Schism: Workload-Driven Partitioning and Replication

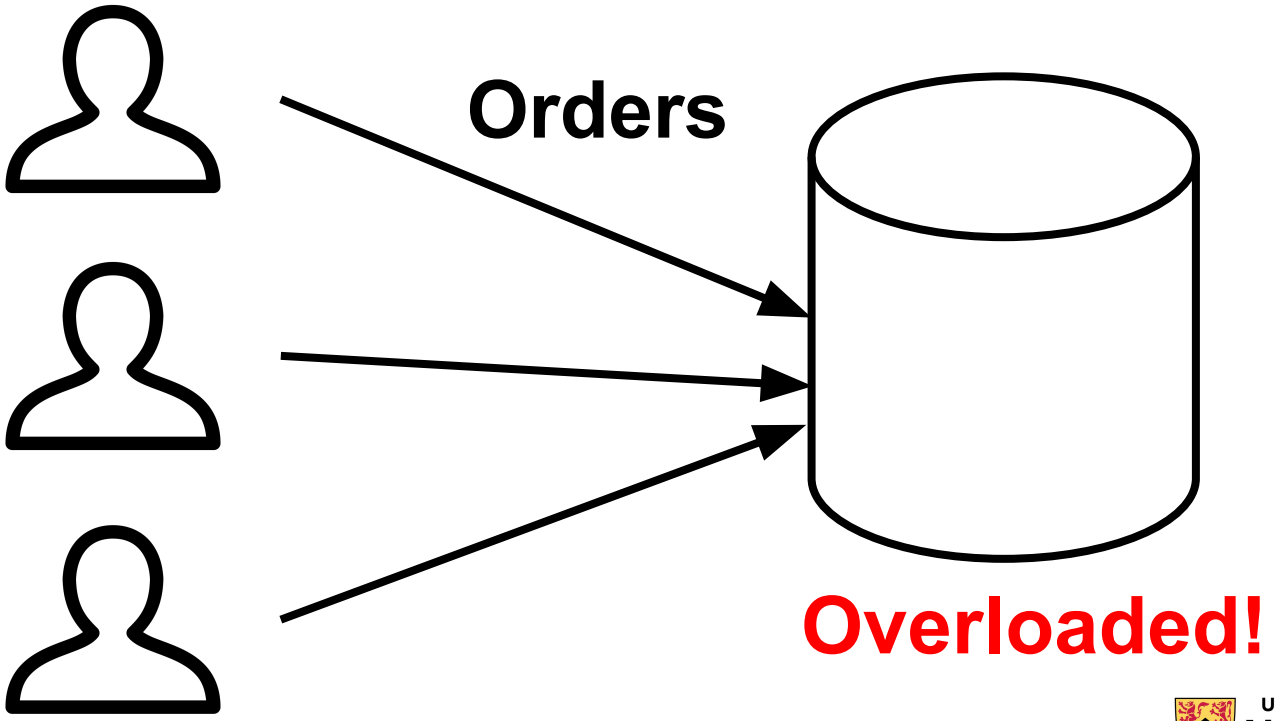
Curino et al., VLDB 2010

Presented By: Brad Glasbergen

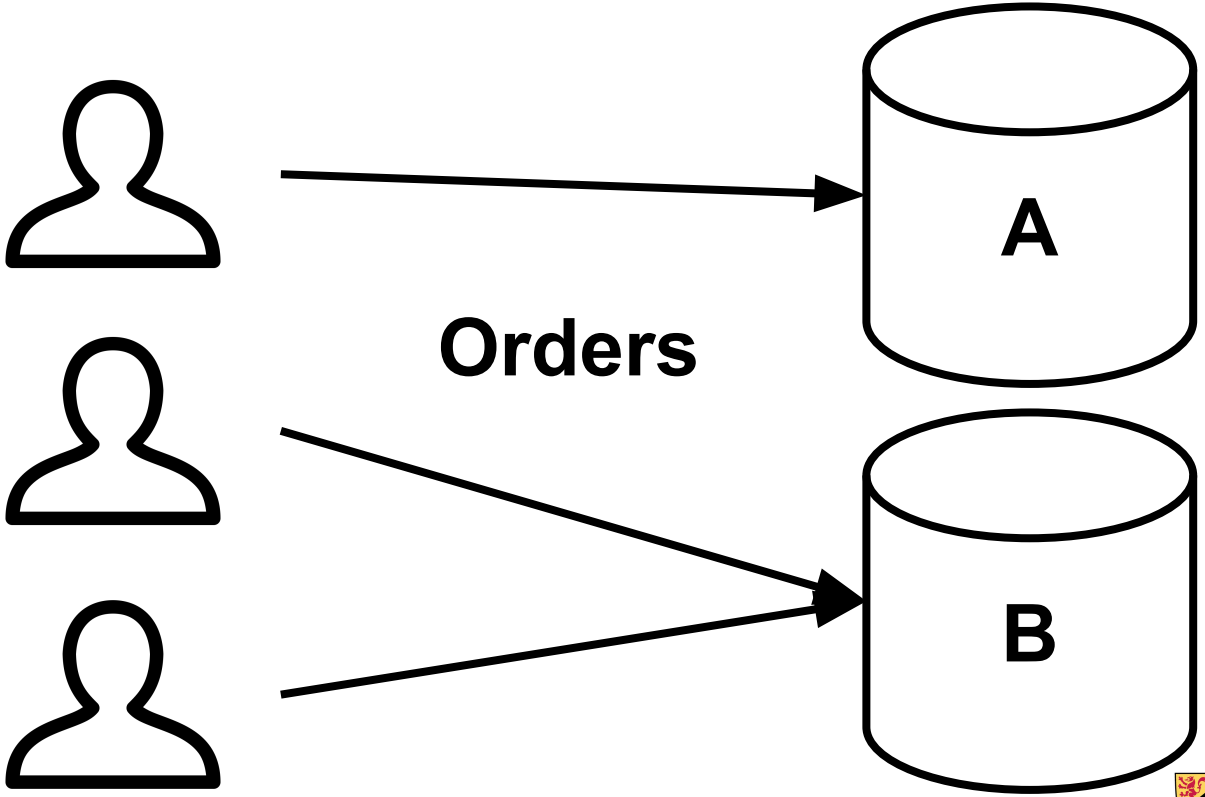
OLTP Workloads

- Short-lived transactions
- Touch few data items
- Write-heavy

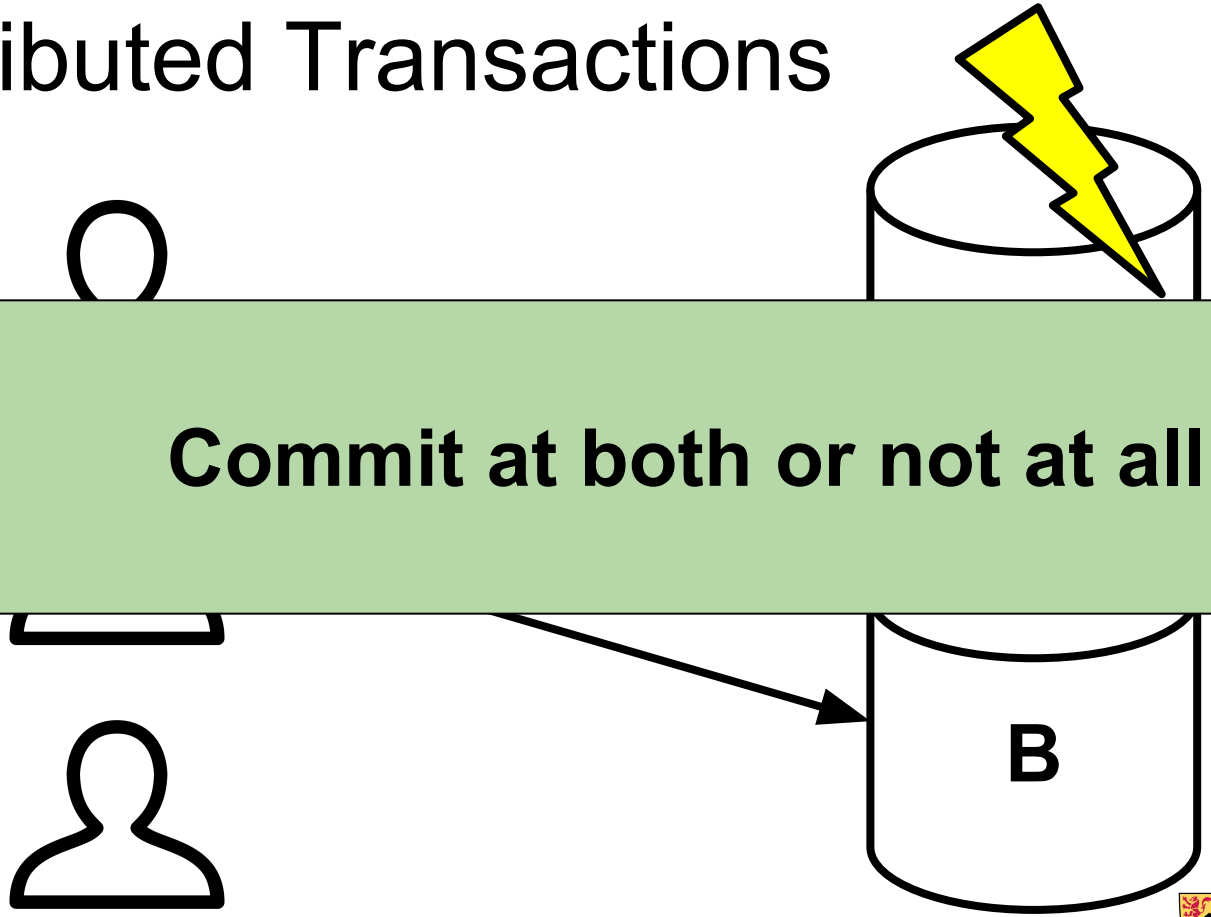
