# Distributed Data Processing

Big Data Management

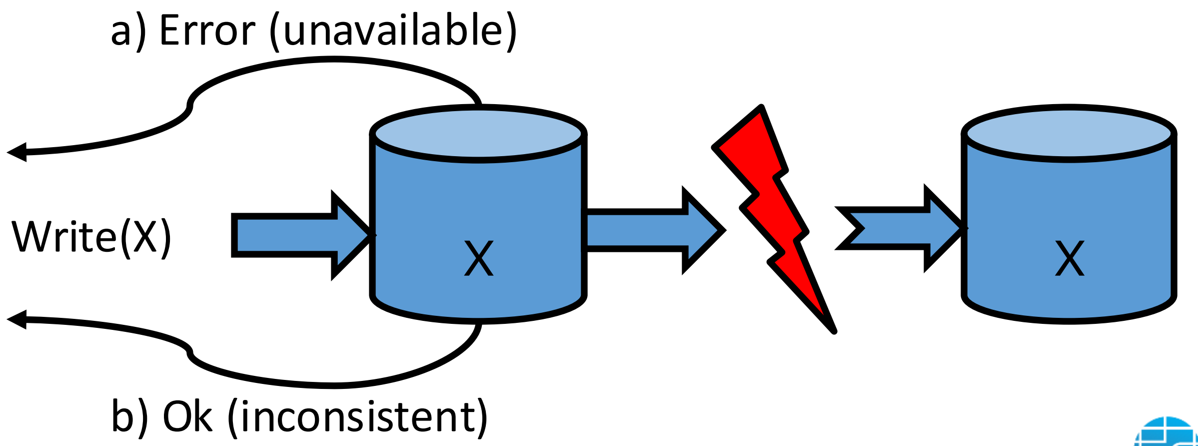
# (Distributed) Transaction Management

# CAP theorem

"We can only achieve two of Consistency, system Availability, and tolerance to network Partition."

Eric Brewer

- Consistency (C) equivalent to a single up-to-date copy of the data
- High availability (A) of the data (for updates)
- Tolerance to network partitions (P).



# Configuration alternatives

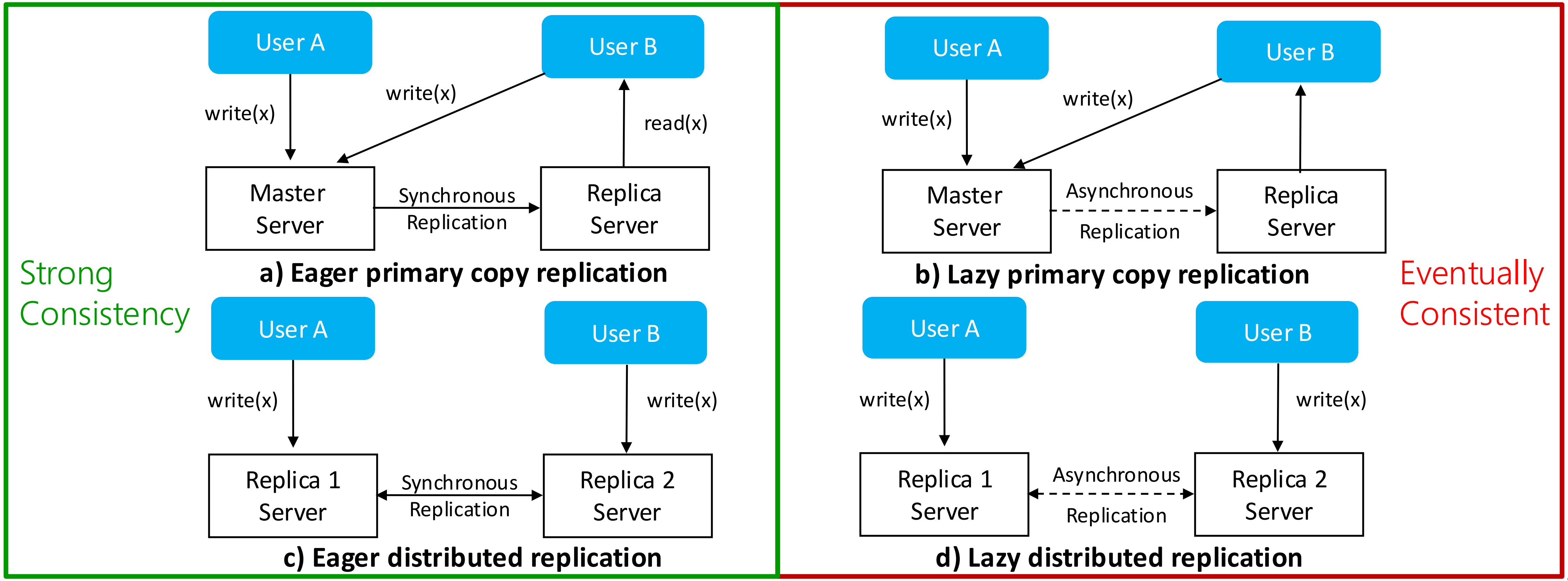
a) Strong consistency
   - Replicas are synchonously modified and guarantee consistent query answering
   - The whole system will be declared not to be available in case of network partition

