Principles of Distributed Databases

Knowledge Objectives

1. Name the two main characteristics of a distributed database
2. Explain what are the benefits of data fragmentation and data replication performance-wise
3. Explain the 7 degrees of transparency a DDBMS might provide
4. Enumerate the 4 kinds of distributed databases attending to their autonomy
5. Justify the need of new schema levels in the ANSI-SPARC architecture due to data distribution
6. Enumerate and justify the need of new components in the functional architecture of a DDBMS
7. Enumerate the contents of the global catalog
8. Explain 7 bottlenecks of traditional relational systems
9. Explain 6 simplifications for RDBMS in an OLTP environment
10. Enumerate 6 environments that entail different access (and consequently storage) characteristics
11. Explain what are the main goals behind the NOSQL wave
12. Enumerate 4 challenges in data distribution database

Understanding Objetives

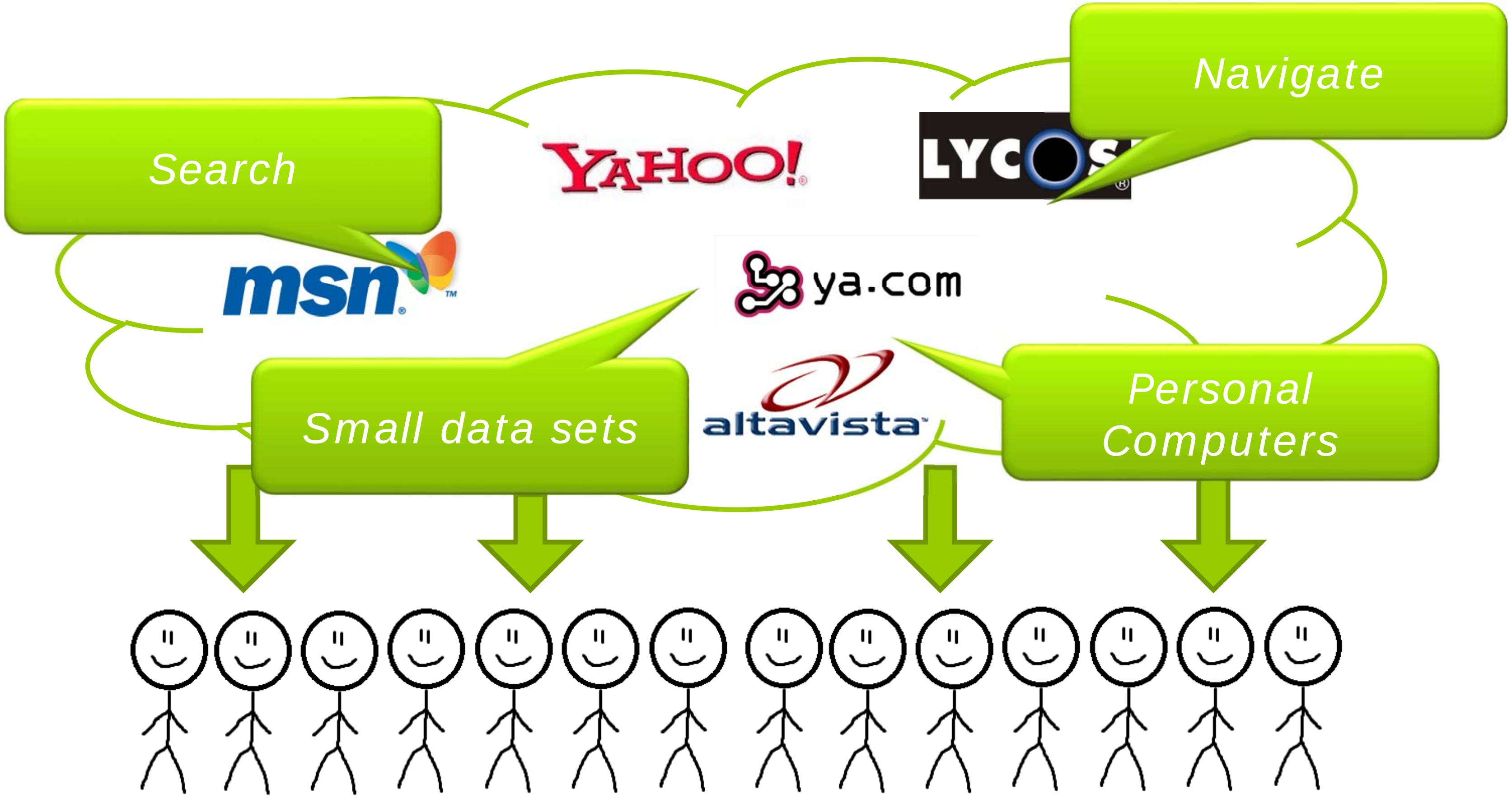
1. Compare the latency and response time of a query in both centralized and distributed storage

