Data Parallelism Part 1: Comprehensive Review

DSC 208R - Data Management for Analytics

Introduction to Data Parallelism

What is Data Parallelism?

Data Parallelism involves partitioning a large data file physically across multiple nodes or workers. Within each worker, data can be managed in DRAM (memory-based) or disk-based storage. The core idea behind scalability in data parallelism is to split a data file (either virtually or physically) and manage the reads and writes of its pages between disk and DRAM. It is the most common approach to combine parallelism and scalability in data systems. Data parallelism generalizes the SIMD (Single Instruction, Multiple Data) and SPMD (Single Program, Multiple Data) concepts from parallel processors to large-scale data and multi-worker/multi-node environments. It can involve distributed-memory or distributed-disk setups.

Paradigms of Multi-Node Parallelism

Data parallelism is often paired with shared-nothing parallelism, though it's technically orthogonal to the three main paradigms of multi-node parallelism.

1. Shared-Nothing Parallelism

- Each CPU (worker) has its own dedicated memory and disk, which are not directly shared with other CPUs.
- Communication between nodes happens explicitly via an interconnect network.
- Partitioning a data file across nodes is also known as **sharding**.
- This approach is commonly used in data processing workflows as part of Extract-Transform-Load (ETL) processes, which prepare data for querying and analysis. Examples of ETL steps include sharding, compression, and file format conversions.

2. Shared-Disk Parallelism

- Multiple CPUs (workers) share access to the same set of disks via an interconnect, but each CPU has its own private memory.
- Contention for shared disk resources can be a challenge.

3. Shared-Memory Parallelism

- Multiple CPUs (workers) share access to a common memory pool and typically a common set of disks, all connected via an interconnect.
- This paradigm is usually limited to a single machine due to memory coherence issues and interconnect bandwidth limitations across multiple machines. Contention for shared memory can be a significant issue.

Data Partitioning Strategies

Row-wise or horizontal partitioning (sharding) is the most common strategy. For k nodes, three common schemes are used:

- **Round-robin**: Tuple i is assigned to node $i \pmod k$.
- **Hashing-based**: Requires a hash partitioning attribute(s). This is most common in practice for Relational Algebra (RA) and SQL workloads.
- **Range-based**: Requires an ordinal partitioning attribute(s). This is often good for range predicates in RA/SQL.

All three strategies are often suitable for many Machine Learning (ML) workloads. Replication of partitions across nodes (e.g., 3x) is common to enable fault tolerance and improve parallel runtime performance.

Other Forms of Data Partitioning

Beyond row-wise partitioning, data can also be partitioned in other ways, similar to disk-aware data layout on a single node.

- **Columnar Partitioning**: Data is partitioned by columns across different nodes. For example, if a table has columns A, B, C, D, Node 1 might store all values for column A, Node 2 for column B, and so on.
- **Hybrid/Tiled Partitioning**: A combination of row-wise and columnar partitioning, where data is divided into tiles or blocks that span a subset of rows and columns, then distributed across nodes.

Cluster Architectures

Clusters require a protocol for nodes to communicate with each other.

Manager-Worker Architecture

- Consists of one (or a few) special nodes called "Manager" (also known as "Server" or "Master") and one or more "Workers".
- The Manager dictates what workers should do and when they should communicate with other nodes.
- This is the most common architecture in data systems like Dask, Spark, and parallel RDBMS. It is a centralized architecture.

Peer-to-Peer Architecture

- In this architecture, there is no special manager node.
- Workers communicate directly with each other.
- An example is Horovod. This is also known as a decentralized architecture.