Big Data Management: The "M" in DBMS

when to use ?

hadoop
parallel

SDSC SAN DIEGO SUPERCOMPUTER CENTER

DBMS-based and non-DBMS-based Approaches to Big Data

SDSC SAN DIEGO SUPERCOMPUTER CENTER

After this video you will be able to

- Explain the various advantages of using a DBMS over a file system
- Specify the differences between a parallel and distributed file system
- Briefly describe a MapReduce-style DBMS

DBMS vs file system
parallel vs DBMS

Storing Data – Files vs. DBMS

- In the old times, database operations were applications in file systems
- Problems
  - Data redundancy, inconsistency and isolation — duplication or inconsistencies (hard to determine)
  - Each task a program < data access / data update
  - Data integrity — constraints : condition as part of condition of program
  - Atomicity of updates

└ changes as single unit (altogether) — everything or nothing

Advantages of a DBMS

→ state what we want without saying how

- **Declarative query languages**
  - No more task-based programs
- **Data independence** : isolate users from record as long as logic is clear (tables & attributes)
  - Applications don't worry about data storage formats and locations
- **Efficient access through optimization**
  - The system automatically finds an efficient way to access data

powerful ← data structures, algorithms, principles

Advantages of a DBMS

- **Data integrity and security**
  - Methods to keep the accuracy and consistency of data despite failure
    - ACID properties of transactions
    - Failure recovery
- **Concurrent access**
  - Many users can simultaneously access data without conflict

· systems fail ! + malicious process
· safety and failure recovery

(single transactions) — even if it has multiple changes
properties ACID :
· atomicity
· consistency : data valid (rules, constraints)
· isolation : concurrency (multiple people update simultaneously)
· durability : data remains even after — power loss, crashes, errors

serially
eg
selling last ticket

Parallel and Distributed DBMS

- **Parallel database system**
  - Improve performance through parallel implementation
  - Often allows data replication
    - Data redundancy against table corruption
    - More concurrent queries
- **Distributed database system: !**
  - Data is stored across several sites, each site managed by a DBMS capable of running independently !

Does your big data problem need these facilities? *?

answer can be negative

eg: parallel oracle
parallel DB2
post SQL XE

· tables spread across machiness
· operations use parallel algorithms
· allow replications
  - data redundancy
  - failure of replicas
  - replicas synched

· network of independently running DBMS that communicate with each other

· one component knows some part of the schema of its neighbour in DBMS and can pass query to

DBMS and MapReduce-style Systems

- Started with a different problem focus
  - DBMSs: efficient storage, transactions and retrieval
    - Partitioned data parallelism — different parts of logical table can physically reside on different machines
    - Account for computation and communication cost $$
    - Not node failure — classical DBMS did not take into account failures
  - Mapreduce-style systems: complex data processing over a cluster of machines — originally developed not for storage and retrieval, but for distributive processing of large amounts of data
    - HDFS-based
    - Analytics – data mining, clustering, machine learning
    - Multi-stage, problem-specific algorithms — hard to develop in standard
    - Operate on wider variety of data including text

· number of machines could go up
· issues automatically accounted for (like node failure)
· complex applications, like data mining, data clustering

Shifting Requirements — tension points

- **Data loading – a new bottleneck** — analysis on the data must be performed within a given time after it's arrival
  - Does the application need data sooner than the loading time?
- **Too much functionality**
  - Does the application use only a few data management features?
- **Combined Transactional and Analytical Capabilities** — optimization ← support for efficient analytical operations meets transactional guarantees

  eg. real time decision support

No Single Solution

combination of traditional requirements → new capabilities and products in the big data management

- **Mixed solutions**
  - DBMS on HDFS — new techniques that use MR → side door for MR-style operations
    → exchange data
    - Hadoop-DBMS interoperation * flexibility to use both forms of data processing
      hadoop subsystem ⟷ DBMS
  - Relational operations in MapReduce systems like Spark ← eg
  - Streaming input to DBMS — large distributed management operations to accept streaming data → design: analysis to perform known before
    └ as new data records arrive, keep record of data in memory to finish computation
  - New parallel programming models for analytical computation within DBMS
    └ large scale distributed algorithms emerge to solve analytics problems
    └ MR style algorithms
    └ evolve