Principles of Distributed Databases

MOTIVATION

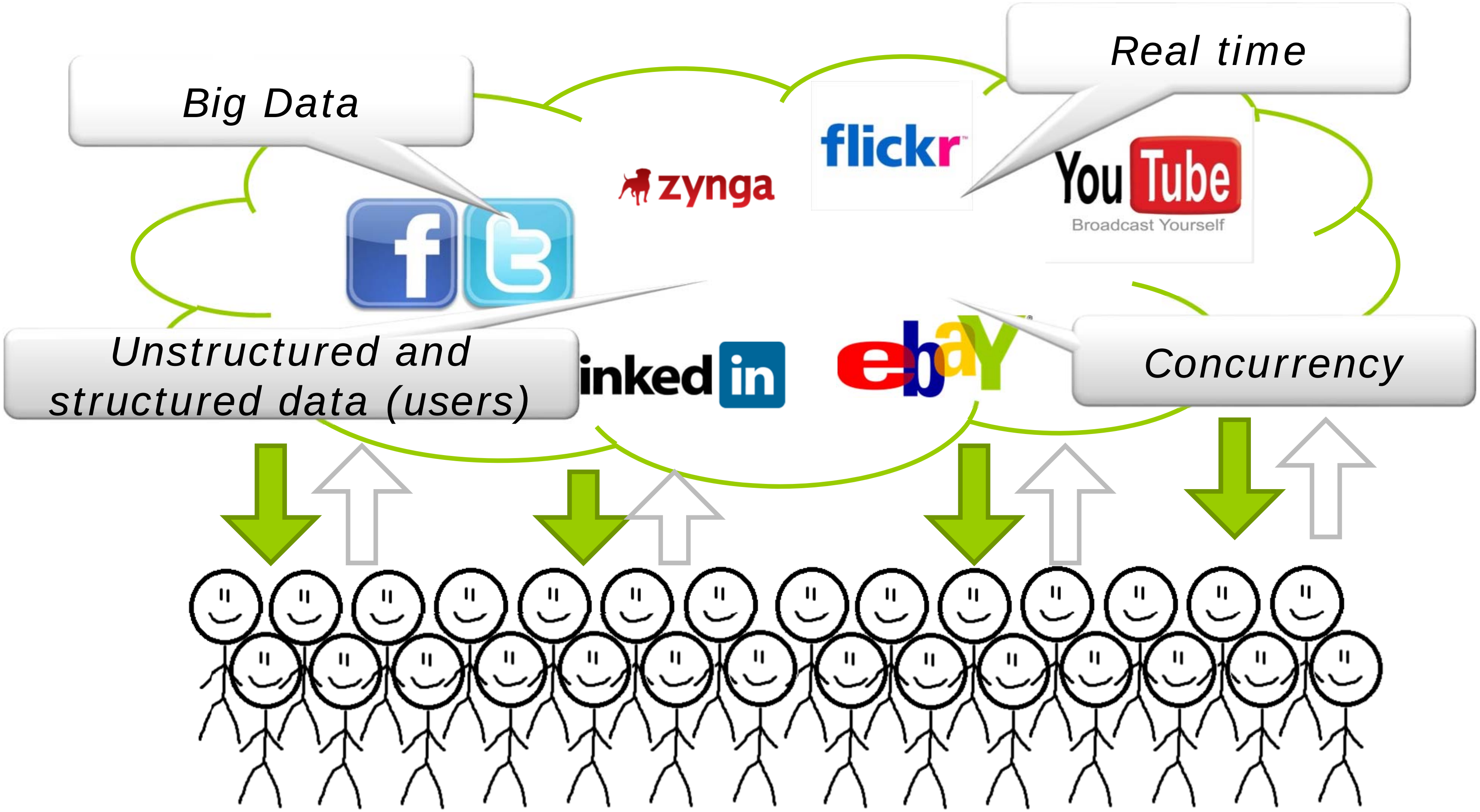
The End of an Architectural Era

WEB 1.0 – Read Era



The End of an Architectural Era

WEB 2.0 – Write Era



RDBMS: One Size Does Fit All

- Mainly write-only Systems (e.g., OLTP)
  - Data storage
    - Normalization
  - Queries
    - Indexes: B+, Hash
    - Joins: BNL, RNL, Hash Join, Merge Join
- Read-only Systems (e.g., DW)
  - Data Storage
    - Denormalized data
  - Queries
    - Indexes: Bitmaps
    - Joins: Star-join
    - Materialized Views

BUT, WHAT IF I NEED TO DEAL WITH MASSIVE READS AND WRITES AT THE SAME TIME?

