Self-Managing DBMS Technology at Microsoft

Surajit Chaudhuri
Microsoft Research

Microsoft

Acknowledgement

SQL Product Unit
AutoAdmin Research Team

Easy Solutions

- Throw more hardware
  - Use this with caution
  - Where do you throw hardware?
- Rules of Thumb approach
  - Finding them is harder than you think
  - May simply not exist – oversimplified wrong solutions are not helpful

Microsoft's Early Focus on Self-Managing Technology

- 1998: SQL Server 7.0 launch towards a self-tuning database system:
  - Eliminate outright many knobs and provide adaptive control
    - Dynamic Memory Management
    - Auto Stats, Auto Parallelism and Space Management
    - Index Tuning Wizard
- 1996: AutoAdmin Project at Microsoft Research – exclusive focus on self tuning

Key Pillars

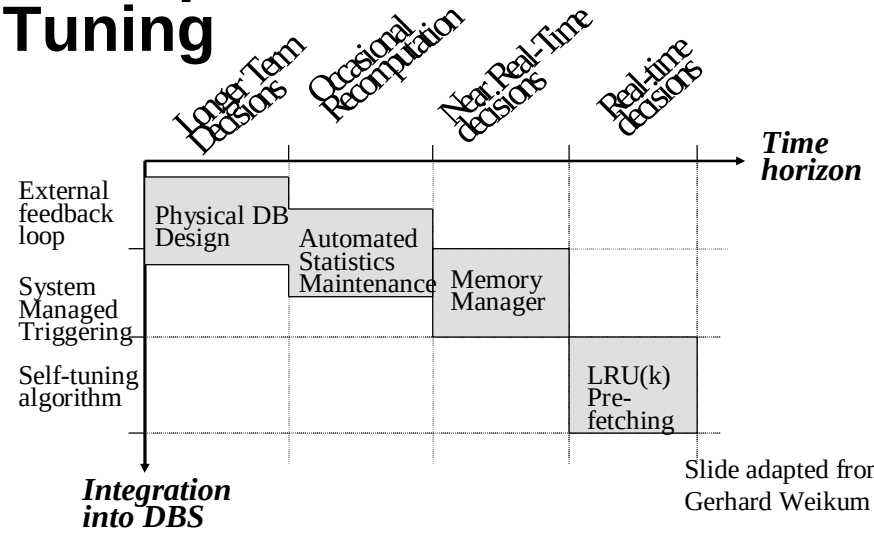
- "Observe-Predict-React" Feedback cycle
  - Powerful Monitoring Framework (useful in itself)
  - Local Models for estimating Target (Predict Phase)
  - What-If functionality is a key component of "React"
- Key Characteristics
  - Robustness
  - Transparency

5

Surajit Chaudhuri

Microsoft

The Spectrum for Self-Tuning



Slide adapted from Gerhard Weikum

6

Surajit Chaudhuri

Microsoft

Monitoring SQL Server Activities

Microsoft

Monitoring Tools

- Microsoft Operations Manager
  - Track Connectivity, Free space, Long Running Jobs, PERFMON
  - Reporting
- Best Practices Analyzer
  - Detect common oversights in managing a SQL Server installation
  - Simple UI, Rules metadata (70+), Reporting
  - File Compression, File Placement, Index frag
- Dedicated Admin connection in SS 2005
  - Connect even to a “hung” server (uses reserved scheduler, port & resources)

Surajit Chaudhuri

Microsoft

SQL Profiler

- SQL Trace
  - Server side component of the auditing engine
  - Pick Events (plan compilations, index usage,..), Data Columns, Filters
- SQL Profiler
  - GUI tool for SQL Trace
- Event log
  - heap where events are logged
- Trace must be stopped to be queried

Need for More Transparency

- Majority of case time is spent diagnosing the problem (allocation errors, perf degradation)
  - 60% in data collection, 40% in data analysis
- Dependence on Repros
  - Difficult to ID some performance issues
  - Unacceptable to many customers
  - End User experience
- Help requested for cases which don't resolve within 30 mins
  - Full dump requested on ~40%

Dynamic Management Views in SQL Server 2005

Simple queries now solve many scenarios (Live in memory stats)

- low level system (server-wide) info such as memory, locking & scheduling
- Transactions & isolation
- Input/Output on network and disks
- Databases and database objects

Populate a Data Warehouse

Example: Dynamic Management Views

Sys.dm_exec_requests – currently running requests

Sys.dm_exec_query_stats

- One row per query plan currently in the cache
  - Min, max, avg, last;
    - Physical reads, logical reads, physical writes;
  - Execution count; First and last execution times

