DynaMast

Adaptive Dynamic Mastering for Replicated Systems

Michael Abebe mtabebe@uwaterloo.ca

Brad Glasbergen

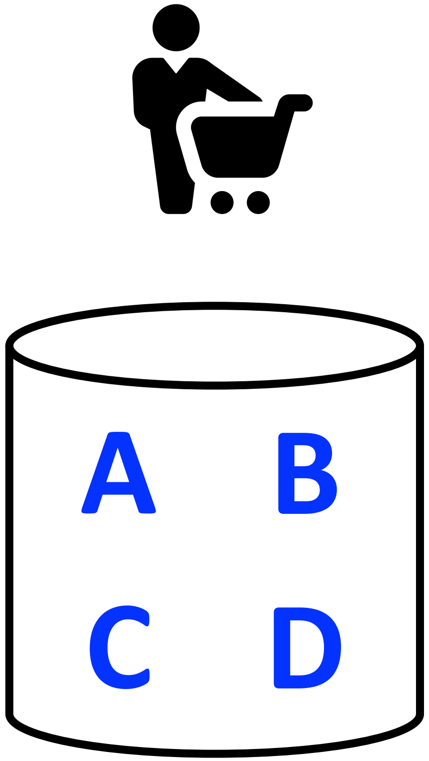
Khuzaima Daudjee

UNIVERSITY OF WATERLOO

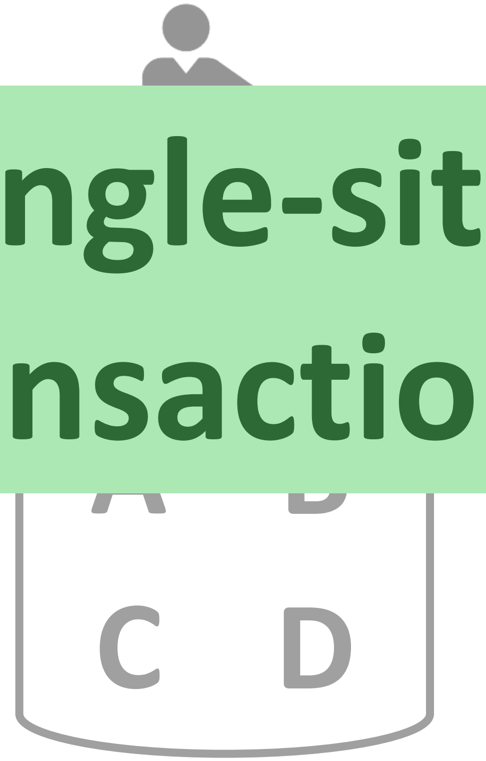
ICDE (April 2020)

tiny.cc/dynamast

Single Database

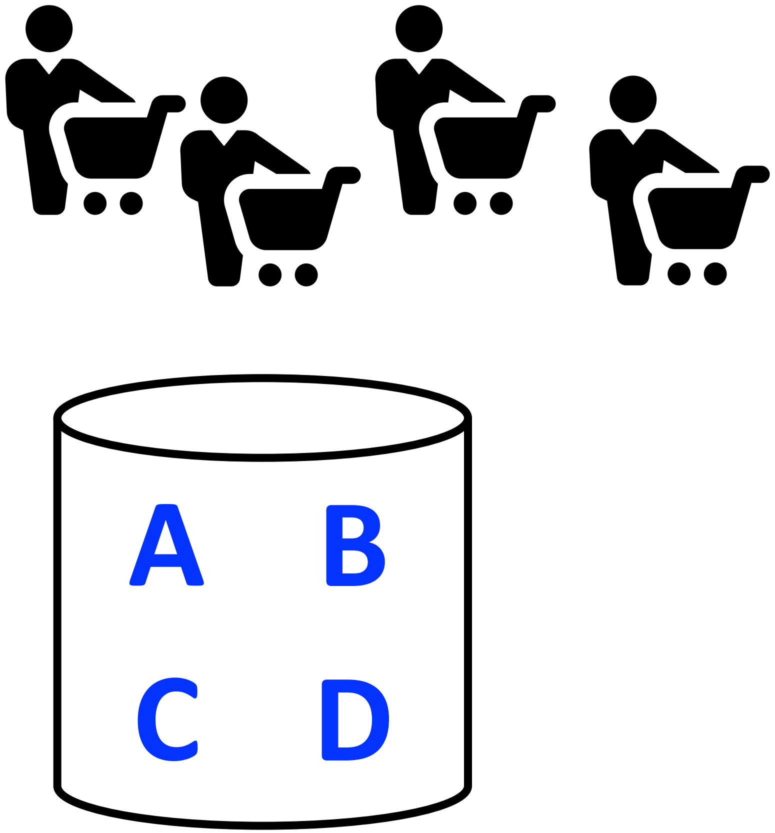


Single Database

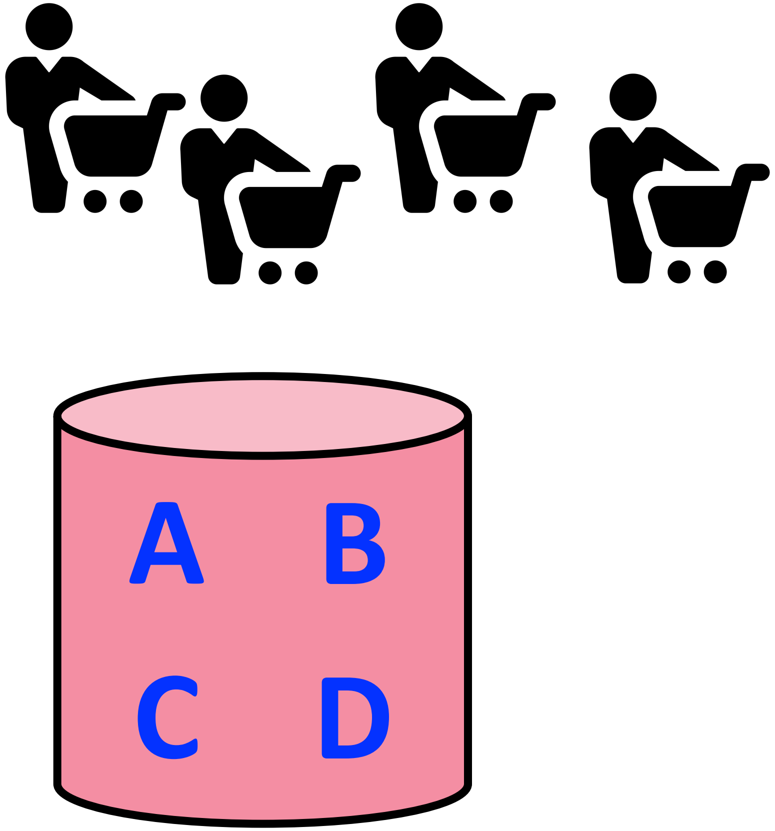


Single-site transactions

Single Database



Single Database



Single Database

Overloads site

How to scale the database?

Single-master

Multi-master

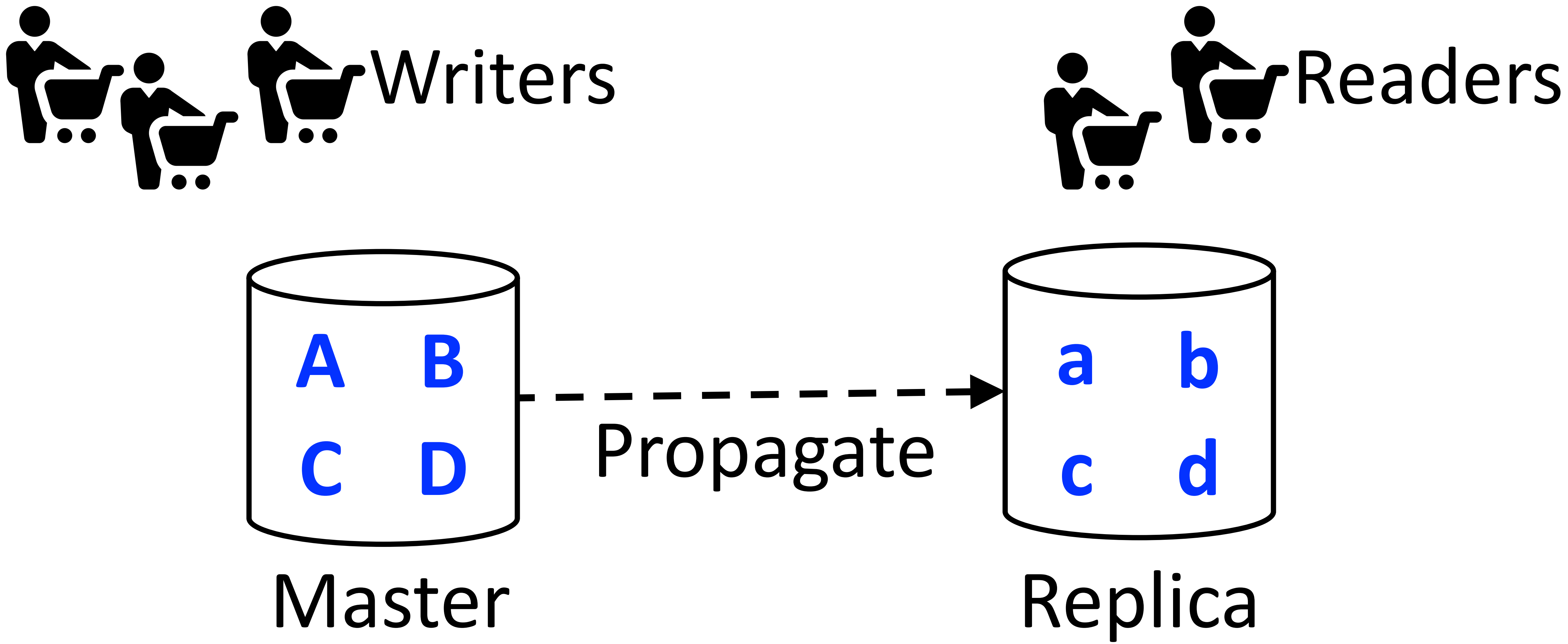
UNIVERSITY OF WATERLOO

How to scale the database?