Scaling OLTP Databases



Scaling **Out** OLTP Databases

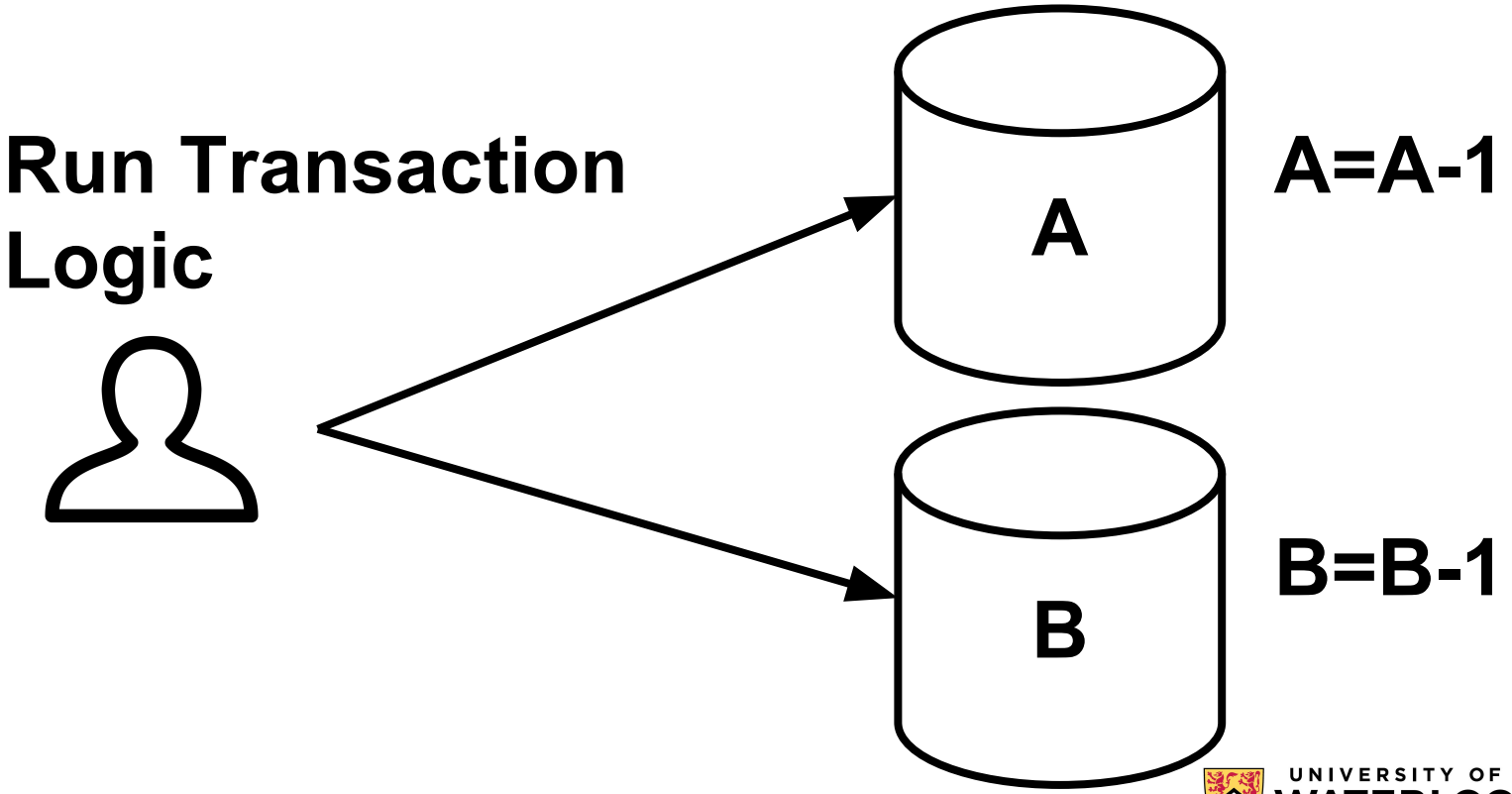


Distributed Transactions



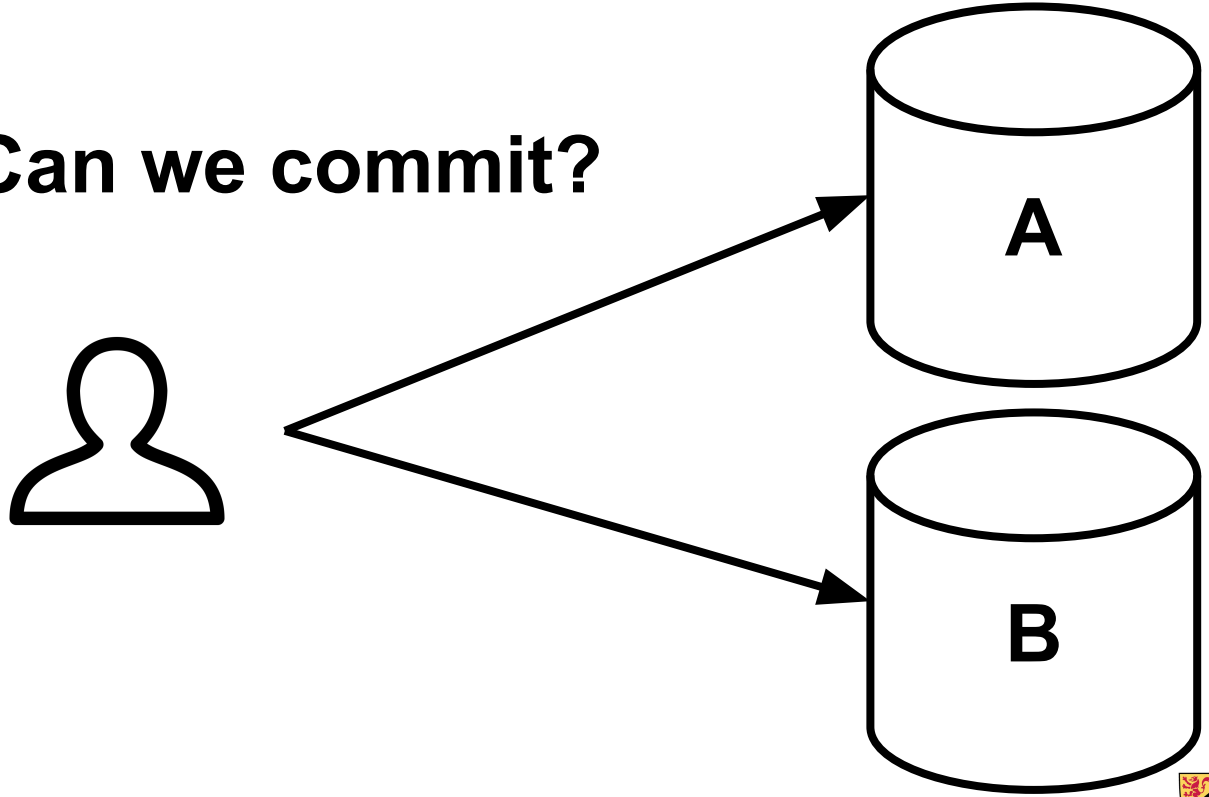
Commit at both or not at all!

Two-Phase Commit

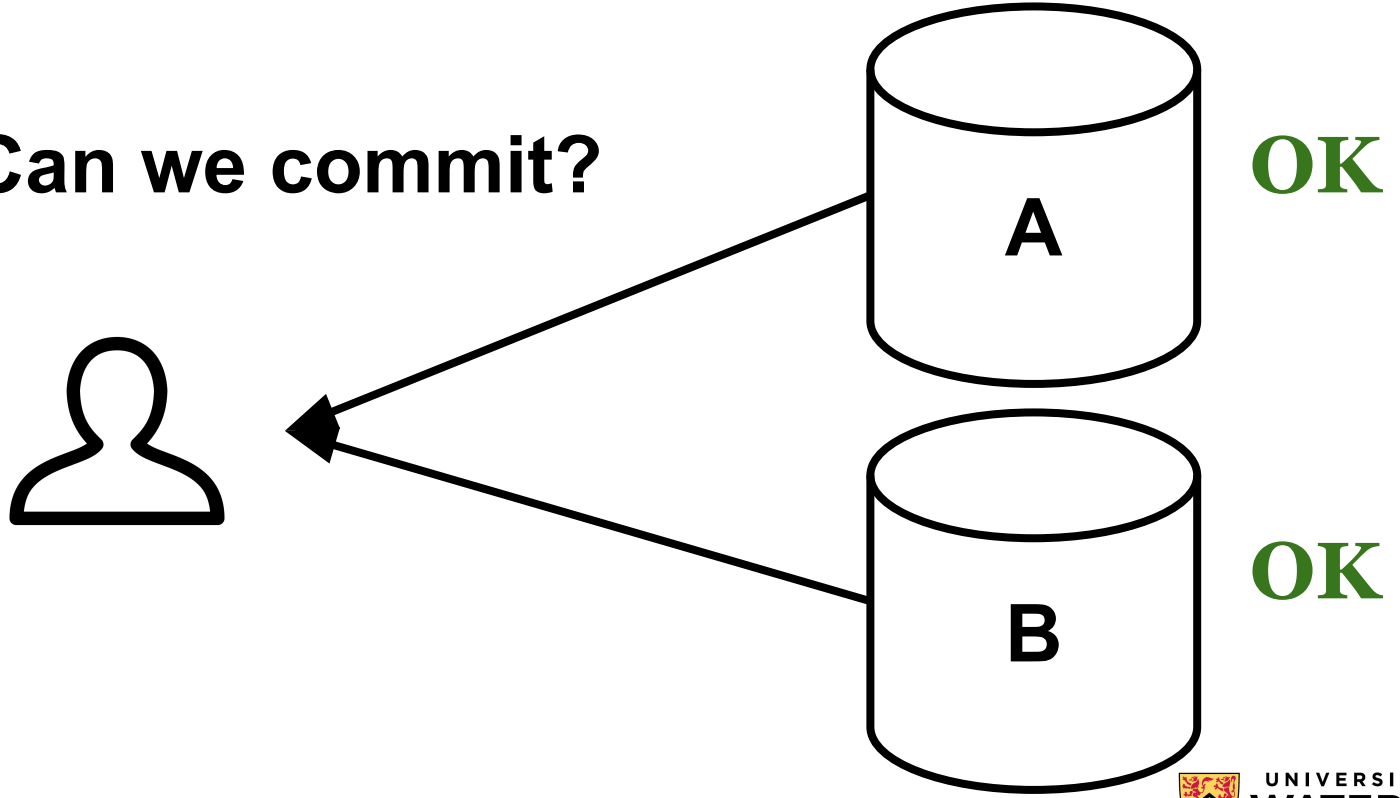


Two-Phase Commit (Phase 1)

