CMSC724: Auto Administration/Self-Tuning Databases

Amol Deshpande

University of Maryland, College Park

April 26, 2011

Motivation

- Eliminating the knobs in a typical database system
  - Hardware capacity planning, physical database design
  - Settings for run-time resource management, inter-system dependencies
    - Need an expert DBA to understand these things
  - Total cost of systems often dominated by the human cost
- Need a feedback control loop
  - However trying to do these automatically is also problematic
  - Much harder to find out what went wrong

Motivation

- Eliminating the knobs in a typical database system
  - Hardware capacity planning, physical database design
  - Settings for run-time resource management, inter-system dependencies
    - Need an expert DBA to understand these things
  - Total cost of systems often dominated by the human cost
- Need a feedback control loop
  - However trying to do these automatically is also problematic
  - Much harder to find out what went wrong
- More recently:
  - More than 190 parameters are specified to control behavior of a Map-Reduce job in Hadoop
  - Many (at least 25) of these parameters do affect the performance significantly
- Data center capacity planning
  - Many interesting design decisions...

Self-Tuning Database Systems

- Quite broad in scope
- Work as early as 1974 (Stonebraker on Indexes)
- COMFORT project (80's)
  - Online feedback control loop
  - Observe: monitor performance metrics and workload parameters like transaction response times
  - Prediction: assess hypothetical adjustment of various candidate knobs
  - React: implement the recommendation from prediction phase
    - Conceptually straightforward, but doing this while actively serving queries a tough engineering challenge

COMFORT Project

- Load Control (Performance Evaluation...; Moenkeberg, Weikum; VLDB 1992)
  - Thrashing due to excessive lock conflicts
  - Parameter to tune: multiprogramming level = No. of transactions that are admitted for concurrent execution
  - Define *conflict-ratio = No. locks held by all transactions/No. locks held by non-blocked transactions*
    - If around 1.3, very high thrashing danger
    - Experimentally observed, but later confirmed by mathematical modeling
- Dynamic Data Placement: data migrations to balance load
- Workflow server configuration

COMFORT Project: Retrospective

- From 2002 Paper: Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering
- **Note: Authors' opinions**
- Obvious improvements in the last 10 years (1992 to 2002)
    - Second-order tuning knobs with minor performance effects (e.g., group commit timers, or log buffer sizes) eliminated
    - Reasonable default values for many tuning knobs

COMFORT Project: Retrospective

- From 2002 Paper: Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering
- **Note: Authors' opinions**
- Obvious improvements in the last 10 years (1992 to 2002)
  - Second-order tuning knobs with minor performance effects (e.g., group commit timers, or log buffer sizes) eliminated
  - Reasonable default values for many tuning knobs
- Relative *simple* recipes
  - Robust rules of thumb (e.g., page size)
  - Throw hardware at the problem if helpful
  - Remove tuning options that offer only marginal gains
    - e.g., drop hash indexes or join indexes altogether
    - at most factor of 2 performance improvement
  - Eliminate tuning options that require black magic
    - Indexes on multi-table clusters – very difficult to assess its effect

COMFORT Project: Retrospective

- From 2002 Paper: Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering
- **Note: Authors' opinions**
- Advanced tuning problems that are *solved*
  - Tuning issues at disk storage level essentially solved
  - Index selection solved
    - Most commercial systems ship with index wizards
  - Selection of materialized views solved

COMFORT Project: Retrospective

- From 2002 Paper: Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering
- **Note: Authors' opinions**
- Advanced tuning problems that are *solved*
  - Tuning issues at disk storage level essentially solved
  - Index selection solved
    - Most commercial systems ship with index wizards
  - Selection of materialized views solved
- Number of challenging problems still left
  - Query optimization still tough
  - Memory management – dynamically adjusting size of query working spaces etc
  - Transaction isolation levels – reasoning about correctness and performance
  - Admission control, scheduling, especially for mixed workloads
  - ...

