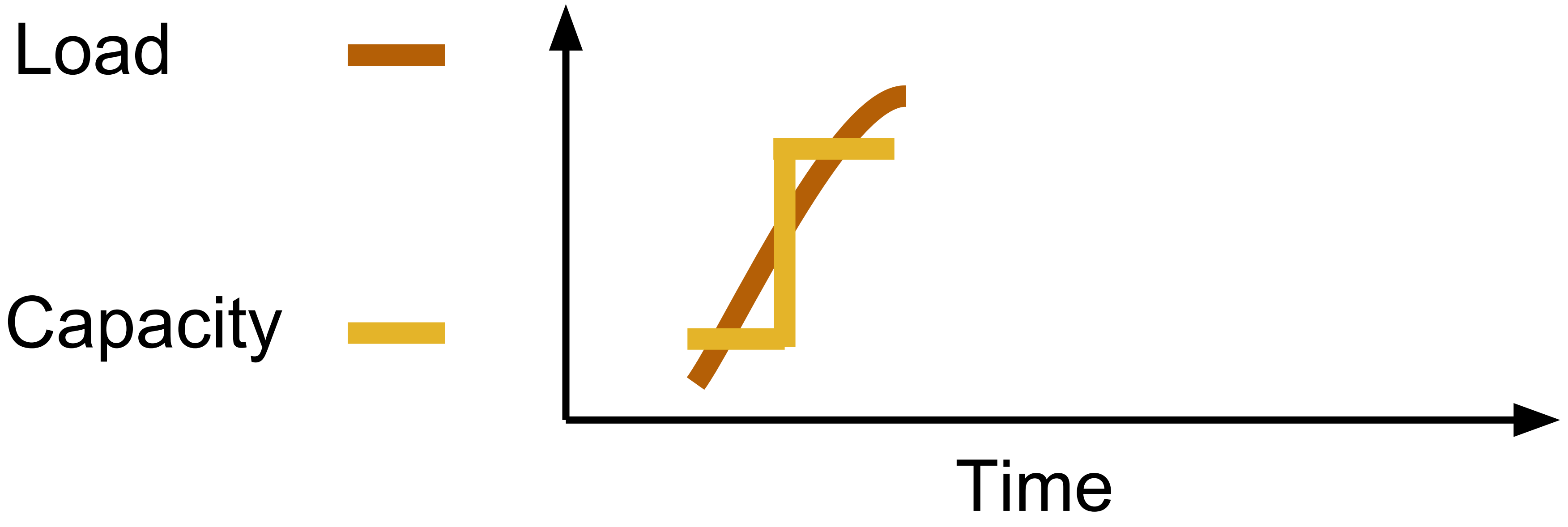
Reactive Elasticity



(Taft et al., VLDB 2015)