RDBMS Approach

Too many reads? → data replication

Too many writes? → data fragmentation

- Distributed RDBMS (DRDBMS) are not flexible enough
  - ACID properties must be preserved when synchronizing nodes
  - Too much logging writes
  - They do not scale well

Principles of Distributed Databases

DISTRIBUTED DATABASES

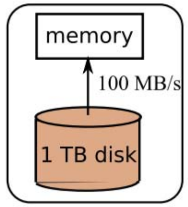
Activity: Why Distribution?

- Objective: Recognize the benefits of distributing data
- Tasks:
  1. (15') By pairs, answer the following questions:
     a) How long would it take to read 1TB with sequential access (fig. a)? (in secs)
        - Can you identify any additional drawback to be considered?
     b) How long would it take to read 1TB with parallel access (fig. b)? Assume 100 disks on the same machine with shared-memory and infinite CPU capacity.
        - Can you identify any additional drawback to be considered?
     c) How long would it take to read 1TB with distributed access (fig. c)? Assume 100 shared-nothing machines in a star-shape LAN in a single rack where all data is sent to the center.
        - Can you identify any additional drawback to be considered?
     d) Now, repeat the exercise considering a single random access. What changes?
  2. (5') Discussion

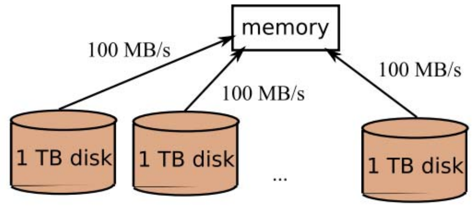
Type	Latency	Bandwidth
Disk	$\approx 5 \times 10^{-3}$s (5 millisec.);	At best 100 MB/s
LAN	$\approx 1 - 2 \times 10^{-3}$s (1-2 millisec.);	$\approx$ 1GB/s (single rack); $\approx$ 100MB/s (switched);
Internet	Highly variable. Typ. 10-100 ms.;	Highly variable. Typ. a few MB/s.;

Bottom line (1): it is approx. one order of magnitude faster to exchange main memory data between 2 machines in a data center, that to read on the disk.

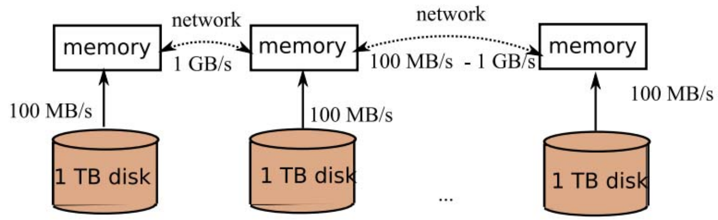
Bottom line (2): exchanging through the Internet is slow and unreliable with respect to LANs.



a. Single CPU, single disk



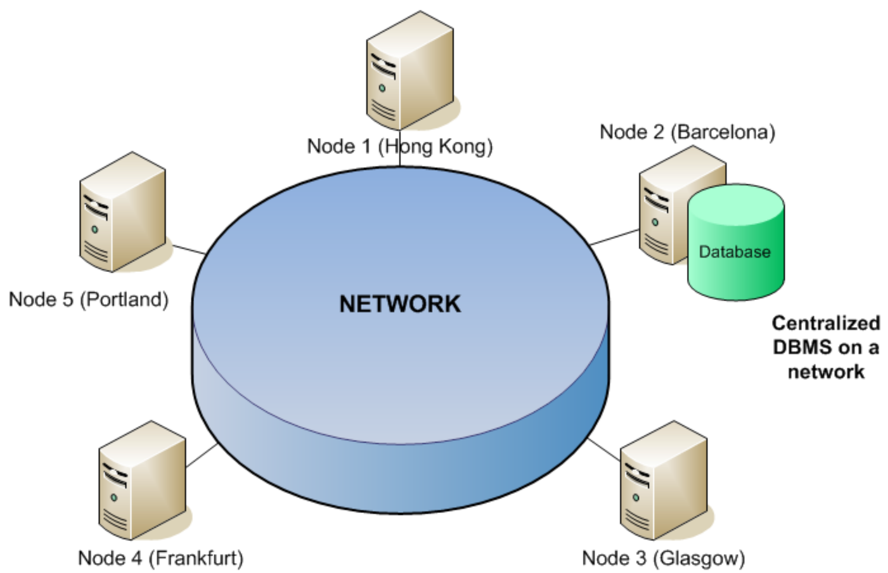
b. Parallel read: single CPU, many disks