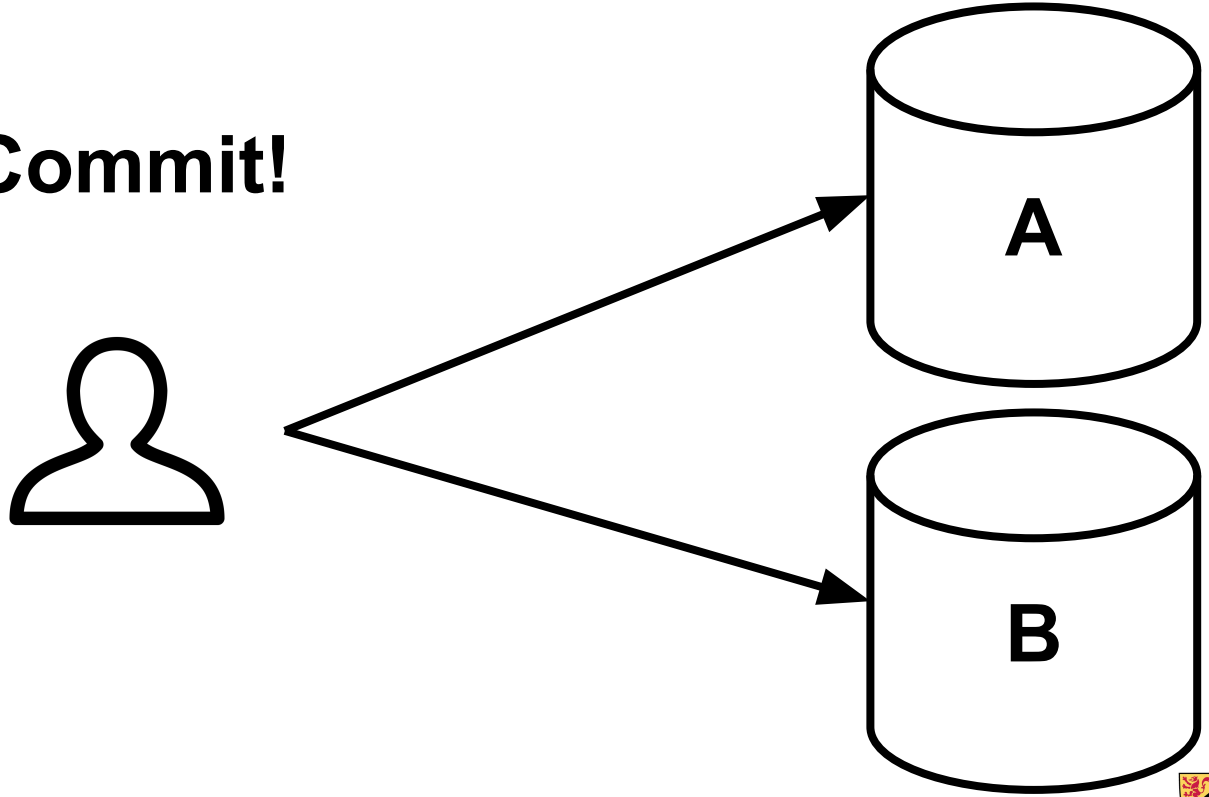
Can we commit?



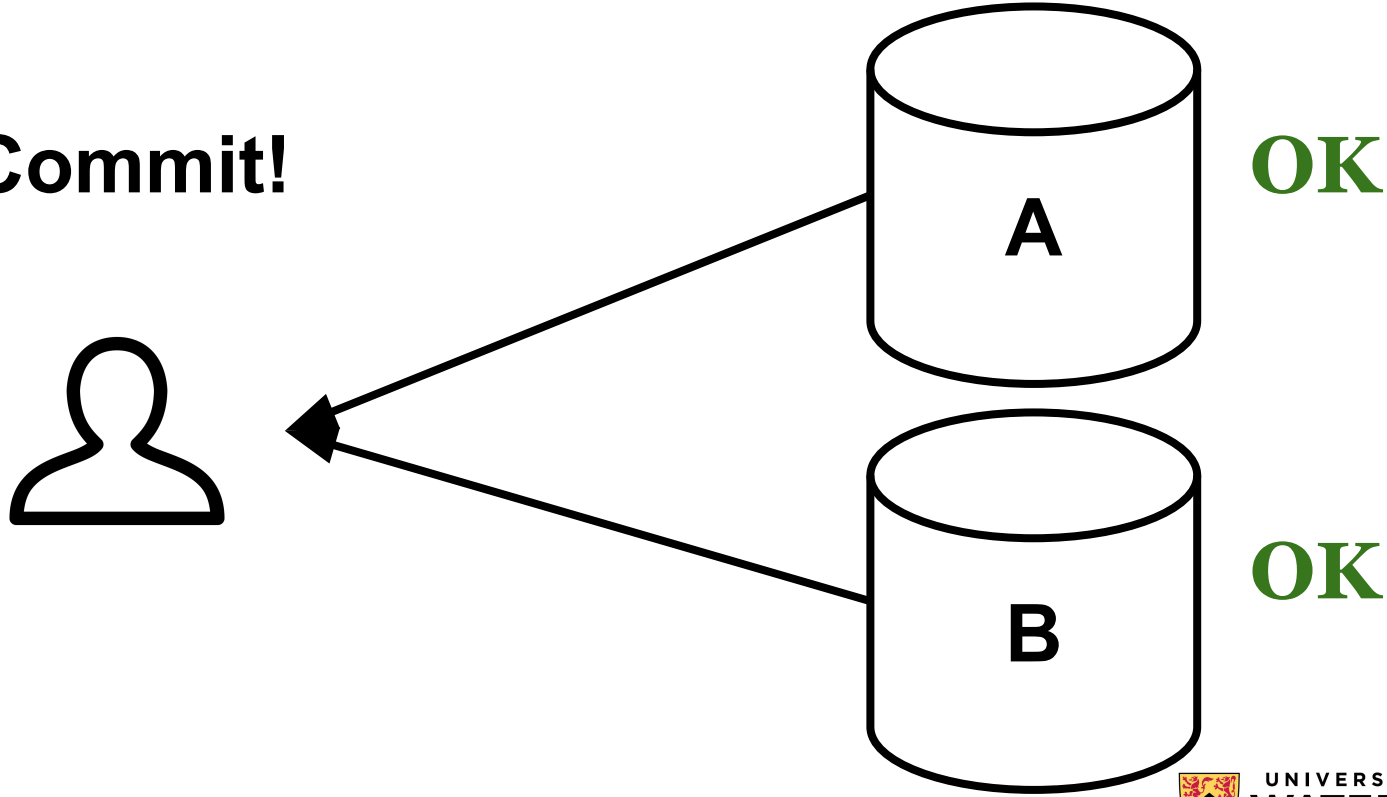
Two-Phase Commit (Phase 1)



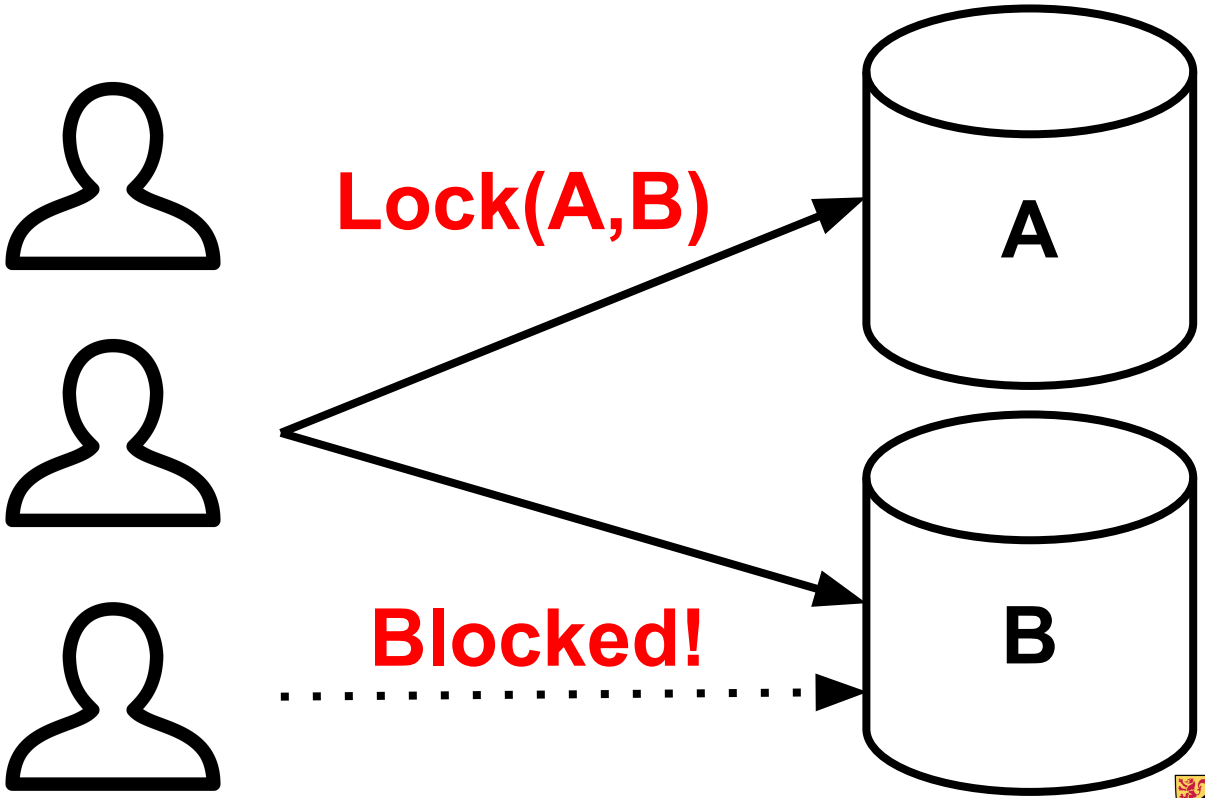
Two-Phase Commit (Phase 2)



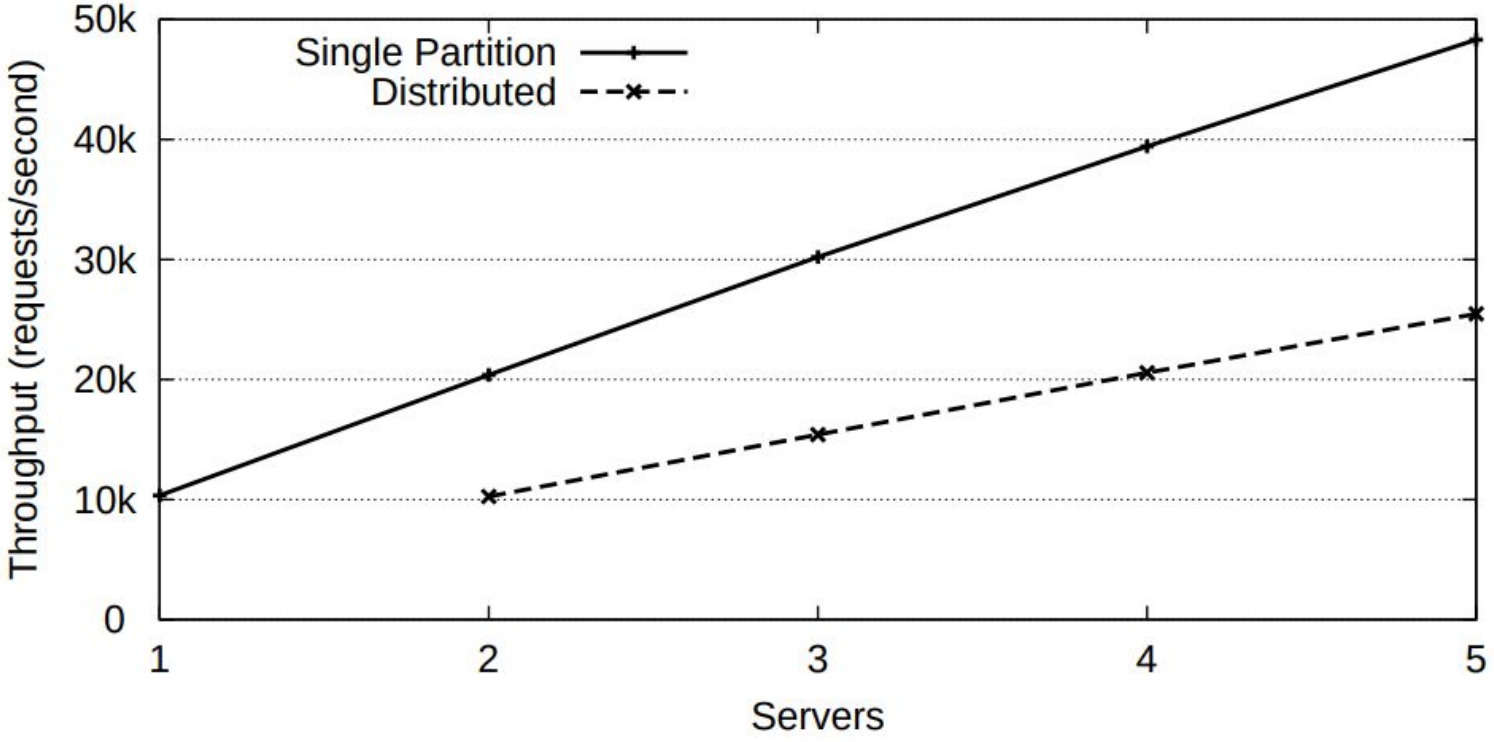
Two-Phase Commit (Phase 2)