Single-master

Multi-master

Single-Master



Writers

Readers

A B
C D

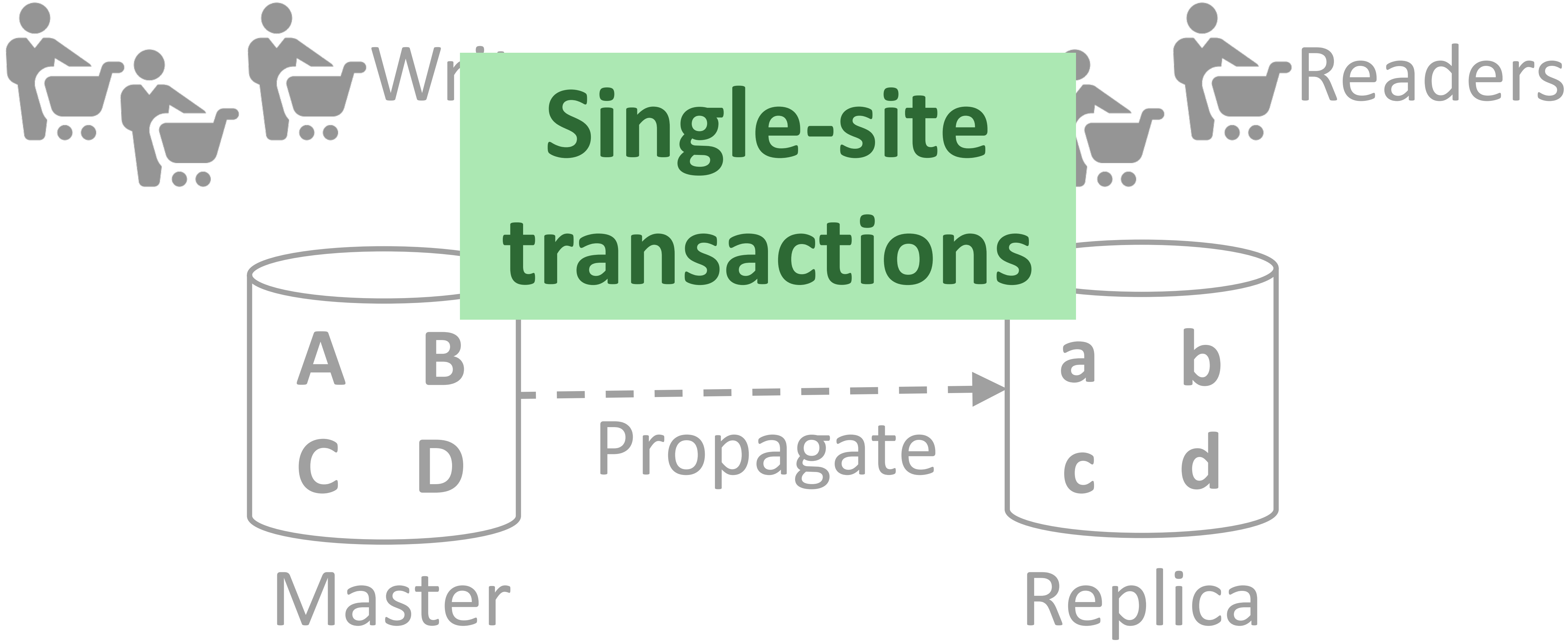
Master

Propagate

a b
c d

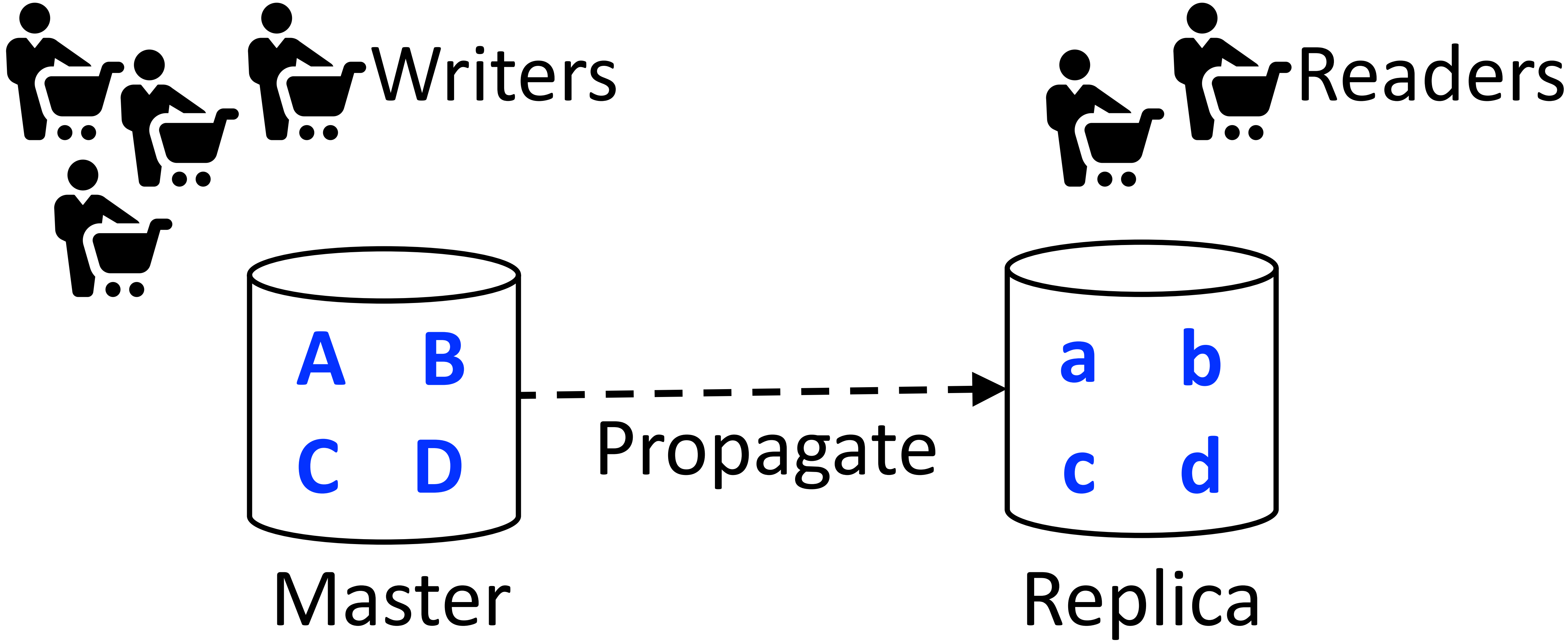
Replica

Single-Master

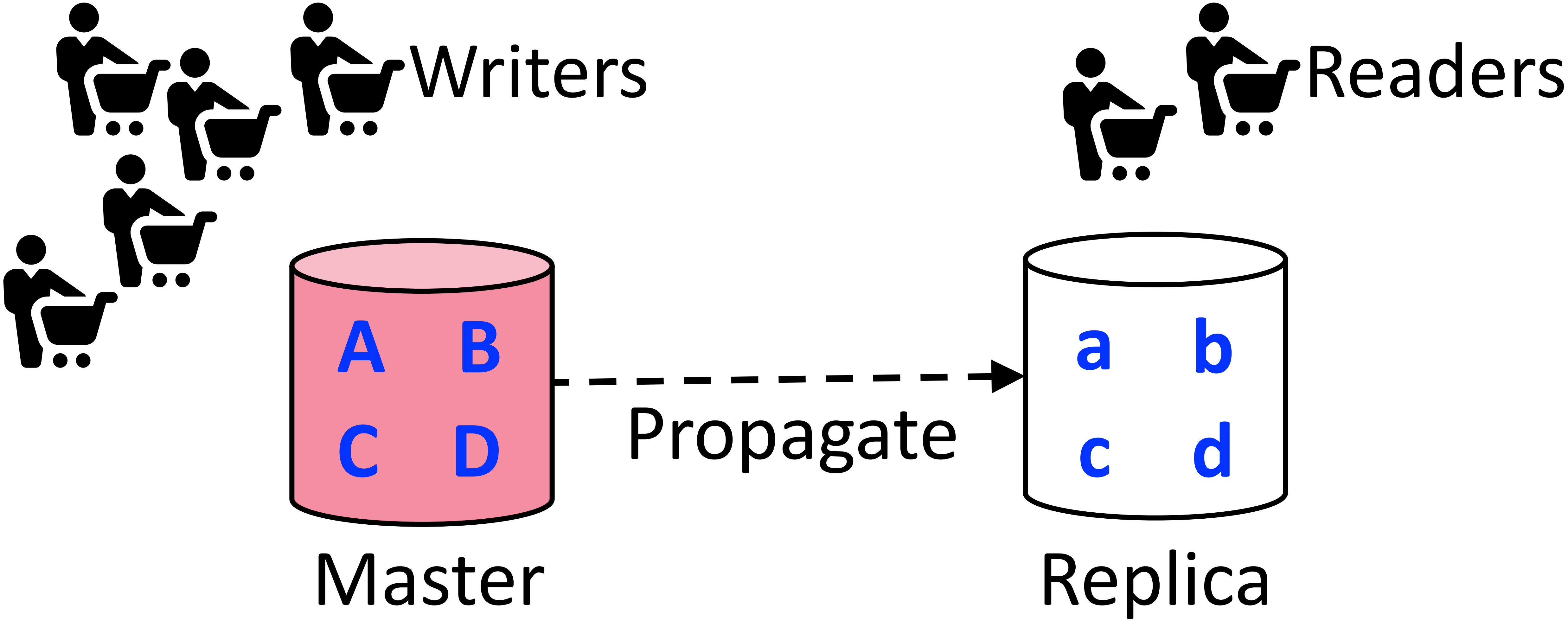


Single-site transactions

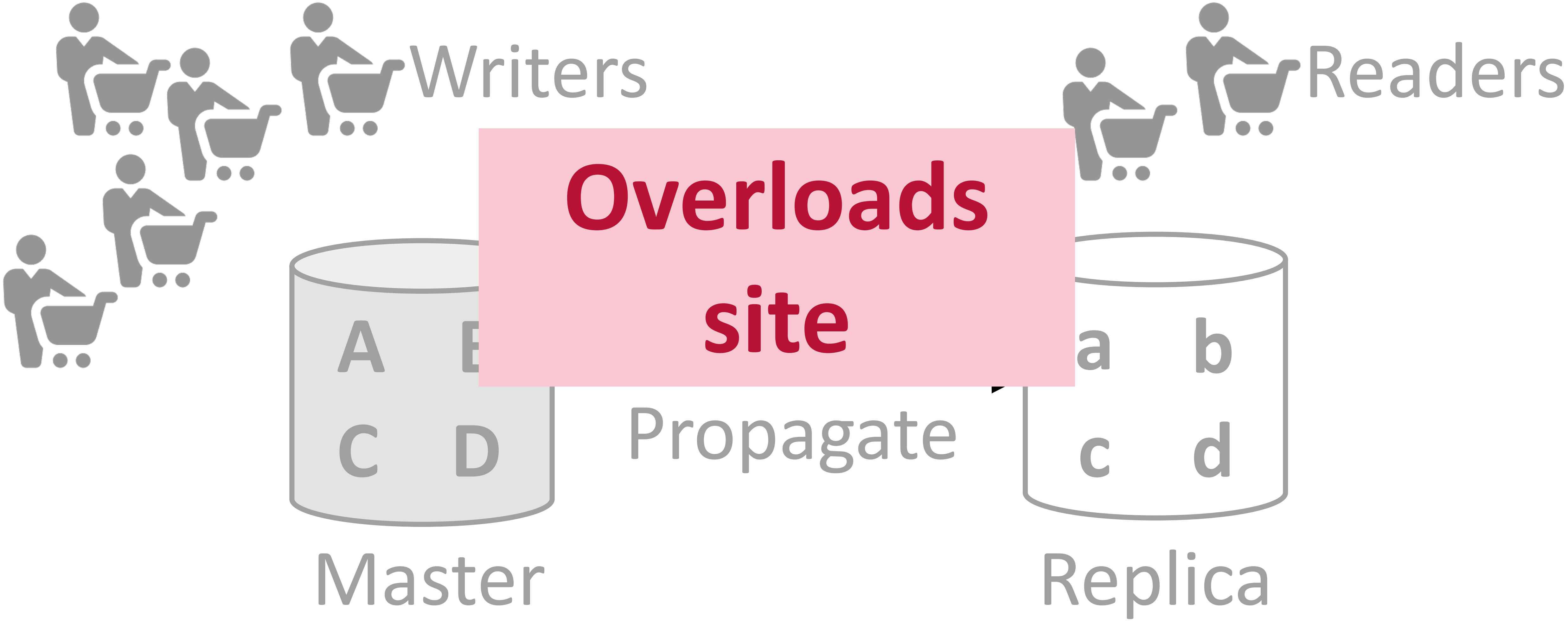
Single-Master



Single-Master



Single-Master



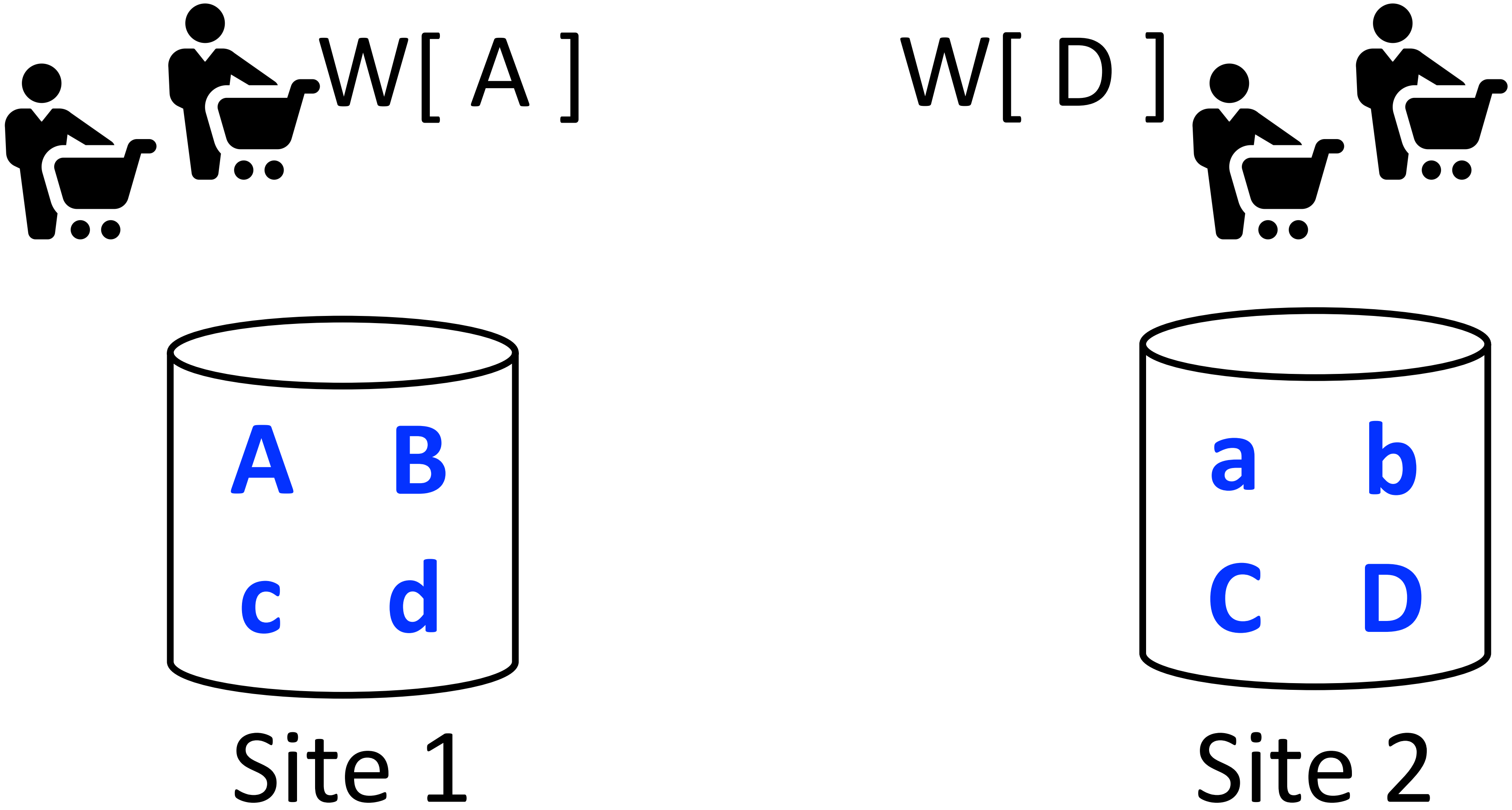
Overloads site

How to scale the database?

Single-master

Multi-master

Multi-Master



W[A]

W[D]

A B

c d

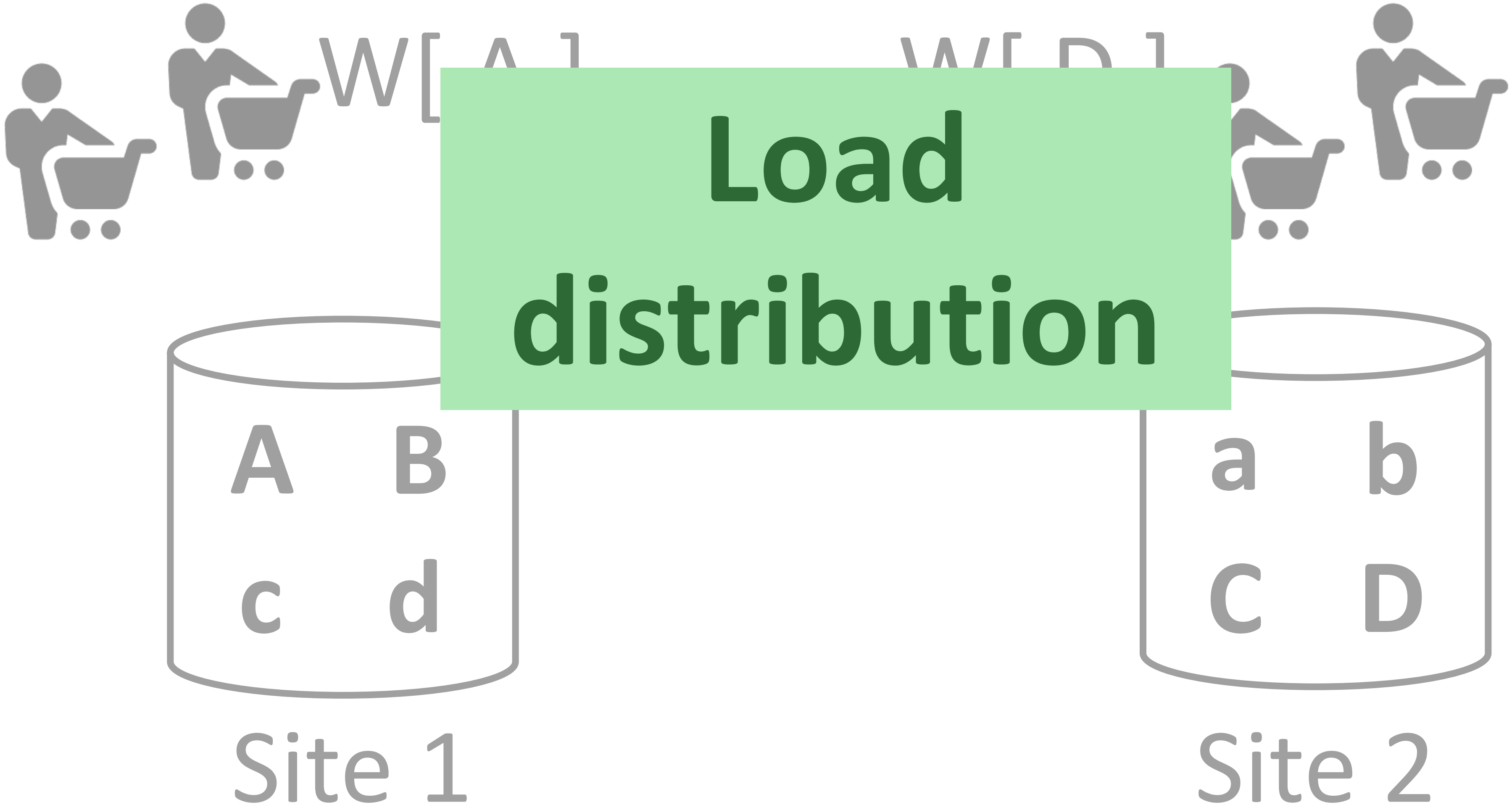
a b

C D

Site 1

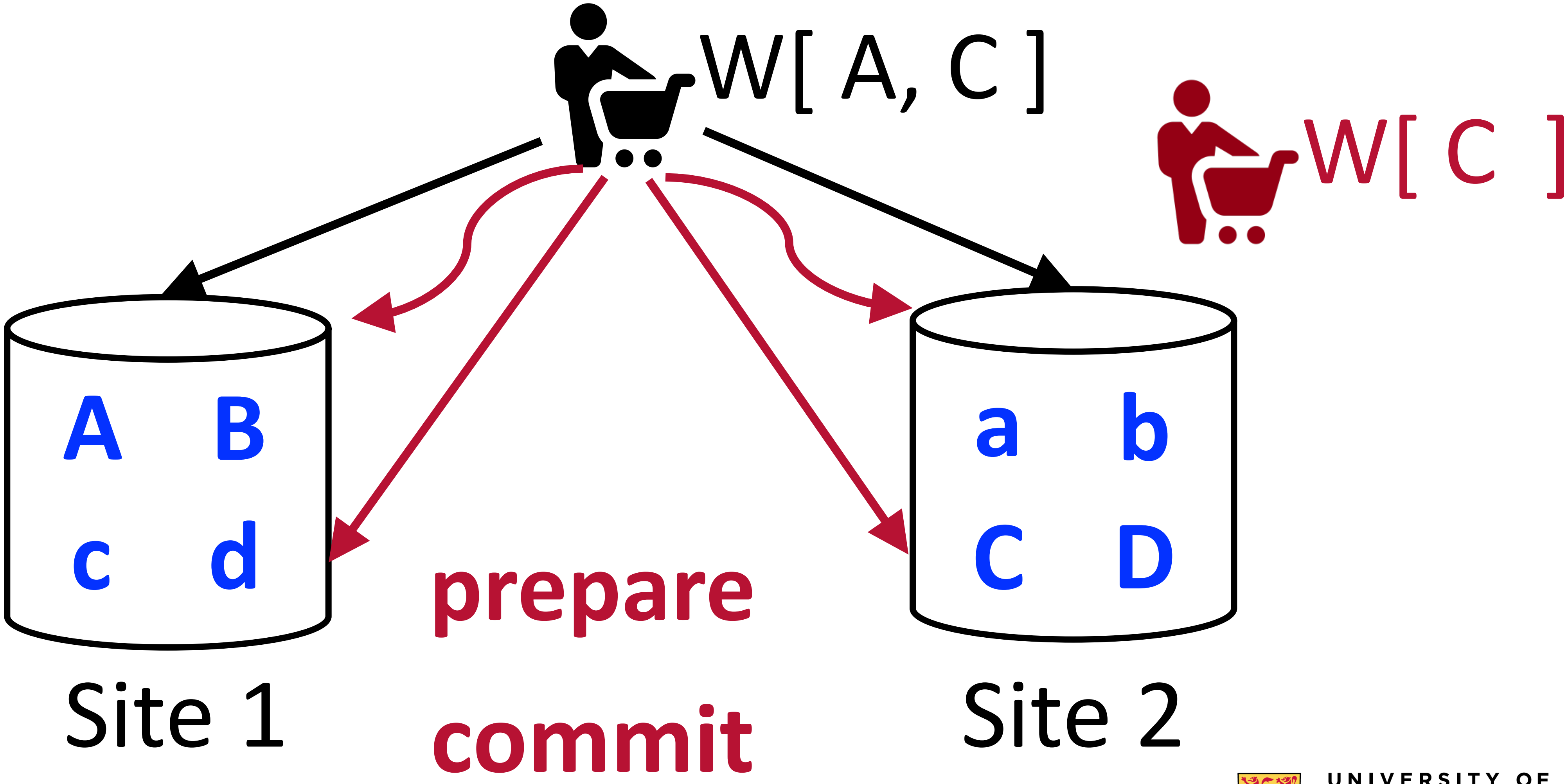
Site 2

Multi-Master

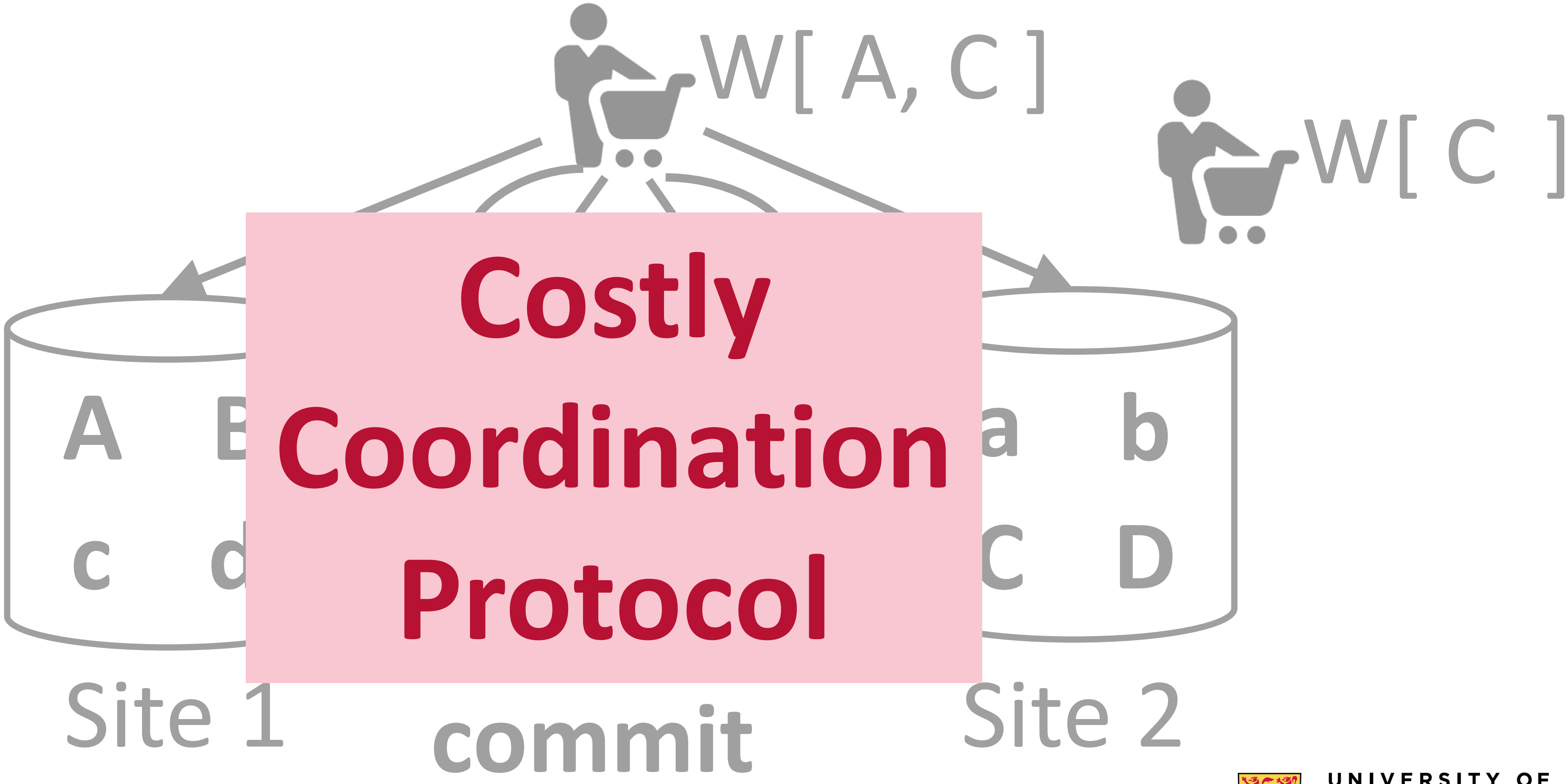


Load distribution

Multi-Master



Multi-Master



Costly Coordination Protocol

Multi-Master	Single-Master
Load distribution	Single-site transactions
Costly Coordination Protocol	Overloads site

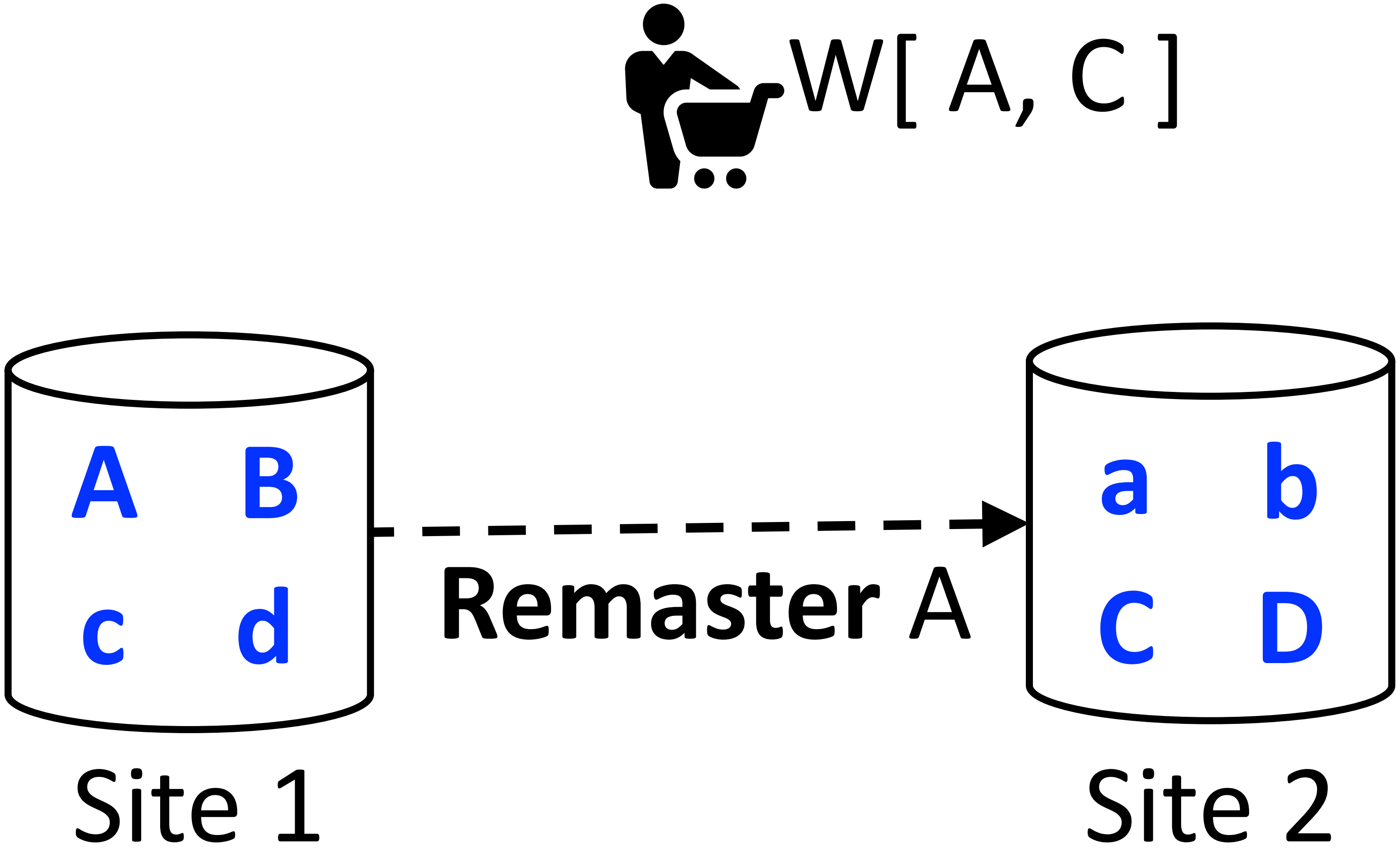
How to provide:

