Automating Physical Database Design in a Parallel Database

Jun Rao, Chun Zhang, Nimrod Megiddo, and Guy M. Lohman

Presentation: Mumtaz Ahmad

Scope

- Shared-nothing Parallel Architecture
- Horizontal Partitioning of Base Data
- DB2 Data Partitioning Advisor
  - Hash-based Partitioning
  - Node Groups

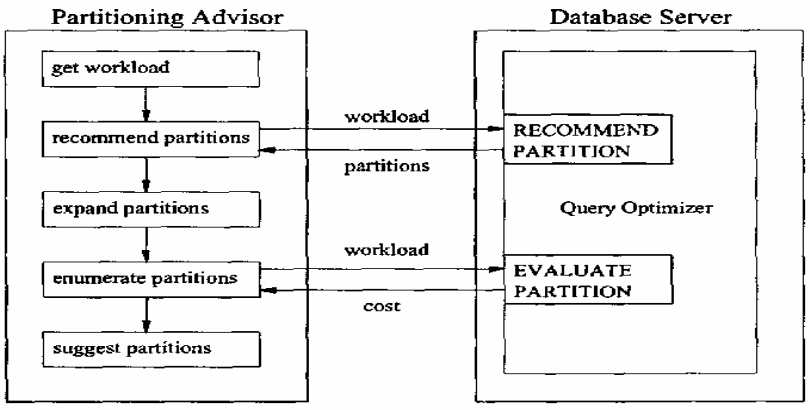
Problem

- Given: Query Workload, Database Statistics, Default Partitions
- Find: The optimal Partition for each table
- Hardness:
  - Different queries, best partitions differ
  - Same query, multiple tables join on different columns
- Why ?
  - Local joins, aggregation etc.
  - Load Balancing
  - Overall Optimal Performance

Solution Approach

- Key Idea
  - Same general framework as used for index /materialized view selection tools – apply to partitioning problem
  - Query Optimizer and its cost model has evolved well
  - Ask it for recommendation
  - Supplement the recommendations
  - Search the candidate plans space (using rank-based enumeration)
  - Finally, any plan is evaluated by Query Optimizer

Architecture



Recommend Partition

- Find optimal partition for each table for each query in work load
- Candidate Partitions considered
  - Columns in equality join; *R.a = S.a*
  - Grouping Columns; *Group by R.a*
  - Equality Predicate ; *R.a = "123"*
  - Replication
  - NodeGroups; Default, Existing

Recommend Partition

- Generate all combinations from candidate partitions
- Regular task of selecting best plan
- Write partitions in best plan to CANDIDATE_PARTITION table along with benefit

Expand Partition

- Existing Partition, if missed
- Subsumed Partitions
  - Q1: <T.a, T.b> ; Q2: <T.a, T.c>
  - Consider <T.a> as well

Evaluating Partitions

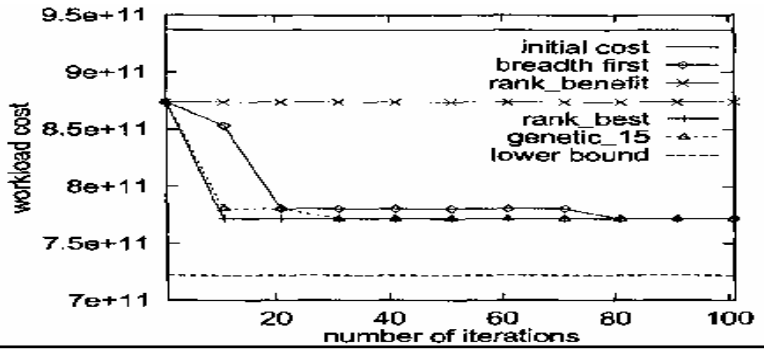
- Find: $C_{optimal}$, where $C = (c_1, c_2, \ldots, c_n)$ and $c_i \in (p_1, p_2, \ldots, p_m)$ for table *i*, for entire workload
- Problem: All candidate plans; large search space; time constraint
- Use Rank-Based Enumeration
  - Start with a root consisting of partitions with maximum benefit, expand to children that differ in one partition, pick next configuration based on a ranking function
  - Rank_Best (C) = -Cost (C') – P.benefit* $(P.tablecard/max\_tablecard)^{1/2}$
  - Cost of parent, benefit of difference from its parent, size of table

Evaluating Partitions

- Call the Query Optimizer to evaluate the selected configuration for entire workload; returns cost
- If better than previous, keep it
- Time constraint

Experimental Results

- **Customer Database with 50 queries, 15 tables,**
- **1-5 partitions /table recommended**
- **500 configurations**
- **Rank_Best converges fastest**
- **Speed up is 22%, 11 out of 15 partitions unchanged**



Related Work

- Partitioning
  - General problem is *NP*-Hard
  - Build a cost model, greedy solution
  - "An actual design tool should use the actual optimizer" [4].
- Load Balancing
  - Can supplement Physical database design at run time.
  - Actual workload mix keeps on changing
  - Strategies like least utilized processors, adaptive least utilized processors, degree of join parallelism [3].

Discussion

- Benefit of a query assigned to every partition
  - No way to measure contribution of each table.
  - So if only one table has different partition and query benefits, the benefit value is assigned to unchanged partitions as well
- Why not more than one partitions; its just replication
- Multiple calls to query Optimizer during evaluation
- No comparison to the results of other cost models
- Why not external tool or cost model during expansion phase
- Cache from recommend mode may be used during evaluation
- Assumptions for Cost derivation for "virtual" partitions
- Paper is well written.

References

[1] Guy M. Lohman, "A DB2 that manages itself ? ", Tutorial at VLDB 2004.

[2] Jun Rao, Chun Zhang, Nimrod Megiddo, and Guy M. Lohman, "Automating Physical Database Design in a Parallel Database", SIGMOD 2002.

[3] R. Marek, E. Rahm, "Analysis of Dynamic Load Balancing Strategies for Parallel Shared Nothing Database Systems", VLDB 1993.

[4] D. Sacca, G. Wiederhold, "Database Partitioning in a cluster of processors", TODS 1985.