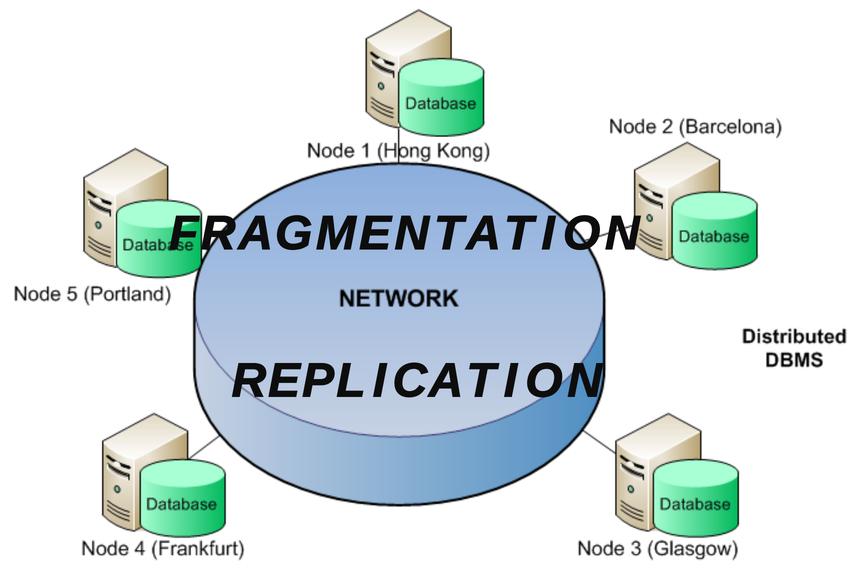


c. Distributed reads: an extendible set of servers

Distributed Database

- A distributed database (DDB) is a database where <u>data management</u> is distributed over several nodes in a network.
  - Each node is a database itself
    - Potential heterogeneity
  - Nodes communicate through the network