Load distribution

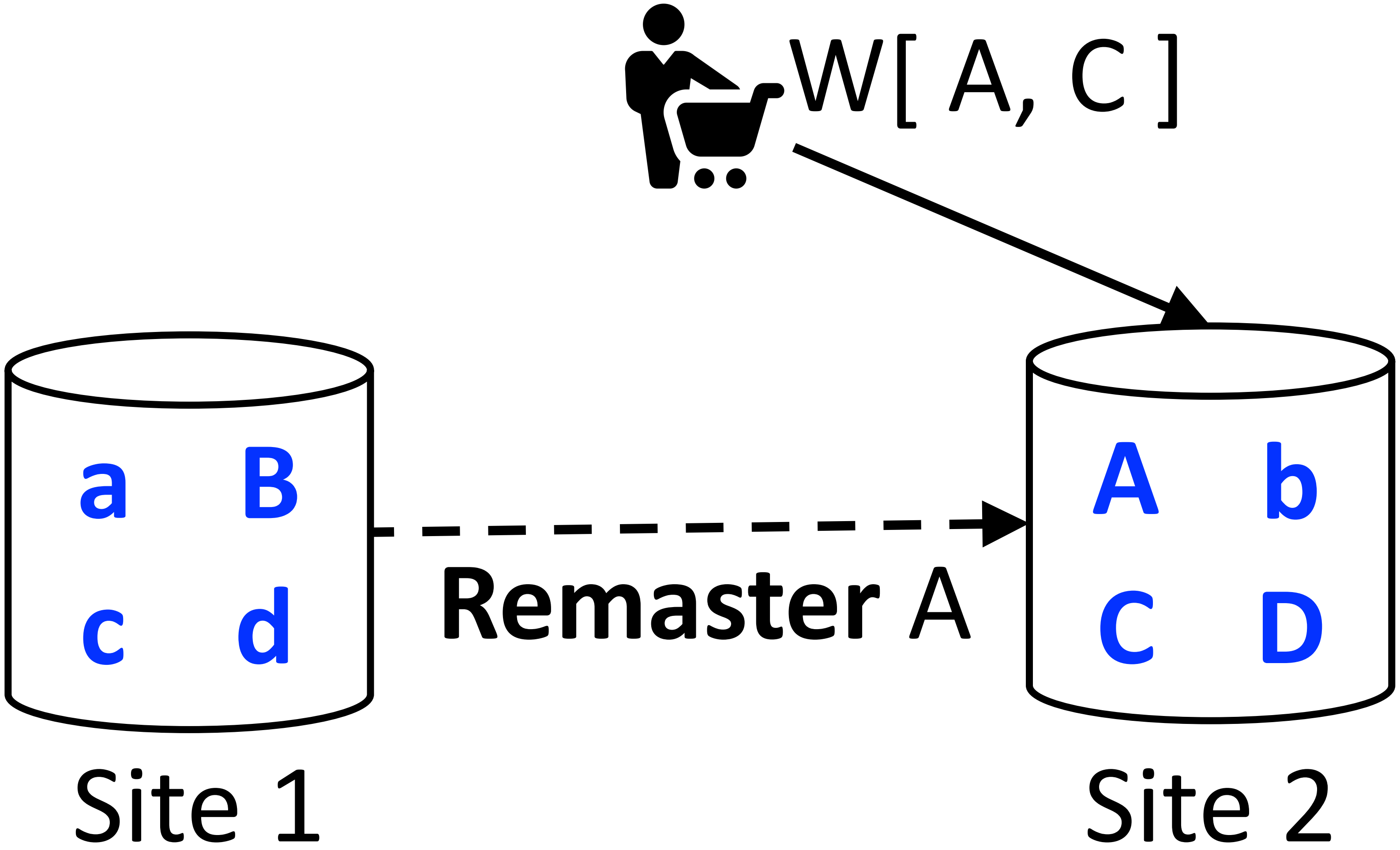
Single-site transactions

Dynamic Mastering

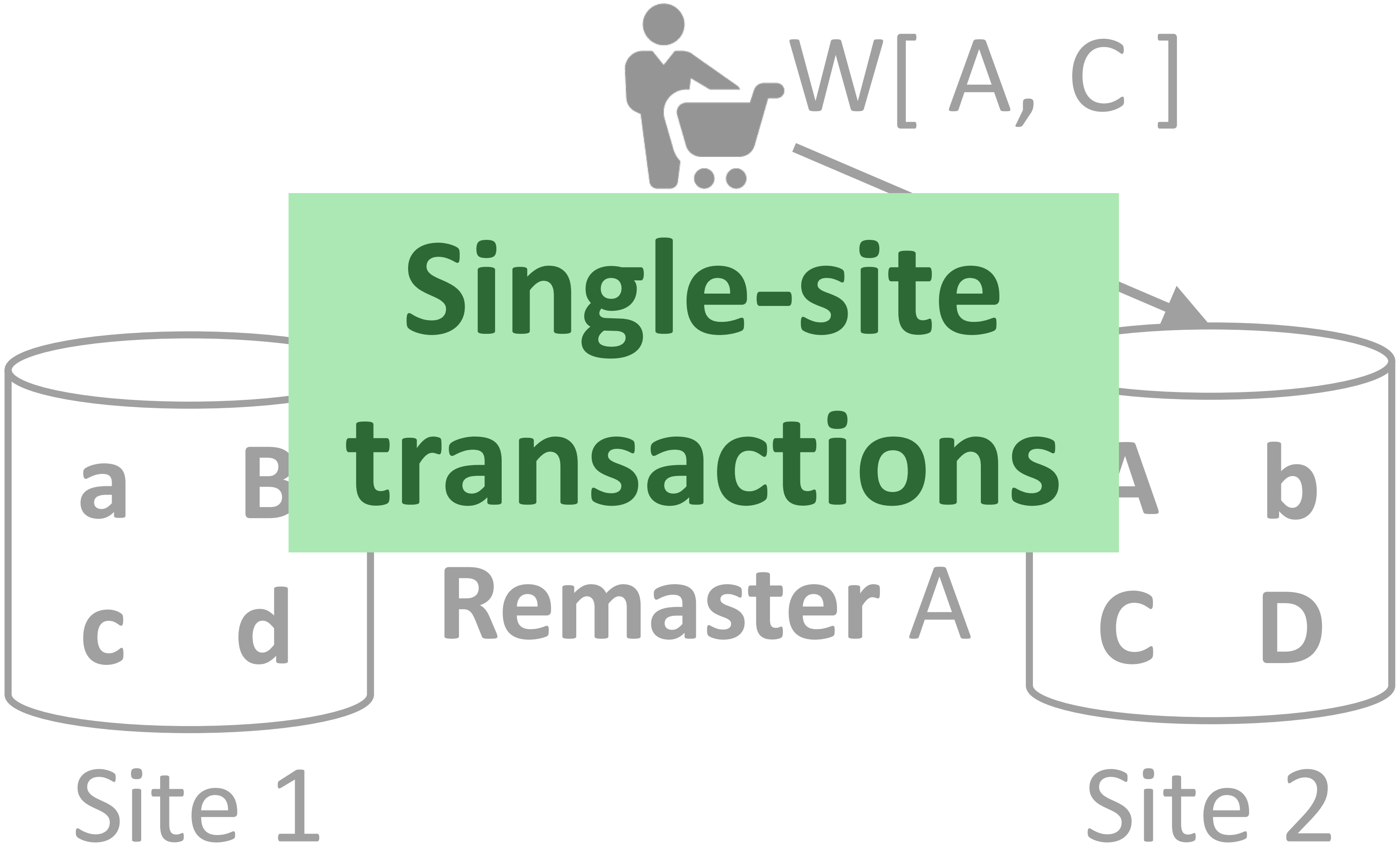
Dynamic Mastering



Dynamic Mastering

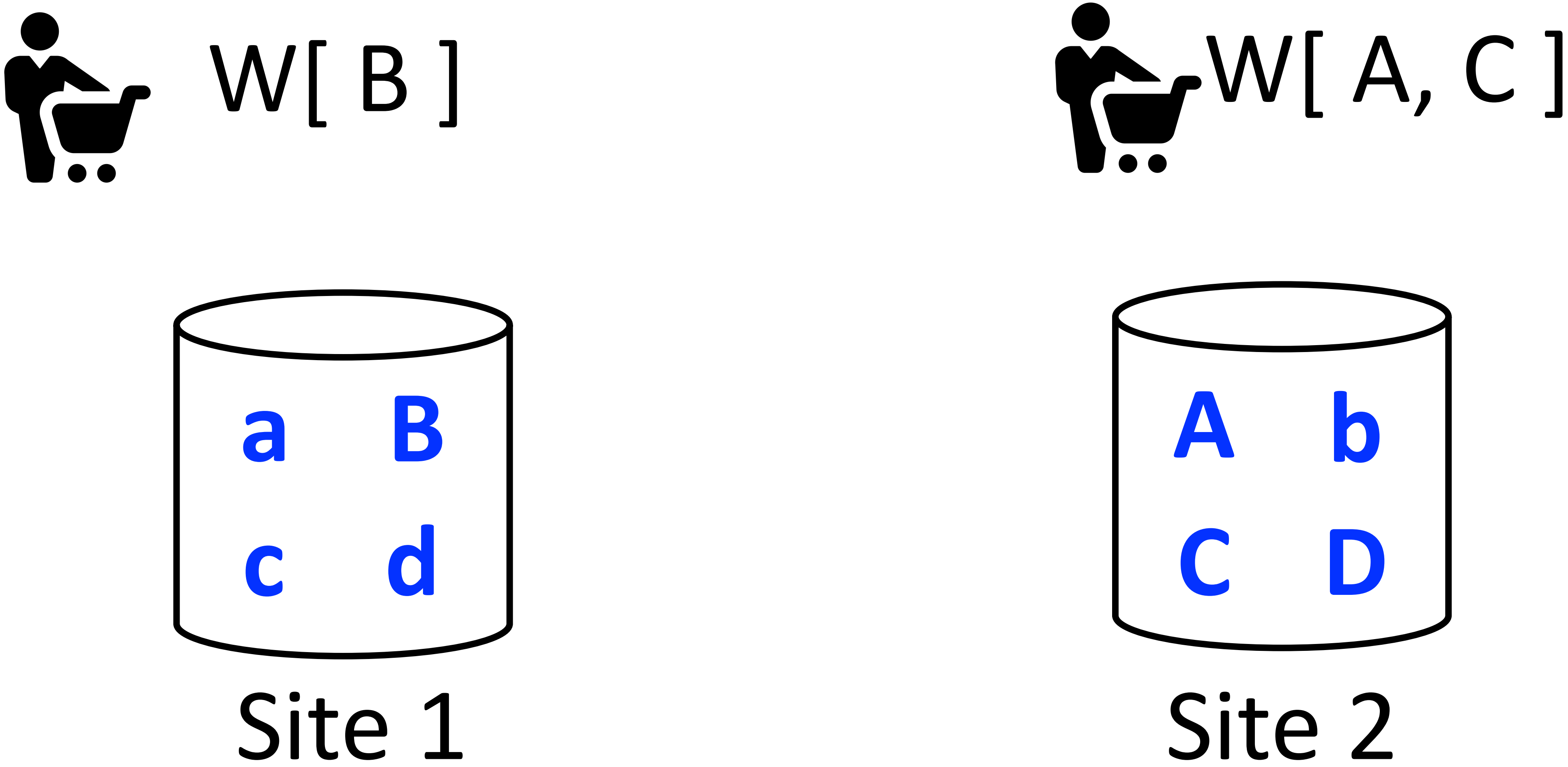


Dynamic Mastering



Single-site transactions

Dynamic Mastering



W[B]

W[A, C]

a B

c d

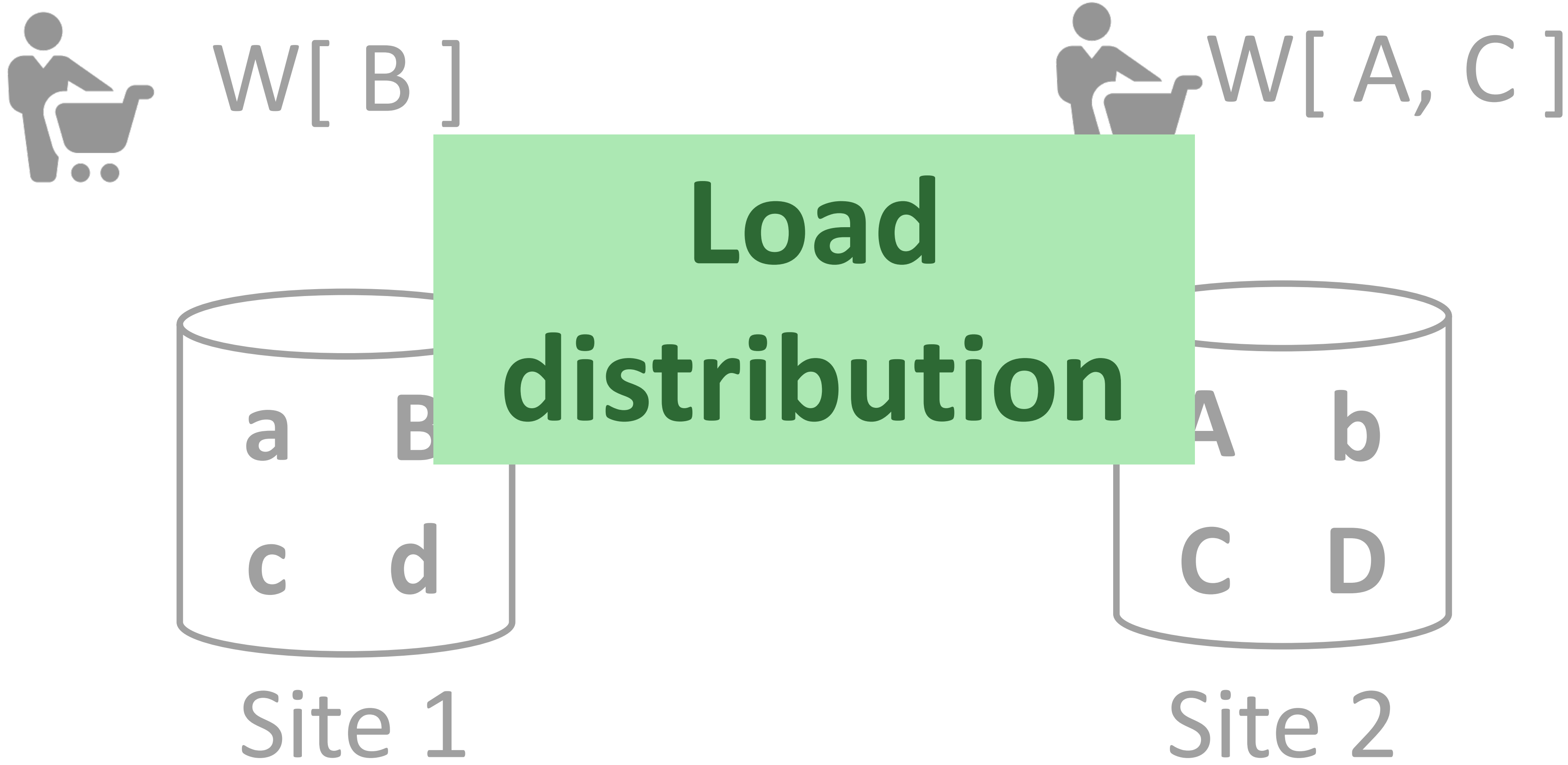
A b

C D

Site 1

Site 2

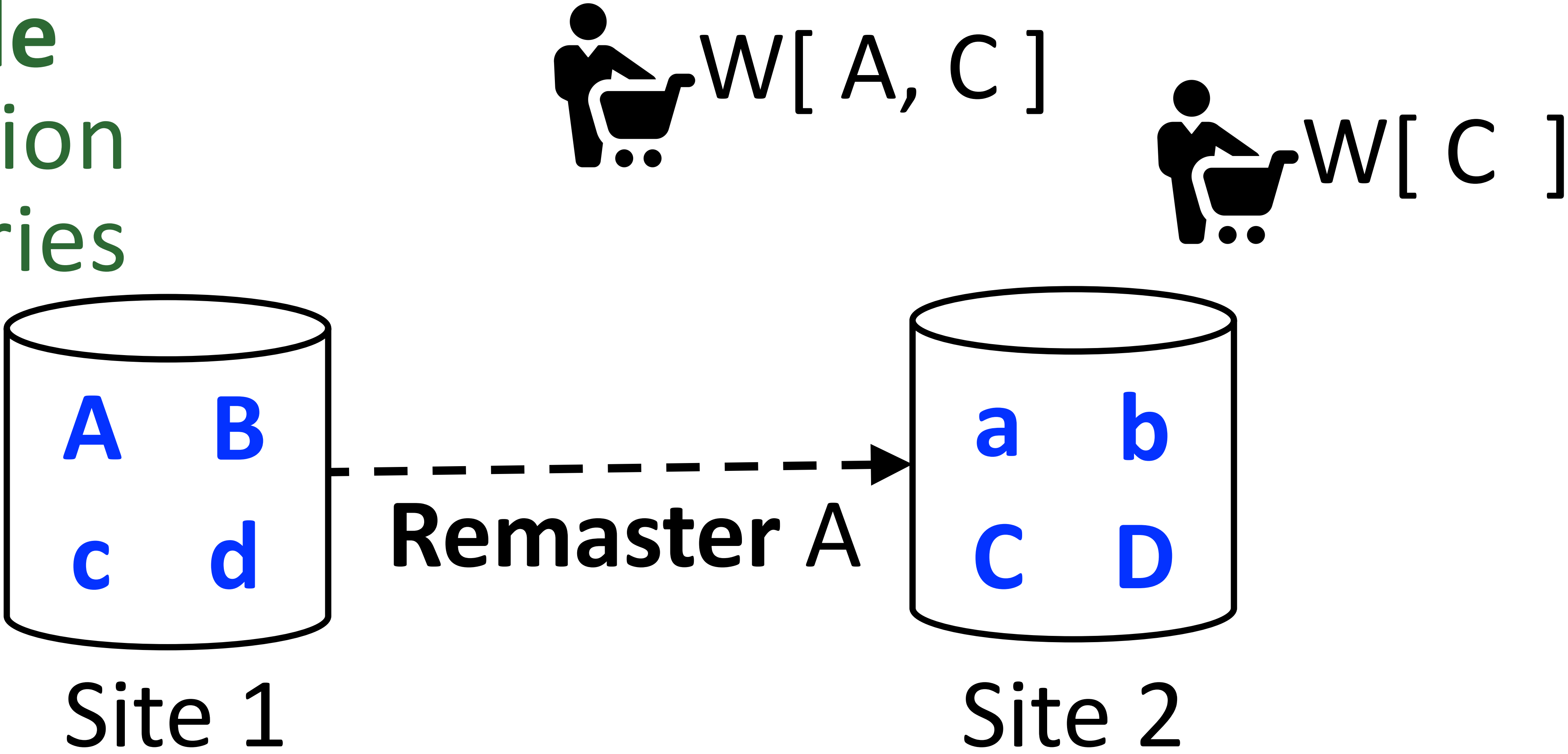
Dynamic Mastering



Load distribution

Dynamic Mastering

Outside transaction boundaries



DynaMast

Distributed and replicated database system

Employs ***adaptive*** **dynamic mastering**

Provides both **single-site transactions** and **load balance**

Dynamic Mastering Challenges

How to perform remastering efficiently?