b) Eventually consistent
   - Changes are asynchronously propagated to replicas so answer to the same query depends on the replica being used
   - In case of network partition, changes will be simply delayed

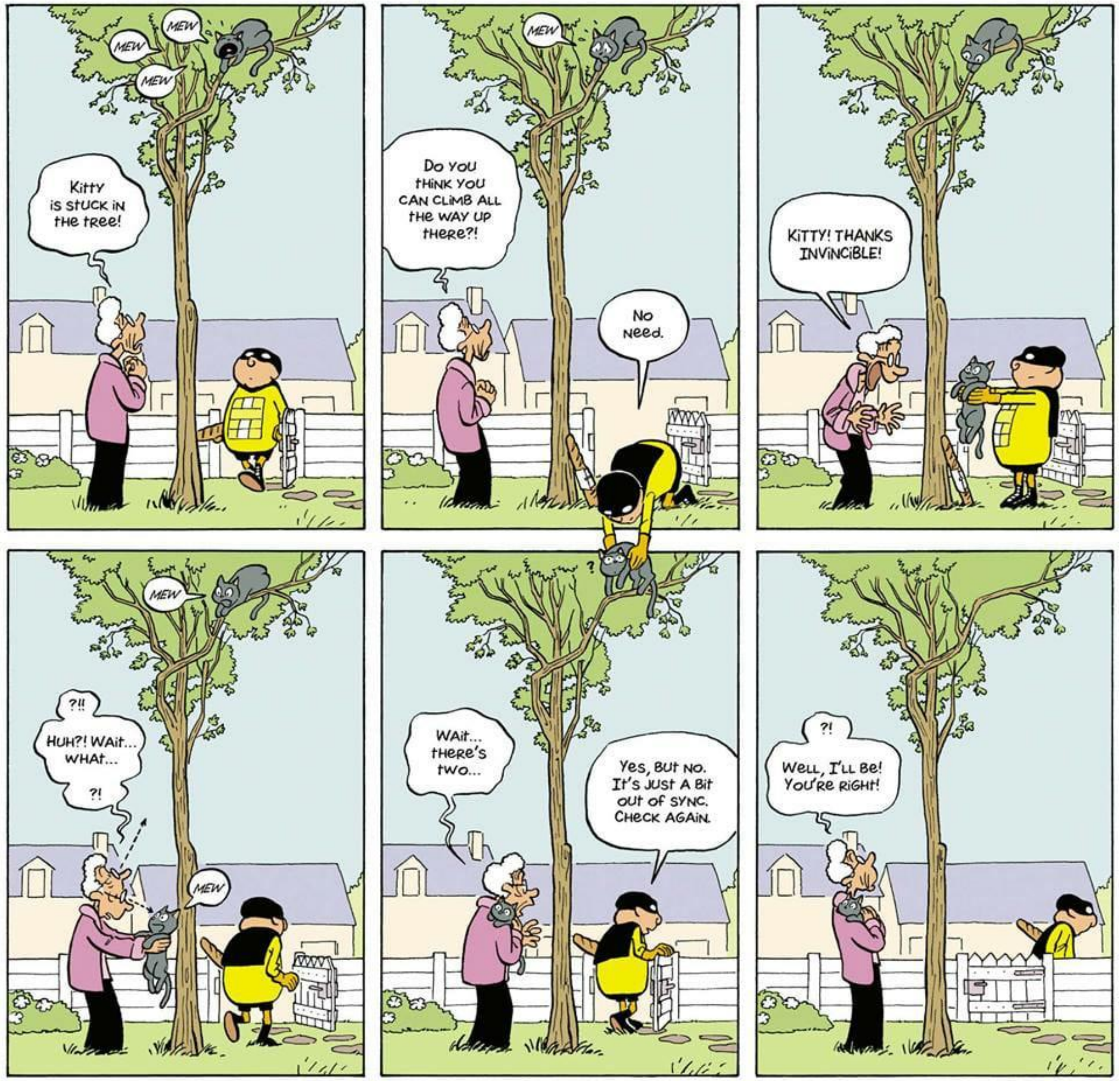
c) Non-distributed data
   - Connectivity cannot be lost
   - We can have strong consistency without affecting availability