AutoAdmin Project at MSR

- AutoAdmin largely focused on physical design
  - Goal: given a query workload, find a *configuration* (e.g., a set of indexes) to minimize the cost
  - Query workload captured as a trace
  - How to measure *goodness* of a configuration?
    - Not feasible to try out the configuration
    - Usually not amenable to analytic cost equations
    - Pretty much the only option to use the *optimizer cost model*

Index Selection; VLDB 1997

- Builds upon a prior work by Finkelstein et al.; 1988
- Three ways to do automated index selection
  - 1. Use semantic information like uniqueness, reference constraints, statistics etc..
  - 2. Rule-based "expert" systems
  - 3. Use optimizer's cost estimates to compare different alternatives
    - Often the optimizer itself is not very good at estimating costs
- Need a "what-if" interface from the optimizer
  - Allows creating a "Hypothetical Index"

Index Selection; VLDB 1997

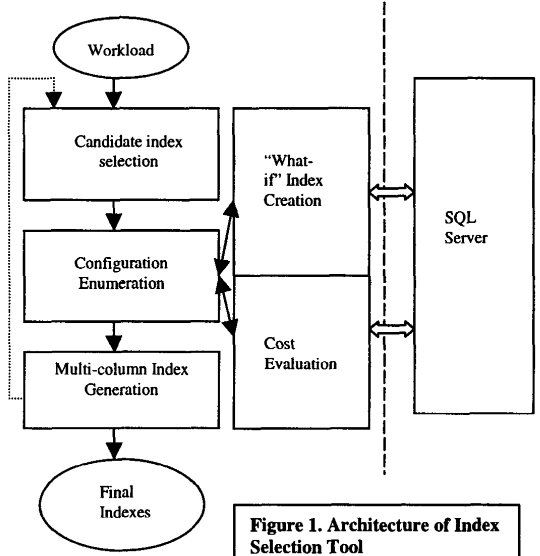

Figure 1. Architecture of Index Selection Tool

Index Selection; VLDB 1997

- Let Cost(Q, C) = cost of executing Q using configuration C
- Atomic configuration for a query
  - A set of indexes that are all used in some plan for the query
- Suppose we have computed the costs for all atomic configurations for a query
- Now consider a non-atomic configuration C'
  - For select/update queries, we will have used a subset of C' that is atomic
    - Hence: Cost(Q, C) = Min { Cost(Q, Ci) }, where $C_i \subset C'$ is atomic
    - Enough to look at maximal atomic configurations
  - For insert/delete queries, more complex because we have to worry about updates to the index itself

Index Selection; VLDB 1997

- How to choose atomic configurations?
  - The space even for a single query is too large
  - Heuristics for reducing the search space
    - A query may use at most 2 indexes per table
    - In a multi-table query, indexes will be used on at most 2 tables

Example 2. Single-join Atomic Configurations

Consider a SELECT query with conditions $T_1.A < 20$, $T_1.A = T_2.B$, $T_3.C$ BETWEEN [30,50], $T_3.C = T_2.B$. In this case, one 3-table atomic configuration is ($T_1.A$, $T_2.B$, $T_3.C$) since all three indexes may be used together to answer the query. However, due to the single-join atomic configuration based pruning step, the above atomic configuration is not evaluated. Rather, the cost of this query for the 3-table configuration is estimated by taking minimum of the costs of the atomic configurations: ($T_1.A$, $T_2.B$), ($T_1.A$, $T_3.C$), and ($T_2.B$, $T_3.C$).

Index Selection; VLDB 1997

- Candidate Index Selection Module
  - Find the best configuration for each query independently
  - Focus only on indexes that appear in those configurations
- Configuration Enumeration
  - Greedy approach based on adding indexes with highest incremental benefit
- Multi-column Indexes
  - Extend the single-column indexes chosen in the earlier steps

AutoAdmin: Retrospective; VLDB 2007

- Other physical design structures
- Materialized Views
  - Larger search space
  - Try to find common subexpressions that occur frequently using frequent itemset mining
- Partitioning, ...
- Many of these interact with each other
  - Staging (doing one after another) not efficient
  - Rather must search intelligently

AutoAdmin: Retrospective; VLDB 2007

- Statistics Management
  - Query optimizers rely fundamentally on the statistics
  - 1. Choosing statistics (i.e., histograms on which columns or column-sets etc)
  - 2. Self-tuning histograms
    - Use runtime feedback to tune a histogram itself
- Query progress estimation
  - Quite tricky to decide how far a query has progressed
- Adaptive query processing
  - Use runtime feedback to tune the query processor
  - We saw some details on this earlier

AutoAdmin: Retrospective; VLDB 2007

- Future Directions
  - Judging quality of physical design tools
  - Designing databases s.t. the physical design can be changed easily
    - e.g., just an "alter table" to insert column is quite heavy-weight
  - Multi-tenancy
- Other things that the paper did not talk about
  - Tuning Map-REduce like architectures
  - See the Starfish paper in CIDR 2011