Databases at scale

DATA 604

Leanne Wu

lewu@ucalgary.ca

Department of Computer Science

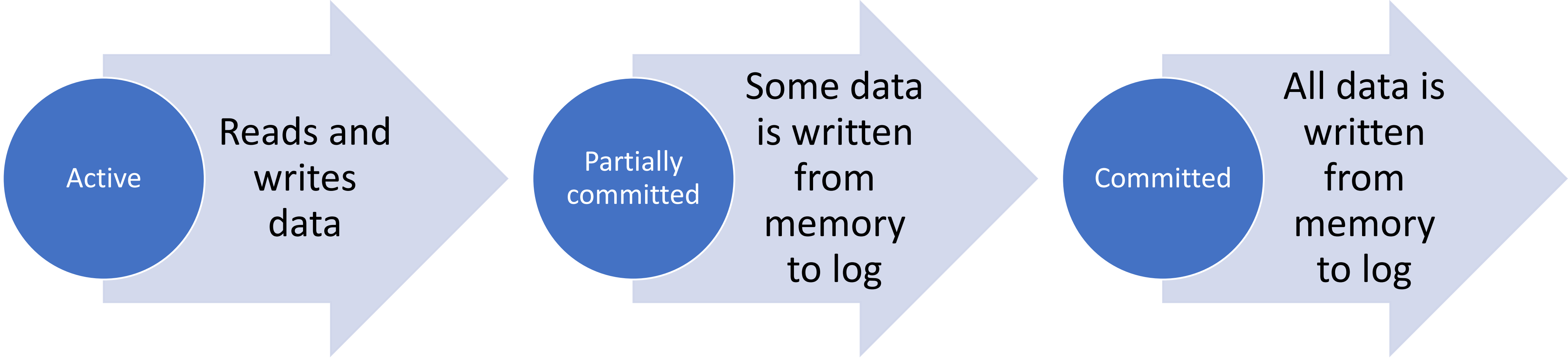
UNIVERSITY OF CALGARY

Adding more users

- Supporting more users means that databases need to support concurrent transactions
  - Writing data
    - ensure that each item that is to be written to the database is written to disk
  - Reading data
    - ensure that each read operation is accurate
  - Updating data
    - ensure that data is read accurately, then changed in a consistent way

The lifecycle of a transaction

Once data is written to the log, it exists somewhere on disk


Active
Reads and writes data
Partially committed
Some data is written from memory to log
Committed
All data is written from memory to log

Transactions may be aborted by the application (user) or otherwise fail

Typical problems

- Lost update
  - one interleaved update overwrites another
- Temporary update (dirty reads)
  - a failed transaction does not roll back its data before another operation reads an update that is now invalid
- Incorrect summary
  - aggregation operation reads some database items before they are updated by another operation, and reads some database items after they are updated by that operation
- Unrepeatable reads
  - successive reads by a transaction results in different values, because another operation has updated data between reads

Isolation levels

- Indicates to the DBMS how tolerant of consistency errors you may be
- Generally set at a database-wide level, but some platforms allow per-transaction isolation levels.

SQL standard isolation levels are:

READ UNCOMMITTED	Any consistency errors
READ COMMITTED	No dirty reads
REPEATABLE READ	No dirty or phantom reads
SERIALIZABLE	No consistency errors

Controlling transactions

- To manage which transactions have access to specific parts of the database, objects called **locks** are assigned to individual data items
    - only one lock per object
- Different locking schemes are possible
    - **binary locks:** items are either locked or unlocked
    - **shared/exclusive (read/write) locks**: shared locks allow for reading, exclusive locks must be used for writing
- **Two phase locking** protocols insure that transactions can be serializable
    - **Growing:** transactions can acquire locks
    - **Shrinking:** transactions can release locks

Deadlocks and lock escalation

- The DBMS also monitors when data items are **in contention**
  - Deadlocks: When transaction A has a lock for item X and needs a lock for item Y, but transaction B has a lock for item Y and needs a lock for item X
    - the system will cause one of these transactions to fail to ensure progress
  - items which are in high demand are said to be in contention
- To reduce contention, some platforms will allow for different granularities of lock
  - row level, block(page) level, table-level
  - if many items are in contention, then the granularity level can be changed to reduce demand

In MySQL

- Explore the performance_schema database
  - What is the isolation level? (look at the global_variable table)
  - Is table-level locking allowed? (lock at the global_variable table)
  - Can you find anything interesting in the global_status table?)

