A DB2 That Manages Itself?

Guy M. Lohman

(lohman@almaden.ibm.com)

Almaden Research Center

The Idea

Wouldn't it be great if your Database (and entire system!) were as easy to maintain and as self-controlled as your refrigerator?
Agenda

- Introduction & Motivation
- DB2 Autonomic Computing Project
- Existing DB2 Autonomic Features
  - Index Advisor
  - Configuration Advisor
  - Health Advisor
- New in “Stinger”
  - Design Advisor
  - Automated Statistics Collection
- Ad. Tech. & Research Projects
  - Progressive Optimization
- Conclusions

DB2 Autonomic Computing

- **Goal** -- Make DB2 Autonomic
- **The Project:**
  - Multi-Platform (Linux, Unix, Windows, mainframe)
  - Multi-Division (Research, Development)
  - Multi-Site (Toronto, Almaden, Silicon Valley, Watson)
  - Part of IBM’s company-wide “Autonomic Computing” initiative
- **Leaders:**
  - Toronto Lab: Sam Lightstone, Randy Horman, Mark Wilding
  - SVL: Jim Teng (z/OS), Bryan Smith (tools)
  - Research: Guy Lohman (ARC), Joe Hellerstein (Watson)
- **History:**
  - Index Advisor prototyped in 1998
  - Project formed in early 2000
    - Previously called Self-Managing And Resource Tuning (SMART)
  - IBM-wide Autonomic Computing initiative
  - Evolutionary: Multi-Release Rollout
- **Refn:** SMART: Making DB2 (More) Autonomic, VLDB 2002
An Autonomic DB2: What’s our Focus?

- Up and Running
  - pre-purchase capacity planning tools
  - automate install and initial configuration
- Design
  - advise on logical and physical design
- Maintenance
  - automatic tuning for queries, resources
  - physical maintenance (statistics collection, reorganization, ...)
- Problem Determination and Resolution
  - detecting existing, and predicting future
  - user notification
  - self-correcting features
- