Lecture 19:
• Data Processing Architectures (intro)

Announcements:
• HW-3 out
• R-3 out

Data Processing Systems

A Maxim: (∗)
"Any successful data analysis system will (eventually be asked to) support SQL"

Plan for the middle 3rd of the semester ...
• High-level architectures for scalable data (query) processing
• Including “traditional” query processing basics (review, for newer archs)
• Popular file formats for data analytics (column-based, hybrid)
• Spark (Spark SQL, Dataproc) and Dremel (BigQuery) architectures

... then move on to Data Mining / ML approaches and data pipelines (last 3rd)

(∗) Including MapReduce ... e.g., Apache Hive and Pig
Data Processing Systems

*Centralized Data Systems:*
- built to run on a single machine (one CPU, memory, disk)
- all data resides on the machine
- can be single-user (e.g., on mobile device) or multi-user
- still widely used, even for enterprise systems
- but scalability requires more powerful machines

*Parallel and Distributed Data Systems:*
- Data and processing spread (partitioned) across many machines
- *Parallel:* nodes close together, fast & reliable communication
- *Distributed:* nodes may not be close, slower & less reliable communication

Parallel and distributed database systems have been studied since the 80’s!

---

Data System Architectures: Shared Everything

Architectures based on degree of shared resources:

1. *Shared Everything*
   - same as single-node, centralized system
   - CPUs, memory, disks are all shared
   - can still leverage parallelism if multiple cores or CPUs

Note on types of parallelism
- **Course-grained:** send one query to each CPU
  - increases throughput (more queries per second)
- **Fine-grained:** split each query into parallel tasks
  - increases latency (execute each query faster)

Centralized largely support course-grained, distributed/parallel can support both
(2) **Shared Memory**
- many processors (CPUs, cores) share same memory and disks
- usually implies high-end servers (with many 10’s of cores, etc.)
- data systems today not typically designed for shared memory

(3) **Shared Nothing**
- each node is a separate machine, but no separate shared storage
- nodes only communicate through network connections
- nodes contain **partitions**, each node works on its own partition
- a centralized (master) node manages data processing
Note that partitioning is also called sharding

**Naive partitioning:**
- each table is a distinct partition (assumes tables can fit on a node)

**Vertical partitioning:**
- create partitions out of a table’s columns
- have to add information (like an offset) to reconstruct rows
- similar to decomposition (normalization)

**Horizontal partitioning:**
- split rows into distinct partitions
- **round-robin:** evenly distribute rows (1st row to node 1, 2nd to node 2, ...)
- **data ranges:** value ranges of “partitioning” attribute (e.g., last names)
- **hashing:** hashcode of partitioning attributes mapped to partition number
Disadvantages:

- hard to add nodes (scale) since have to move data to new nodes
- adding nodes typically requires repartitioning
- this makes system unavailable or worse inconsistent

Summary – Things to Know

1. Centralized vs Parallel vs Distributed Data Systems
2. Types of parallelism (course vs fine grained)
3. Shared everything systems
4. Shared memory systems
5. Shared nothing systems
6. Naive, vertical, horizontal partitioning
7. High-level approaches for horizontal partitioning
8. Challenge of shared nothing for scalability