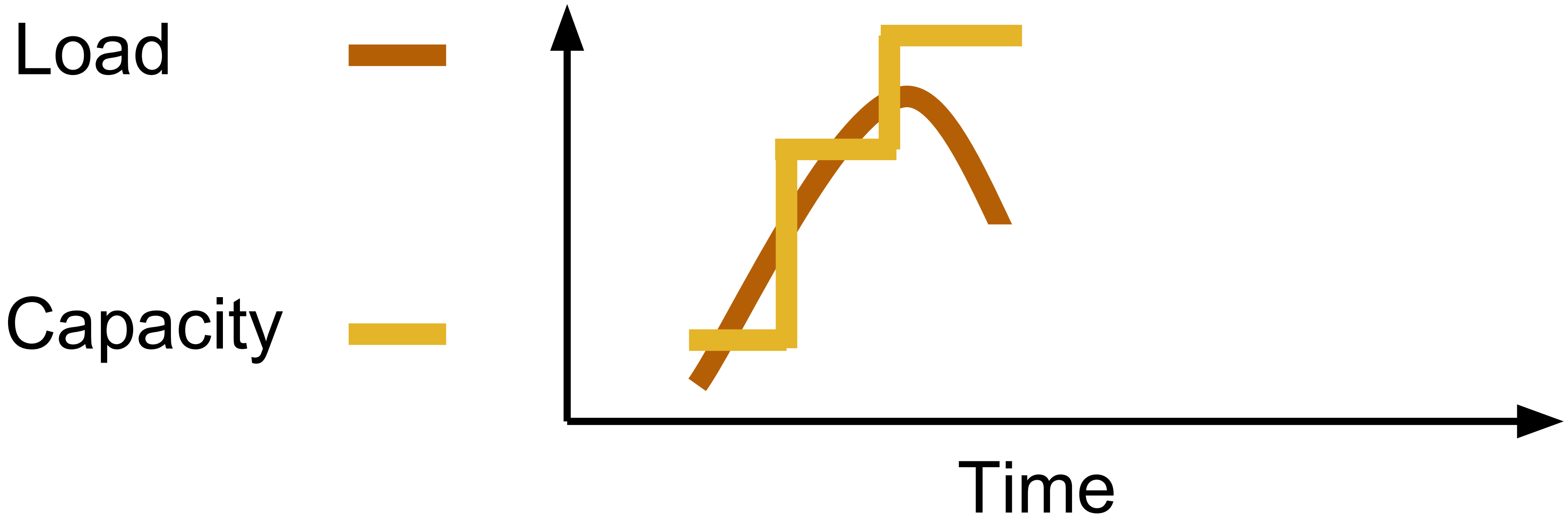
UNIVERSITY OF WATERLOO

Reactive Elasticity



(Taft et al., VLDB 2015)

Reactive Elasticity

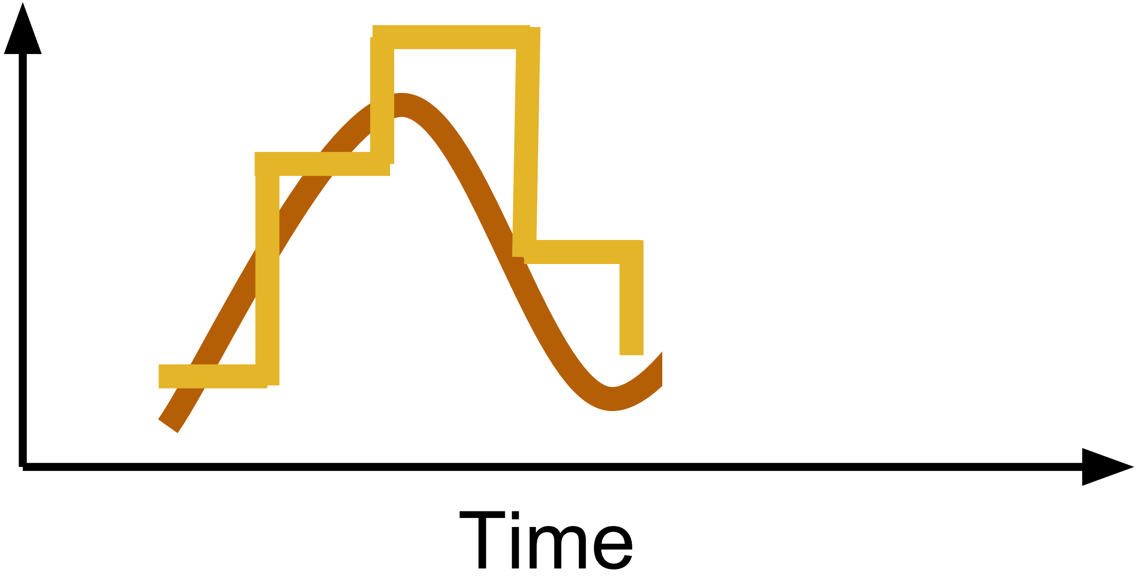


(Taft et al., VLDB 2015)