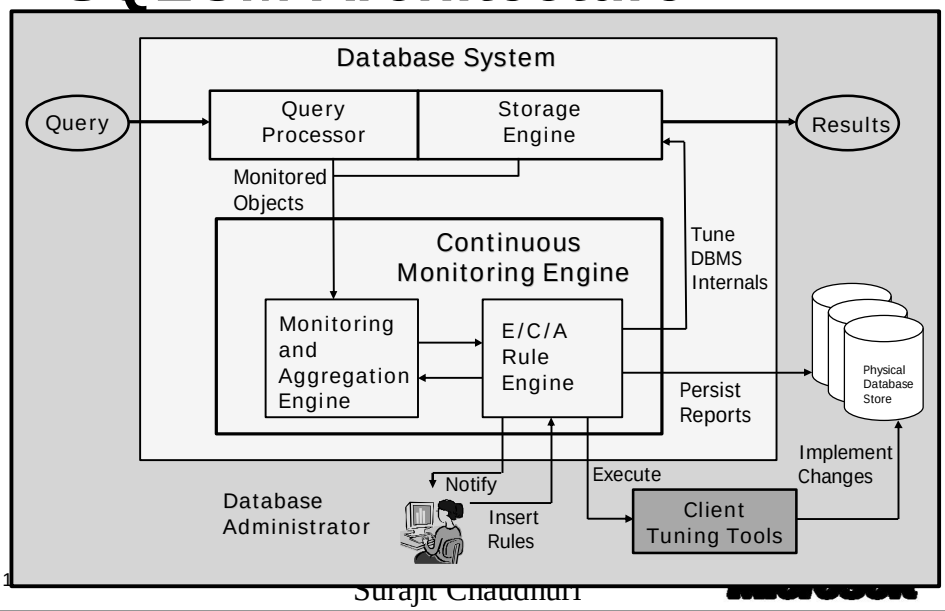
"Performance Statistics" Trace event

- Log "query_stats" for plans which are removed from the cache

SQLCM Research Project

- SQLCM is implemented inside the DB server
- Grouping/Aggregation can be processed inside the server
  - Actions based on monitored data allow modifications in server behavior
- The programming model to specify monitoring tasks is ECA rules
  - Rules are interpreted, dynamic
  - Expressiveness limited ⇒ low and controllable overhead
- Overcomes problems with push and pull

SQLCM Architecture



Key Ideas in SQLCM

- Logical Query Signature:
  - Extracts tree structure
  - Exact match between signatures
  - Signature cached with query plan
- Lightweight Aggregation Table (LAT) :
  - A set of grouping attributes, Aggregation functions
  - A memory-constraint (in terms of rows/bytes)
  - An ordering column used for eviction
  - LAT-counters may age over time
- Status: AutoAdmin research prototype. Technical details in IEEE ICDE 2003)

15 Surajit Chaudhuri Microsoft

Workload Analysis

- Variety of tasks leverage workload
  - DBA (ad-hoc analysis)
  - Physical design tuning tools
  - Approximate query processing
- Workload typically gathered by logging events on server
- Workloads can be very large
  - Few DBAs can eyeball 1GB workload file!
  - Few tools can scale
- Need infrastructure for summarizing and analyzing workloads

16 Surajit Chaudhuri Microsoft

Approaches to Workload Analysis

- Populate a schematized database
- Model as multi-dimensional analysis problem
  - Good for ad-hoc analysis using SQL and OLAP
  - Insufficient support for summarization
- Summarizing Workload:
  - Random sampling
  - Application specific workload clustering (SIGMOD 2002)
    - Plug-in "distance" function, adapt K-Mediod clustering
  - Novel declarative primitives (VLDB 2003)

Estimating Progress of SQL Query Execution

- Decision support systems need to support long running SQL queries
- Today's DBMS provides little feedback to DBA during query execution
- Goal: Provide reliable progress estimator during query execution
  - Accuracy, Fine Granularity, Low Overhead, Monotonicity, Leverage feedback from execution
  - Status: AutoAdmin Research Project and prototype: technical details in SIGMOD 2004