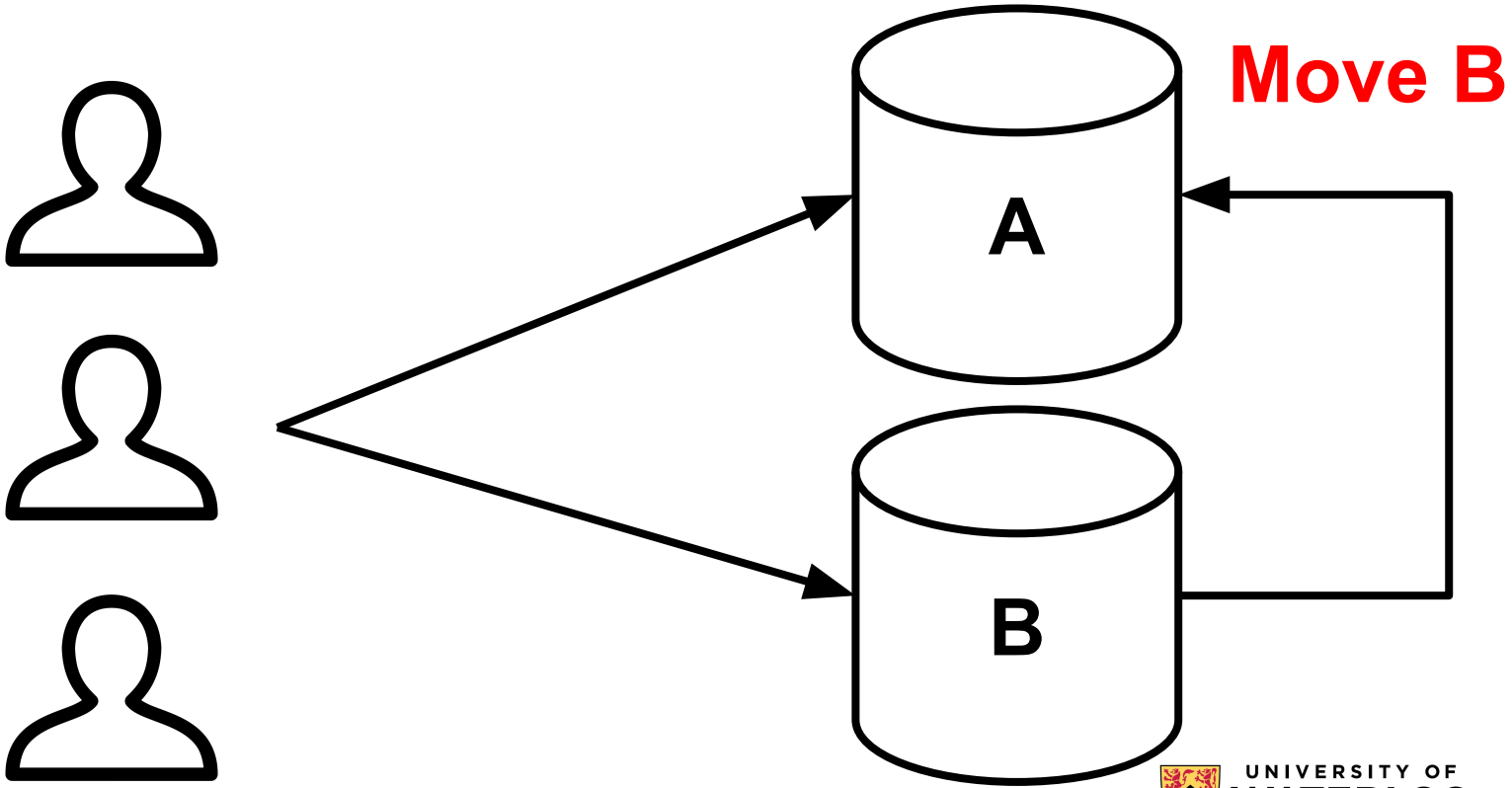
2PC Overheads



2PC Overheads

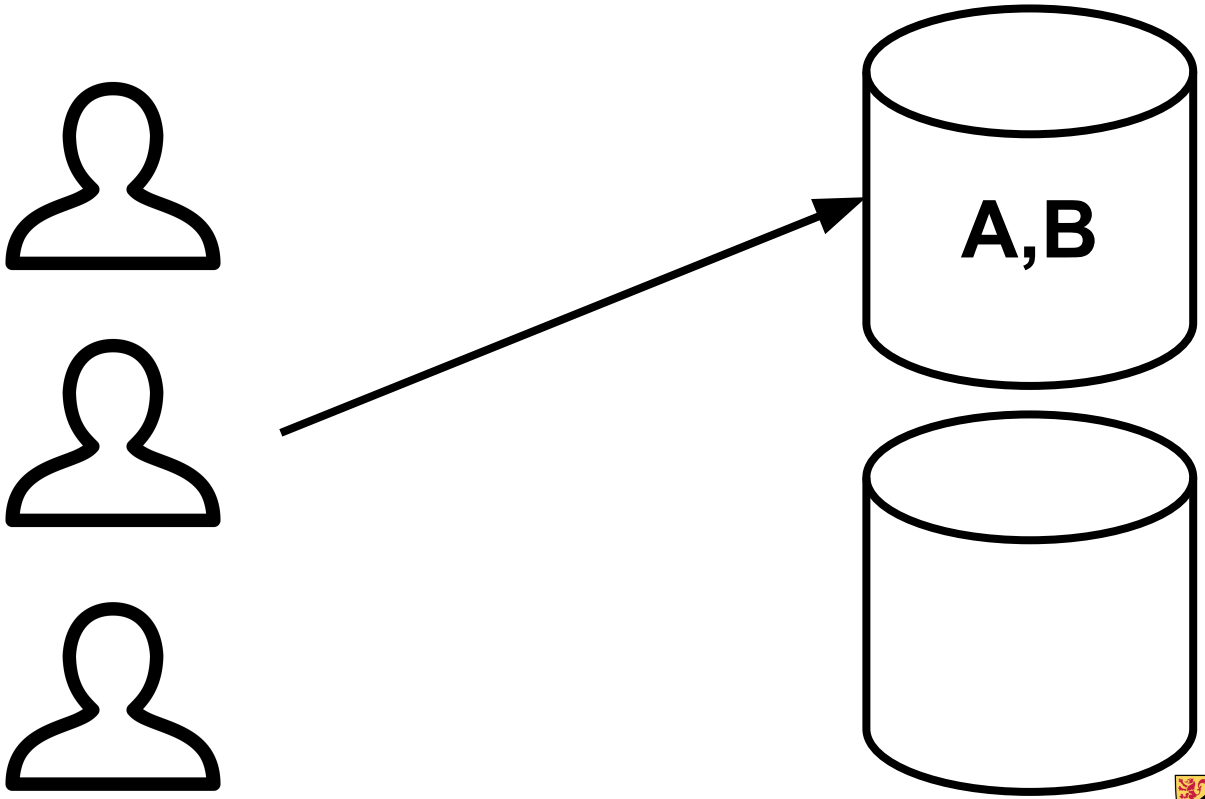


Avoiding 2PC



Avoiding 2PC

Overloaded?

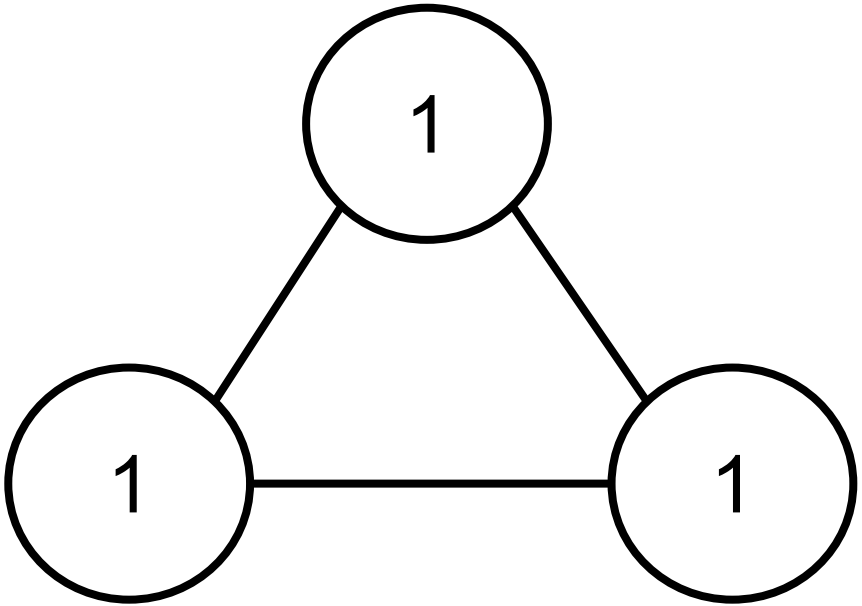


Objectives

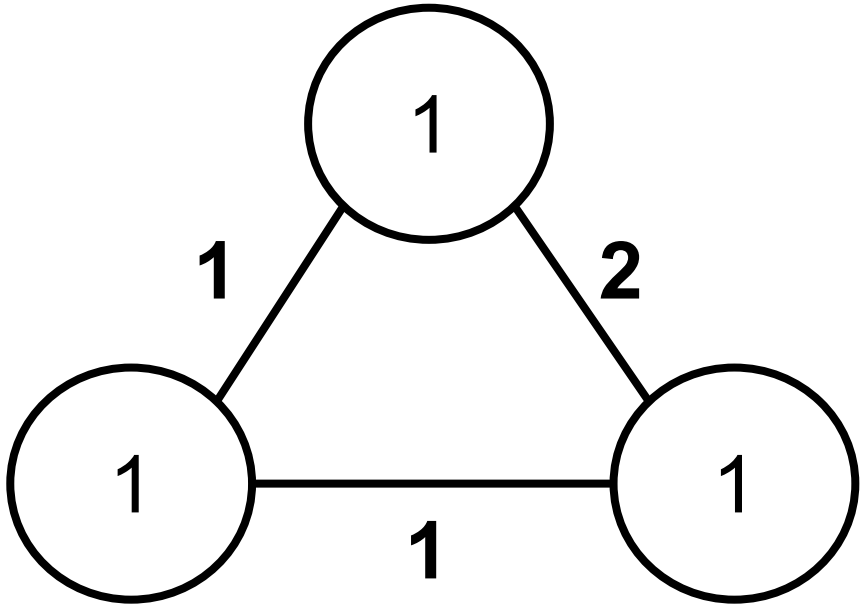
1. **Minimize distributed transactions**
2. **Roughly balance load/data**