Reactive Elasticity

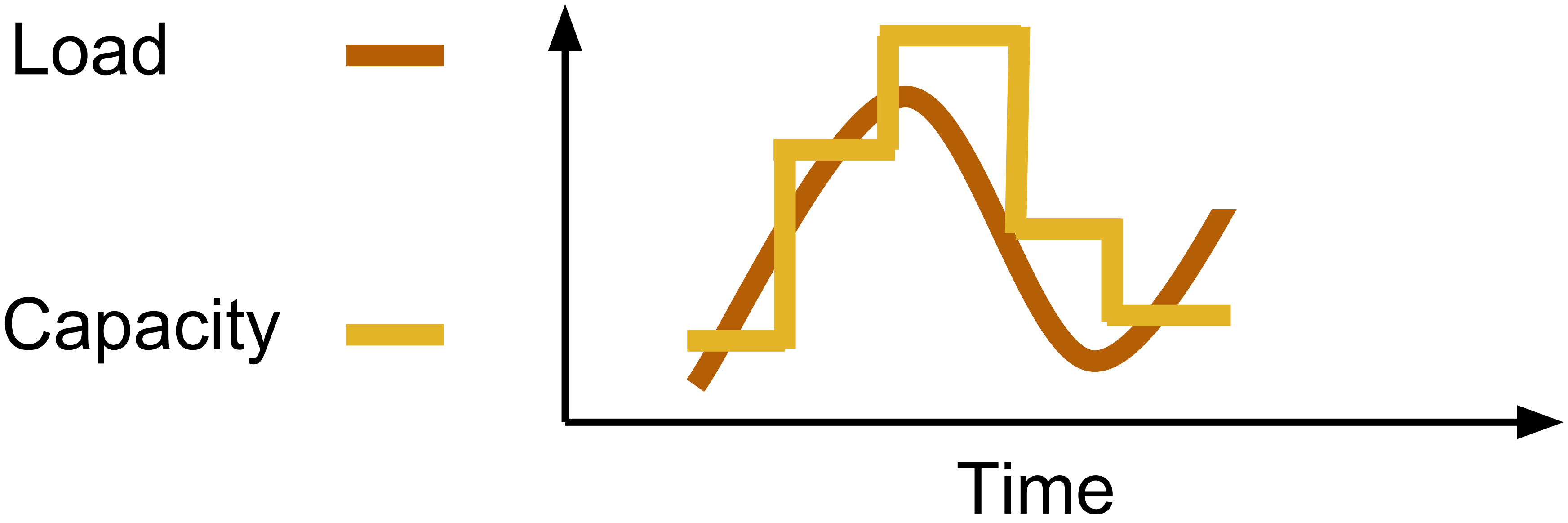
Load

Capacity



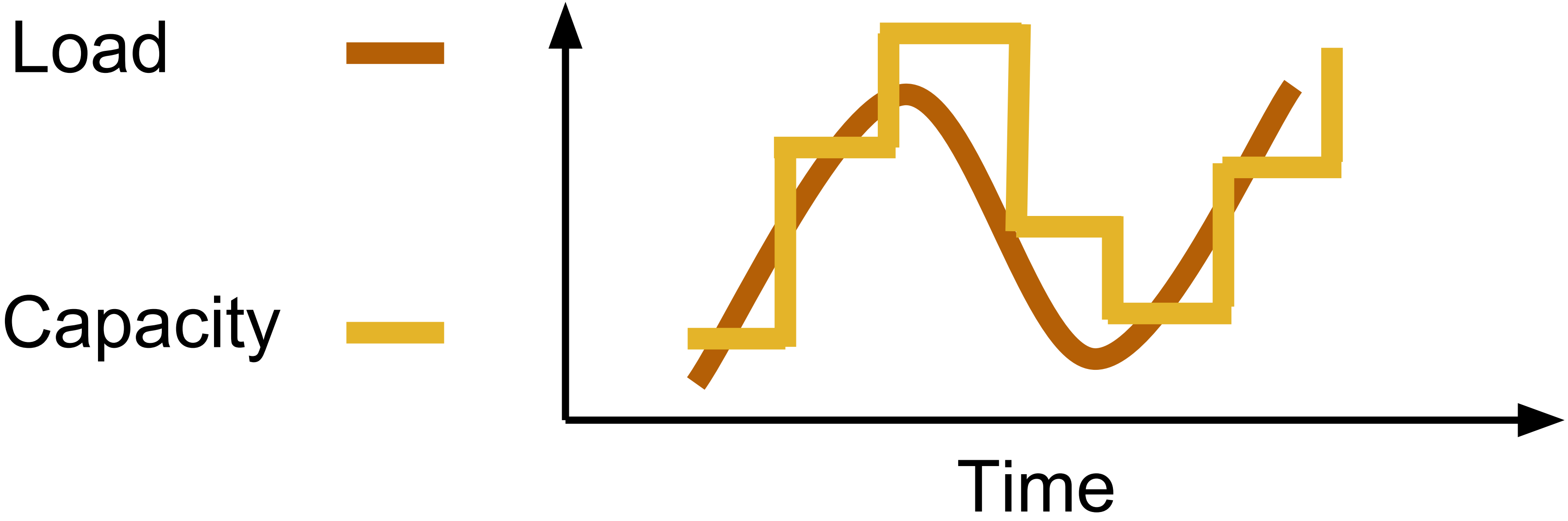