- Hint: You don’t have to switch databases to access tables from another schema. Try:

```
SELECT count(*) FROM performance_schema.global_variables;
```

Adding more data

- Adding more tables
  - How do we use more complex schemata in interesting ways?
  - Can we combine data from different databases in meaningful ways?
- Adding more records
  - What is the impact of big data on databases?
    - Consider relational databases
      - cost of joins
      - cost of constraints
      - cost of transaction management

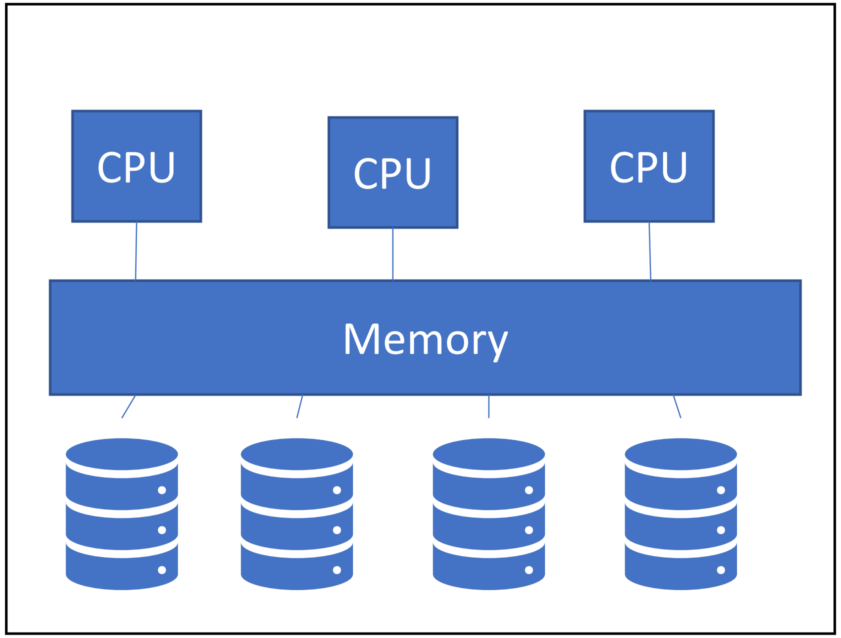
Adding more computing power and disk

- More complexity in your system architecture results in more complex logical and physical design
  - How do you spread data across different disks and CPUs?

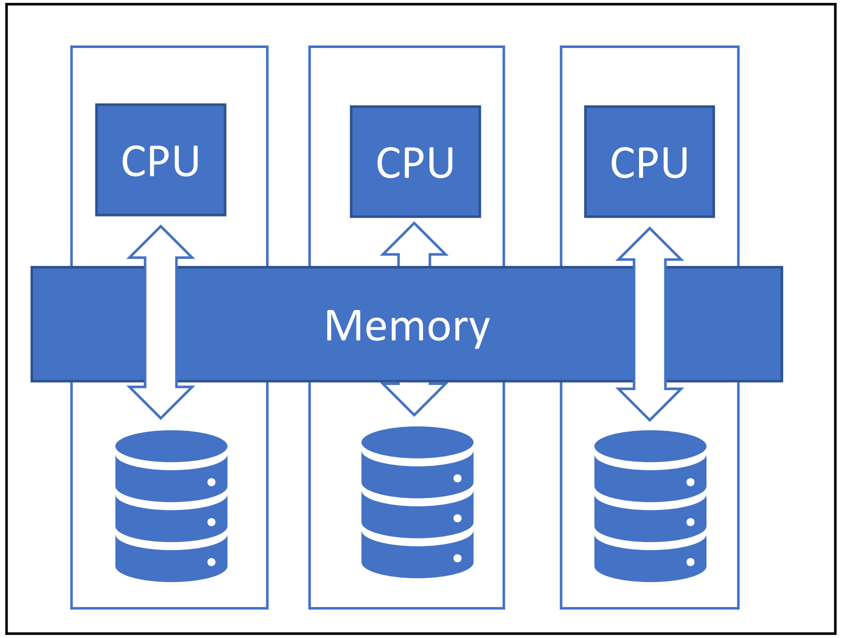
- Consider if you have a single database server, and you’re able to add more computing power and disk
  - Naively, you might treat the entire system as a single processing unit
  - CPUs share the same disk and the same memory area
  - A single query might be assigned (in turn) to different CPUs to process