Similar to Graph Partitioning problem!
Minimize Edge-cuts subject to imbalance factor

K-Way Graph Partitioning

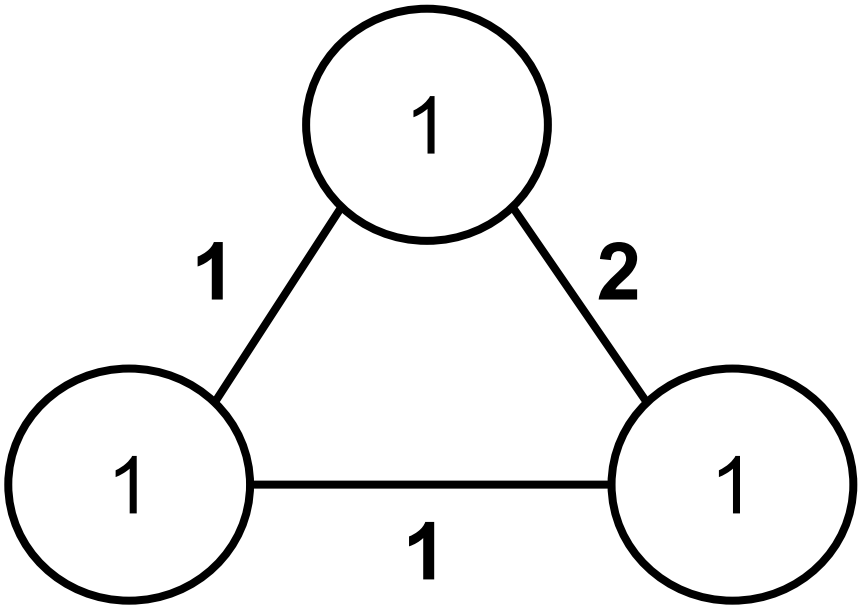


K-Way Graph Partitioning



Each partition is at most twice the weight of another

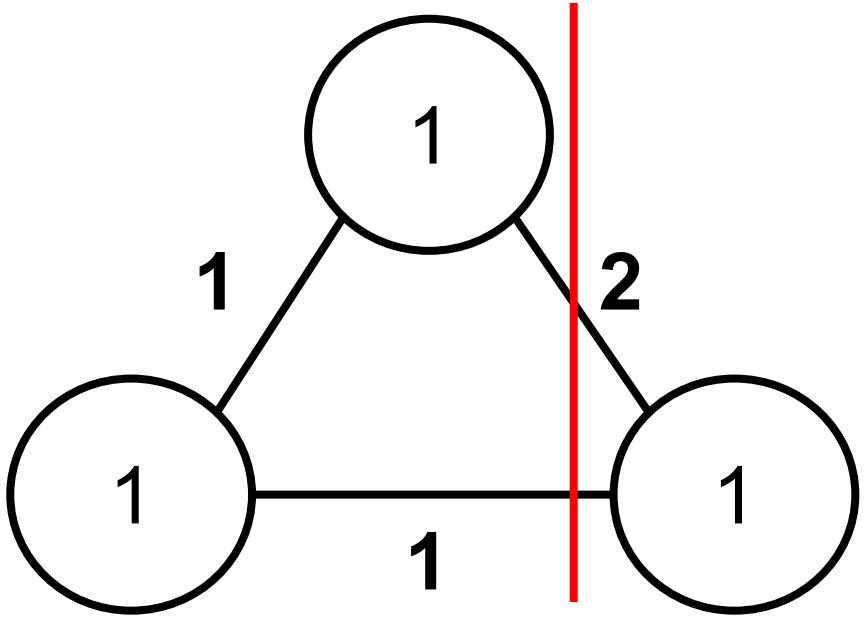
K-Way Graph Partitioning



Each partition is at most twice the weight of another

UNIVERSITY OF WATERLOO

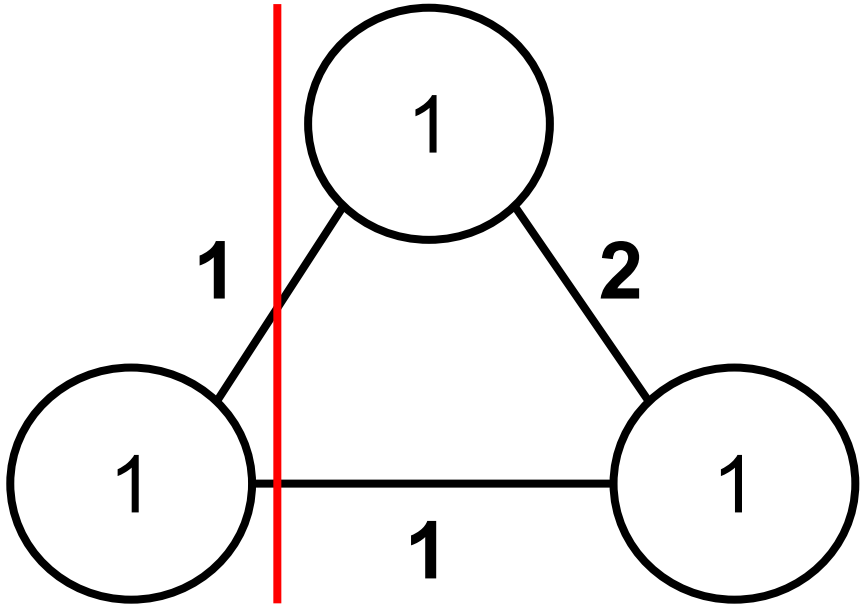
K-Way Graph Partitioning



Each partition is at most twice the weight of another

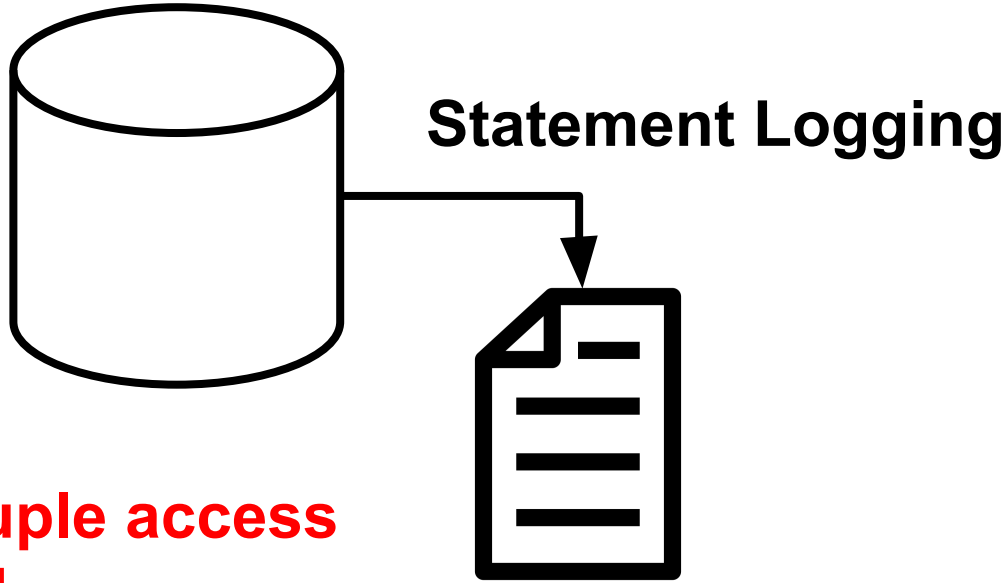
UNIVERSITY OF WATERLOO

K-Way Graph Partitioning



Each partition is at most twice the weight of another

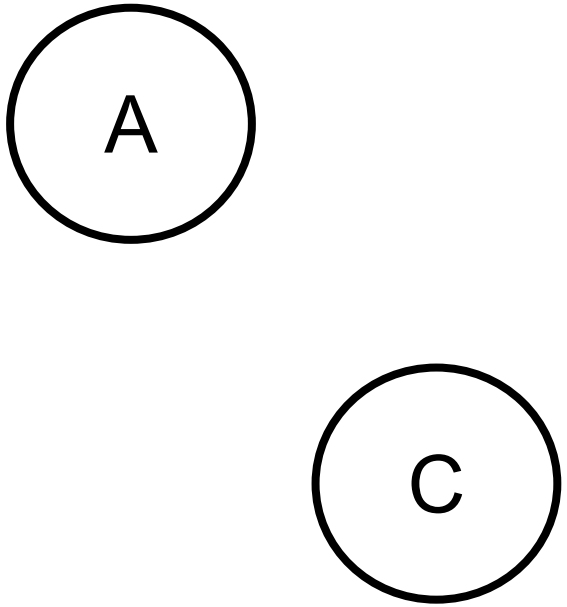
Mapping Partitioning to Graphs



Determine tuple access patterns and frequencies!