# Managing replicas

- Replicating fragments improves query latency and availability
  - Requires dealing with consistency and update (a.k.a., synchronization) performance
- Replication protocols characteristics
  - Primary – Distributed versioning
  - Eager – Lazy replication



# Eventual consistency




Justin Travis Waith-Mair

# (Distributed) Query Processing

# Challenges in distributed query processing

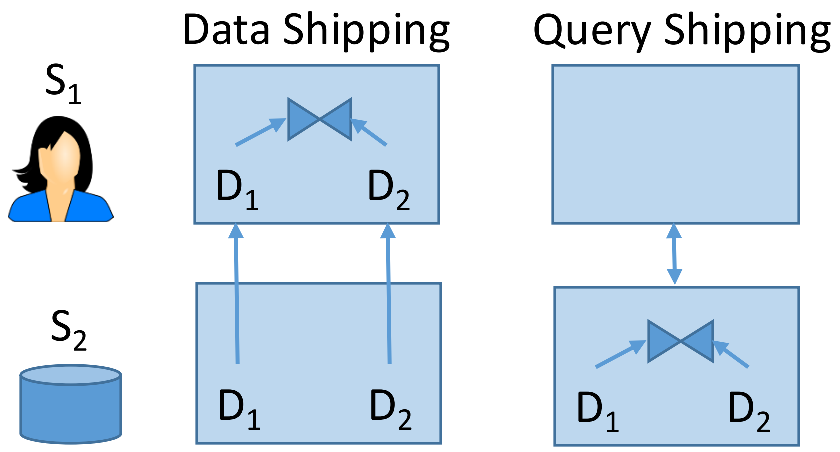
- Communication cost (data shipping)
  - Not that critical for LAN networks
    - Assuming high enough I/O cost
- Fragmentation / Replication
  - Metadata and statistics about fragments (and replicas) in the global catalog
- Join Optimization
  - Joins order
  - Semi-join strategy
- How to decide the execution plan
  - Who executes what
  - Exploit parallelism (!)