(Taft et al., VLDB 2015)

Reactive Elasticity



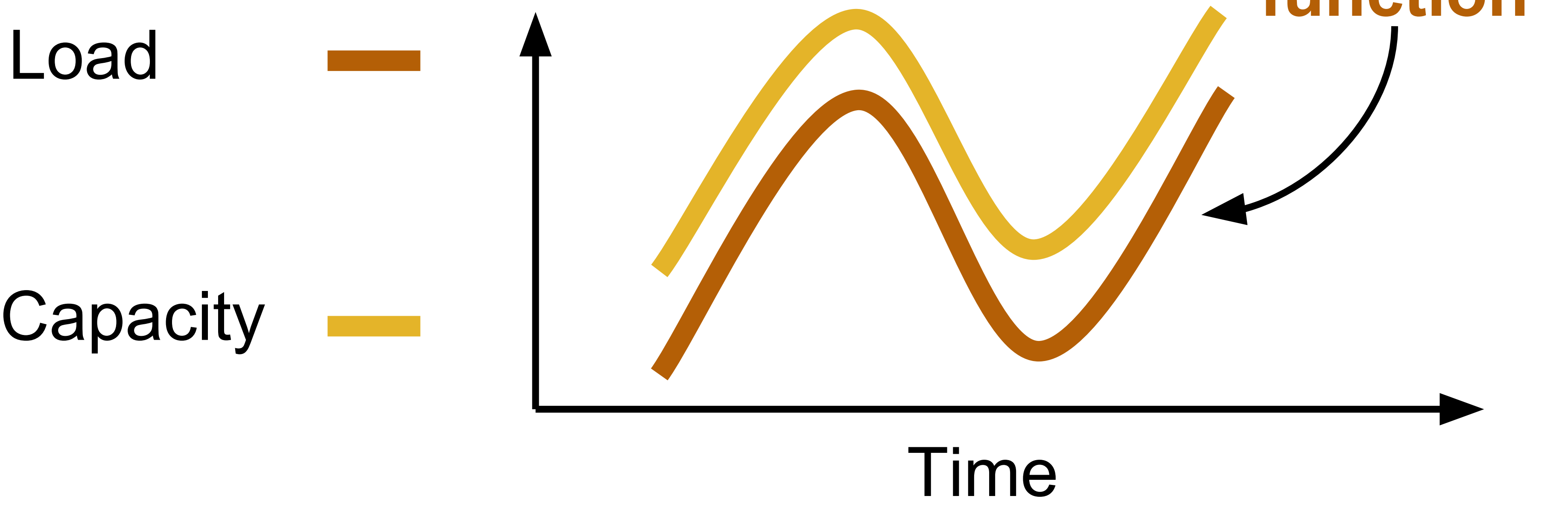
(Taft et al., VLDB 2015)