Modeling Total Work

Want a simpler model than query optimizer's cost estimate

Query execution engines use iterator model

Total work = Total number of GetNext() calls

Let N_i be total number of GetNext() calls for Op_i

Let K_i be total number of GetNext() calls for Op_i thus far

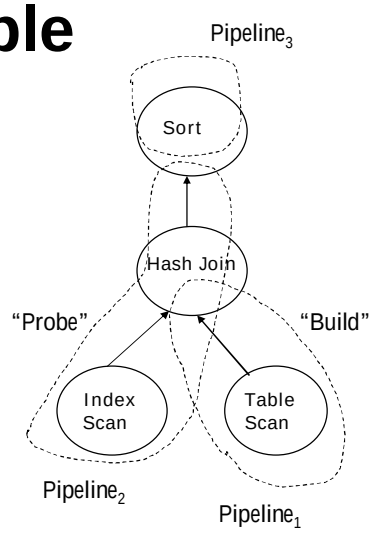
Estimator

$$gnm = \frac{\sum_i c_i \cdot K_i}{\sum_i c_i \cdot N_i}$$

where c_i is relative weight of Op

Problem: Estimating N_i during query execution

Example



K_i of each operator can be observed exactly during execution

Problem: Estimating N_i (in particular for Hash Join, Sort operators)

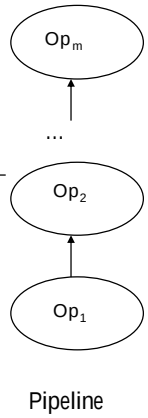
Single-Pipeline Queries

Driver Node: Operator that is "source" of tuples for the pipeline (leaf node)

Estimator: dne = $\frac{K_1}{N_1}$

Driver node hypothesis: $\frac{K_1}{N_1} \approx \frac{\sum_i K_i}{\sum_i N_i}$

- Estimate of N_1 is usually more accurate
- N_1 may dominate other N_i's, e.g., TPC-H queries
- Work done per tuple does not vary significantly



Other Key Considerations

Leverages execution information
- Observed cardinalities (K_i's)
- Algebraic properties of operators
- Internal state of the operator

Spills due to insufficient memory
- Model as a new (runtime) pipeline

Trade-off between guaranteeing monotonicity and accuracy

Non-uniform weights of operators

Recap of Monitoring Highlights

- Transparency of current server state crucial for easing DBA tasks, supported by DMVs
- Online aggregation of server state can support a monitoring framework (SQLCM)
- Logging of workloads as well as server events using SQL Profiler is crucial for offline analysis
- Tool to estimate progress of queries

Self-Tuning Memory Management

Dynamic Self Tuning Memory Manager

- **SQL 7.0 pioneered idea of dynamic self-tuning memory**
  - Sufficient memory set aside so that Windows and other applications can run without hiccups
  - Amount depends on system load
- **Observe:**
  - Query Windows for the amount of free physical memory periodically
  - Considers page life expectancy for the buffer pool

Self-Tuning Memory Manager

- **Predict:** Available memory compared to required threshold of Target Pages (PERFMON values consulted)
  - No explicit model-based prediction
  - Takes physical memory size into account
- **React:**
  - Keep a given number of free pages (for new allocation requests) at all times
  - Grab if low page life expectancy
  - If memory pressure from OS, free up buffers

Memory Management by Query Execution Engine

- Among competing queries
- Within a query
  - Among parallel threads
  - Nodes of a plan
  - Phases within an operator
- Give each query, once admitted to execution, adequate memory
  - Waiting memory, Waiting operators
  - Preempt on demand

Resolving Memory Pressure

- Identifying Memory Pressure
  - OS level clues not so useful
  - Cache hit ratio, Low Page life expectancy in buffer pool, Free list stalls/s, Physical disk, Memory Grant request queue
- Dig for the cause before adding memory
  - Recompilations, poor physical design – lack of indexes, excessive de-normalization, sloppy SQL update code

Examples of Self-Tuning Features in Storage Engine

29

Microsoft

Automatic Checkpointing

- Uniform time interval is not ideal
  - Based on number of records in the log
  - Specified recovery interval – max time SQL Server should take for restart
- Log manager estimates if it is time for checkpointing
- For simple recovery model
  - Log 70% full
  - Restart may take more than recovery interval

30

Surajit Chaudhuri

Microsoft

Storage Engine

- Expanding and Shrinking a Database
  - Specify initial, max sizes and the growth rates
  - Proportional allocation of extents in a filegroup
  - Autoshrink invokes shrink with a free space threshold
- Read-ahead depth for pre-fetching/Write-behind depth for bulk write
- Lock escalation
- Online index creation in SQL Server 2005

Query Engine

- Compilation efficiency
  - Use of Procedure Cache
    - Initial cost based on compilation cost
    - Lazywriter sweep for maintenance
  - Conservative Auto-parameterization
    - Select fname, lname, sal from emp where eid = 6
    - Select fname, lname, sal from emp where eid = @e
- Degree of Parallelism dynamically chosen based on runtime conditions
  - CPU, concurrency, memory
- Auto-select exhaustiveness of optimization

Self-Tuning for Statistics Management

Microsoft

Why Statistics Management?