A centralized optimizer minimizes the number of accesses to disk

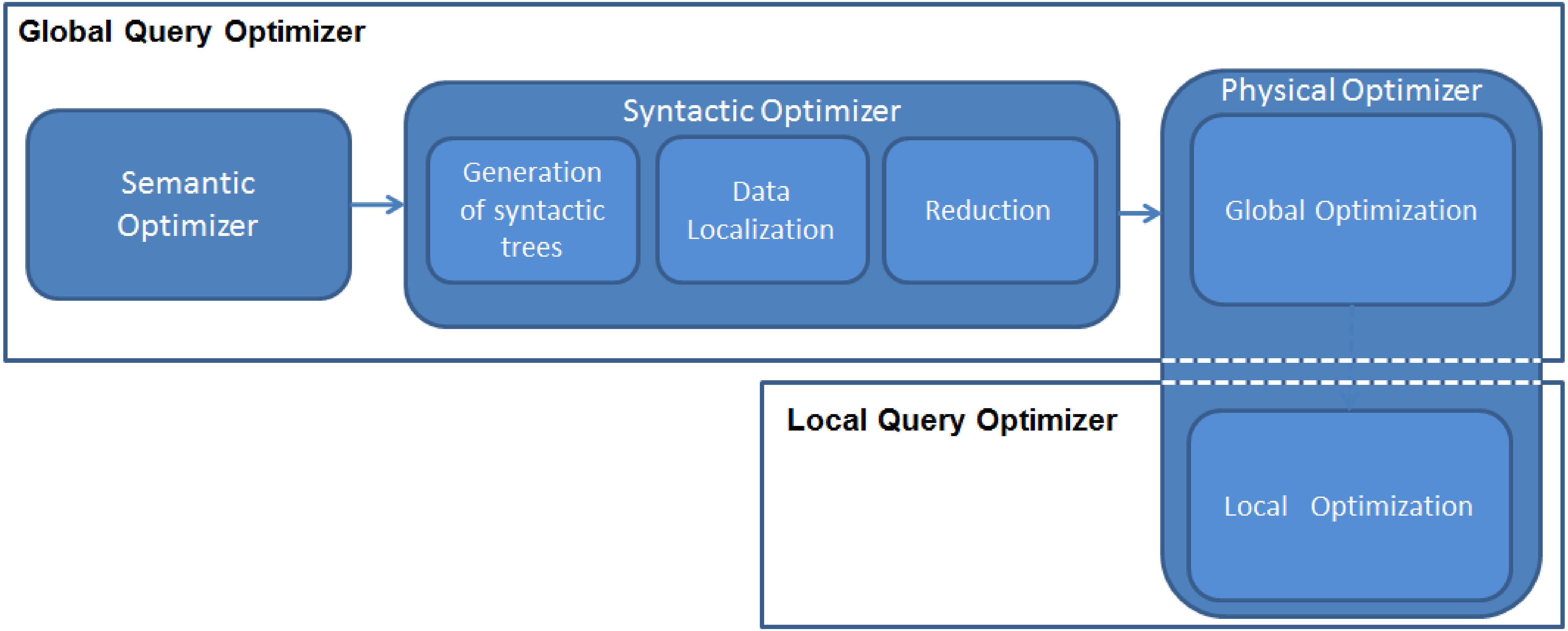
A distributed optimizer minimizes the use of network bandwidth

# Allocation selection

- Data shipping
  - The data is retrieved from the stored site to the site executing the query
    - Avoid bottlenecks on frequently used data
- Query shipping
  - The evaluation of the query is delegated to the site where it is stored
    - To avoid transferring large amounts of data
- Hybrid strategy
  - Dynamically decide data or query shipping