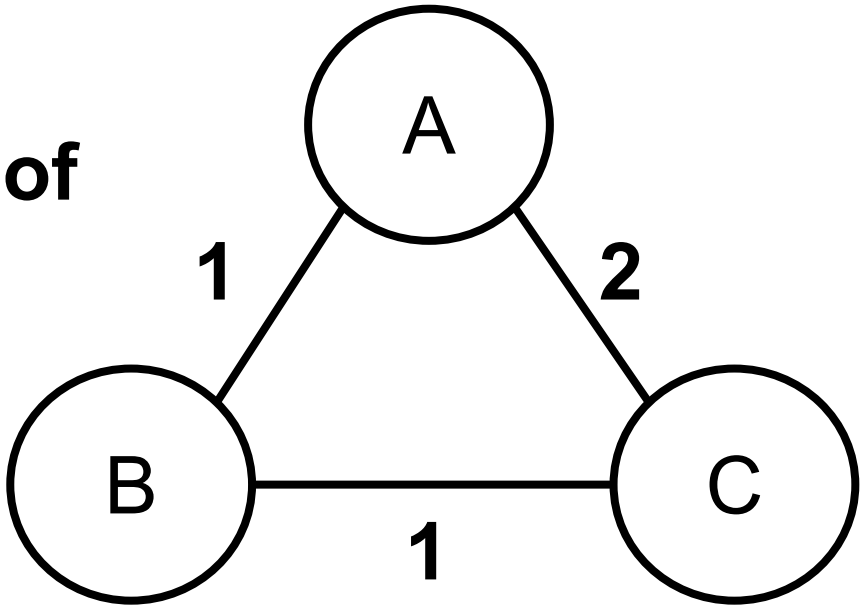
Mapping Partitioning to Graphs

Accessed tuples become nodes



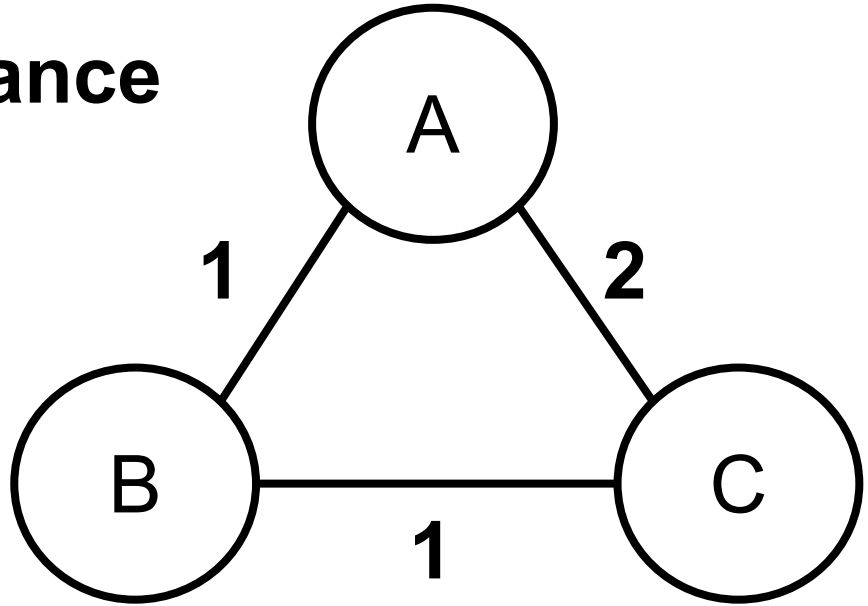
Mapping Partitioning to Graphs

Edges: Frequency of tuples accessed together



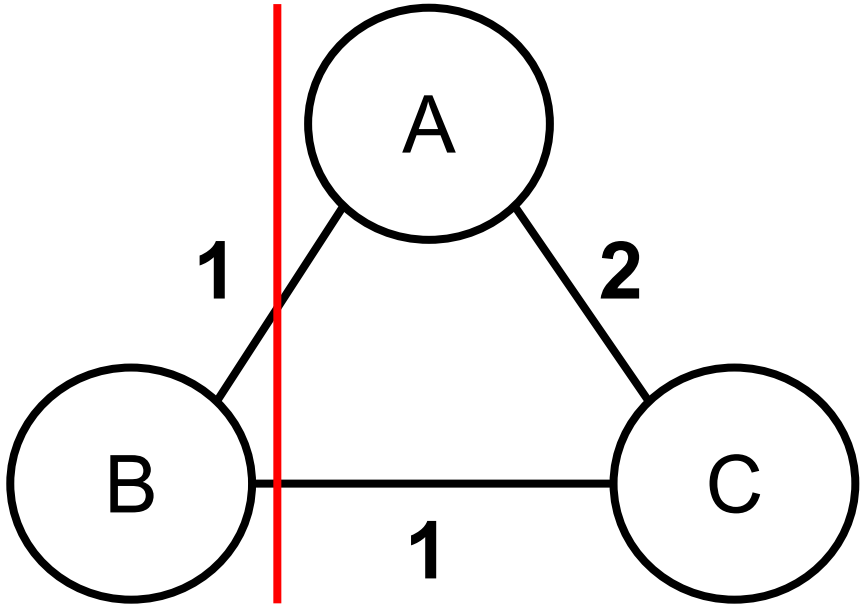
Mapping Partitioning to Graphs

Decide balance factor



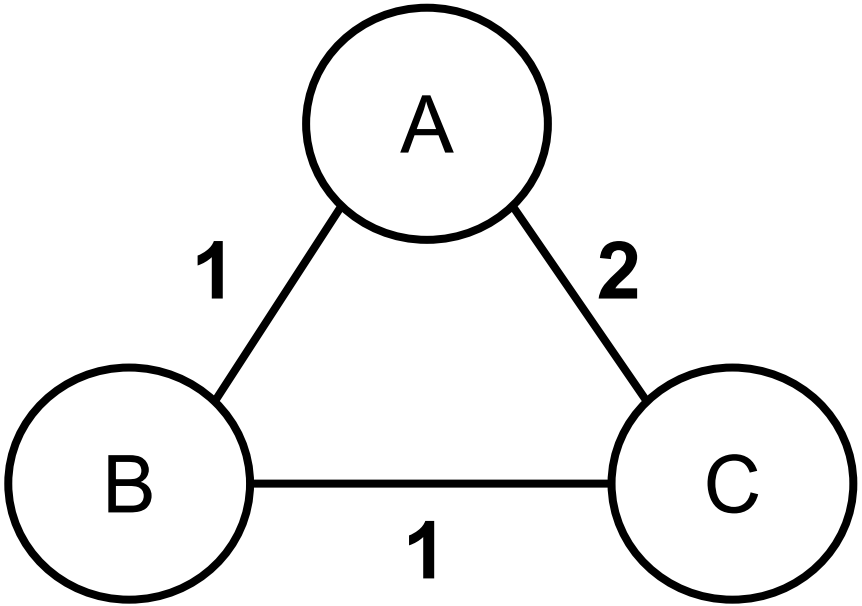
Each partition is at most twice the weight of another