Having “right” statistics is crucial for good quality plans.

- When to build statistics?
- Which columns to build statistics on?
- How to build statistics on any column efficiently?

Surajit Chaudhuri

Microsoft

Auto Statistics in SQL Server

- Created dynamically at query compilation time
- On single table columns for which optimizer needs to estimate distribution
- Uses sampling of data to create statistics
- Statistics auto-maintained
- Novel feature supported since SQL Server 7.0

Uniform vs. Block-Level Sampling

- Uniform random sampling is too expensive.
- Block-level sampling:
  - Pick a few blocks at random and retain all tuples in those
- Block level sampling is efficient but tuples may be placed in blocks arbitrarily
  - Reduced quality of the resulting estimate

AutoUpdate of Statistics

- Triggered by Query Optimization
  - Involves only a subset of the columns in the query
- Refreshed when a certain fraction (roughly) of the rows have been modified
  - Uses rowmodctr information to check if threshold has been reached
- Statistics that are auto-created are aged and retired if appropriate.

Lazy Scheduling

- AutoStat and its refresh adds to the cost of the query compilation
- For some applications with large tables, this presents a choice between a poor plan and a high cost of compilation
- SQL Server 2005 offers asynchronous auto stats
  - The “current” query will be optimized with the existing statistics
  - However, an asynchronous task to build the statistics will be posted

Frontiers for Further Thinking

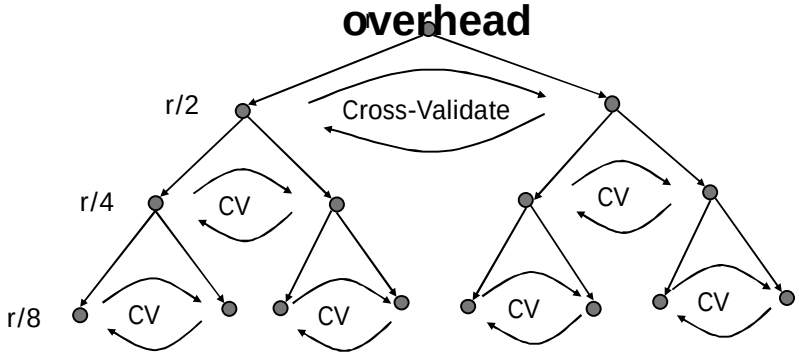
- Determining the appropriate Block Level Sampling
- Identifying the interesting subset of statistics for a query
- Statistics on views and query expressions
- Leveraging execution feedback
- Remaining slides in this part are on some research ideas being pursued at Microsoft

Adaptive 2-phase approach for Block Level Sampling

- Get initial sample
- While sorting get error estimate for *r/2*, *r/4*, *r/8* … etc.
- Find the best-fit curve of the form c/sqrt(r) through these points
  - Read off the required sample size
  - Experimentally found to almost always reach the error target or very close.
- AutoAdmin research prototype, SIGMOD 2004

Cross-Validation and Sorting

A way to get lots of estimates at little overhead



Recommending Base-Table Statistics

Find subset as good as having all statistics ("essential" set)

- Depends on workload, data distribution, optimizer…

Determining an essential set is non-trivial.

- "Chicken-and-egg" problem: cannot tell if additional statistics are necessary until we actually build them!
- Need a test for equivalence *without* having to build any new statistics

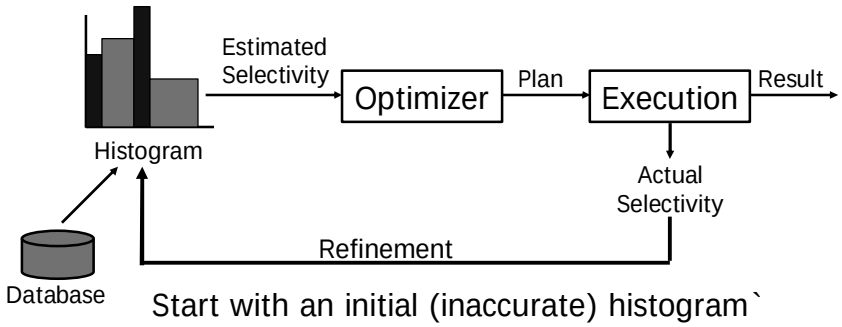
Our Contribution: MNSA

Research Prototype: [IEEE ICDE 2000]

Builds essential sets of statistics.