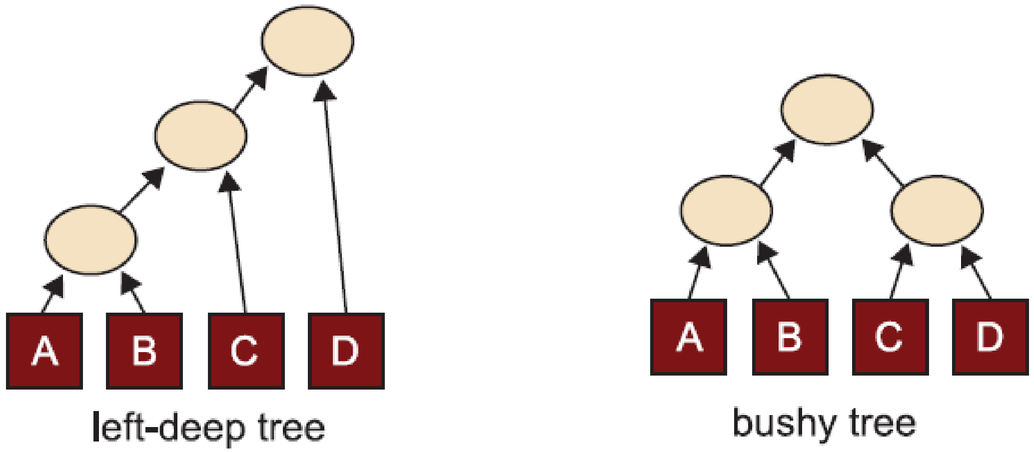


# Phases of distributed query processing



# Syntactic optimizer

- Ordering
  - Left or right deep trees
  - Bushy trees




Hard to parallelize

Easy to parallelize

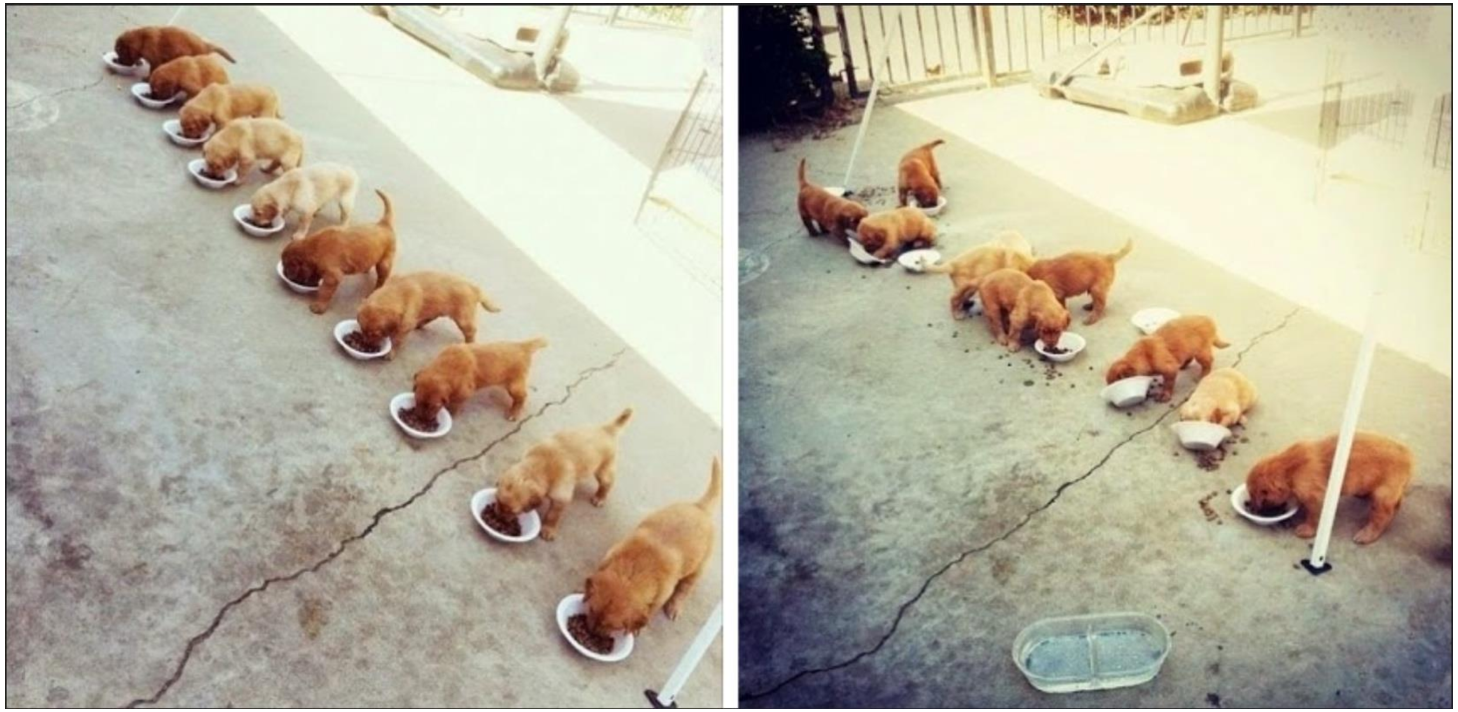
- Added difficulties
  - Consider multi-way joins
  - Consider the size of the datasets
    - Specially the size of the intermediate joins

# Physical optimizer

- Transforms an internal query representation into an <u>efficient plan</u>
  - Replaces the logical query operators by specific algorithms (plan operators) and access methods
  - Decides in which order to execute them
    - **<u>Parallelism (!)</u>**
  - Selects where to execute them (exploit **<u>Data Location</u>**)
    - More difficult for joins (multi-way joins)
- This is done by...
  - Enumerating alternative but equivalent *plans*
  - Estimating their costs
  - Searching for the best solution
    - Using available statistics regarding the physical state of the system