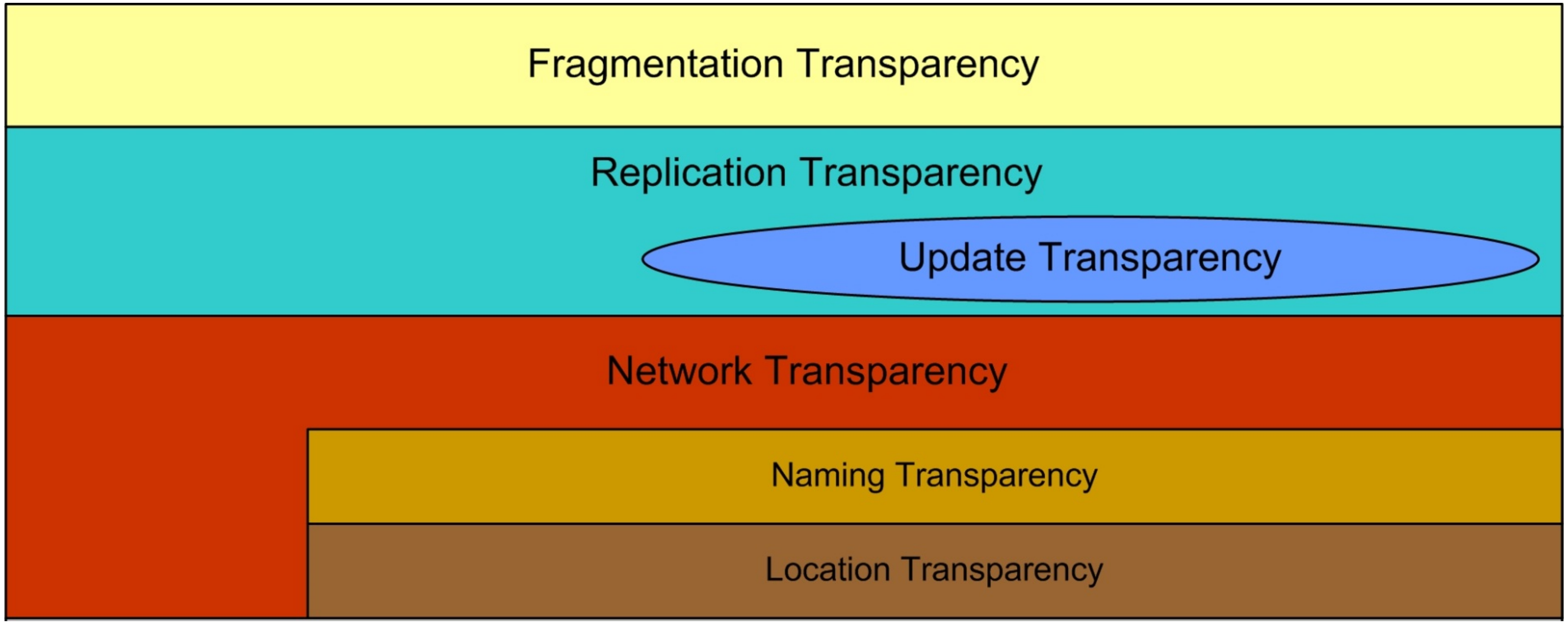




Distributed Architectures

- Main objective: hide implementation (i.e., physical) details to the users
  - Data independency at the logical and physical level must be guaranteed
    - Inherited from centralized DBs (ANSI SPARC)
  - Network transparency
    - Data access must be independent regardless where data is stored
    - Each data object must have a unique name
  - Replication transparency
    - The user must not be aware of the existing replicas
  - Fragmentation transparency
    - The user must not be aware of partitioning