- t-Optimizer-Cost equivalence:Cost (Q, All-stats) and Cost (Q, Current-stats) are within t% of each other.
- Varies magic numbers using monotonicity property.
- If cost differ => need more statistics => choose stats for more expensive operators.

Exploiting Execution Feedback: Self-tuning Histograms



Start with an initial (inaccurate) histogram` and refine it based on feedback

Self Tuning Histograms: STGrid and STHoles

Assume uniformity and independence until execution feedback shows otherwise (no data set examination)

Exploit workload to allocate buckets.

Query feedback captures uniformly dense regions

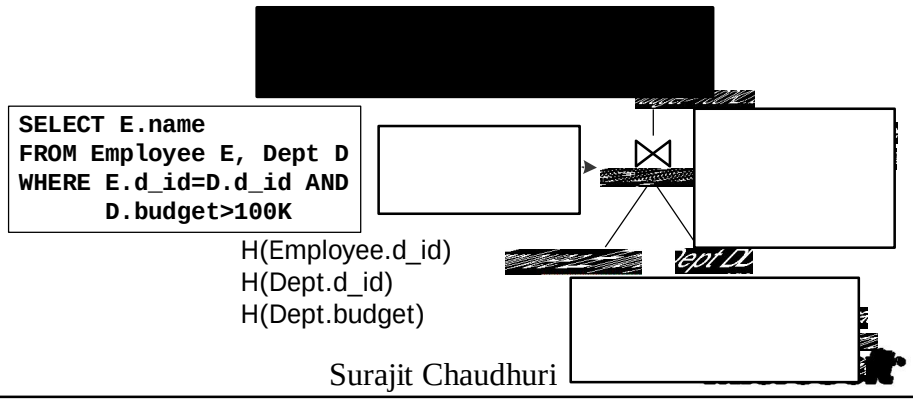
Differences: Bucket structure and refining

- STGrid: Multidimensional Grid [SIGMOD'99].
- STHoles: Bucket nesting [SIGMOD'01].

Microsoft

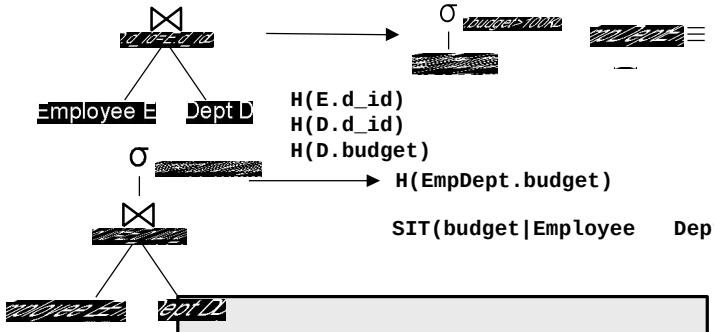
Are base-table statistics sufficient?

Statistics are usually propagated through complex query plans.



[SIGMOD'02, ICDE'03, VLDB'03, SIGMOD'04]

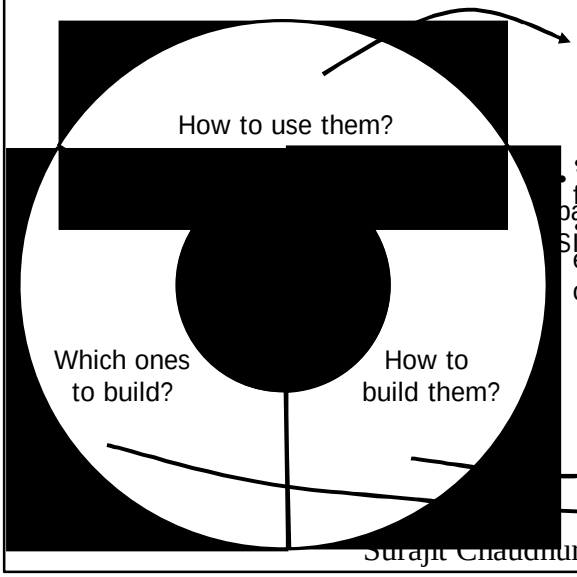
Statistics on Views (Query Expressions)



We do not need to materialize EmpDept!
(We just need the histogram)

Microsoft

SoV/SIT Challenges



Leverages view matching techniques to incorporate SITS into an existing optimizer

- Online based on execution feedback
- Sweep algorithms to efficiently materialize classes of SITs