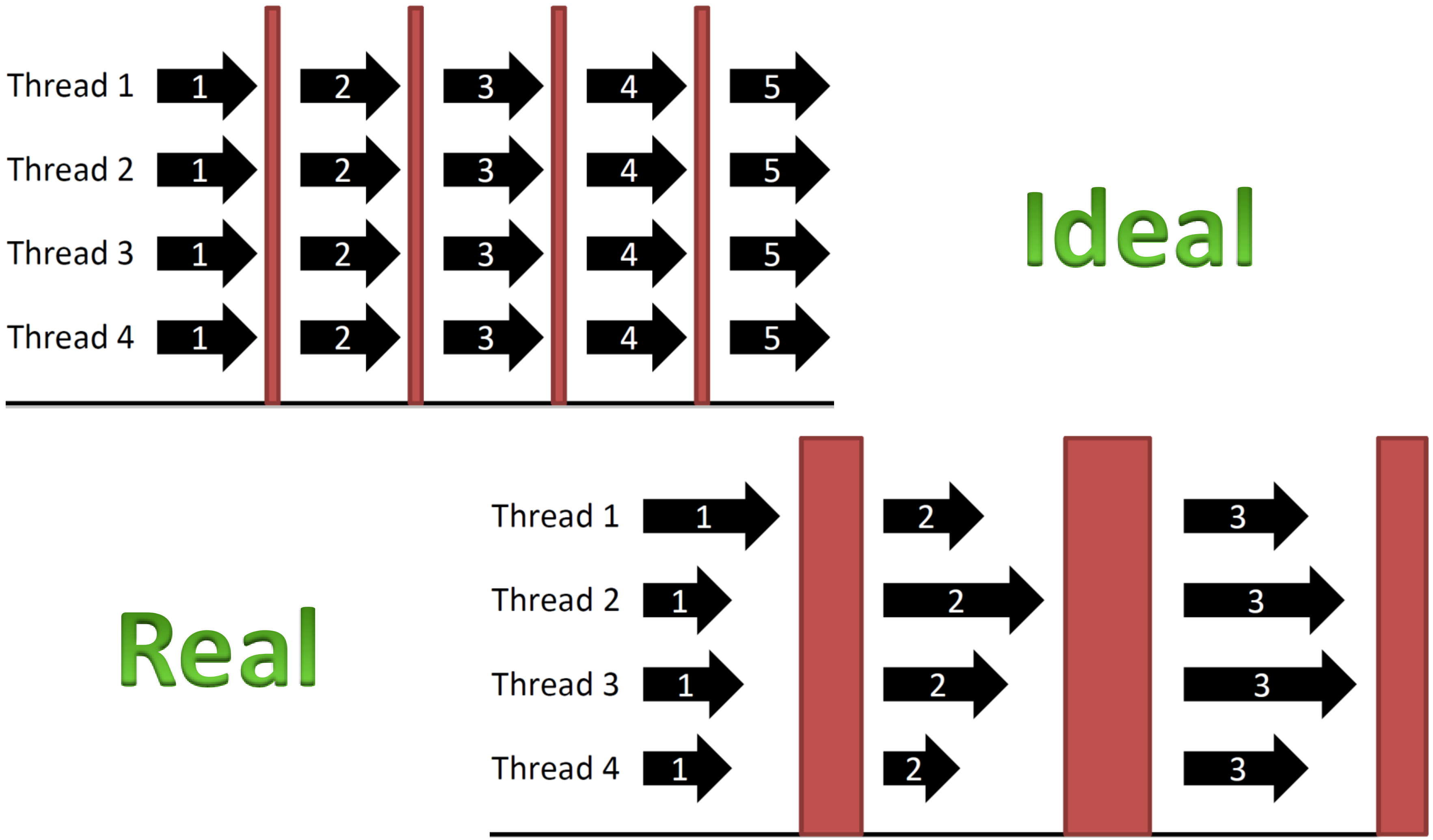
# The problem of parallelism

Theory Practice



Samuel Yee

# Bulk Synchronous Parallel Model




SAILING lab slides

# Measures for parallelism

Ahmdal's law and the Universal Scalability Law

# Parallel processing

- Goal
  - Reduce response time / increase throughput
    - Employ parallel hardware effectively
- Means
  - Process pieces of input in different processors
    a) Divide the dataset into (disjoint) subsets
    b) Adapt serial algorithms to multi-thread environments