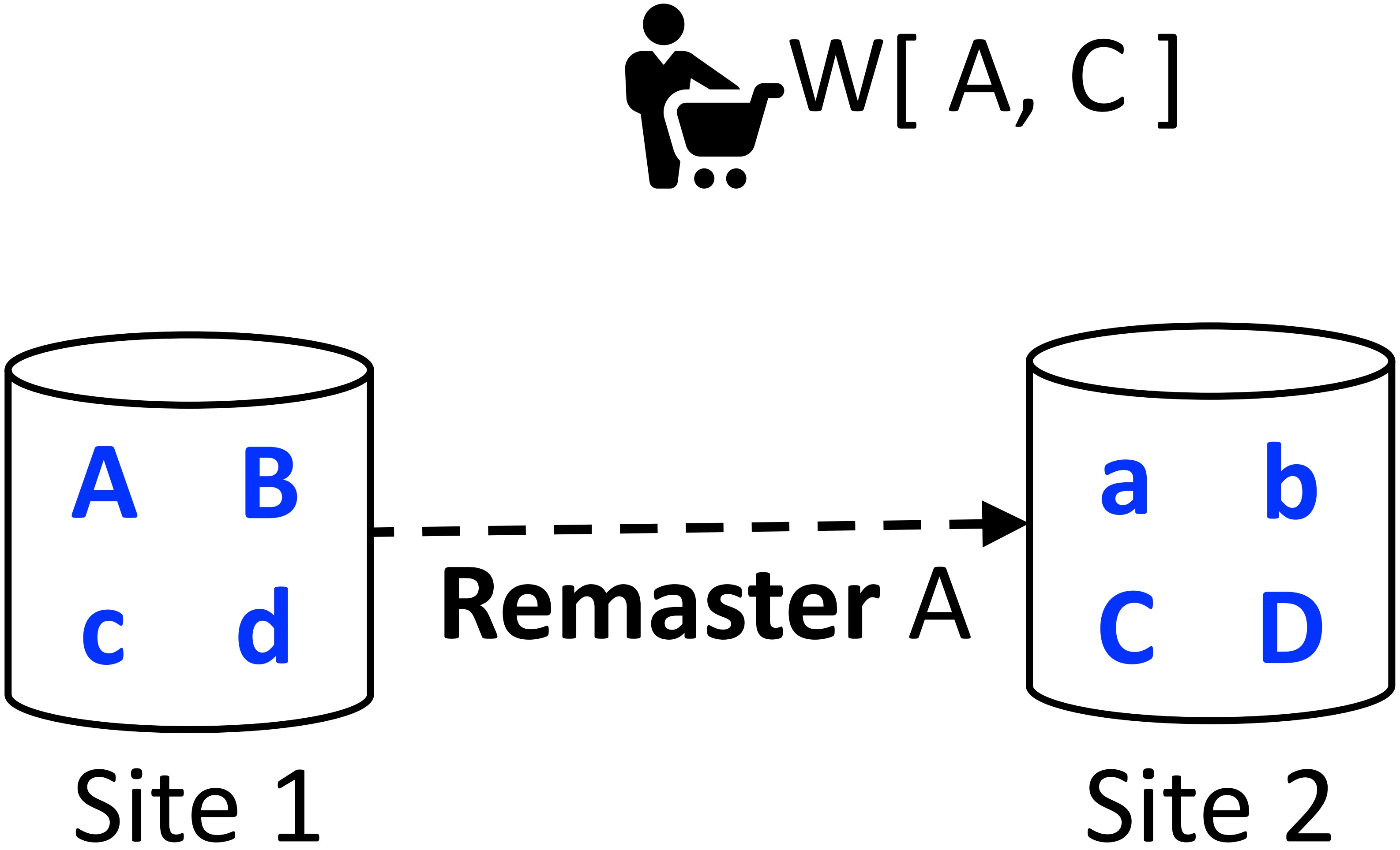
How to decide where to master data?

Dynamic Mastering Challenges

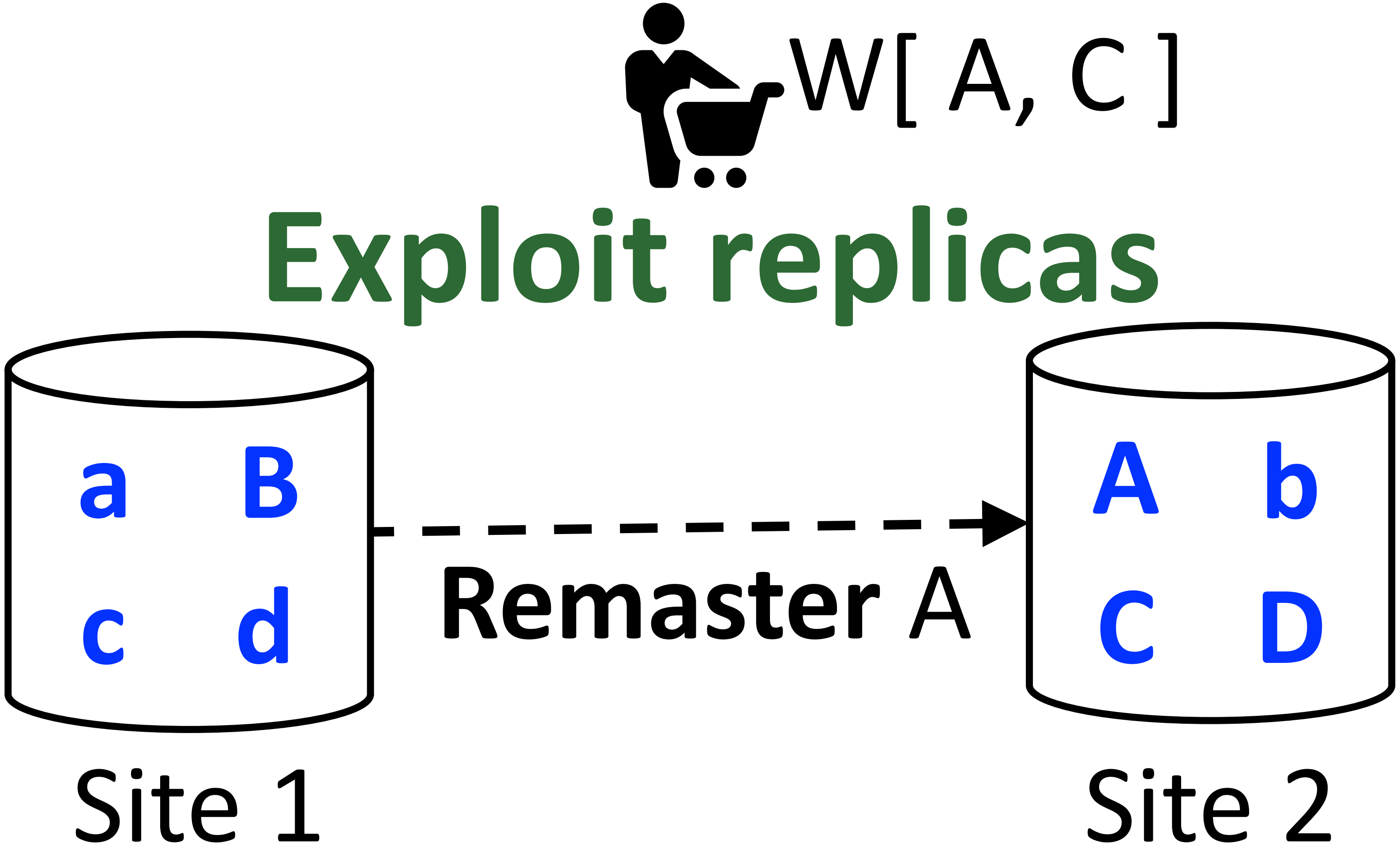
How to perform remastering efficiently?

How to decide where to master data?

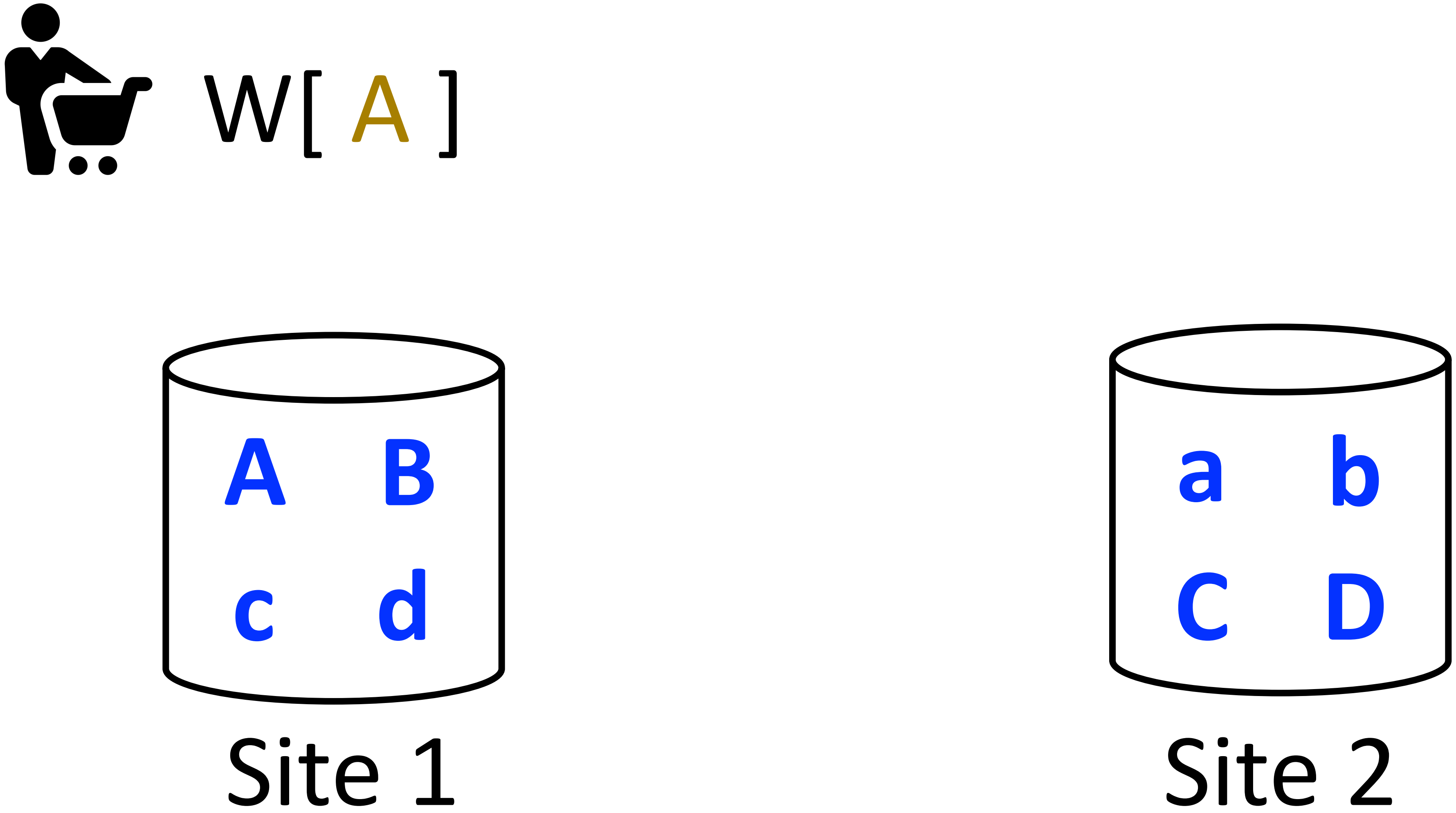
Dynamic Mastering



Dynamic Mastering



Exploiting Replicas



W[A]

A B

c d

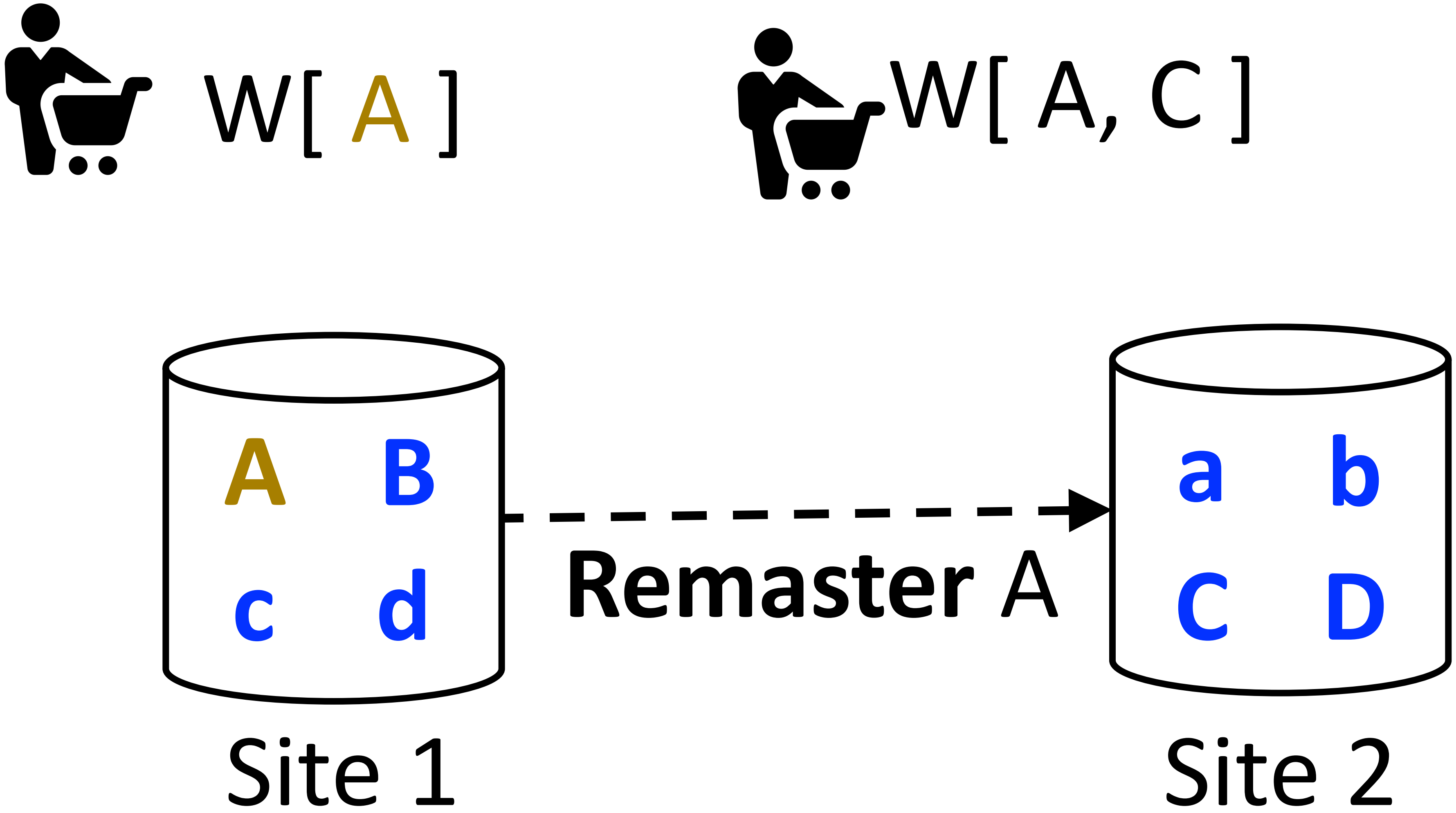
Site 1

a b

C D

Site 2

Exploiting Replicas



W[A]

W[A, C]

A B c d

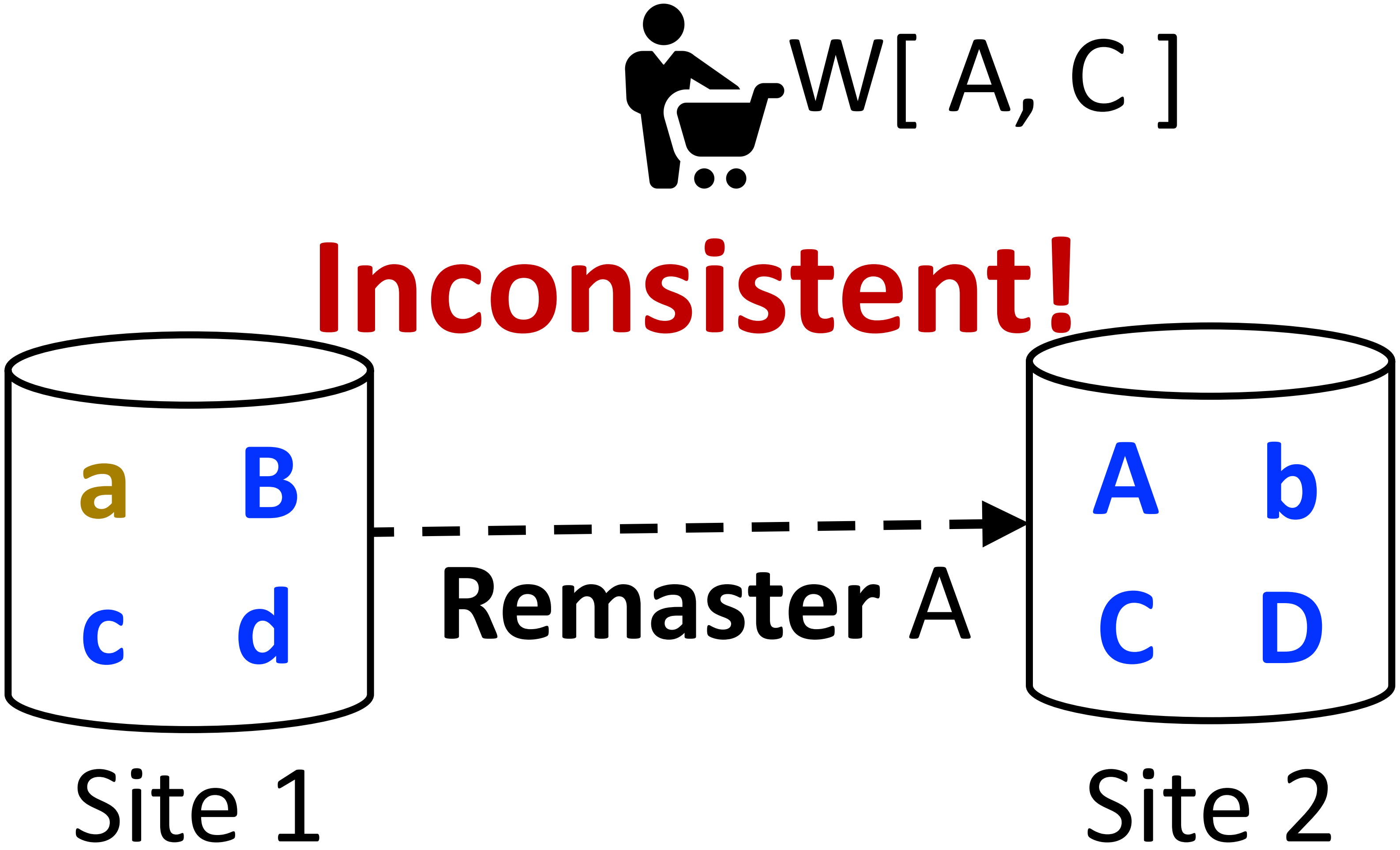
Site 1

Remaster A

a b C D

Site 2

Exploiting Replicas



Ensuring Consistency

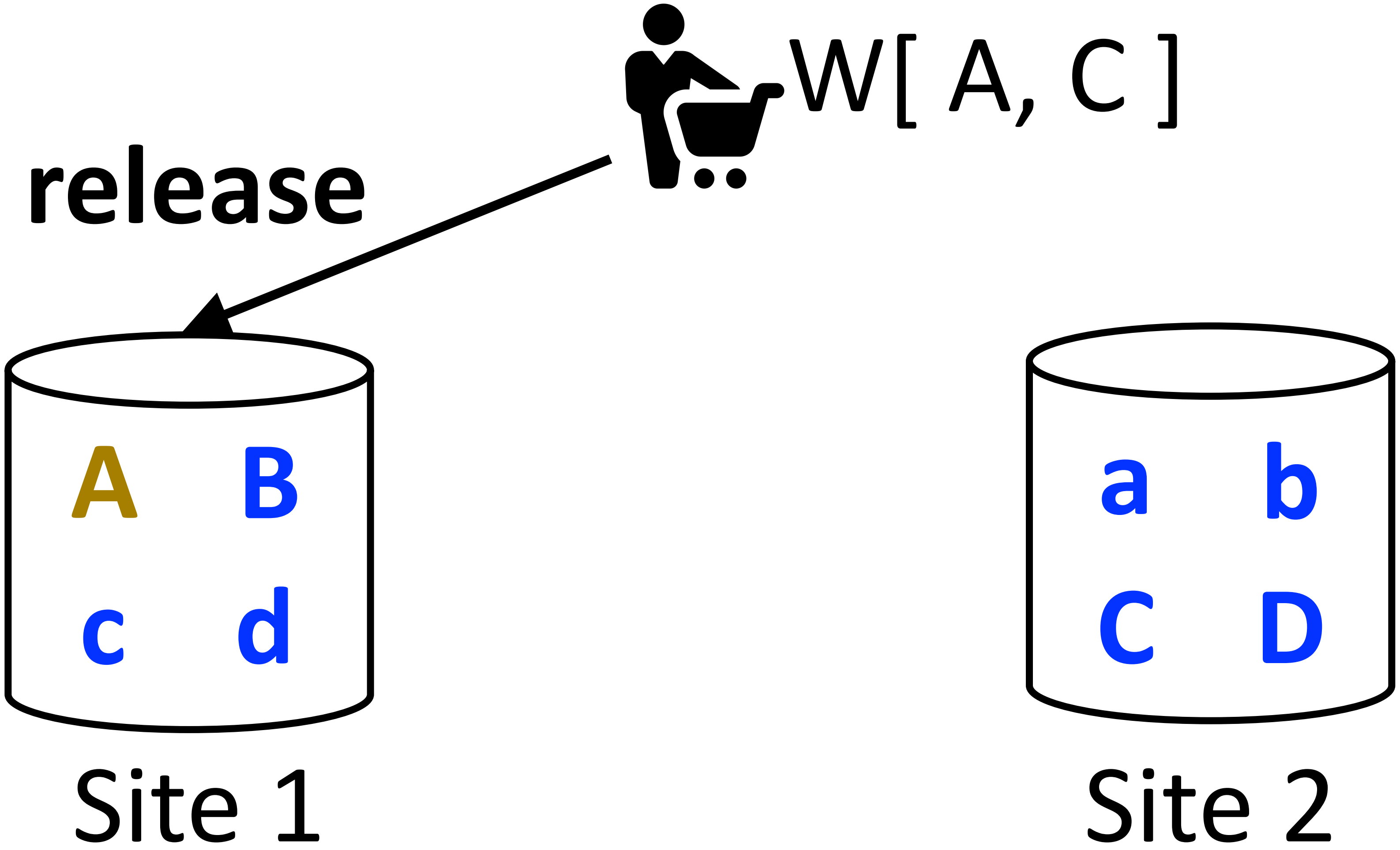
Old master must **not allow new updates**

