Data-Parallel Data Science Operations

Comprehensive Review

DSC 208R – Parallel Data Processing and the Cloud

Contents

1 Motivation

Many data science pipelines run a small set of operations on tables or matrices that are too large for one machine. Data parallelism lets us shard the data, run node-local work in parallel, and then merge partial results. The slides illustrate this Bulk Synchronous Parallel (BSP) pattern using a manager-worker architecture.:contentReference[oaicite:0]index=0

2 Representative Operations

2.1 Non-deduplicating Project

- **Goal**: drop unneeded columns, keep all rows.
- **Algorithm**: the manager broadcasts the projection query; each worker reads its shard and writes a local file with only the requested columns; the manager unions the files.:contentReference[oaicite:1]index=1

2.2 Simple SQL Aggregates

- Workers compute local MIN, MAX, SUM, or COUNT and send one value to the manager.
- The manager reduces the partials to produce the final answer.:contentReference[oaicite:2]index=2

Aggregate classes

1. Distributive: MIN, MAX, COUNT, SUM (1 value per shard).
2. Algebraic: AVG, VARIANCE, STDEV (O(1) stats per shard).
3. Holistic: MEDIAN, PERCENTILE (may need large partials).:contentReference[oaicite:3]index=3

2.3 Group-By Aggregates

Workers build local hash tables keyed by the GROUP BY column and send them to the manager, which merges them. Network cost depends on the domain size of the key. If the manager RAM is too small, two-level aggregation or repartitioning is needed.:contentReference[oaicite:4]index=4

2.4 Matrix Sum and Norms

Treat the matrix as a large table or as 2x2 tiles; each worker sums its tile or computes local norms, sending partials to the manager. The pattern matches simple aggregates.:contentReference[oaicite:5]index=5

3 Common Optimizations

- **Caching**: keep hot shards in worker memory or SSD.
- **Replication**: store a shard on multiple workers to unlock more parallelism.
- **Asynchrony**: used more in ML (parameter servers) than in SQL.
- **Approximation**: sample data when exact answers are not required.

Using ML to decide data placement and caching across the memory hierarchy is an active research area.:contentReference[oaicite:6]index=6

4 Manager-Worker Diagram

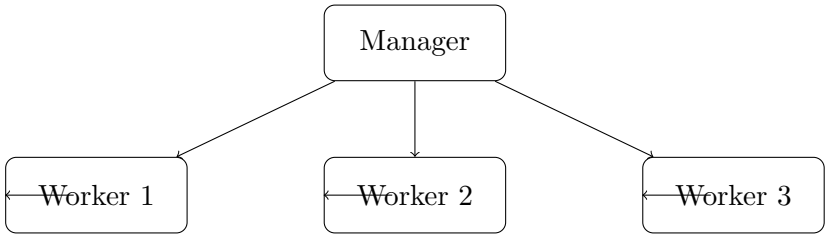

Figure 1: Manager-worker BSP pattern for data-parallel operations.

5 Pros and Cons

- **Pros**
  - Simple mental model; reuse single-node code inside each worker.
  - Scales linearly until network shuffle dominates.
- **Cons**
  - Manager may become a bottleneck for large partials (e.g., holistic aggs).
  - Data skew can cause load imbalance across shards.

6 Future Work

- Adaptive shard sizing based on runtime load.
- Hierarchical aggregation to relieve manager memory pressure.
- Automatic selection of approximate or exact algorithms depending on query SLAs.

Conclusion

Non-dedup projection, simple and group-by aggregates, and matrix sums cover a wide swath of data science workloads. All fit naturally into a manager-worker BSP framework, but careful attention to caching, replication, and skew handling is essential for high performance at scale.:contentReference[oaicite:7]index=7