# Measuring scalability

- Scalability is normally measured in terms of
  - Speed-up – measuring performance when adding hardware for a constant problem size
    - Linear speed-up means that N sites solve in T/N time, a problem solved in T time by a sequential version of the code
  - Scale-up - measuring performance when the problem size is altered with resources
    - Linear scale-up means that N sites solve a problem N times bigger in T time, the same code run in 1 site solves the same problem also in T time

# Amdahl's law

- Principle - parallel computing with many processors is useful only for highly parallelizable programs
- Amdahl's law measures the maximum improvement possible by improving a particular part of a system
  - Sequential/serial processing vs.
  - Parallel processing
- $S(p, N)$ - theoretical speed-up, where $p$ is the fraction of parallelizable code using $N$ processors (hardware)

# Amdahl's law

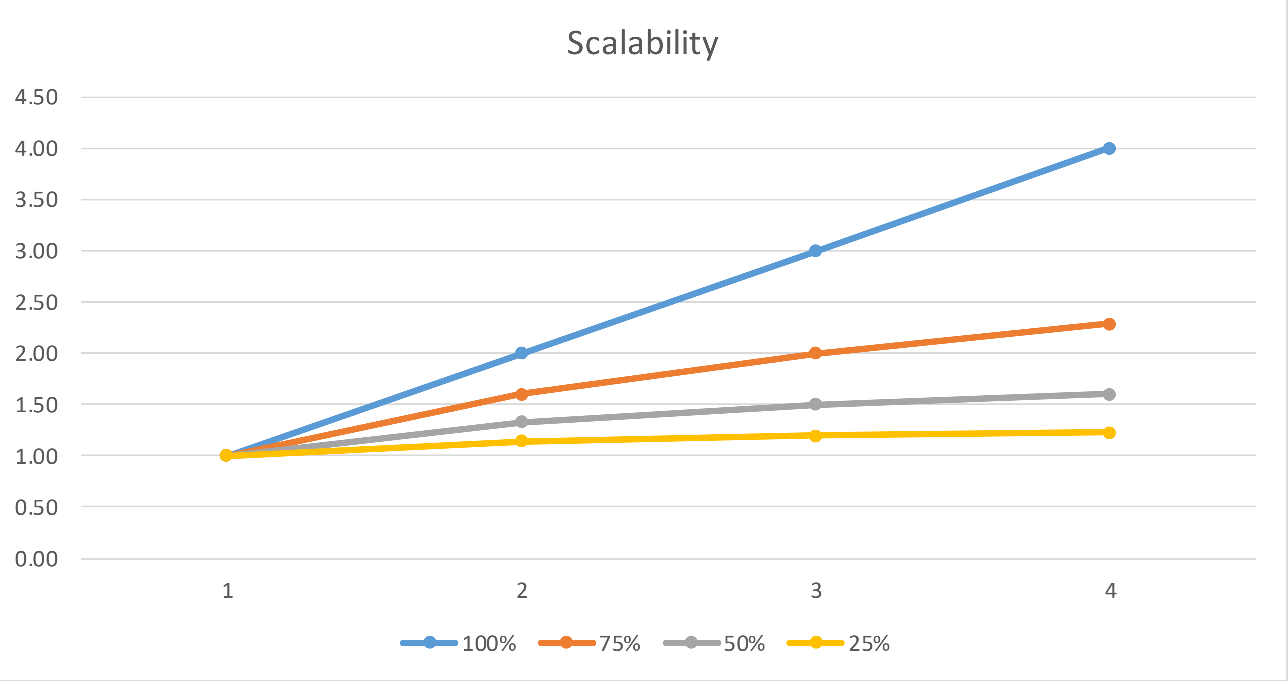
$$S(p,N) = \frac{N}{N - p(N-1)}$$

$$S(p,N) = \frac{N}{N - pN + p}$$

$$S(p,N) = \frac{1}{\frac{N - pN + p}{N}}$$

$$S(p,N) = \frac{1}{1 - p + \frac{p}{N}}$$

| N\p | 100% | 75% | 50% | 25% |
|---|---|---|---|---|
| 1 | 1.00 | 1.00 | 1.00 | 1.00 |
| 2 | 2.00 | 1.60 | 1.33 | 1.14 |
| 3 | 3.00 | 2.00 | 1.50 | 1.20 |
| 4 | 4.00 | 2.29 | 1.60 | 1.23 |