release mastership

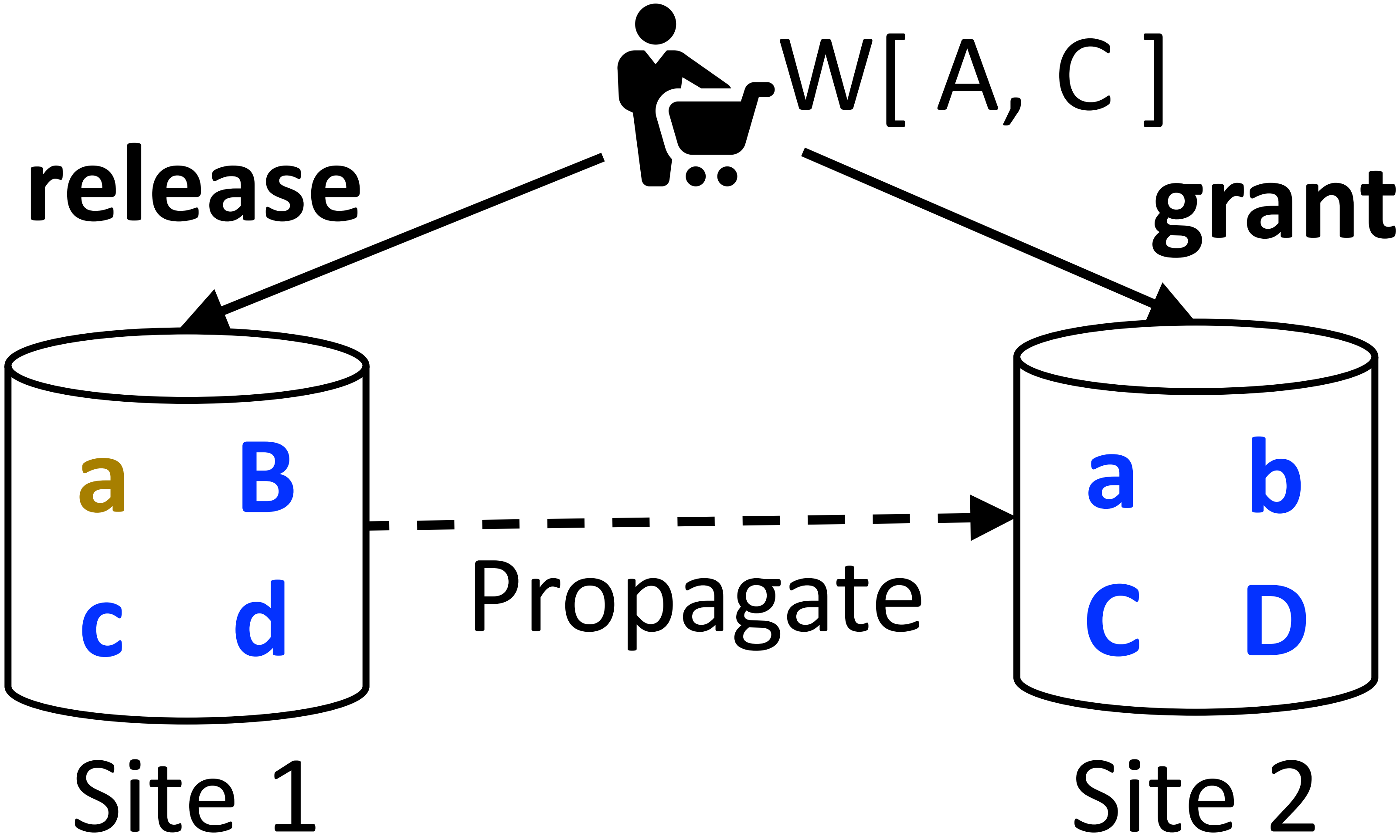
New master must have **all previous updates**

grant mastership

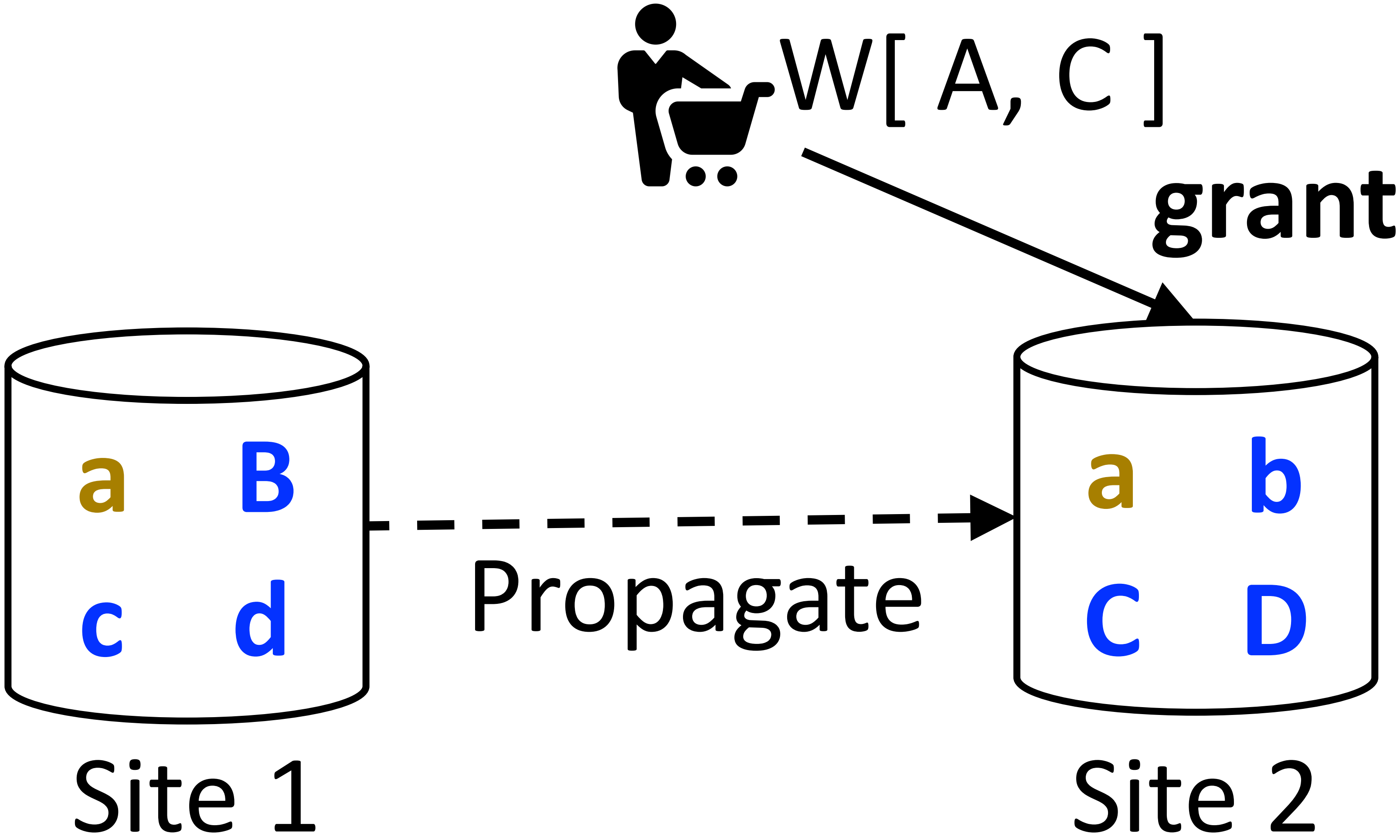
Ensuring Consistency



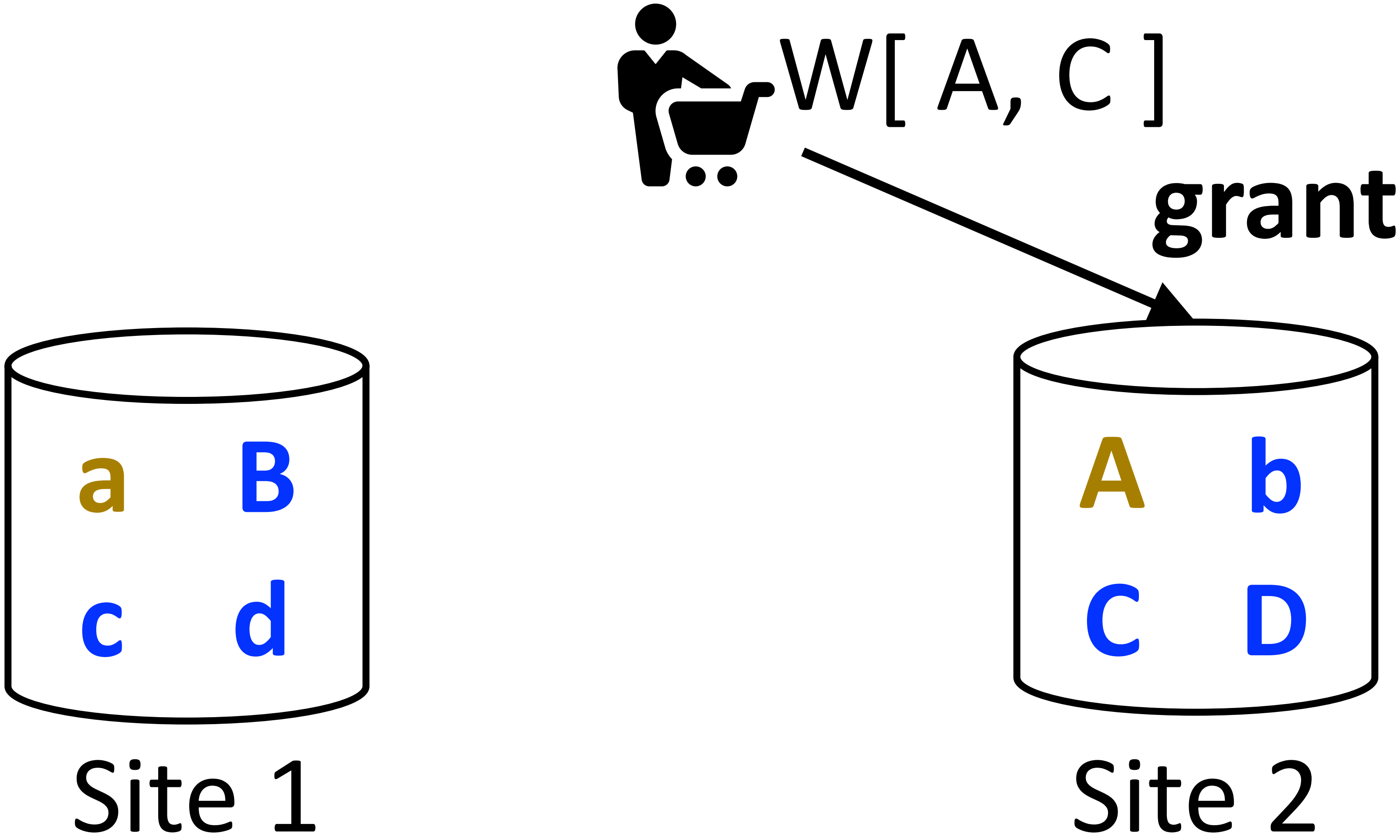
Ensuring Consistency



Ensuring Consistency



Ensuring Consistency



Ensuring Consistency

New master must have **all previous updates**

New master was **a lazy replica of old master**

Little time spent waiting for updates

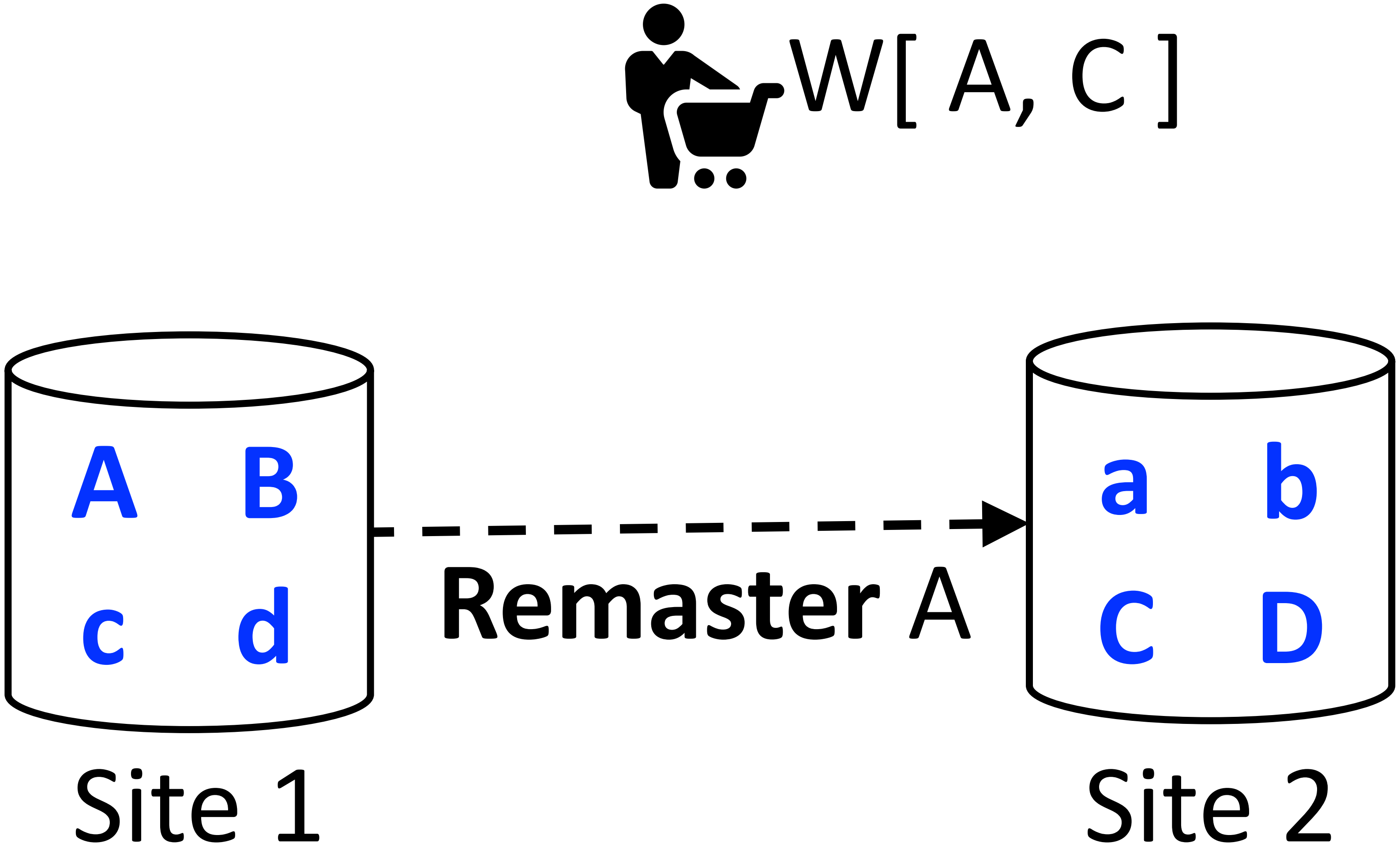
tiny.cc/dynamast

Dynamic Mastering Challenges

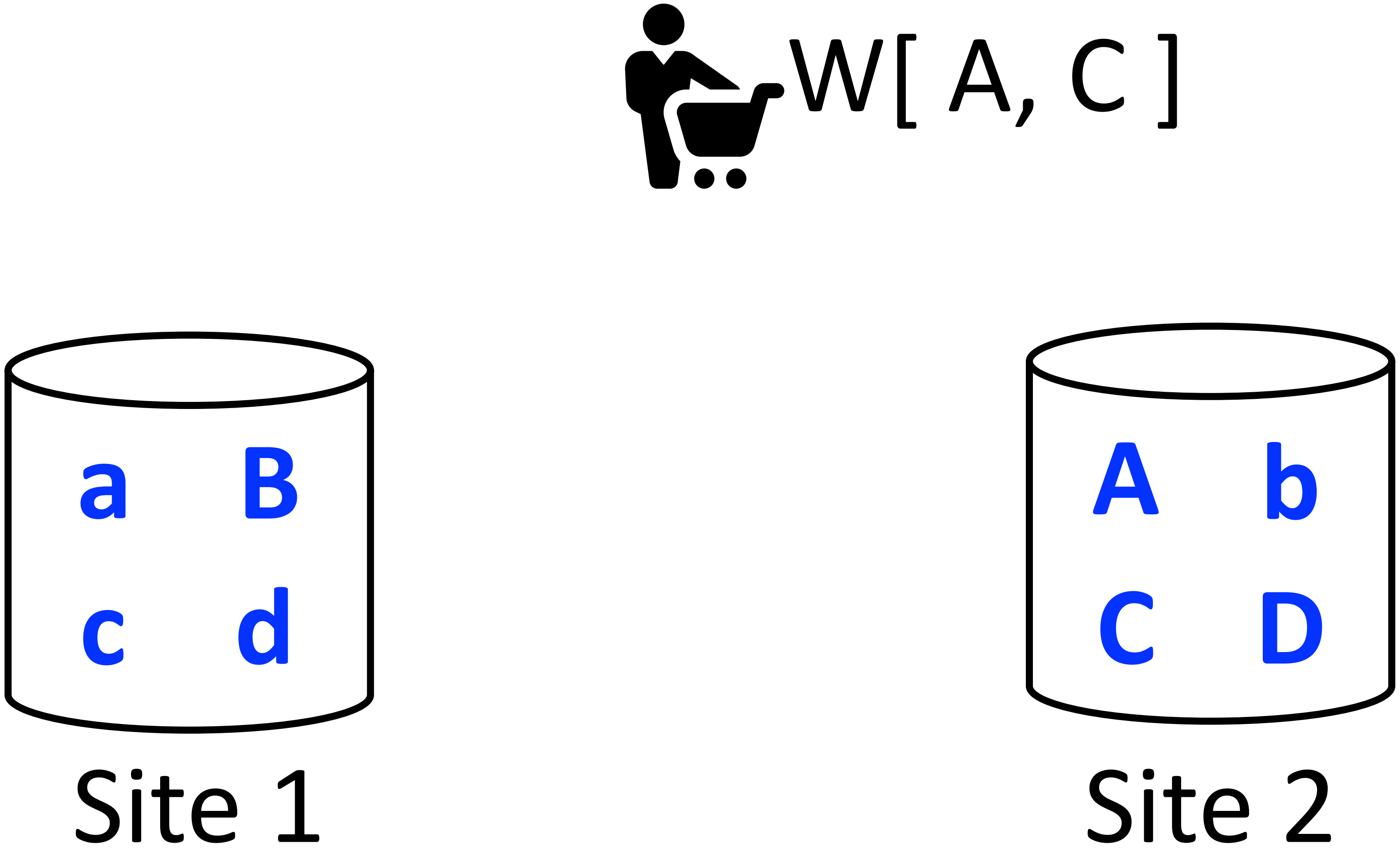
How to perform remastering efficiently?

How to decide where to master data?

Where to master?



Where to master?