Reactive Elasticity



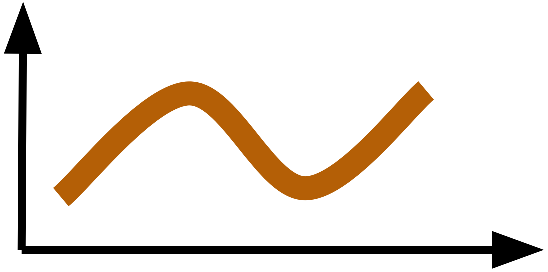
(Taft et al., VLDB 2015)

Ideal Elasticity

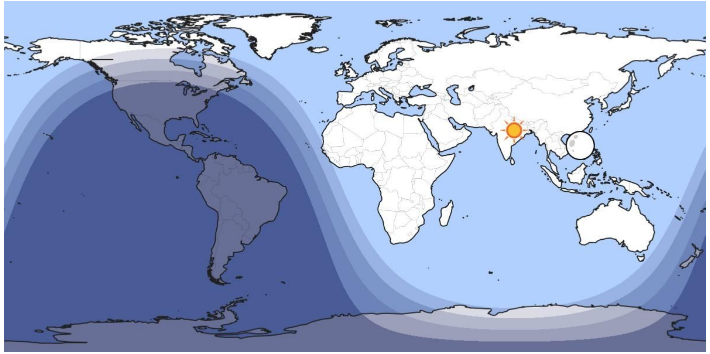


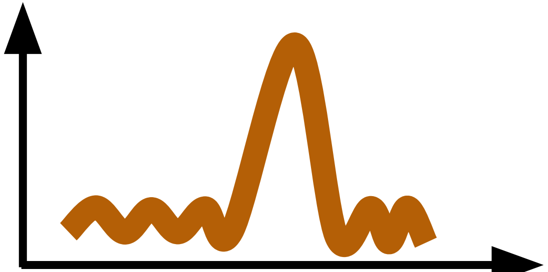
(Taft et al., SIGMOD 2018)

Periodic Workloads



Daily load variations





Seasonal load spikes

(Taft et al., SIGMOD 2018)

