Data Parallelism (Part 1)

Comprehensive Review

DSC 208R – Parallel Data Processing and the Cloud

Contents

1 Motivation

Data parallelism partitions large data sets across multiple workers so that each worker operates on its own shard. This strategy is the backbone of scalable systems that handle data sets too large for a single machine.:contentReference[oaicite:0]index=0

2 System Context

Classical multi-node architectures are shared-nothing, shared-memory, and shared-disk. Data parallelism aligns best with *shared-nothing* clusters, where each node owns its local shard and communicates with others only when necessary.:contentReference[oaicite:1]index=1

Shared-Nothing Cluster Illustration

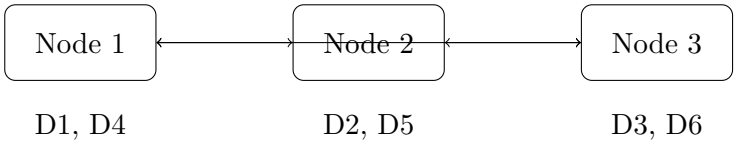


Figure 1: Shared-nothing cluster with six data shards.

3 Row-Wise Partitioning Strategies

For k workers the main schemes are: contentReference[oaicite:2]index=2

1. Round-robin: tuple i goes to worker (i mod k).
2. Hash: apply a hash to one or more attributes.
3. Range: assign contiguous key ranges to workers.

Notes. Hashing dominates SQL engines; range can accelerate range predicates; round-robin is simplest but ignores skew.

4 Replication for Fault Tolerance

Replicating every shard (for example, three copies) improves durability and enables locality-aware scheduling at the cost of extra storage.:contentReference[oaicite:3]index=3

5 Column and Hybrid Layouts

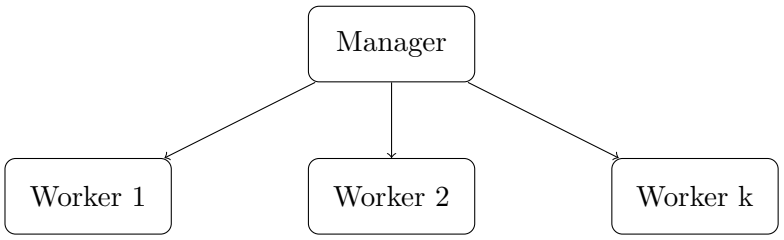
Single-node disk layouts generalize to clusters:contentReference[oaicite:4]index=4

- Column layouts store each column (or group) across workers.
- Hybrid or tiled layouts combine row and column slicing.

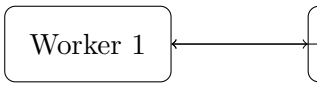
Such layouts help workloads that read only a subset of columns.

6 Cluster Coordination Styles

- **Manager-worker** (centralized): one node assigns tasks. Examples include Dask and Spark.:contentReference[oaicite:5]index=5
- **Peer-to-peer** (decentralized): workers coordinate directly, as in Horovod.:contentReference[oaicit



(a) Manager-worker



(b

Figure 2: Two coordination styles for data-parallel clusters.

7 Pros and Cons

- **Pros**
  - Horizontal scalability beyond one machine.
  - Clear mental model: each worker processes local data.
  - Works naturally with replication.
- **Cons**
  - Cross-shard joins and shuffles can dominate runtime.
  - Skewed partitions lead to load imbalance.
  - Coordination overhead grows with cluster size.

8 Future Directions

- Adaptive, workload-aware partitioning rather than static hash or range.
- Dynamic replication levels driven by failure probability and data locality.
- Tighter integration with cloud object storage tiers for cost efficiency.

Conclusion

Data parallelism divides data across many workers so that each can process a shard in parallel. The shared-nothing model with hash or range partitioning remains the default, but ongoing work targets smarter partitioning, adaptive replication, and better cloud integration.:contentReference[oaicite:7]index=7