W[A, C]

a B

c d

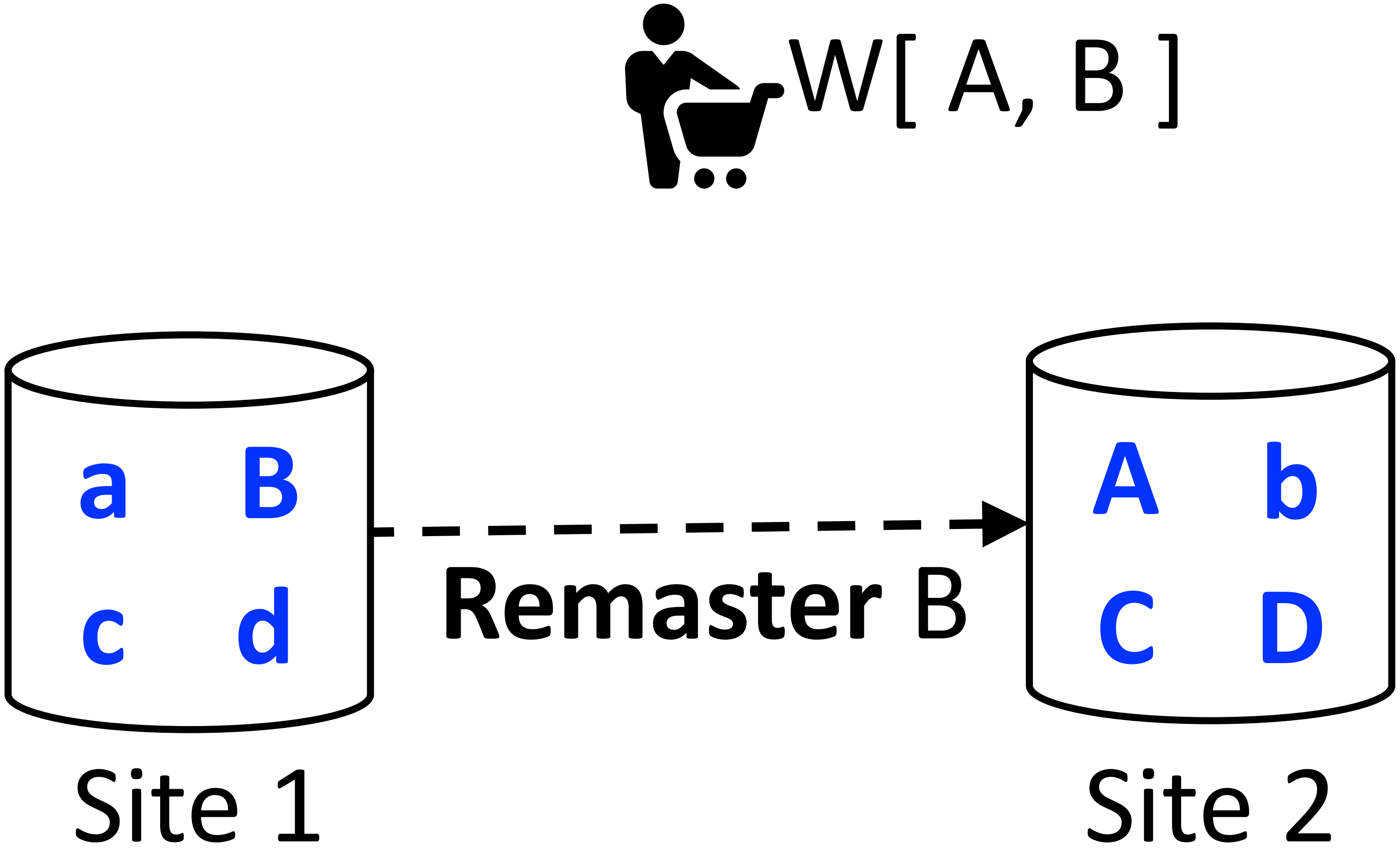
Site 1

A b

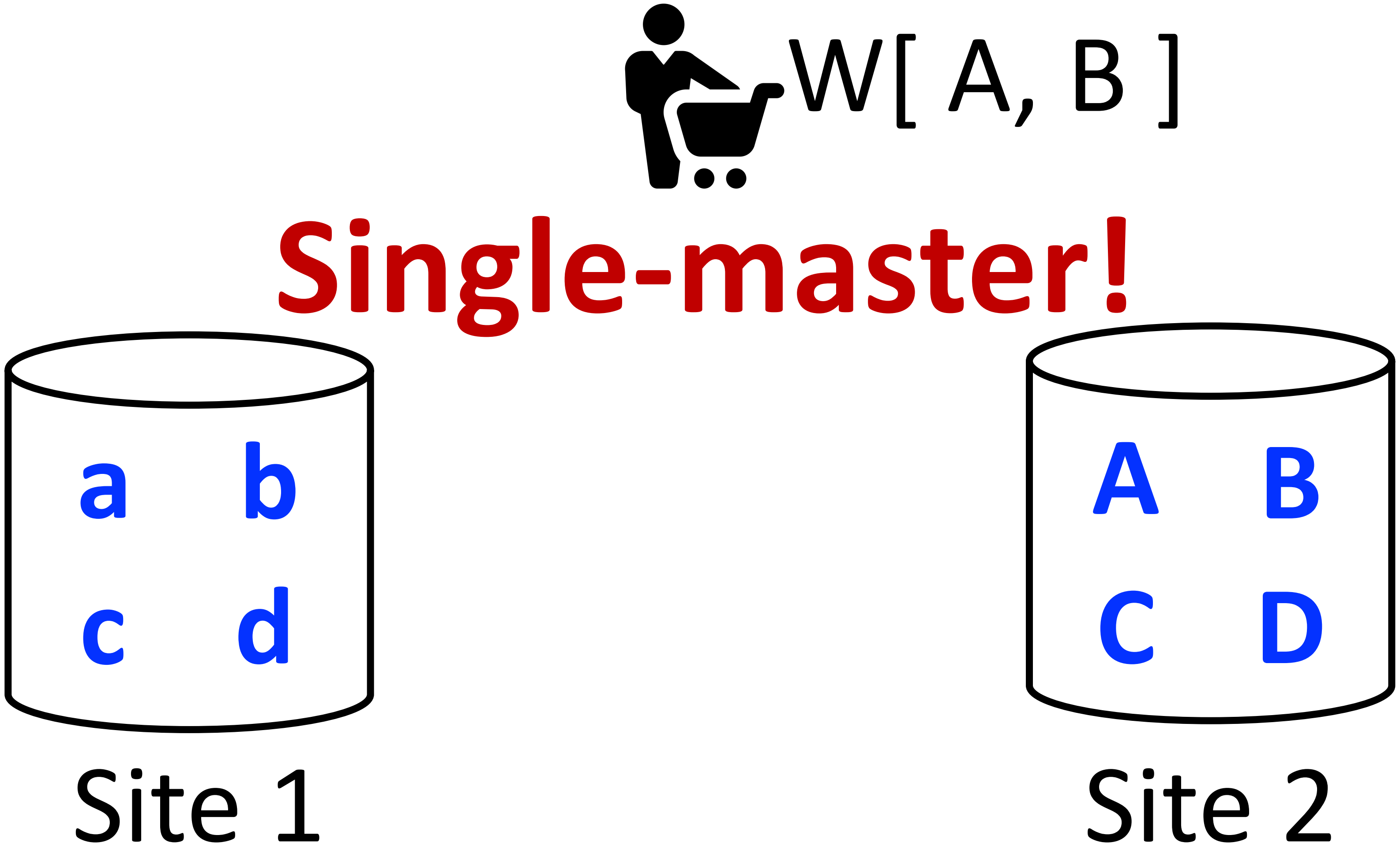
C D

Site 2

Where to master?

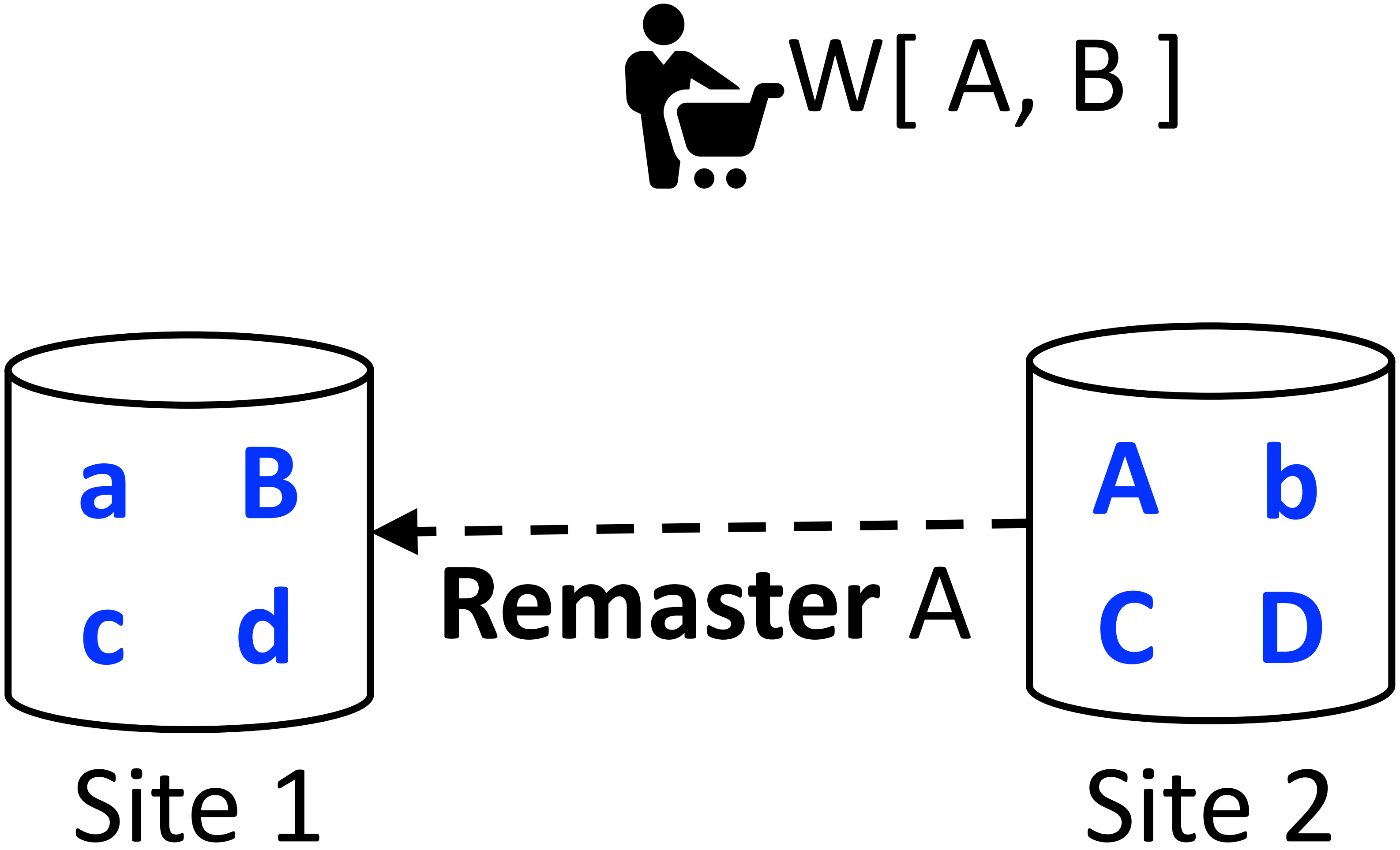


Where to master?

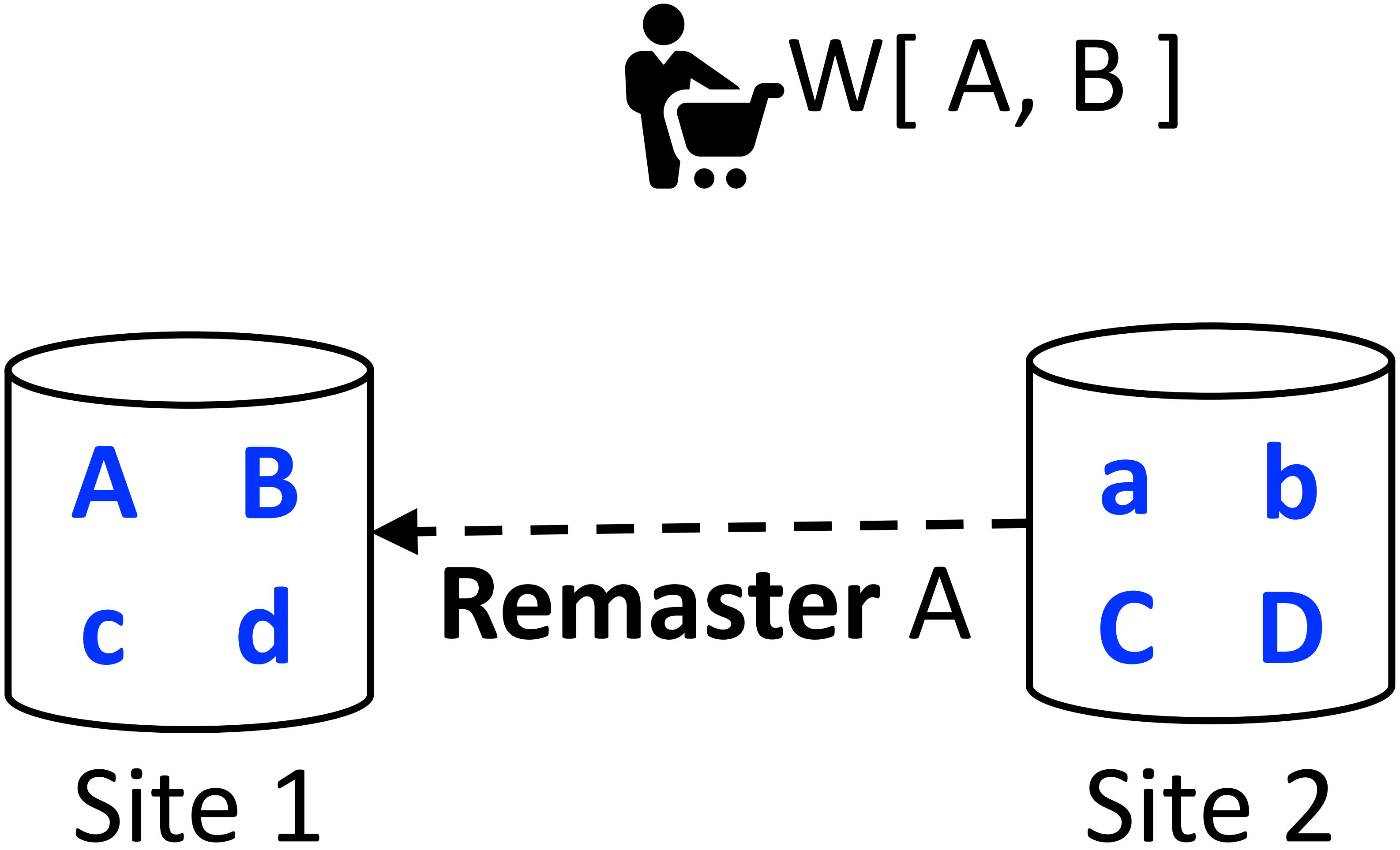


UNIVERSITY OF WATERLOO

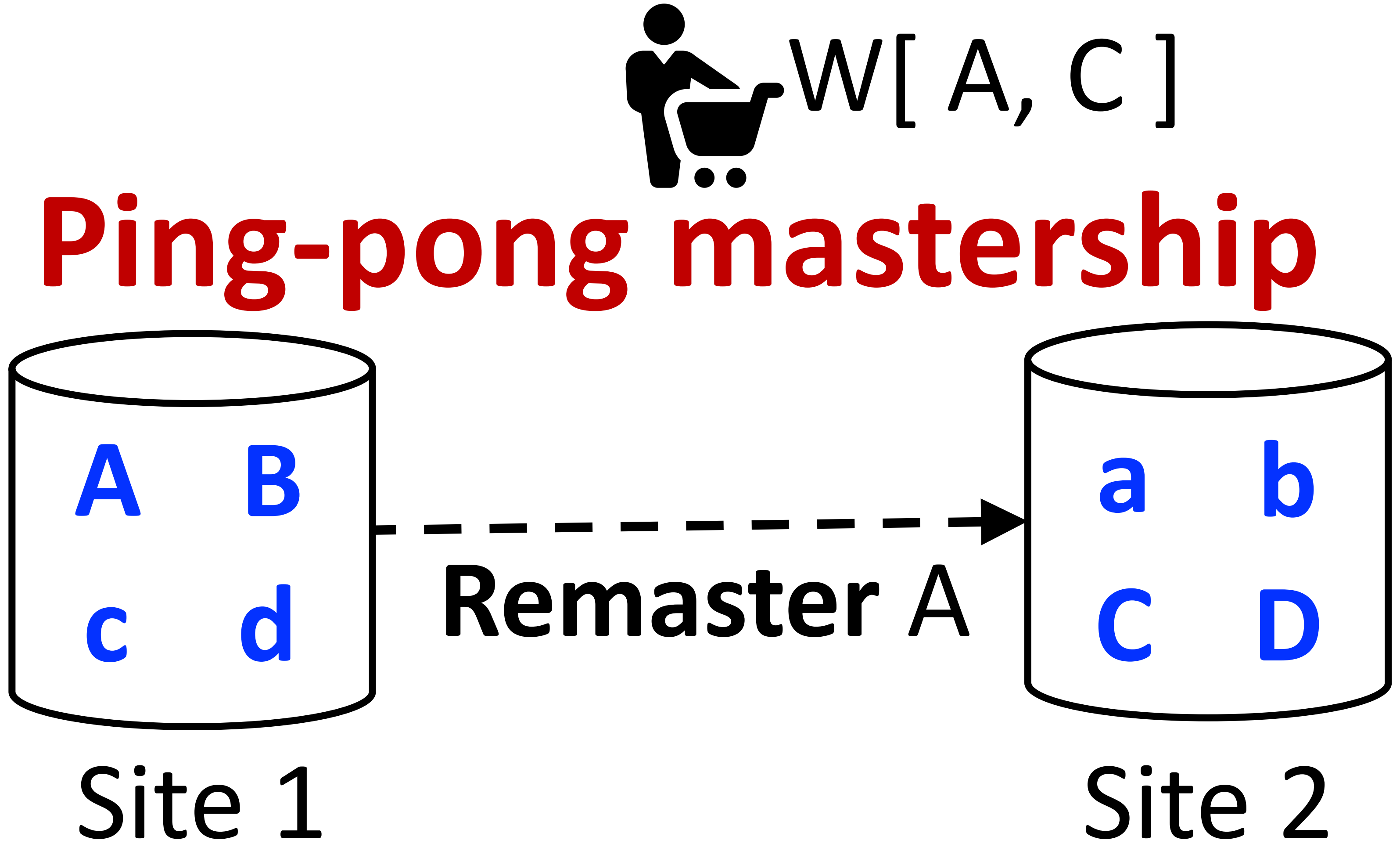
Where to master?



Where to master?



Where to master?



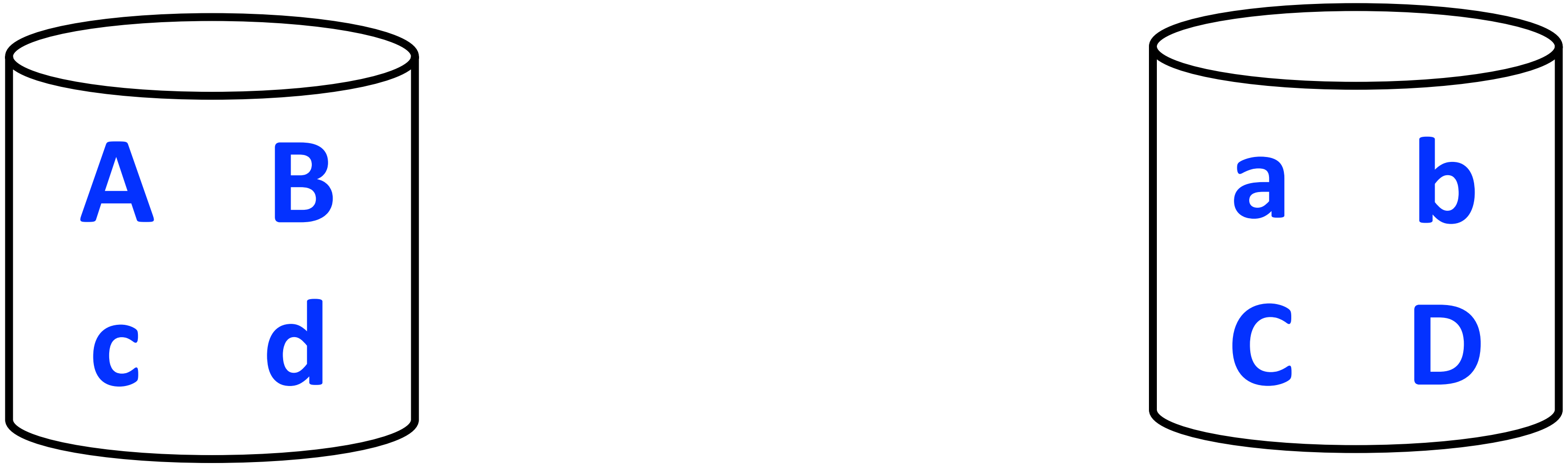
W[A, C]

Ping-pong mastership

Dynamic Mastering Strategies

Site Selector

Makes **adaptive** decisions



Site 1

Site 2

Dynamic Mastering Strategies

Track: data access patterns, site load

Quantify **benefit** of remastering to each site

Load distribution

Future remastering

Update propagation

Remaster to site that **maximizes benefit**

tiny.cc/dynamast

How well does it work?