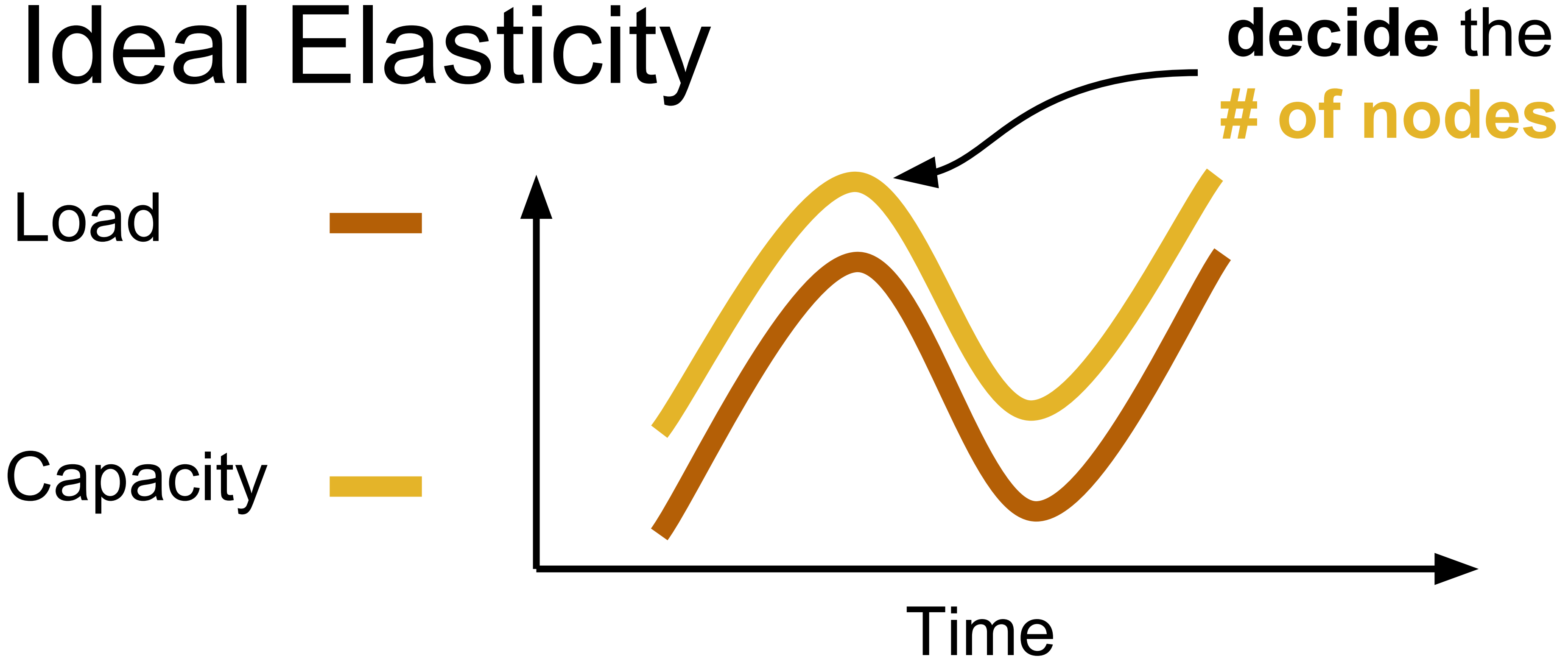
How to Predict Load

SPAR: Sparse Periodic Auto-Regression

$$\mathbf{load}(t) = \mathbf{avg\_load}(t - p_i) + \mathbf{change\_in\_load}(t - j_i)$$

(Taft et al., SIGMOD 2018)

Ideal Elasticity



(Taft et al., SIGMOD 2018)

Number of Nodes

Assuming **partitionable**

$$\text{\# of nodes} = \frac{\text{Predicted Load}}{\text{Load per Server}}$$

(Taft et al., SIGMOD 2018)

Partitioning Decisions

- **How to form partitions?**

  Heuristically (Clumping versus 2 Tier)

- **Where to place partitions?**

  React or predict based on load

- **How to execute multi-partition operations?**

  2PC

(Serafini et al., VLDB 2016)

(Taft et al., SIGMOD 2018)

(Taft et al., VLDB 2015)

UNIVERSITY OF WATERLOO

Road Map

- Adaptive Replication
- Adaptive Partitioning
- Outlook

Outlook

Adaptive Systems

How to **make** a partitioning or replication **decision** **when** access patterns **change**?

Adaptively replicate and partition

UNIVERSITY OF WATERLOO

Partitioning Decisions

- **How to form partitions?**

  Iterative, Temporarily, Graph partitioning, Heuristic

- **Where to place partitions?**

  Sorted, Leader, At requester, Graph partitioning, Reactively, Predictively

- **How to execute multi-partition operations?**

  Novel protocols, 2PC

UNIVERSITY OF WATERLOO

Replication Decisions

- **How many replicas?**
  Decentralized, Client workload, Cost-based, Predictive, Fault tolerance
- **Where to place replicas?**
  At requester, Heuristic, Cost-based, Predictive, Dynamic
- **How to propagate updates?**
  Synchronous, Quorums, Single-master, Cache

Decisions

- How **many** replicas? **Predictively**
- Where to **place** replicas? **Predictively**
- Where to **place** partitions? **Predictively**

Adaptive & Predictive Systems

How to **make** a partitioning or replication **decision when** access patterns **change**?