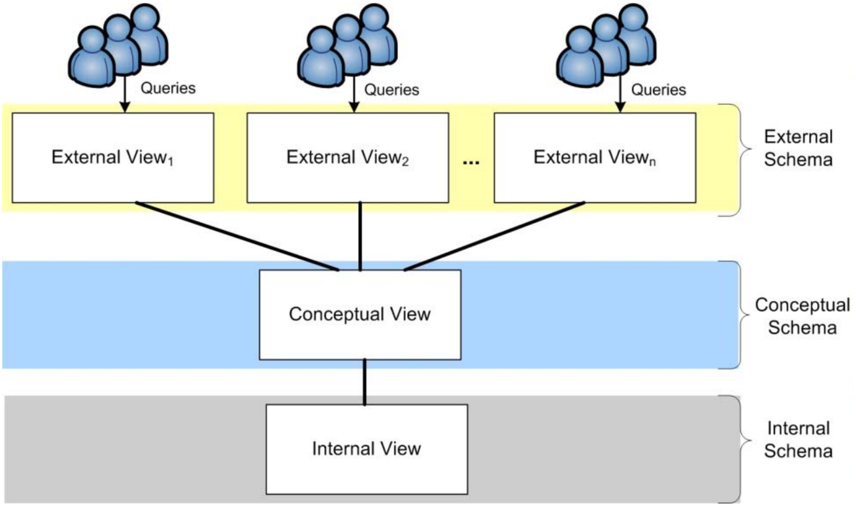
Distributed Architectures

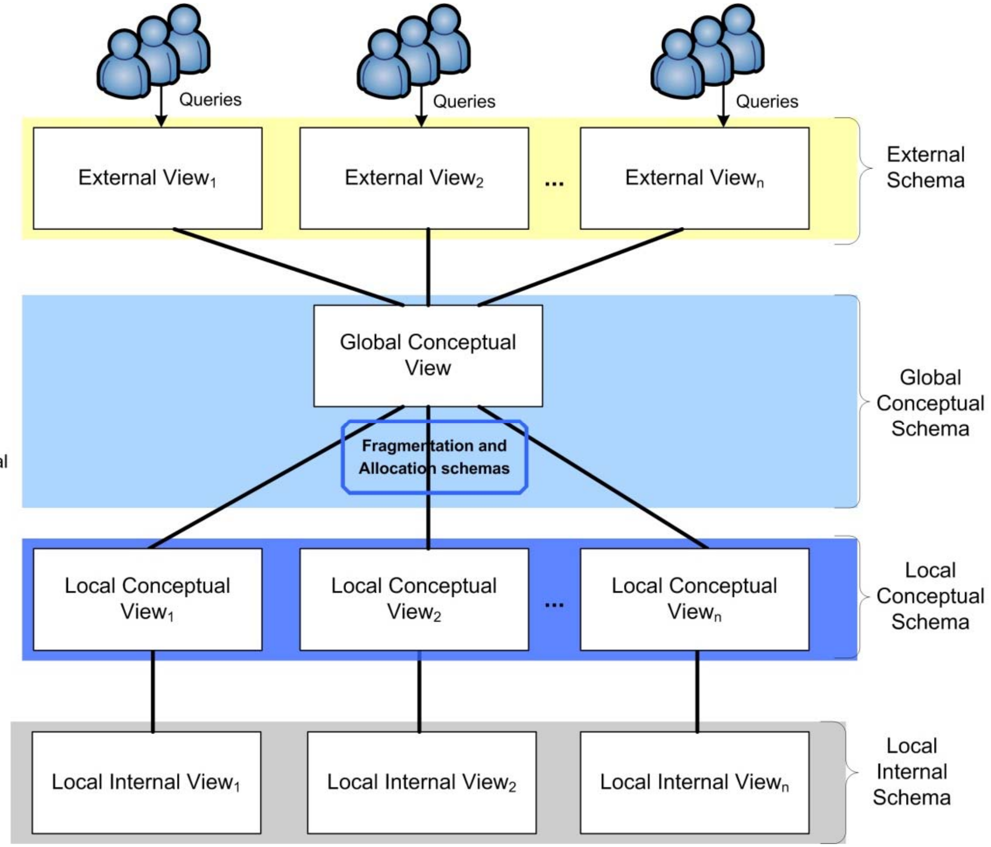


Comparison of DDBs

	Autonomy	Central Schema	Query Transparency	Update Transparency
Homogeneous DBs	*No*	*Yes*	*Yes*	*Yes*
Tightly Coupled Federated DBs	*Low*	*Yes*	*Yes*	*Limited*
Loosely Coupled Federated DBs	*Medium*	*No*	*Yes*	*Limited*
Multi-databases	*High*	*No*	*No*	*No*

