Principles of NOSQL

And their strong ties with distributed databases

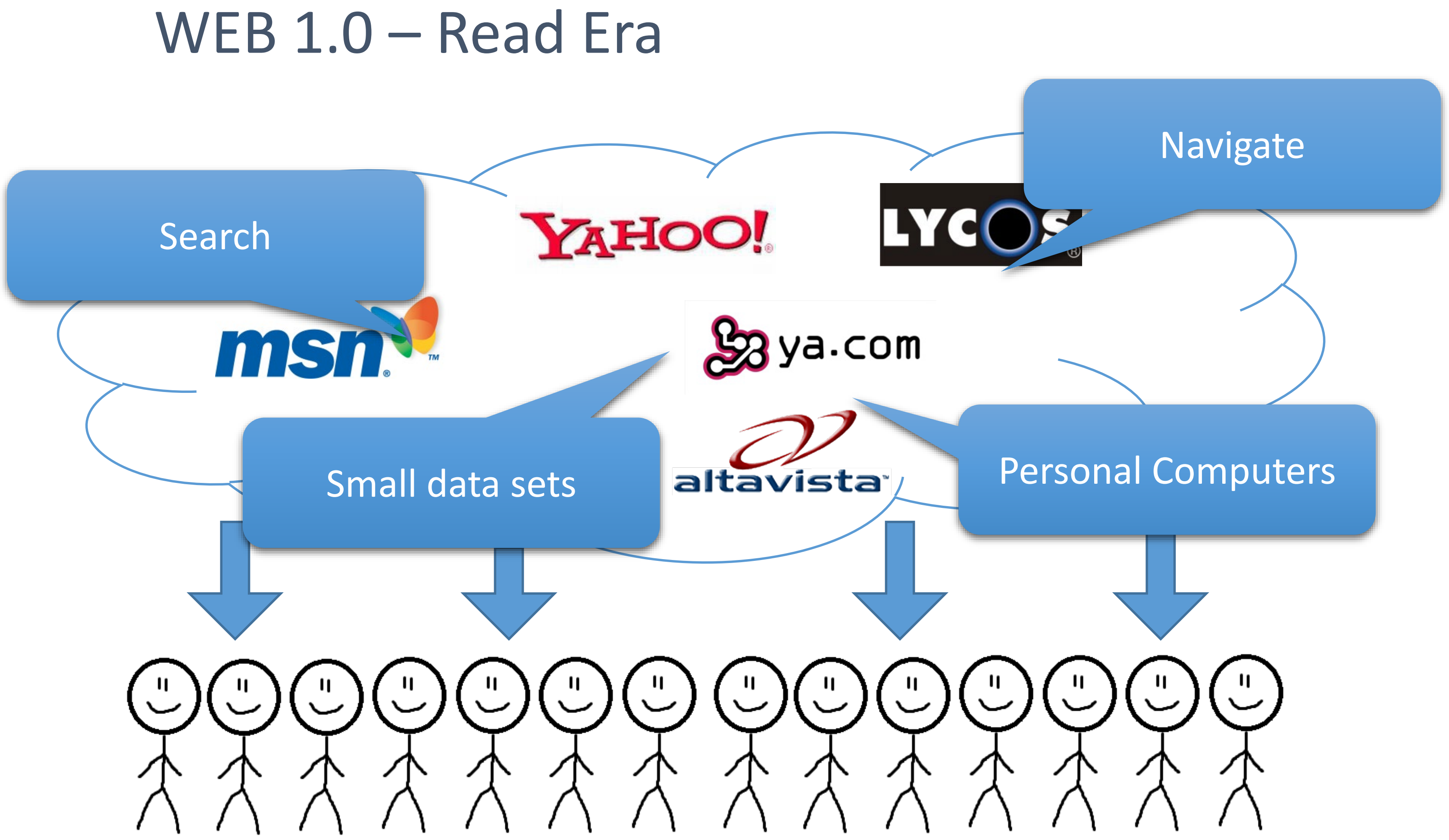
MOTIVATION

Why NOSQL?