Adaptively and predictively
replicate and partition

UNIVERSITY OF WATERLOO

Predicting the Future

How can your system **predict** its future workload?

Apollo: Predict future queries (Markov Model)

P-Store: Predict future load (SPAR)

Predicting the Future **QB5000**

When, how many, and **what** queries will arrive?

SELECT * **FROM** C **WHERE** id = “C1”

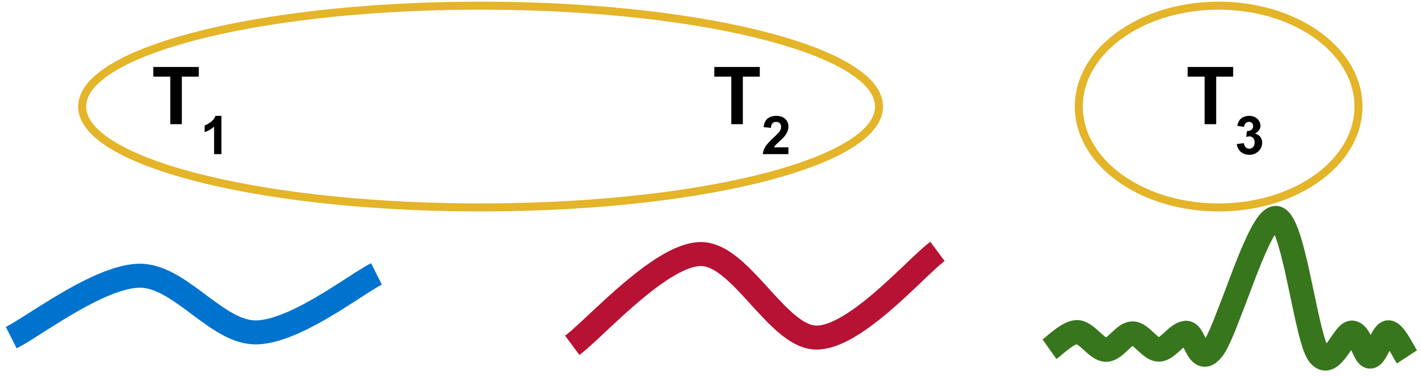
SELECT * **FROM** C **WHERE** id = $

Pre-process: remove parameters, creating templates

(Ma et al., SIGMOD 2018)

Predicting the Future **QB5000**

When, how many, and what queries will arrive?

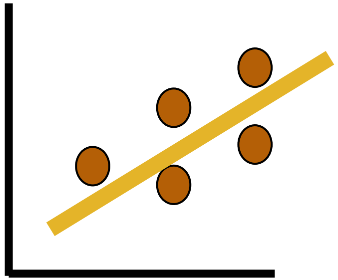


Cluster: group templates by arrival rate

(Ma et al., SIGMOD 2018)

Predicting the Future **QB5000**

When, how many, and **what** queries will arrive?



Forecast: Predict clusters arrival rate (Ensemble of RNN, LR, KR)

(Ma et al., SIGMOD 2018)

Predicting the Future **QB5000**

When, how many, and **what** queries will arrive?

Pre-process: remove parameters, creating templates

Cluster: group templates by arrival rate

Forecast: Predict clusters arrival rate (Ensemble of RNN, LR, KR)

(Ma et al., SIGMOD 2018)

Predicting the Future

How can your system **predict** its future workload?

Apollo: Predict future queries (Markov Model)

P-Store: Predict future load (SPAR)

QB5000: Predict query workloads (Ensemble of RNN, LR, KR)

Predicting the Future

How can your system **predict** its future workload?

If your system **knew the future workload**, how could it **partition and replicate** data?