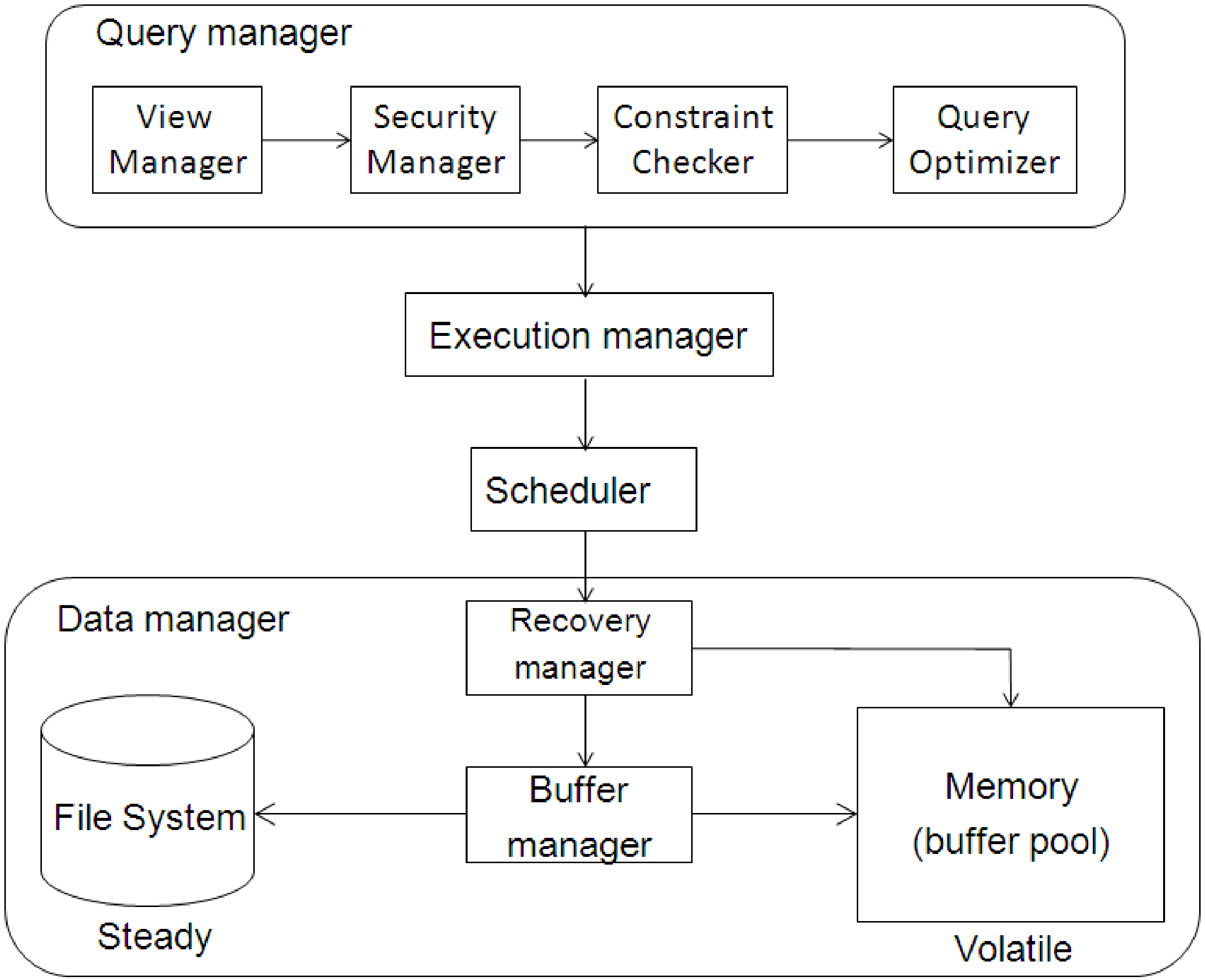
Extended ANSI-SPARC Architecture



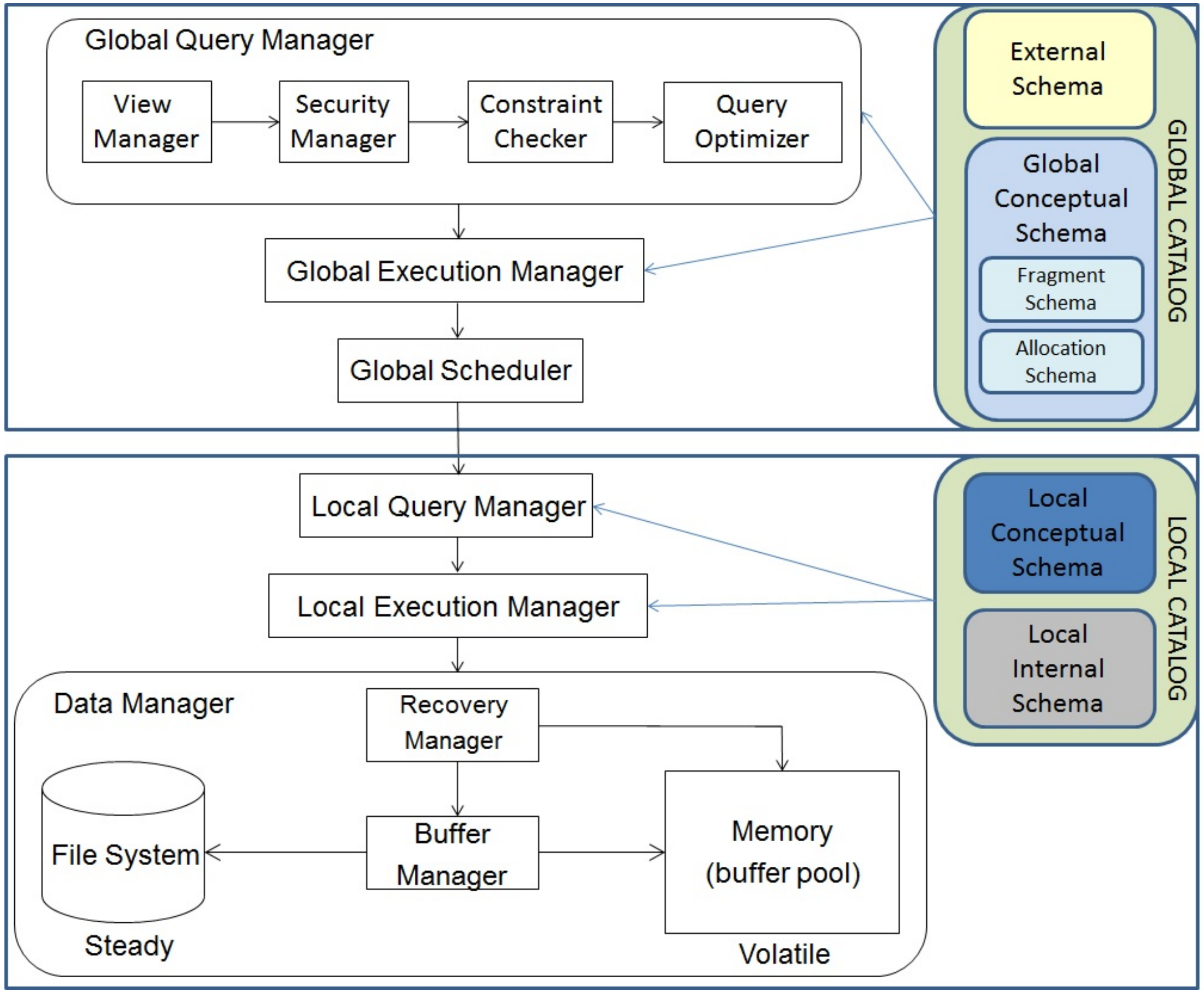


- Global catalog
  - Mappings between ESs – GCS and GCS – LCSs
- Each node has a local catalog
  - Mappings between LCS_i – IS_i

Centralized DBMS Architecture



Distributed DBMS Architecture



Principles of Distributed Databases

NOSQL

Different applications

Not Only SQL (different problems entail different solutions)

- OLTP
  - Object-Relational
    - Distributed databases
    - Parallel databases
- Scientific databases and other Big Data repositories
  - Key-value stores
- Data Warehousing & OLAP
  - MOLAP
  - Column stores
  - Multidimensional features
- Text / documents
  - Document databases
    - XML/JSON databases
- Stream processing
  - Stream processor
- Semantic Web and Open Data
  - Graph databases