Microsoft

Self-Tuning Physical Database Design

Microsoft

Microsoft SQL Server Milestones

SQL Server 7.0: Ships index tuning wizard (1998): Industry's first

SQL Server 2000: Integrated recommendations for indexes and materialized (indexed) Views: Industry's first

SQL Server 2005: Integrated recommendations for indexes, materialized views, and partitioning, offering time bound tuning, Industry's first

Results of collaboration between AutoAdmin Research and the SQL Server teams

Microsoft

Key Insights

Robustness was a design priority

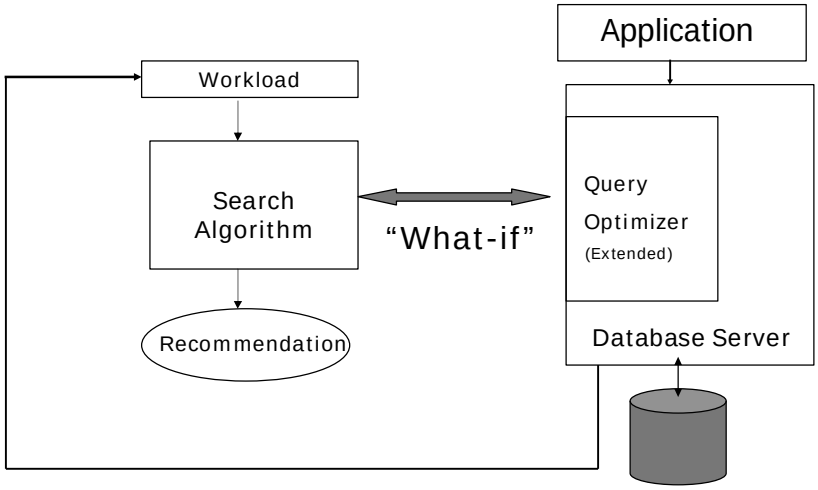
Every system is different – track workloads (VLDB 1997)

"What-If" API for DBMS (SIGMOD 1998) is key to driving selection of physical design

Efficient search for physical design (VLDB 1997, 2000, 2004)

Significant thinking on system usability (VLDB 2004)

"What-If" Architecture Overview



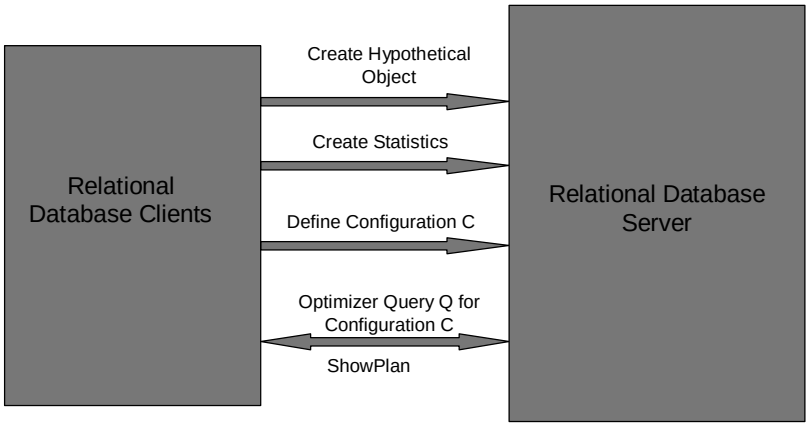
"What-If" Analysis of Physical Design

- Estimate quantitatively the impact of physical design on workload
  - e.g., if we add an index on T.c, which queries benefit and by how much?
- Without making actual changes to physical design
  - Time consuming
  - Resource intensive
- Search efficiently the space of hypothetical designs

Realizing "What-If" Indexes

- Query Optimizer decides which plan to choose given a physical design
- Query optimizer does not require physical design to be materialized
  - Relies on statistics to choose right plan
    - Sampling based techniques for building statistics
- Sufficient to fake existence of physical design