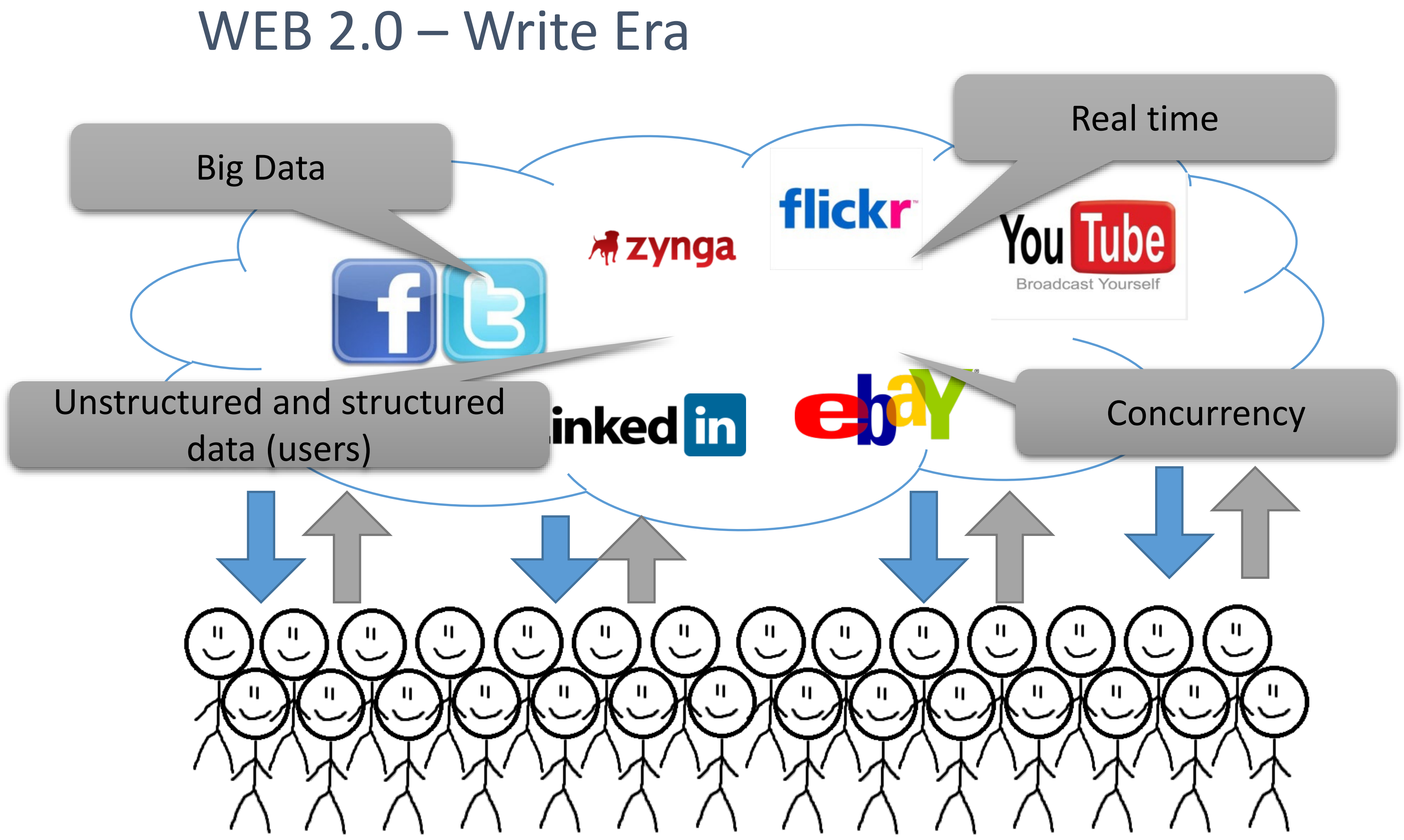
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

DTIM www.essi.upc.edu/dtim

The End of an Architectural Era



The End of an Architectural 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? $\Rightarrow$ data replication

Too many writes? $\Rightarrow$ data fragmentation

- But distributed RDBMS (DRDBMS) are not flexible enough
  - They guarantee the ACID properties when synchronizing nodes
    - Too much logging writes
    - Blocking operations to preserve consistency

  As result, DRDBMS do not scale well
- Further, they are not flexible when ingesting data with unknown schema (the database schema must be predefined at table creation time)

NOSQL

New Problems New Solutions

Why NOSQL?