Different approaches to processing

Symmetric multi-processing (SMP)



Massively Parallel Processing (MPP)



Adding more servers

- Consider how to spread data (and tasks) amongst each node

- Heterogeneous vs Homegeneous nodes
  - Are nodes identical, or are they different?

- Physical location
  - are your nodes located close together, or geographically far apart?
  - **Latency** (the time it takes to transmit data between nodes) quickly becomes a factor

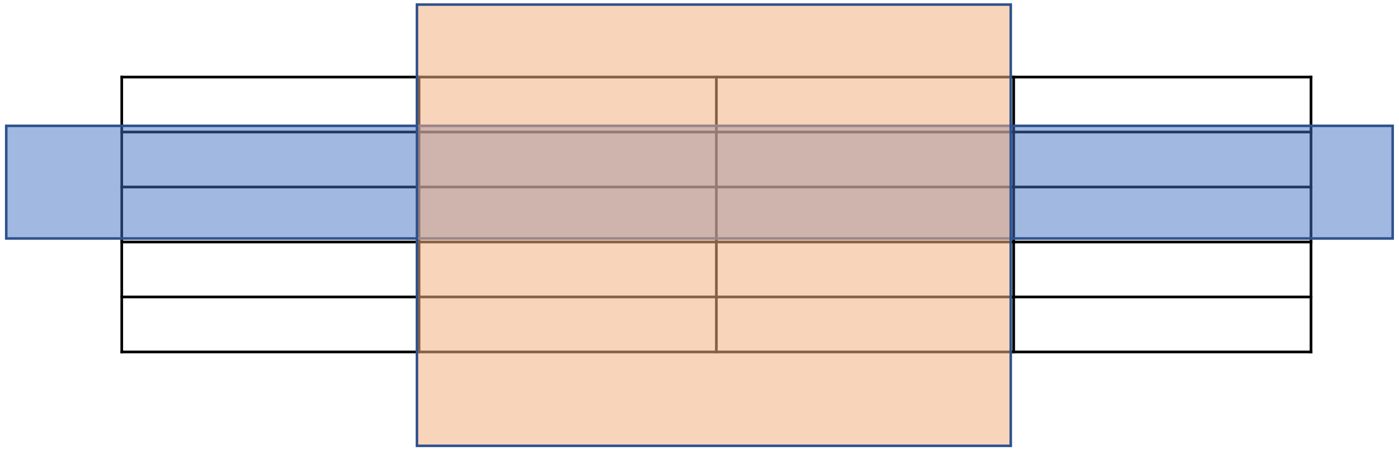
How should clusters share resources?

- There are three typical styles for resource sharing in a distributed database management system

	Memory	Disk
Shared all	Shared	Shared
Shared disk	Not shared	Shared
Shared nothing	Not shared	Not shared

Partitioning

- How should data (and operations) be distributed amongst each node?