Mapping Partitioning to Graphs

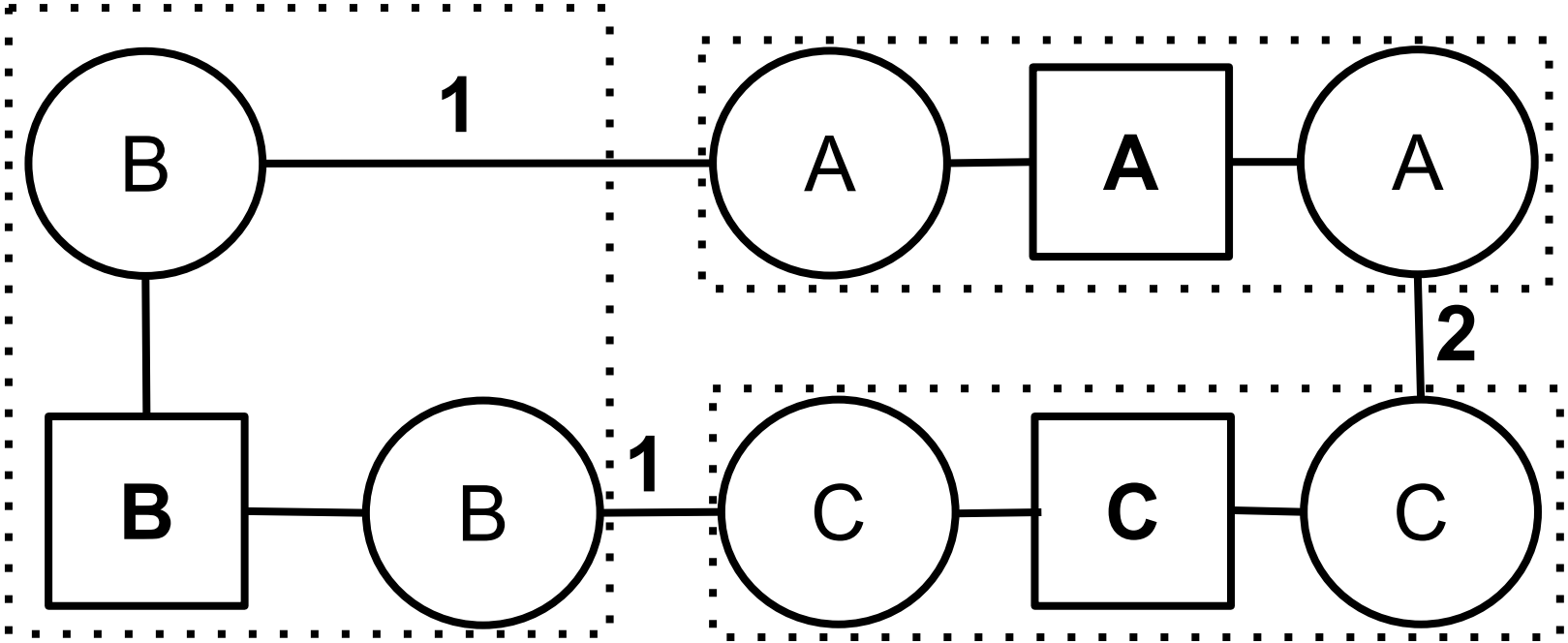


Each partition is at most twice the weight of another

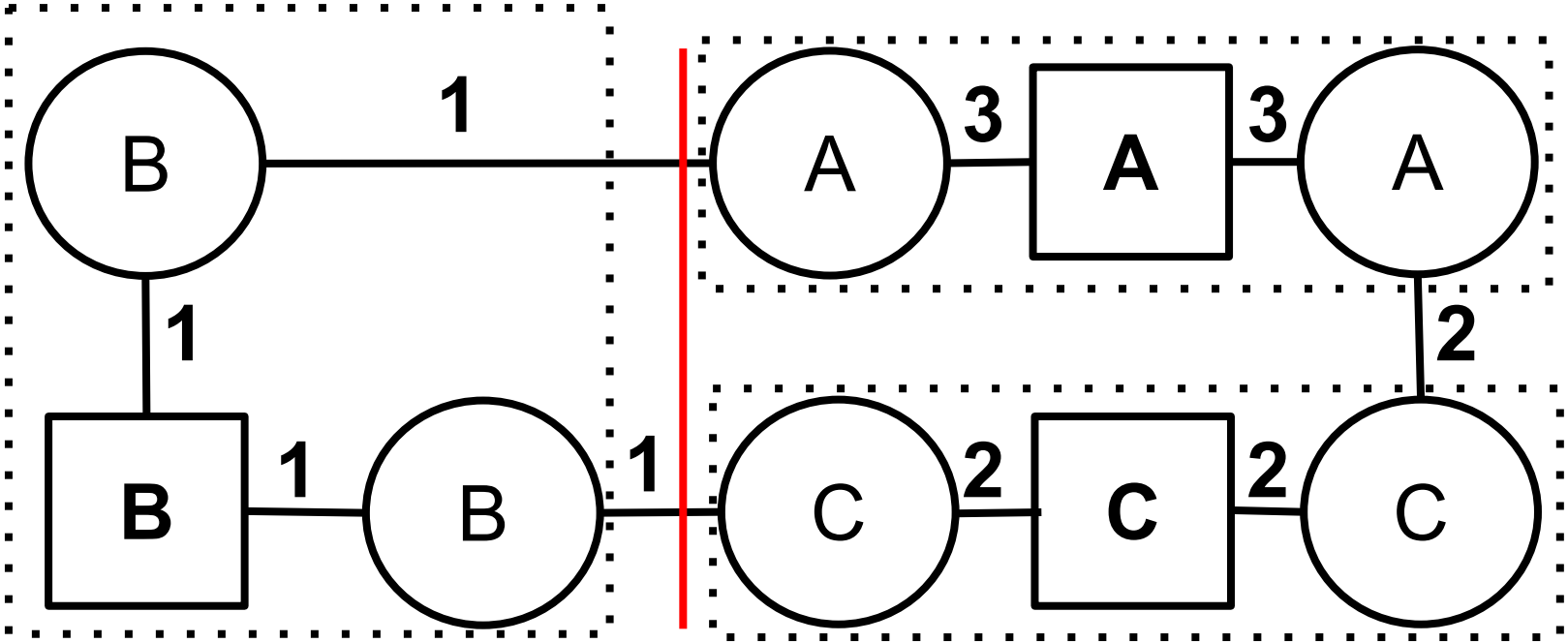
Incorporating Replication



Incorporating Replication



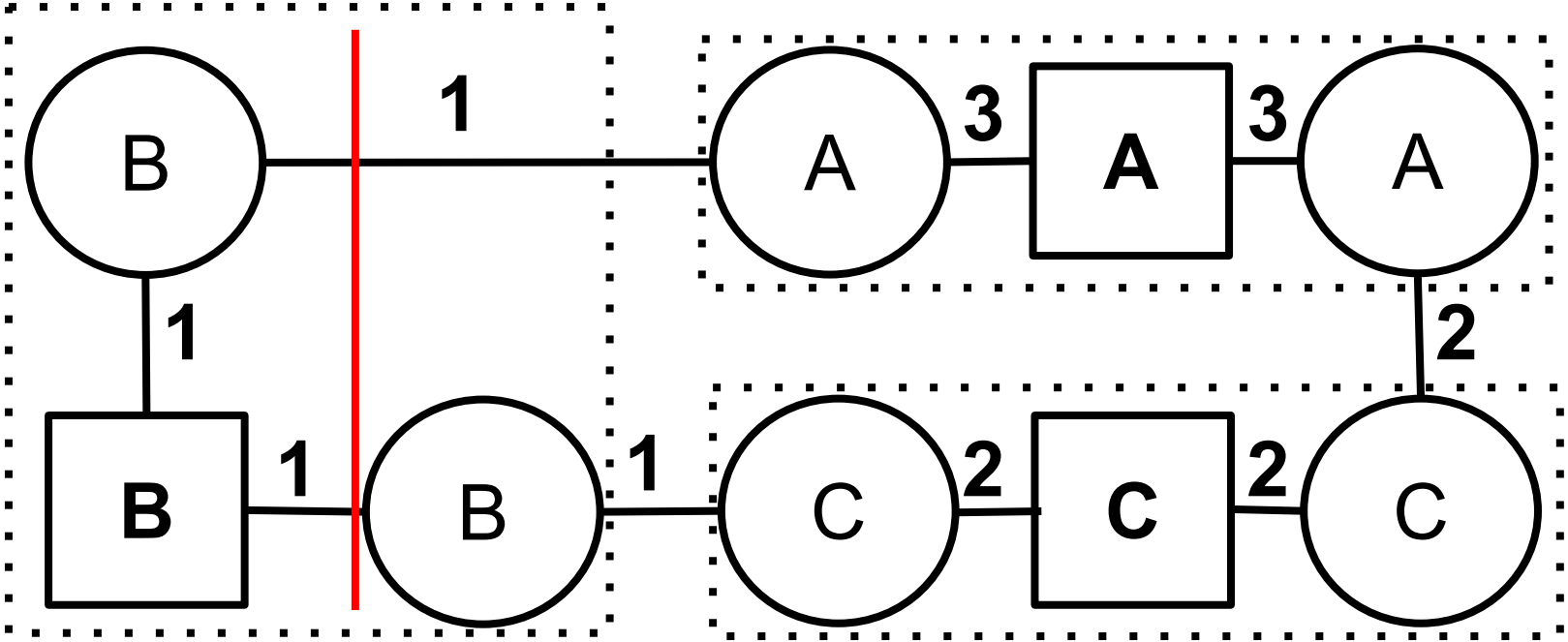
Incorporating Replication



Internal edges: update count!

UNIVERSITY OF WATERLOO

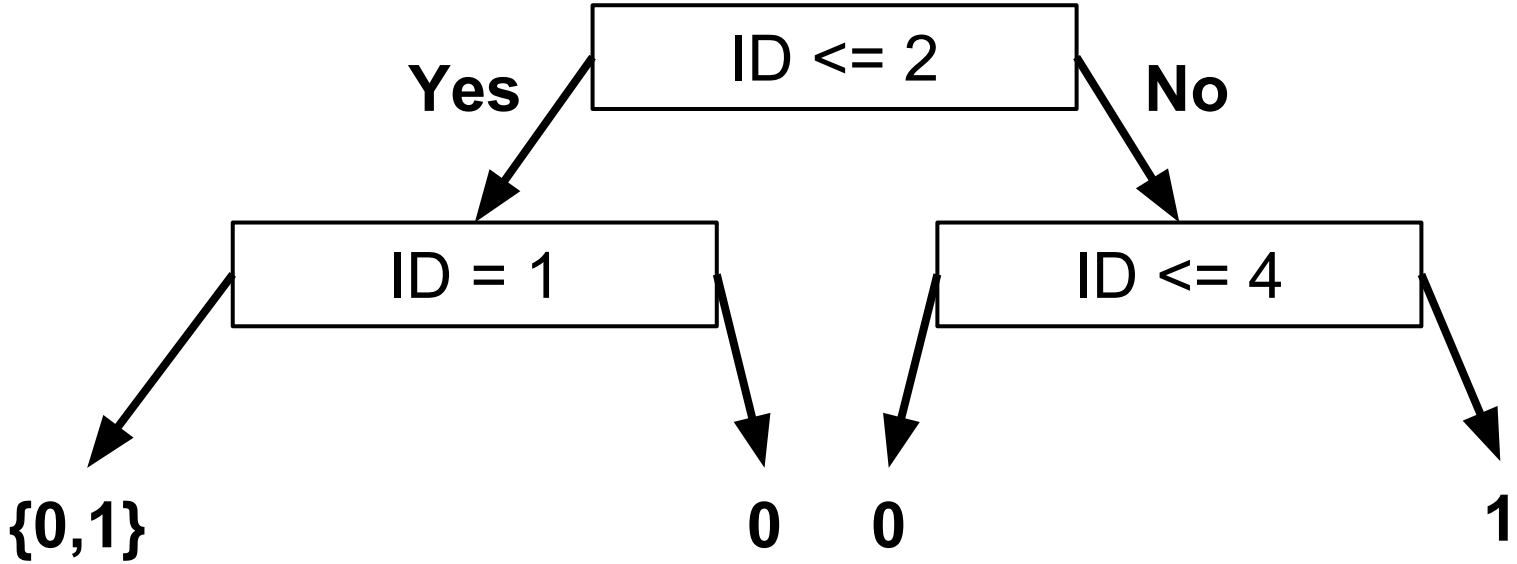
Incorporating Replication



Internal edges: update count!

UNIVERSITY OF WATERLOO

Explaining the Partitioning



Prefer simple explanations

Schism Partitionings

- **YCSB-Default:** Hash Partitioning
- **YCSB-Range:** Range Partitioning
- **TPC-C:** Hash on warehouse, replicate items table
- **TPC-E:** Lookup table, no good known manual partitioning
- **Epinions:** Lookup table, beats manual partitioning

Longest Partitioning: 12 minutes!

The VLDB Journal (2014) 23:845–870
DOI 10.1007/s00778-014-0362-1

SPECIAL ISSUE PAPER

SWORD: workload-aware data placement and replica selection for cloud data management systems

K. Ashwin Kumar · Abdul Quamar · Amol Deshpande · Samir Khuller

Received: 23 September 2013 / Revised: 6 April 2014 / Accepted: 4 June 2014 / Published online: 24 June 2014

Abstract Cloud computing is increasingly being seen as a way to reduce infrastructure costs and add elasticity, and is being used by a wide range of organizations. Cloud data management systems today need to serve a range of different workloads, from analytical read-heavy workloads to transactional (OLTP) workloads. For both the service providers and the users, it is critical to minimize the consumption of resources like CPU, memory, communication bandwidth, and energy, without compromising on service-level agreements if any. In this article, we develop a *workload-aware* data placement and replication approach, called SWORD, for minimizing resource consumption in such an environment. Specifically, we monitor and model the expected workload as a *hypergraph* and develop *partitioning* techniques that minimize the average *query span*, i.e., the average number of machines involved in the execution of a query or a transaction. We empirically justify the use of *query span* as the metric to optimize, for both analytical and transactional workloads, and develop a series of replication and data placement algorithms by drawing connections to several well-studied graph theoretic concepts. We introduce a suite of novel techniques to achieve high scalability by reducing the overhead of partitioning and query routing. To deal with workload changes, we propose an *incremental repartitioning* technique that modifies data placement in small steps without resorting to complete repartitioning. We propose the use of *fine-grained* quorums defined at the level of *groups of data items* to control the cost of distributed updates, improve throughput, and adapt to different workloads. We empirically illustrate the benefits of our approach through a comprehensive experimental evaluation for two classes of workloads. For analytical read-only workloads, we show that our techniques result in significant reduction in total resource consumption. For OLTP workloads, we show that our approach improves transaction latencies and overall throughput by minimizing the number of distributed transactions.