The Problem is Not SQL

- Relational systems are too generic...
  - OLTP: stored procedures and simple queries
  - OLAP: ad-hoc complex queries
  - Documents: large objects
  - Streams: time windows with volatile data
  - Scientific: uncertainty and heterogeneity
- ... But the overhead of RDBMS has nothing to do with SQL
  - Low-level, record-at-a-time interface is not the solution

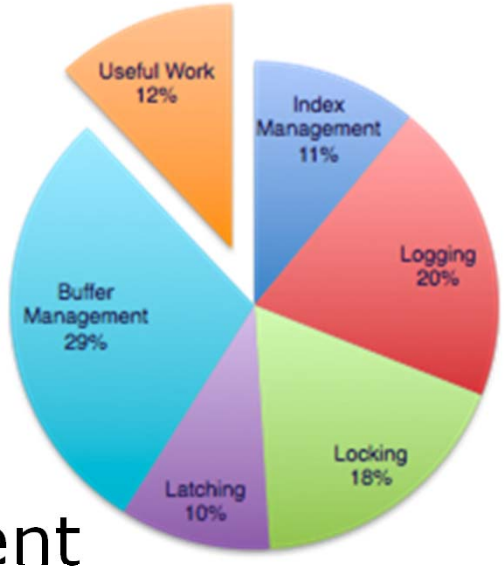
SQL Databases vS. NoSQL Databases
Michael Stonebraker
Communications of the ACM, 53(4), 2010

Activity: RDBMS Bottlenecks

- *Objective: Identify RDBMSs main flaws when used to implement massive distributed systems*
  - *Thus, assume* main memory storage, built-in high availability, no user stalls, and useful transaction work under 1 millisecond
- *Tasks:*
  1. *(5') Read the bottlenecks assigned to you:*
     - I. *Logging*
     - II. *JDBC connection and latching*
     - III. *Concurrency control and distributed commit protocols*
  2. *(15') In group of 3 discuss:*
     1. *Each of you start explaining the bottlenecks assigned to you assuming the kind of massive distributed system described above*
     2. *Now, try to generalize them (do not assume a massive distributed system anymore). Can I always get rid of these bottlenecks? In which scenarios could I? In which I could not? Discuss it in terms of storage, running time for transactions and system availability*
  3. *(10') Think tank*
- *Roles for the team-mates during task 2:*
  - a) *Explains his/her material (one at a time)*
  - b) *Asks for clarifications, raises doubts, answers doubts (all)*
  - c) *Mediates and controls time (one at a time)*

RDBMS Bottlenecks

- Buffers management (cache disk pages)
- Logging (WALP)
  - Persistent redo log
  - Undo log
- Concurrency control (locking)
- Latching for multi-threading
- CLI interfaces (JDBC, ODBC, etc.)
- Variable length records management
  - Locate records in a page
  - Locate fields in a record
- Two-phase commit protocol in distributed transactions



NewSQL: A New Architecture for OLTP

But even for distributed, massive OLTP systems RDBMS can be outperformed!

- Main memory DB
  - A DB less than 1Tb fits in memory
    - 20 nodes x 32 Gb (or more) costs less than 50,000US$
  - Undo log is in-memory and discarded on commit
- One thread systems
  - Perform incoming SQL commands to completion, without interruption
    - One transaction takes less than 1ms
  - No isolation needed
- Grid computing
  - Enjoy horizontal partitioning and parallelism
    - Add new nodes to the grid without going down
- High availability
  - Cannot wait for the recovery process
    - Multiple machines in a PeerToPeer configuration
- Reduce costs
  - Human costs are higher than Hw and Sw
    - An expert DBA is expensive and rare
  - Alternative is brute force
    - Automatic horizontal partitioning and replication
      - Execute queries at any replica and updates to all of them
- Optimize queries at compile time

Summary

- What is a distributed DBMS
- Distributed architecture for a DDBMS
  - Distribution transparency
  - Replication transparency
  - Fragmentation transparency
- Benefits of distributed systems
- NOSQL main goals and features

Bibliography

- M.T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Second edition. Prentice Hall, 1999
- Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart. *Web Data Management*. Cambridge Press, 2011.
- L. Liu, M.T. Özsu (Eds.). *Encyclopedia of Database Systems*. Springer, 2009

Recommended Read

- **SQL Databases vS. NoSQL Databases**
  Michael Stonebraker
  Communications of the ACM, 53(4)
  April 2010

- This paper discusses the RDBMS bottlenecks and underlines the fact that NOSQL is indeed an inappropriate term.
  - This is the main reason why, in this course, you will see NOSQL everywhere instead of NoSQL. NOSQL is not the SQL (or RDBMSs) negation but a complementary view meeting the *one size does not fit all* principle