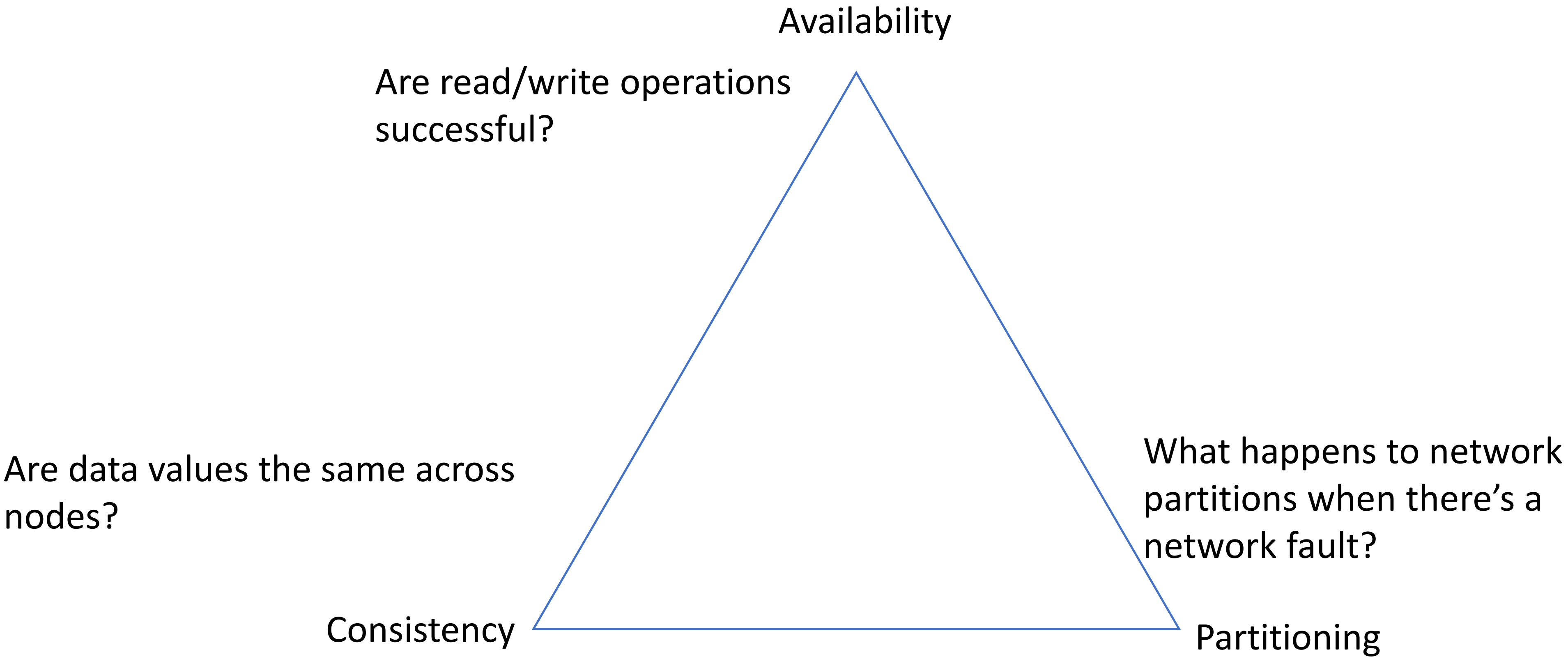


- Horizontally (each node receives a subset of records)
- Vertically (each node receives a piece of each record)
- Often referred to as **sharding** in NoSQL systems

NoSQL and Distribution

- Because NoSQL database management is much simpler, they can be easier to manage at scale across many nodes. Consider the following:
  - Joins
  - Constraints
  - Transactions
- NoSQL databases provide more flexibility when nodes are distributed across a network

The CAP Theorem



Availability

Are read/write operations successful?

Are data values the same across nodes?

Consistency

What happens to network partitions when there’s a network fault?

Partitioning

The coming weeks

	Topic	What we scale
November 25	Data Mining and Warehousing	Breadth and depth of data
November 27	Big Data	Adding more servers
December 2	Specialty databases	Kinds of data
★ December 4	Cloud computing	Networks

USRIs