Squall: Scalable Real-time Analytics

Open-source project, available at https://github.com/epfldata/squall


[Load Balancing and Skew Resilience for Parallel Joins. A. Vitorovic, M. ElSeidy and C. Koch, ICDE 2016]


Big Data Analytics

- Processing large data in a scalable way
- Terabytes of logs and clickstreams
- Processing large scientific data, e.g., LHC
- Emergence of online analytics
- Online Anomaly and fraud detection
- Finding patterns in customer and sales data
- Reach a potential customer during the active session
- Complex analytics can be represented as SQL

Theta (θ) Joins

- Analytics go beyond the equality "=" join predicate
- Band & spatial joins
- Similarity & fuzzy joins
- Range & interval joins
- Required by emergent applications
  - Hive plans to support theta joins
  - https://cwiki.apache.org/confluence/display/Hive/Theta+Join

Grid Partitioning

- Key Partitioning
  - incoming data is randomly shuffled and replicated across regions for skew resilience
  - Regions have the same area for load balance
  - Objective: minimize region semi-perimeter (ILF)
  - Different arrangement → minimized ILF
  - Requires knowledge about relation cardinalities

Low-selectivity joins

- Problem with grid partitioning: high replication
- Our Equi-weight histogram join reduces input costs by covering only candidates
- Minimize the maximum join work per node
- Computation geometry algorithms for rectangle tiling
- Efficient parallel scheme for capturing output distribution

Multi-way joins

- Hash-Hypercube: hash partitioning with join keys as hypercube dimensions
- Random-Hypercube: random partitioning with relations as dimensions
- Hybrid-Hypercube: hash partitioning; skew; random partitioning; skew-free
- Hybrid-Hypercube brings 1.08X and 1.92X improvement
- Efficient local multi-way joins: DBToaster
- Materializes intermediate tuples and reuses them

Demonstration

- Apply the new scheme to the operator: incurs scheme transition & state relocation
- Explore and trigger optimal partitioning and data routing schemes

Evaluation

- Throughput (tuples/sec)
- Total Execution Time