Workloads

YCSB Scans & Multi-Key Read-Modify-Write

Uniform and Skew Access Patterns

TPC-C Complex updates and reads

How well does it work?

Comparisons

Single-Master

Multi-Master

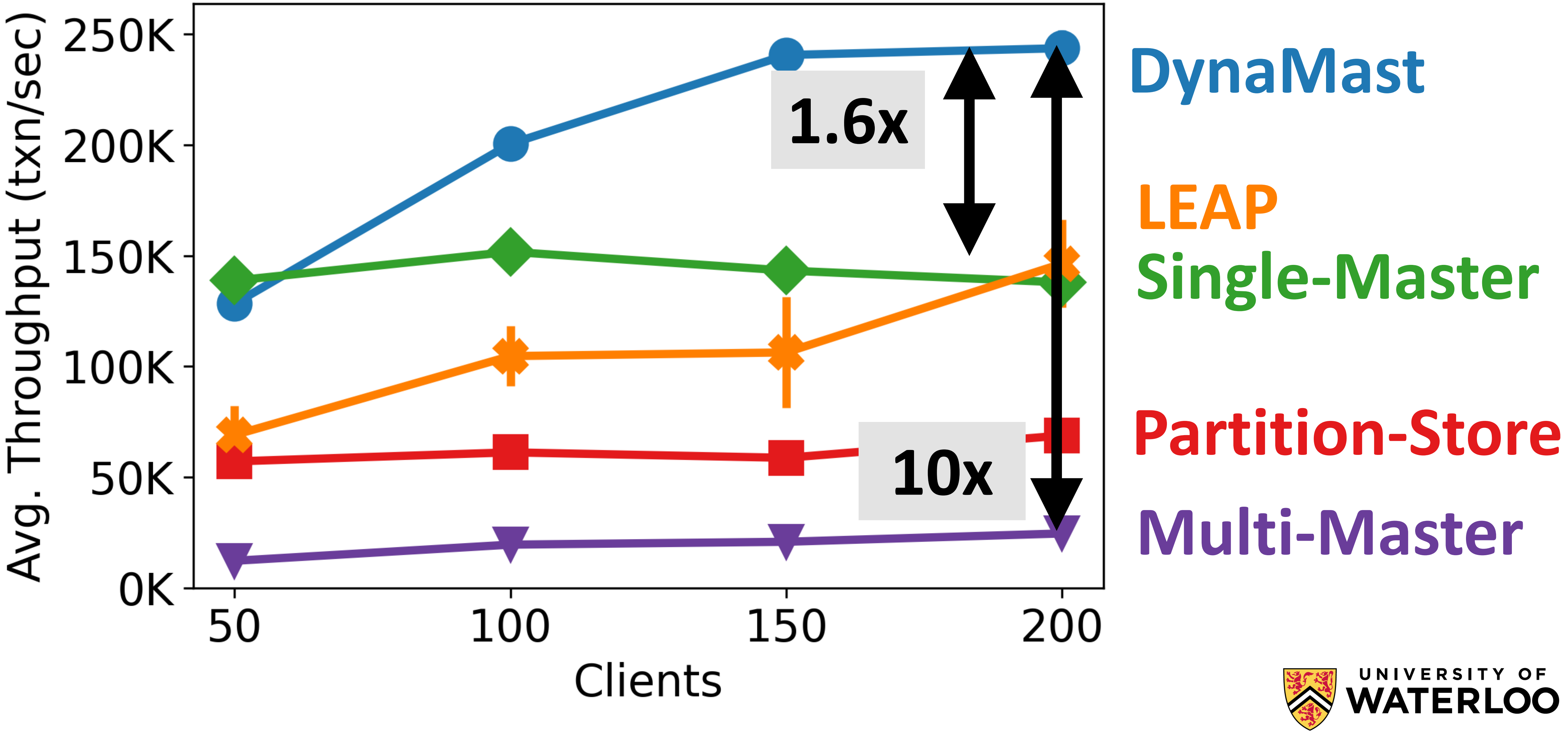
} *Replicated*

Partition-Store

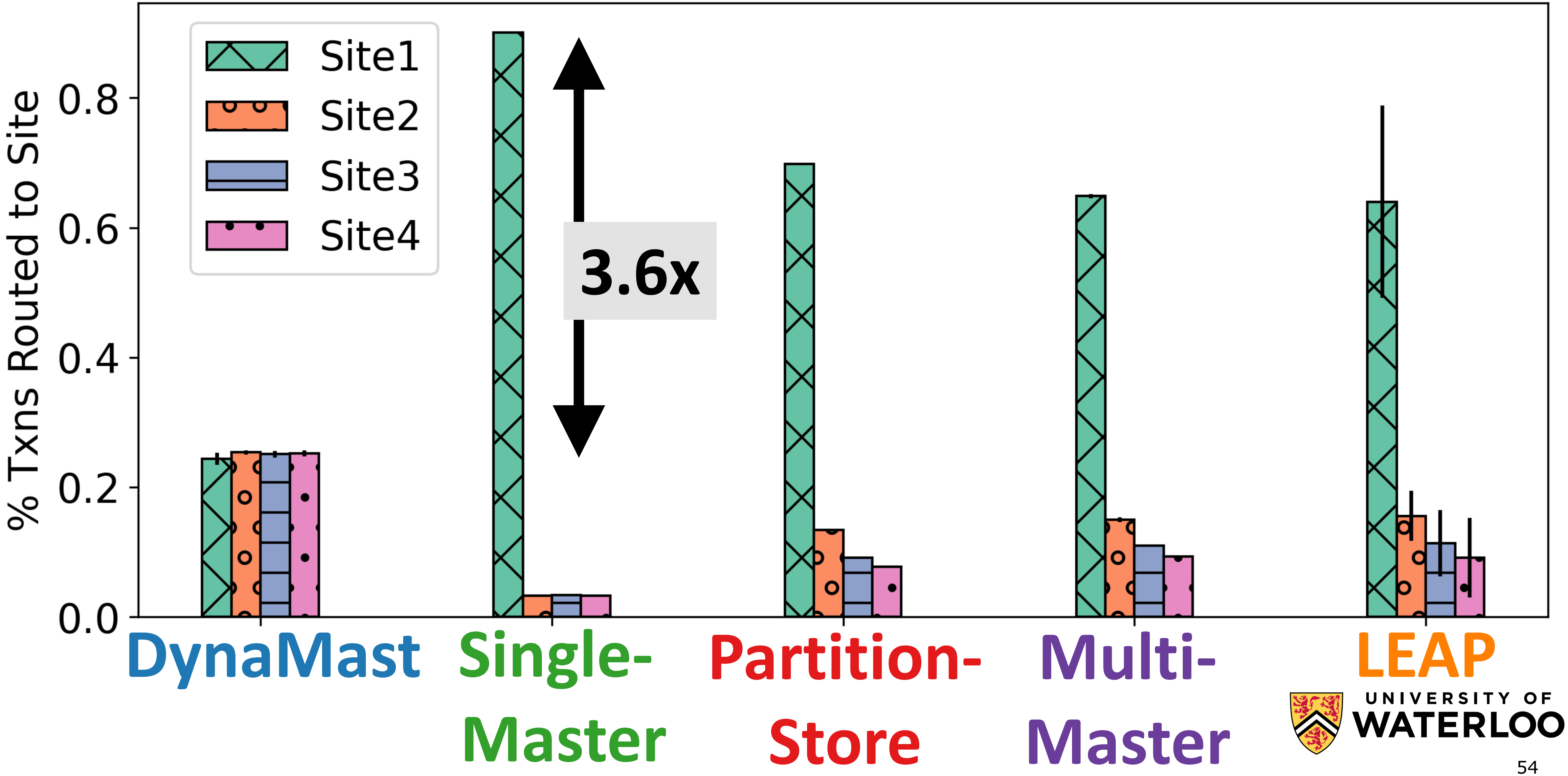
LEAP

} *Single Copy*

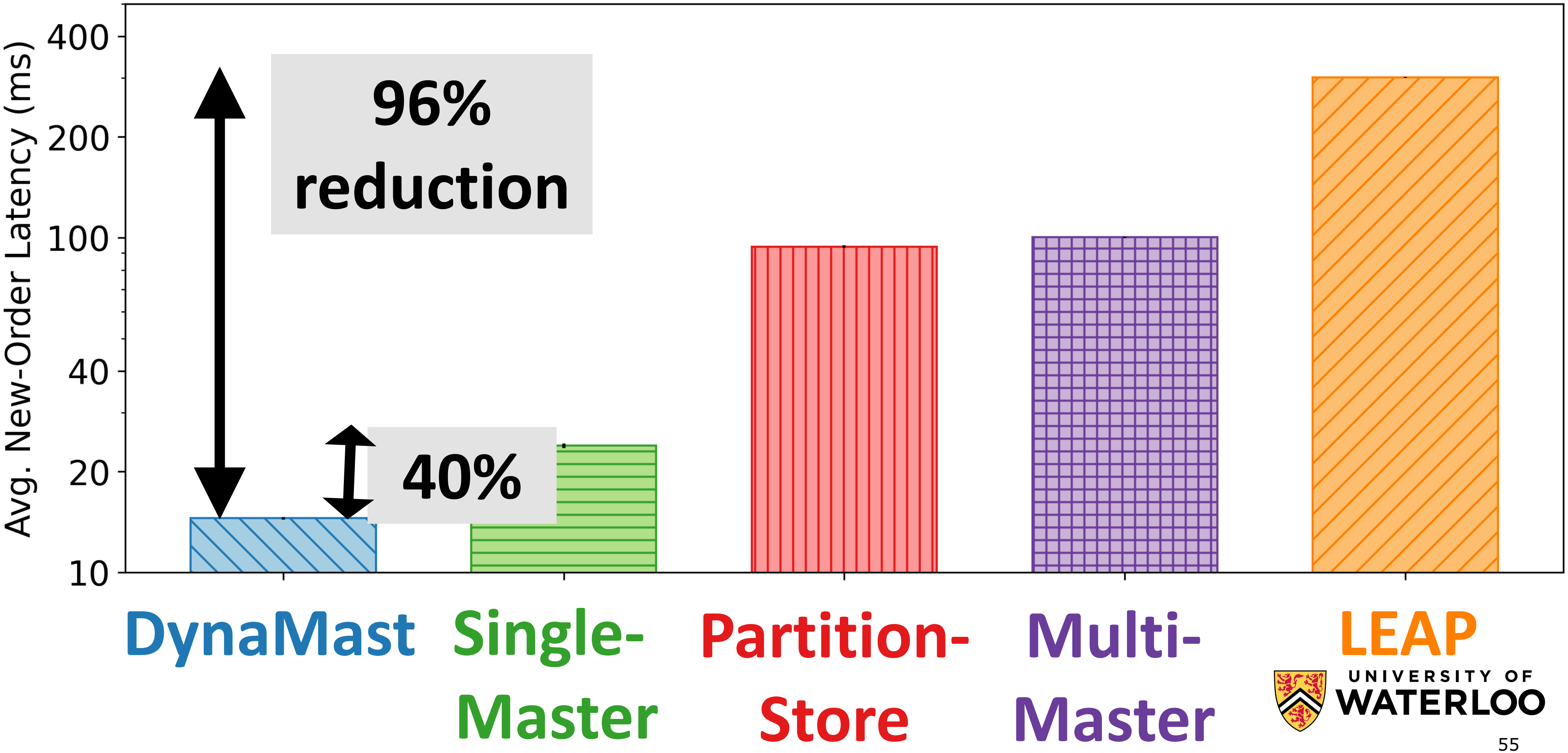
YCSB with Skew - Throughput



YCSB with Skew - Routing



TPC-C – New-Order Latency



DynaMast Takeaways

Dynamic mastering guarantees **single-site transactions**

tiny.cc/dynamast

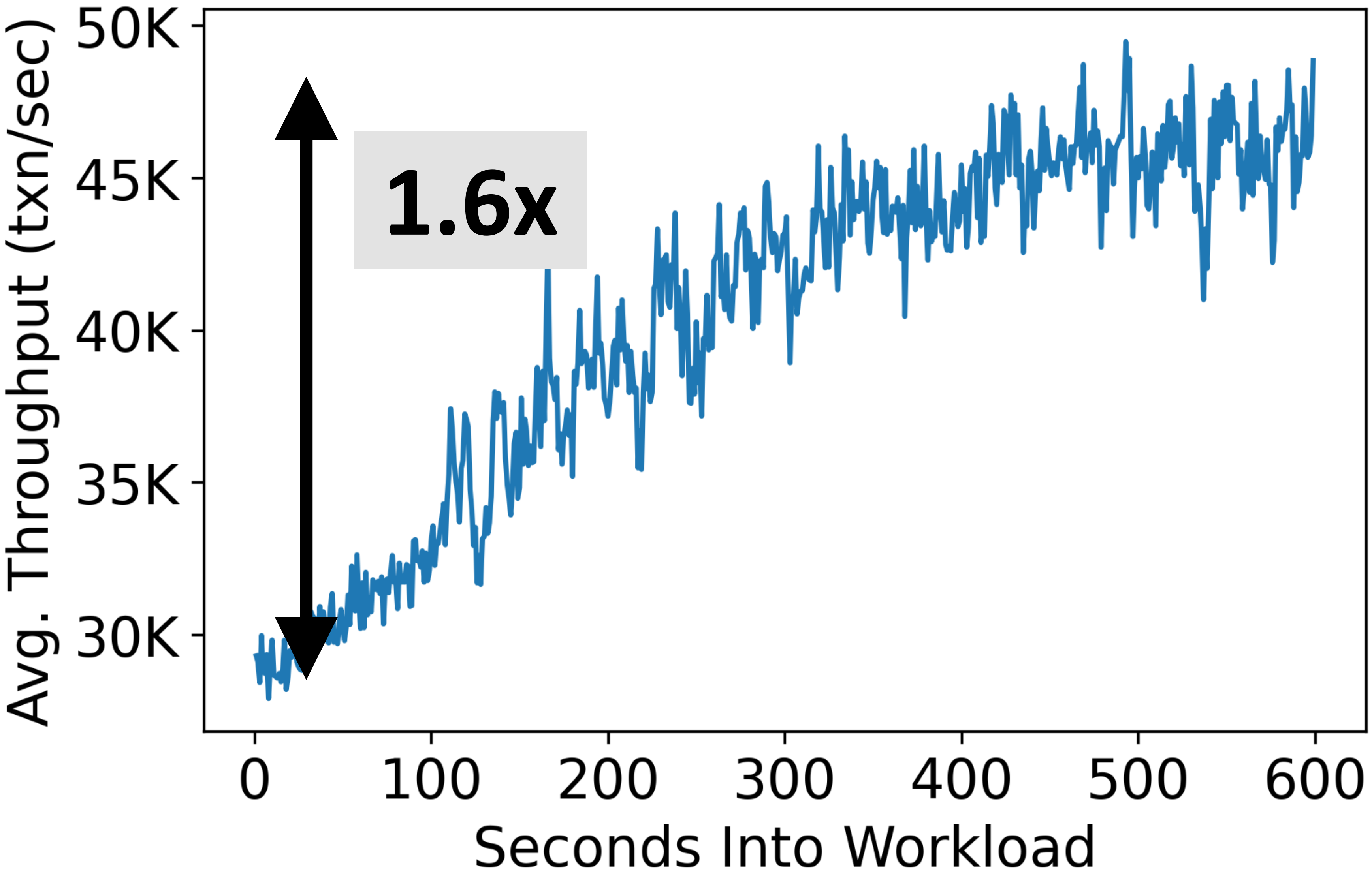
Use replicas to remaster **efficiently** **outside transaction boundaries**

Adaptive site selector strategies **balance load** and **minimizes future remastering**