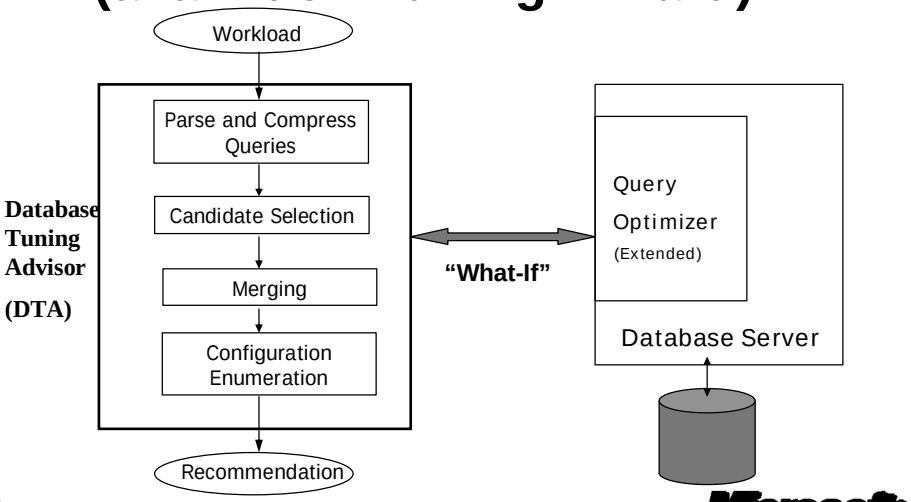
Using What-If Analysis



Physical Database Design: Problem Statement

- Workload
  - queries and updates
- Configuration
  - A set of indexes, materialized views and partitions from a search space
  - Cost obtained by “what-if” realization of the configuration
- Constraints
  - Upper bound on storage space for indexes
- Search: Pick a configuration with lowest cost for the given database and workload.

Database Tuning Advisor (aka Index Tuning Wizard)



57

Surajit Chaudhuri

Microsoft

Some Key Ideas

- Prefiltering of search space
  - Adapt cost-based frequent itemset idea from data mining (VLDB 2000)
- Quantitative analysis at per query level to isolate candidates
- Watch out for over-fitting
  - View Merging
- Search Efficiency crucial
  - Server bears the cost of "searching" as we ping the optimizer,
- Robustness – unaffected by most optimizer changes

58

Surajit Chaudhuri

Microsoft

DTA for Microsoft SQL Server 2005

- Time bound tuning
  - Complete tuning in batch window
- Range partitioning recommendations
  - Integrated Recommendation with Indexes and MVs
  - Manageability: Can recommend “Aligned” partitioning
- User-specified configuration (USC)
  - Exposes “What-if” analysis
  - Manageability: Allows specifying partial configuration for tuning
- Input/Output via XML
  - Public schema:
    http://schemas.microsoft.com/sqlserver/2004/07/dta/dtaschema.xsd
  - More scriptable
  - Easy for ISVs to build value added tools on top

DTA: Microsoft SQL Server 2005

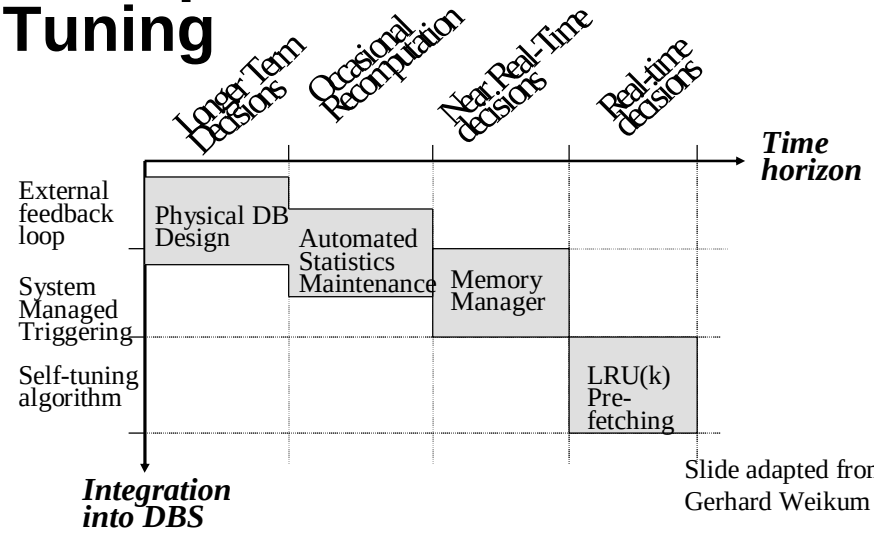
- Production/Test Server Tuning
  - Exploit test server to reduce tuning load on production server
  - Recommendation same as if tuning done on production server
  - Servers need not be H/W identical
- Improved Performance and Scalability
  - Workload compression
  - Reduced statistics creation
  - Exploit multiple processors on server
  - Scaling to large schema
  - Multi-database tuning
- Recommends online indexes
- Drop-only mode
  - Clean up unused indexes, MVs
- More details in VLDB 2004 paper

Lessons for Self-Tuning and Rethinking System Design

61

Microsoft

The Spectrum for Self-Tuning



Slide adapted from Gerhard Weikum

62

Surajit Chaudhuri

Microsoft

Principles for Self Tuning

- Complex problems have simple, easy to understand wrong answers
- "Observe-Predict-React" cycle can only be implemented locally
  - Develop self-tuning, adaptive algorithms for individual tuning tasks
  - Need robust models – when and how
- Global knowledge necessary for identification of bottlenecks
- Watch out for too many Tuning parameters

Rethinking Systems: Wishful Thinking?