Keywords Cloud data management · Hypergraph partitioning · Data placement · Replication · Resource minimization · Scalability

1 Introduction

Cloud computing is increasingly embraced by a wide range of organizations because of its promise to reduce infrastructure costs and provide elastic scalability on demand. This has led to a proliferation of cloud-based data management systems to enable such services, and data centers to provide the computational infrastructure for them. Cloud data management systems today need to serve a range of different workloads. These include mostly read-only analytical workloads that need to process large volumes of data in a resource-efficient manner, as well as transactional OLTP-style workloads that need to support high throughputs with low latencies. For both the service provider and the users, it is crucial to minimize the total resource consumption in executing these workloads, without compromising on service-level agreements if any. For the service provider, lower resource consumption will enable it to serve a larger number of users without further investment into resources, whereas for the users, lower

K. A. Kumar (✉) · A. Quamar · A. Deshpande · S. Khuller
University of Maryland, College Park, MD, USA
e-mail: ashwin@cs.umd.edu

A. Quamar
e-mail: abdul@cs.umd.edu

A. Deshpande
e-mail: amol@cs.umd.edu

S. Khuller
e-mail: samir@cs.umd.edu

UNIVERSITY OF WATERLOO

Accordion: Elastic Scal
Supporting Dis

Marco Serafini
Qatar Computing Research Institute
mserafini@qf.org.qa

Ess
Qatar C
eman

Kenneth Salem
University of Waterloo
kmsalem@uwaterloo.ca

tahara

ABSTRACT

Providing the ability to elastically use more or fewer servers
demand (scale out and scale in) as the load varies is essential
database management systems (DBMSes) deployed on today's
tributed computing platforms, such as the cloud. This requires s
ing the problem of dynamic (online) data placement, which ha
far been addressed only for workloads where all transactions are
cal to one sever. In DBMSes where ACID transactions can ac
more than one partition, distributed transactions represent a m
performance bottleneck. Scaling out and spreading data acro
larger number of servers does not necessarily result in a linear
crease in the overall system throughput, because transactions
used to access only one server may become distributed.

In this paper we present Accordion, a dynamic data placen
system for partition-based DBMSes that support ACID transacti
(local or distributed). It does so by explicitly considering the *aj*
ity between partitions, which indicates the frequency in which t
are accessed together by the same transactions. Accordion
mates the capacity of a server by explicitly considering the imp
of distributed transactions and affinity on the maximum through
of the server. It then integrates this estimation in a mixed-inte
linear program to explore the space of possible configurations
decide whether to scale out. We implemented Accordion and e
uated it using H-Store, a shared-nothing in-memory DBMS.
results using the TPC-C and YCSB benchmarks show that Ac
dion achieves benefits compared to alternative heuristics of u
an order of magnitude reduction in the number of servers used
in the amount of data migrated.

1. INTRODUCTION

Today's distributed computing platforms, namely clusters
public/private clouds, enable applications to effectively use reso
in an on demand fashion, e.g., by asking for more servers when
load increases and releasing them when the load decreases. S
elastic applications fit well with the pay-as-you-go cost mode

Proceedings of the VLDB Endowment, Vol. 7, No. 12

Skew-Aware Automa
Shared-Nothing,

Andrew Pavlo
Brown University
pavlo@cs.brown.edu

krl@

ABSTRACT

The advent of affordable, shared-nothing computing systems
tends a new class of parallel database management systems (DB
for on-line transaction processing (OLTP) applications that s
without sacrificing ACID guarantees [7, 9]. The performanc
these DBMSs is predicated on the existence of an optimal datal
design that is tailored for the unique characteristics of OLTP w
loads [43]. Deriving such designs for modern DBMSs is difficu
especially for *enterprise-class* OLTP systems, since they imp
extra challenges: the use of stored procedures, the need for
balancing in the presence of time-varying skew, complex scher
and deployments with larger number of partitions.

To this purpose, we present a novel approach to automatic
partitioning databases for enterprise-class OLTP systems that
nificantly extends the state of the art by: (1) minimizing the nu
distributed transactions, while concurrently mitigating the eff
of temporal skew in both the data distribution and accesses, (2)
tending the design space to include replicated secondary inde
(4) organically handling stored procedure routing, and (3) sca
of schema complexity, data size, and number of partitions.
effort builds on two key technical contributions: an analytical
model that can be used to quickly estimate the relative coordina
cost and skew for a given workload and a candidate database
sign, and an informed exploration of the huge solution space b
on large neighborhood search. To evaluate our methods, we i
grated our database design tool with a high-performance para
main memory DBMS and compared our methods against both
ular heuristics and a state-of-the-art research prototype [17]. U
a diverse set of benchmarks, we show that our approach impro
throughput by up to a factor of 16× over these other approache

Categories and Subject Descriptors

H.2.2 [**Database Management**]: Physical Design

Keywords

OLTP, Parallel, Shared-Nothing, H-Store, KB, Stored Procedu

1. INTRODUCTION

The difficulty of scaling front-end applications is well known
DBMSs executing highly concurrent workloads. One approac

SIGMOD'12, May 20–24, 2012, Scottsdale, Arizona, USA.

E-Store: Fine-Gr
Distributed Trans

Rebecca Taft♦, Essam Mansour♣,
Ashraf Aboulnaga♦
♦MIT CSAIL, ♣Qatar Computing Rese
{rytaft,
{emansour, mserafini, aaboul
aelmore@cs.

ABSTRACT

On-line transaction processing (OLTP) database manag
tems (DBMSs) often serve time-varying workloads du
weekly or seasonal fluctuations in demand, or becaus
growth in demand due to a company's business succe
dition, many OLTP workloads are heavily skewed to "h
or ranges of tuples. For example, the majority of NYS
involves only 40 stocks. To deal with such fluctuations
DBMS needs to be elastic; that is, it must be able to e
contract resources in response to load fluctuations and d
balance load as hot tuples vary over time.

This paper presents E-Store, an elastic partitioning
for distributed OLTP DBMSs. It automatically scales re
response to demand spikes, periodic events, and gradual
an application's workload. E-Store addresses localized h
through a two-tier data placement strategy: cold data is
in large chunks, while smaller ranges of hot tuples ar
explicitly to individual nodes. This is in contrast to
single-tier hash and range partitioning strategies. Our e
tal evaluation of E-Store shows the viability of our app
its efficacy under variations in load across a cluster of
Compared to single-tier approaches, E-Store improves t
by up to 130% while reducing latency by 80%.

1. INTRODUCTION

Many OLTP applications are subject to unpredictable
in demand. This variability is especially prevalent in
services, which handle large numbers of requests who
may depend on factors such as the weather or social me
As such, it is important that a back-end DBMS be resili
spikes. For example, an e-commerce site may become ov
during a holiday sale. Moreover, specific items within th
can suddenly become popular, such as when a review of
a TV show generates a deluge of orders in on-line books

Proceedings of the VLDB Endowment, Vol. 8, No. 3

Clay: Fine-Grained Adaptive Partitioning for General Database Schemas

Marco Serafini△, Rebecca Taft♦, Aaron J. Elmore♠,
Andrew Pavlo♠, Ashraf Aboulnaga△, Michael Stonebraker♦
△Qatar Computing Research Institute - HBKU, ♦Massachusetts Institute of Technology,
♠University of Chicago, ♠Carnegie Mellon University
mserafini@qf.org.qa, rytaft@mit.edu, aelmore@cs.uchicago.edu,
pavlo@cs.cmu.edu, aaboulnaga@qf.org.qa, stonebraker@csail.mit.edu

ABSTRACT

Transaction processing database management systems (DBMSs)
are critical for today's data-intensive applications because they en-
able an organization to quickly ingest and query new information.
Many of these applications exceed the capabilities of a single server,
and thus their database has to be deployed in a distributed DBMS.
The key factor affecting such a system's performance is how the
database is partitioned. If the database is partitioned incorrectly, the
number of distributed transactions can be high. These transactions
have to synchronize their operations over the network, which is
considerably slower and leads to poor performance. Previous work
on elastic database repartitioning has focused on a certain class of
applications whose database schema can be represented in a hierar-
chical tree structure. But many applications cannot be partitioned
in this manner, and thus are subject to distributed transactions that
impede their performance and scalability.

In this paper, we present a new on-line partitioning approach,
called Clay, that supports both tree-based schemas and more com-
plex "general" schemas with arbitrary foreign key relationships.
Clay dynamically creates blocks of tuples to migrate among servers
during repartitioning, placing no constraints on the schema but tak-
ing care to balance load and reduce the amount of data migrated.
Clay achieves this goal by including in each block a set of hot tuples
and other tuples co-accessed with these hot tuples. To evaluate our
approach, we integrate Clay in a distributed, main-memory DBMS
and show that it can generate partitioning schemes that enable the
system to achieve up to 15× better throughput and 99% lower la-
tency than existing approaches.

1. INTRODUCTION

Shared-nothing, distributed DBMSs are the core component for
modern on-line transaction processing (OLTP) applications in many
diverse domains. These systems partition the database across mul-
tiple nodes (i.e., servers) and route transactions to the appropriate
nodes based on the data that these transactions touch. The key to
achieving good performance is to use a partitioning scheme (i.e., a
mapping of tuples to nodes) that (1) balances load and (2) avoids
expensive multi-node transactions [5, 23]. Since the load on the
DBMS fluctuates, it is desirable to have an *elastic* system that auto-
matically changes the database's partitioning and number of nodes
dynamically depending on load intensity and without having to stop
the system.

The ability to change the partitioning scheme without disrupt-
ing the database is important because OLTP systems incur fluctu-
ating loads. Additionally, many workloads are seasonal or diurnal,
while other applications are subject to dynamic fluctuations in their
workload. For example, the trading volume on the NYSE is an
order of magnitude higher at the beginning and end of the trading
day, and transaction volume spikes when there is relevant breaking
news. Further complicating this problem is the presence of *hotspots*
that can change over time. These occur because the access pattern
of transactions in the application's workload is skewed such that
a small portion of the database receives most of the activity. For
example, half of the NYSE trades are on just 1% of the securities.

One could deal with these fluctuations by provisioning for ex-
pected peak load. But this requires deploying a cluster that is over-
provisioned by at least an order of magnitude [27]. Furthermore, if
the performance bottleneck is due to distributed transactions caus-
ing nodes to wait for other nodes, then adding servers will be of
little or no benefit. Thus, over-provisioning is not a good alterna-
tive to effective on-line reconfiguration.

Previous work has developed techniques to automate DBMS re-
configuration for unpredictable OLTP workloads. For example,
Accordion [26], ElasTras [6], and E-Store [28] all study this prob-
lem. These systems assume that the database is partitioned a pri-
ori into a set of *static blocks*, and all tuples of a block are moved
together at once. This does not work well if transactions access
tuples in multiple blocks and these blocks are not colocated on the
same server. One study showed that a DBMS's throughput drops
by half from its peak performance with only 10% of transactions
distributed [23]. This implies that minimizing distributed transac-
tions is just as important as balancing load when finding an optimal
partitioning plan. To achieve this goal, blocks should be defined
dynamically so that tuples that are frequently accessed together are
grouped in the same block; co-accesses within a block never gener-
ate distributed transactions, regardless of where blocks are placed.

Another problem with the prior approaches is that they only work
for tree schemas. This excludes many applications with schemas
that cannot be transposed into a tree and where defining static blocks
is impossible. For example, consider the Products-Parts-Suppliers
schema shown in Figure 1. This schema contains three tables that
have many-to-many relationships between them. A product uses
many parts, and a supplier sells many parts. If we apply prior ap-
proaches and assume that either Products or Suppliers is the root

Proceedings of the VLDB Endowment, Vol. 10, No. 4

Discussion Points

- Schism is offline/periodic. How important is online partitioning, ***really***?
- Would an OLAP workload change how we partition? Consider parallel query execution.