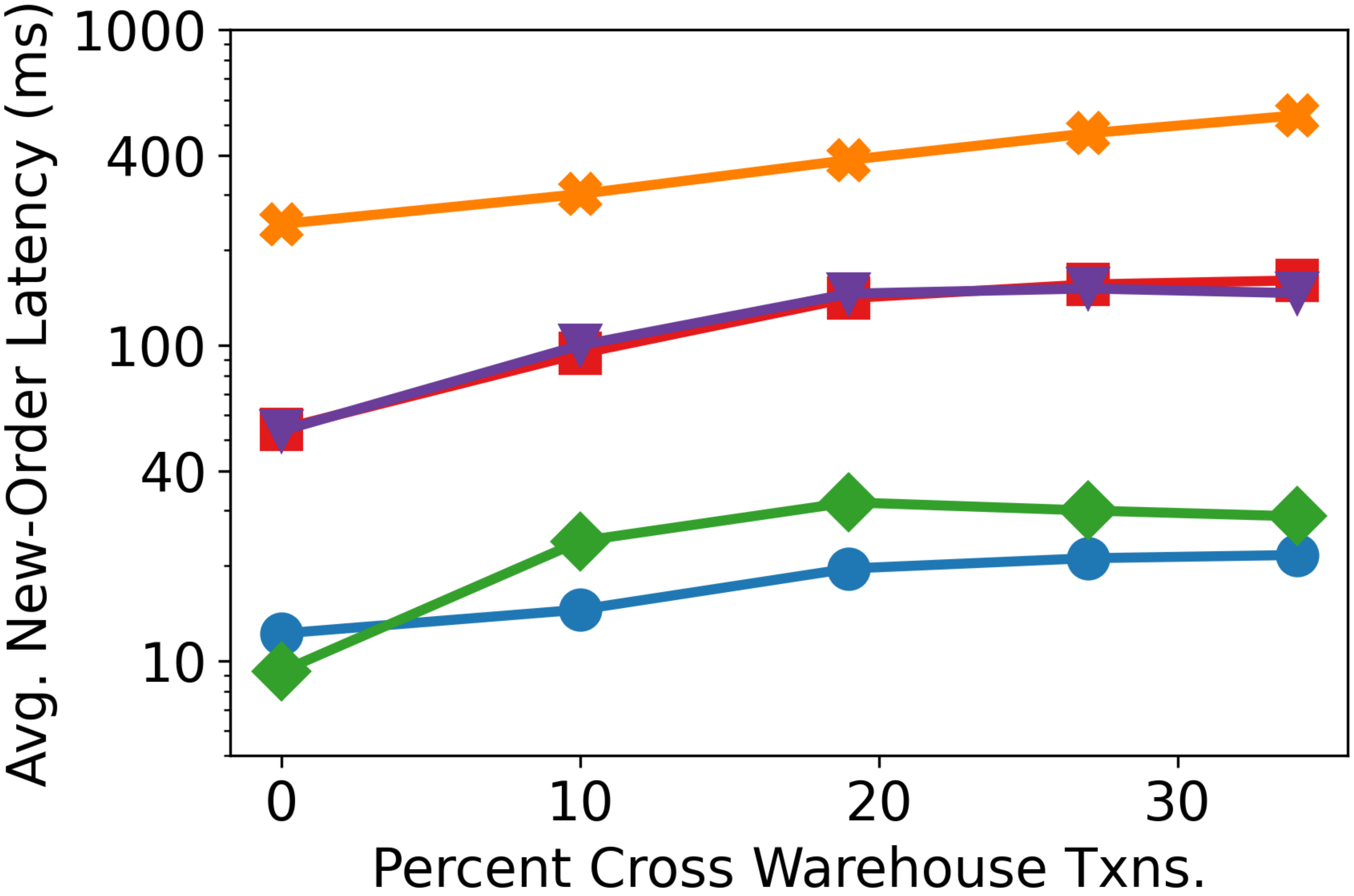
UNIVERSITY OF WATERLOO

Extra slides

DynaMast Learns Over Time



TPC-C – New-Order Latency



LEAP

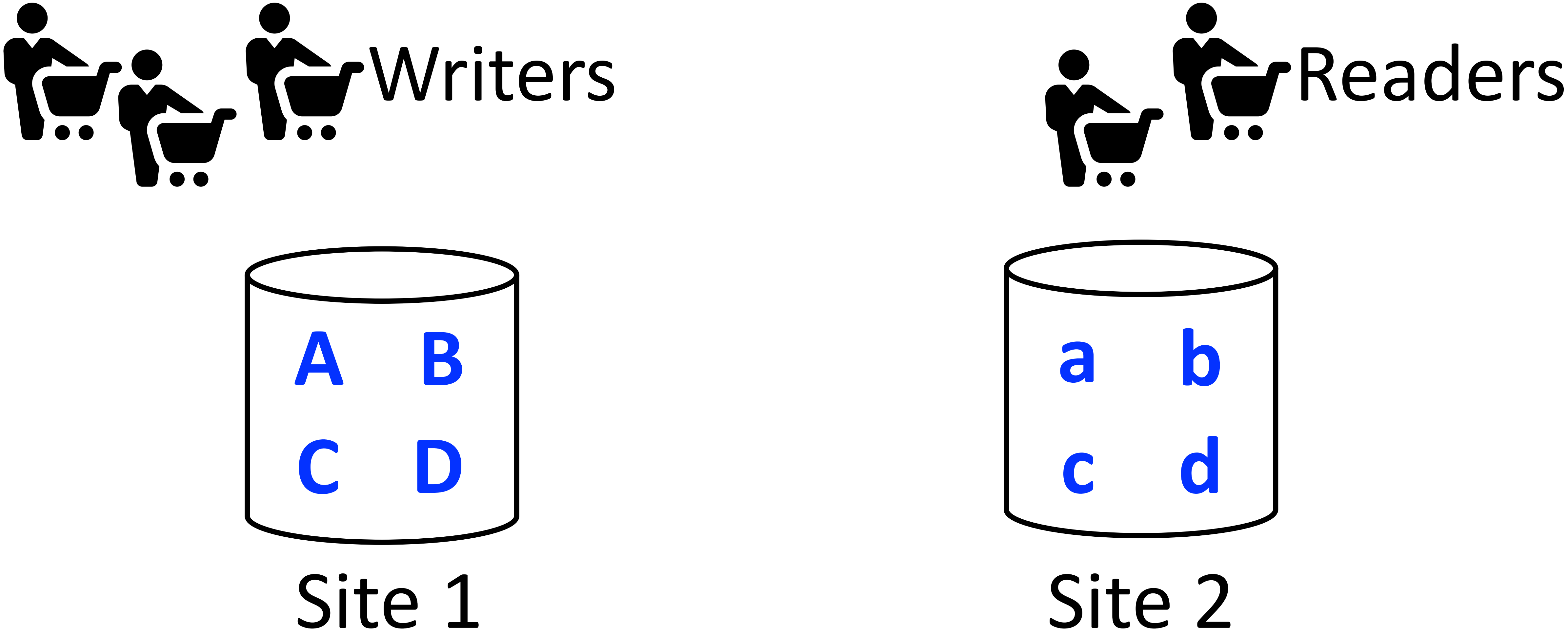
Partition-Store

Multi-Master

Single-Master

DynaMast

Comparisons – Single-Master



Writers

Readers

A B

C D

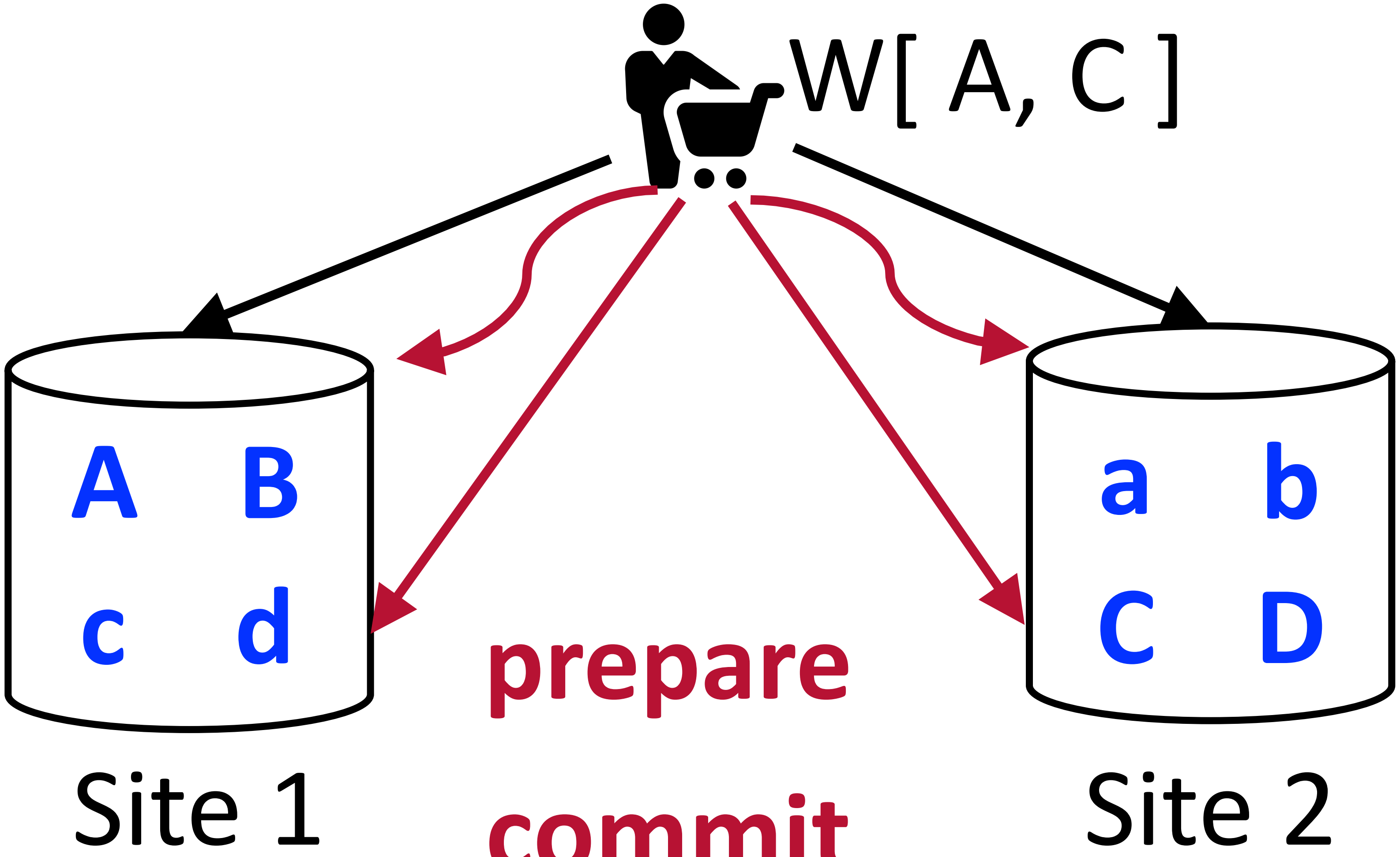
a b

c d

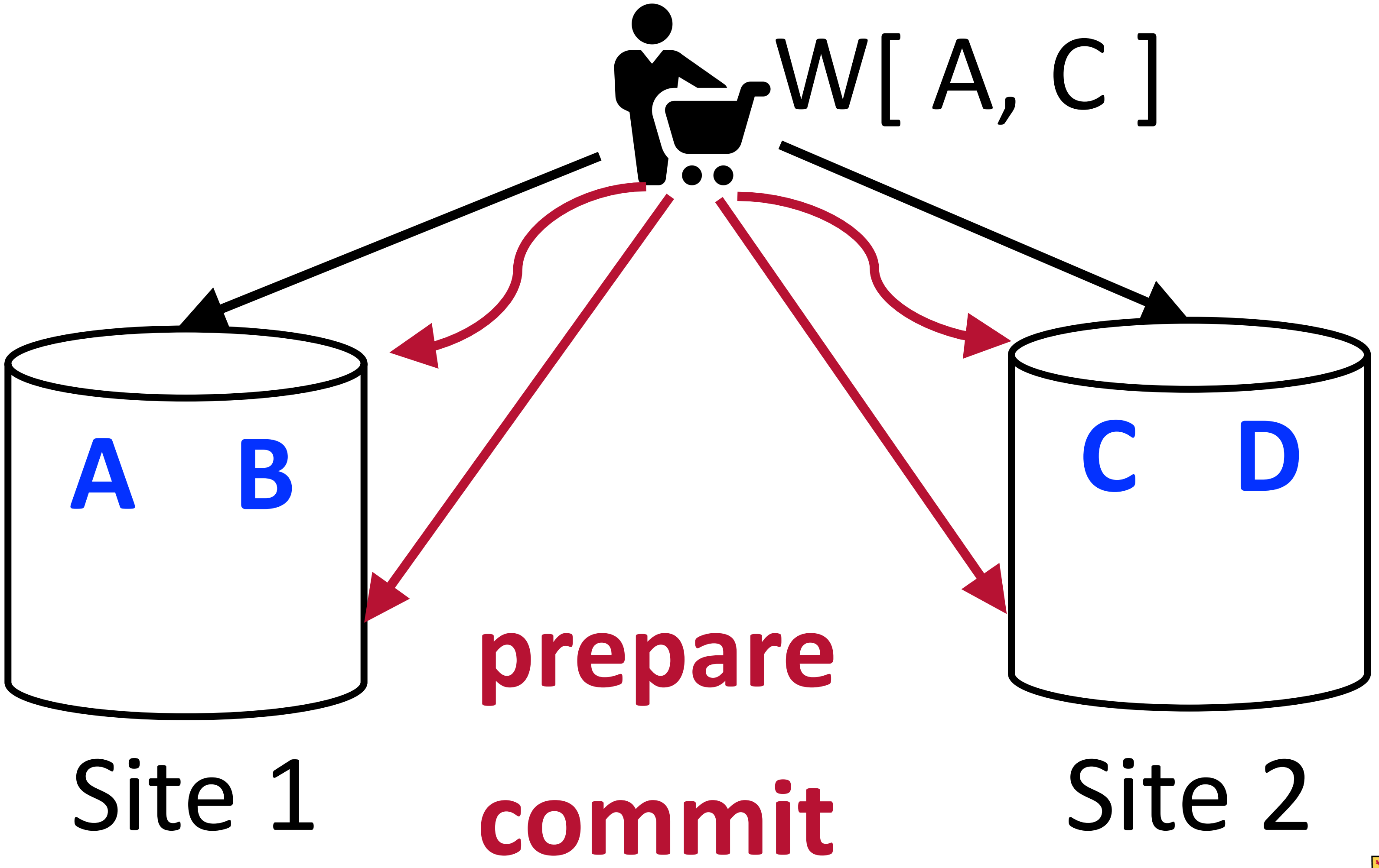
Site 1

Site 2

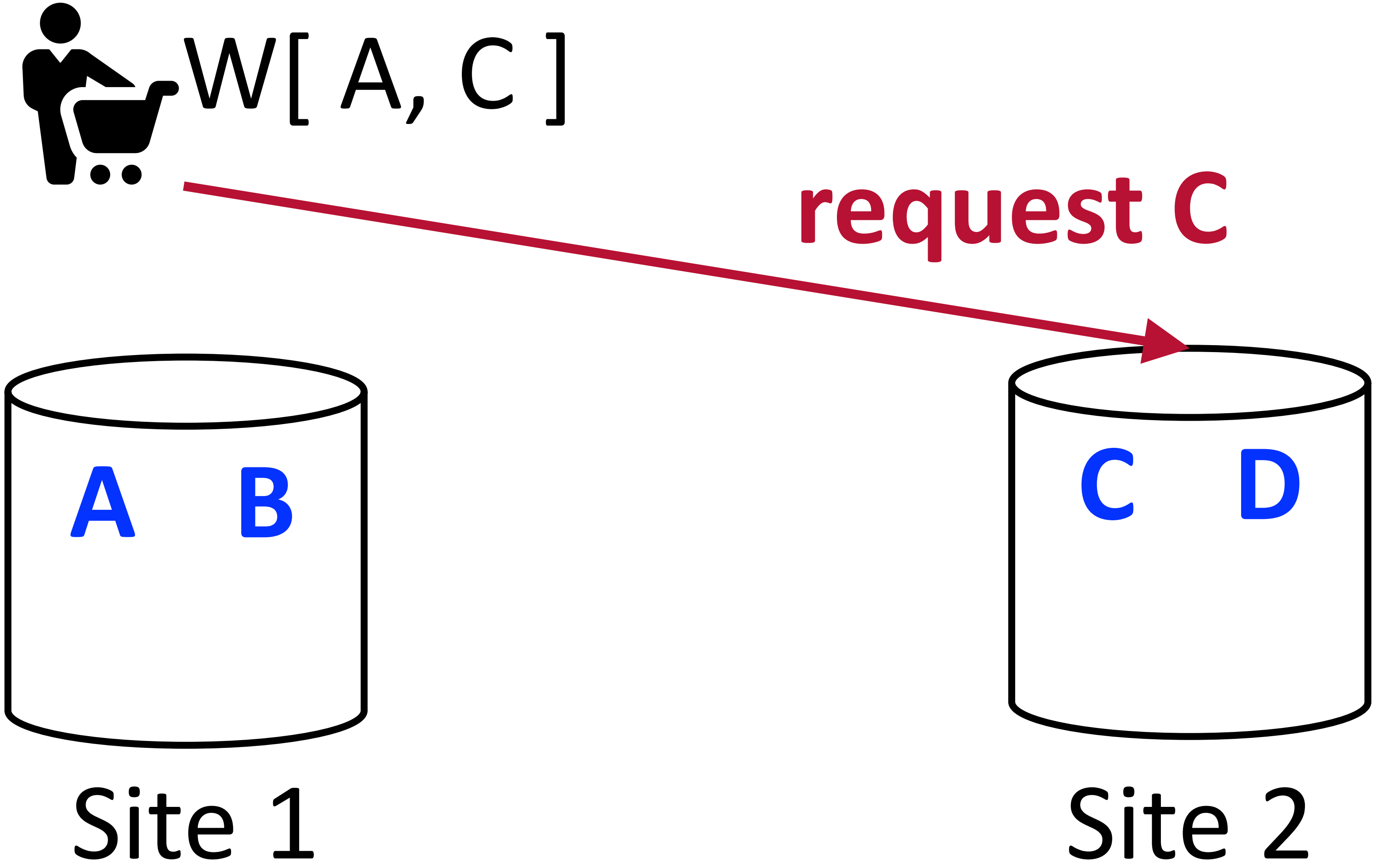
Comparisons – Multi-Master



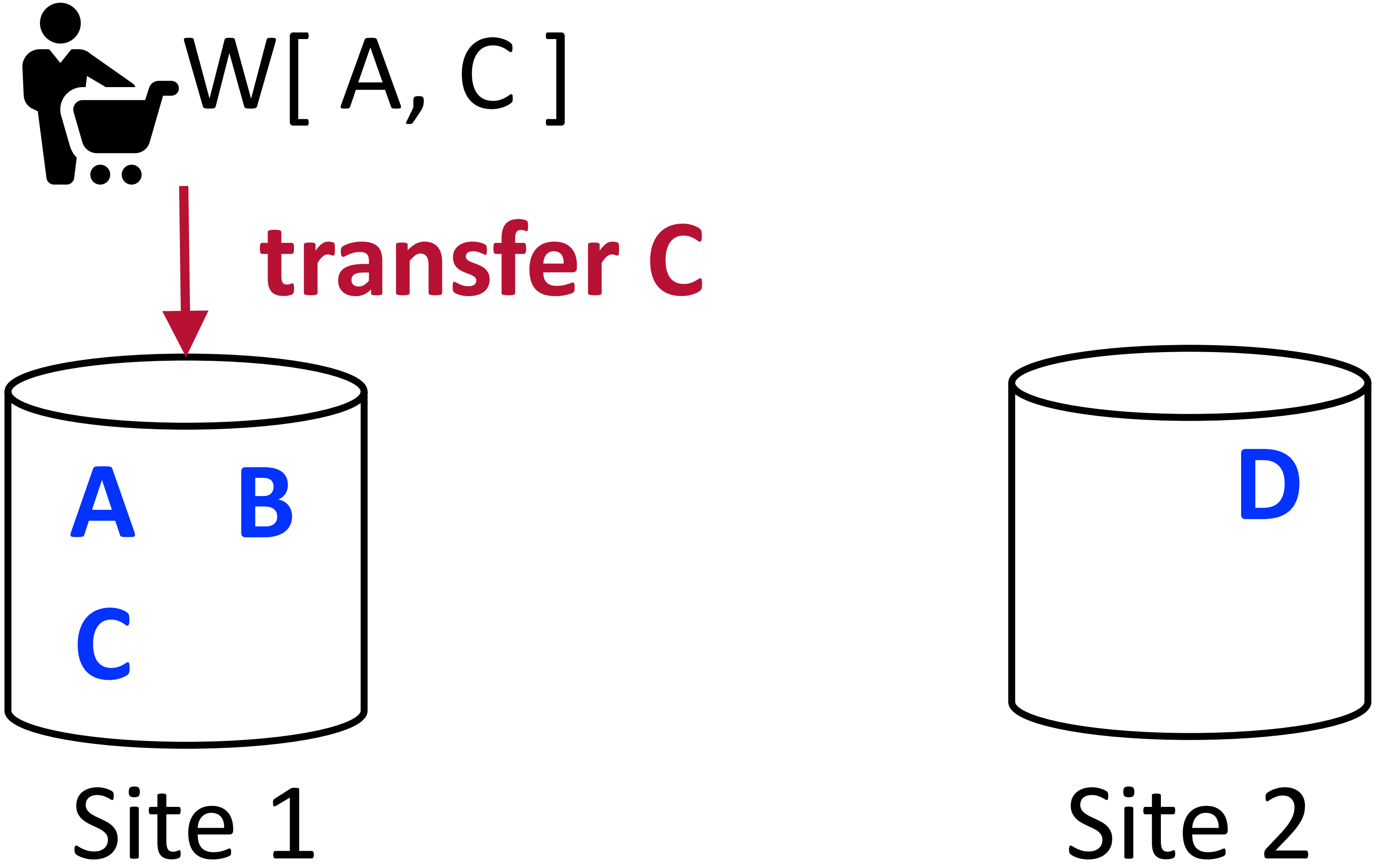
Comparisons – Partition-Store



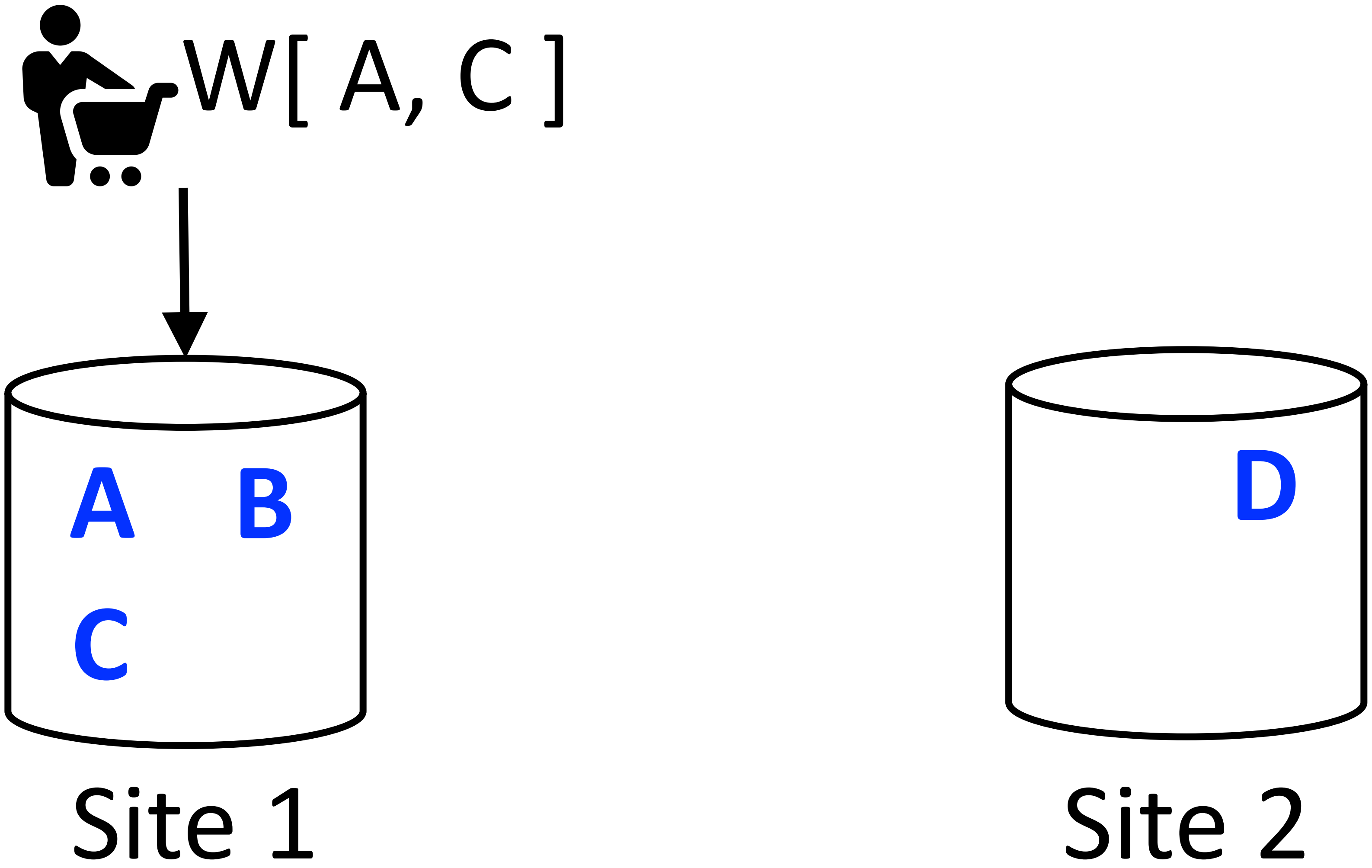
Comparisons – LEAP



Comparisons – LEAP



Comparisons – LEAP


W[A, C]
A
B
C
D
Site 1
Site 2