- The Web 2.0 represented the first use case ever where RDBMS struggled
  - **Volume**: large volumes of data read / updated at the same time
  - **Variety**: many sources where not relational, but text, JSONs, or any other format
  - **Velocity**: the arrival ratio of data plus freshness requirements when querying are the reason for huge performance drops when problems appear (e.g., a read buffer being filled fester than processed, or lively querying very large datasets when being updated at the same time)
- But many other scenarios came later: e.g., IoT Systems, Industry 4.0, etc. As of today, you may expect to (easily) find projects where these data characteristics, known as the three V's, hold

What is NOSQL?

- NOSQL stands for **Not Only SQL**
- It is an alternative family of database technologies to RDBMS
- They break the one size fits all motto in databases by two diferent means:
  - Firstly, handling the three V's as first-class citizen problems
  - Secondly, promoting the adoption of specialized database solutions for specific problems. For example:
    - OLTP
      - Object-Relational Databases (natively storing objects)
    - Scientific databases and other Big Data repositories
      - Key-value stores (scalability is a must)
    - Data Warehousing & OLAP
      - Column stores (query performance for OLAP-like queries)
    - Text / documents
      - Document databases (native support for XML / JSON)
    - Stream processing
      - Stream processor (data arriving in the form of an infinite stream to be processed)
    - Semantic Web and Highly-Relational Datasets
      - Graph databases (allow to represent relationships as first-class citizens and exploit them natively)

NOSQL Goals

- Schemaless: Allow flexible (even runtime) schema definition
- Reliability / availability: Keep delivering service even if its software or hardware components fail
- Scalability: Continuously evolve to support a growing amount of tasks
- Efficiency: How well the system performs, usually measured in terms of response *time* (latency) and *throughput* (bandwith)

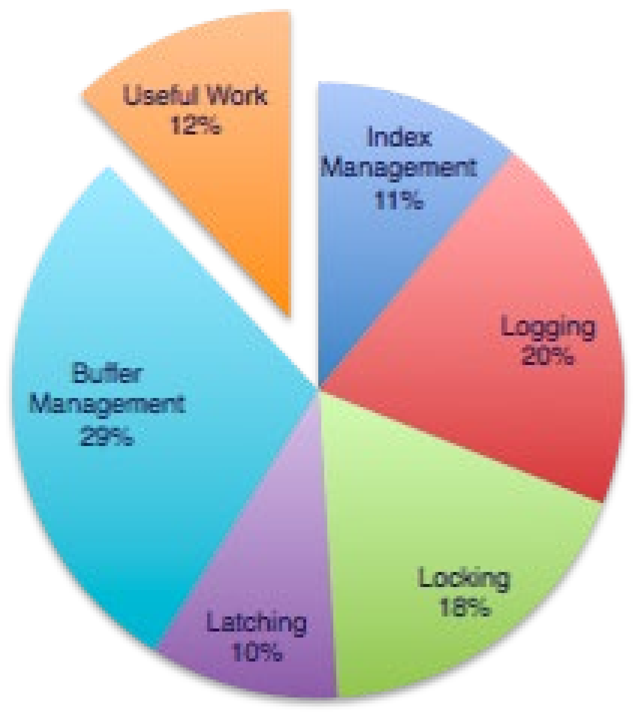
In 2007, a seminal paper identified the main bottlenecks preventing RDBMS from performing well when confronted to these goals

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



The Problem is NOT SQL

- NOSQL promotes alternative data models and system architectures
- …but the RDBMS bottlenecks we discussed and the solutions proposed have nothing to do with SQL!
- Indeed, NOSQL is a cool idea! It is a declarative language that the RDBMS translates into a procedural program automatically
  - Facilitates adoption for non-IT people
  - The query optimizer is king and one of the main contributions of databases to the software industry

Summary

- RDBMS bottlenecks and their lack of generalizability
- NOSQL databases as an alternative to relational databases
- NOSQL promotes two axis of innovation
  - Alternative data models
  - Alternative system architectures
- SQL is **not** the problem

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