# Universal Scalability Law (I) – Generalization of Amdahl's law

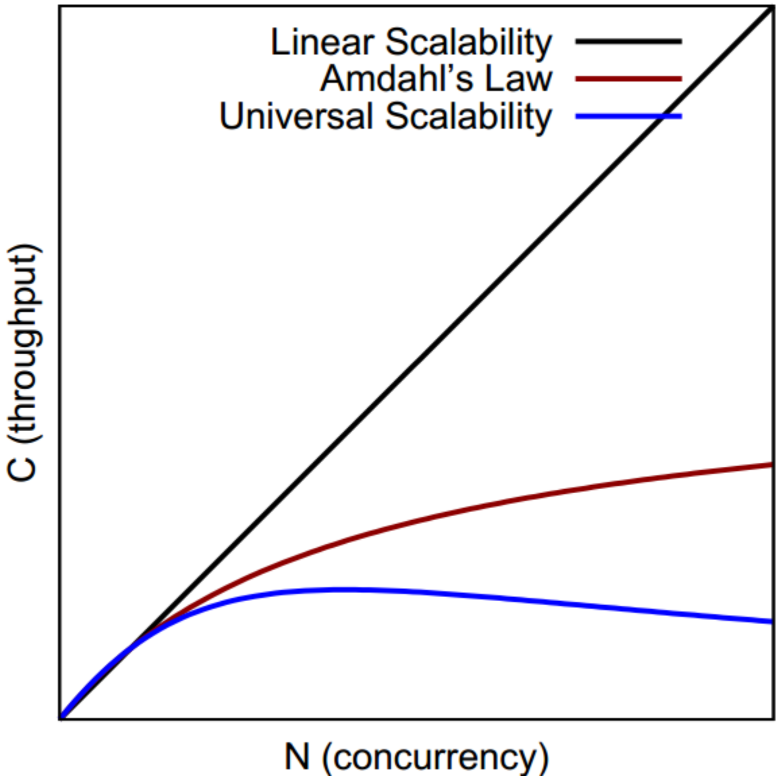
- Mathematical definition of scalability (both for SW or HW scalability)
  - Shows that linear scalability is hardly achievable
- USL is defined as follows

$$C(N) = \frac{N}{1 + \sigma(N-1) + \kappa N(N-1)}$$

- C: System's capacity (i.e., throughput) improvement (e.g., increment of #queries per second)
- N: System's size (SW - number of concurrent threads)/(HW - number of CPUs)
- σ: System's contention (performance degradation due to serial instead of parallel processing)
  - Could be avoided (i.e., σ = 0) if our code has no serial chunks (everything parallelizable)
- κ: System's consistency delay (aka coherency delay), extra work needed to keep synchronized shared data (i.e., inter-process communication)
  - Could be avoided (i.e., κ = 0) if replicas can be synchronized without sending messages

# Universal Scalability Law (II)

- If κ = 0, it simplifies to Amdahl's law
- If both σ = 0 and κ = 0, we obtain linear scalability




https://www.percona.com/sites/default/files/white-paper-forecasting-mysql-scalability.pdf

# Closing

# Summary

- Distributed Transaction Management
  - CAP theorem
  - Eventual consistency
- Distributed Query Processing
  - Kinds of Parallelism
  - Cost estimation
- Scalability measures
  - Amdahl's law
  - Universal scalability law

# References


- G. Graefe. *Query Evaluation Techniques*. In ACM Computing Surveys, 25(2), 1993
- L. Liu, M.T. Özsu (Eds.). *Encyclopedia of Database Systems*. Springer, 2009
- M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*, 3rd Ed. Springer, 2011
- L. G. Valiant. *A bridging model for parallel computation*. Commun. ACM. August 1990