VLDB 2000 Vision paper (Chaudhuri and Weikum 2000)

Enforce Layered approach and Strong limits on interaction (narrow APIs)

- **Package as components of modest complexity**
- **Encapsulation must be equipped with self-tuning**

Featurism can be a curse

- **Don't abuse extensibility - Eliminate 2nd order optimization**

Final Words

Self-Tuning servers crucial for bounding cost

- Policy based adaptive control "observe-predict-react"
- Monitoring infrastructure
- Leveraging Workload
- What-if analysis
- Deep understanding of local systems

Microsoft SQL Server encapsulates significant self-tuning technology

Ongoing work in SQL Server and AutoAdmin research projects

Microsoft SQL Server Self Tuning Technology Talks

Vivek Narasayya "Database Tuning Advisor for Microsoft SQL Server 2005" (Industrial Session 4, Thu)

David Campbell "Production Database Systems: Making Them Easy is Hard Work" (industrial Session 6, Thu)

Self-Tuning Overview Papers

Chaudhuri S., Christensen E., Graefe G., Narasayya V., and Zwilling, M. Self-Tuning Technology in Microsoft SQL Server. *IEEE Data Eng. Bull. 22(2): 20-26 (1999)*

Chaudhuri S. and Weikum G., Rethinking Database System Architecture: Towards a Self-tuning, RISC-style Database System. *VLDB 2000*.

AutoAdmin Research Project Website: http://research.microsoft.com/dmx/AutoAdmin

SQL Product Home http://www.microsoft.com/sql

Self-Tuning Physical Design

Chaudhuri S. and Narasayya V., An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. *VLDB 1997.*

Chaudhuri, S. and Narasayya V., AutoAdmin "What-If" Index Analysis Utility. *SIGMOD*, 1998.

Chaudhuri S. and Narasayya V. , Index Merging. *ICDE 1999*.

Agrawal S., Chaudhuri S. and Narasayya V., Automated Selection of Materialized Views and Indexes for SQL Databases. *VLDB 2000.*

Agrawal S., Narasayya V., and Yang, B. Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design. *SIGMOD 2004*.

Agrawal S., Chaudhuri S., Kollar L., Marathe A., Narasayya V. and Syamala M. Database Tuning Advisor for Microsoft SQL Server 2005. *VLDB 2004.*

Statistics Management

Aboulnaga, A. and Chaudhuri, S., Self-Tuning Histograms: Building Histograms Without Looking at Data. *SIGMOD* 1999.

Chaudhuri S. and Narasayya V. , Automating Statistics Management for Query Optimizers. *ICDE 2000.*

Bruno N., Chaudhuri S. and Gravano L. STHoles: A Multidimensional Workload-Aware Histogram. *SIGMOD 2001*.

Bruno N., and Chaudhuri S. Exploiting Statistics on Query Expressions for Optimization. *SIGMOD 2002*.

Bruno N., and Chaudhuri S. Efficient Creation of Statistics over Query Expressions. *ICDE 2003.*

César A. Galindo-Legaria, Milind Joshi, Florian Waas, Ming-Chuan Wu, Statistics on Views. *VLDB 2003*

Bruno N. and Chaudhuri S. Conditional Selectivity for Statistics on Query Expressions. *SIGMOD 2004.*

Chaudhuri S., Das G., and Srivastava U. Effective Use of Block-Level Sampling in Statistics Estimation. *SIGMOD 2004.*

Monitoring and Workload Analysis and Management

Chaudhuri S., König, A., and Narasayya V. SQLCM: A Continuous Monitoring Framework for Relational Database Engines. *ICDE 2004.*

Chaudhuri S., Narasayya V., and Ramamurthy, R. Estimating Progress of Execution for SQL Queries. *SIGMOD 2004.*

Chaudhuri S., Gupta A., and Narasayya V. Compressing SQL Workloads. *SIGMOD* 2002.

Chaudhuri S., Ganesan P., and Narasayya V. Primitives for Workload Summarization and Implications for SQL. *VLDB 2003*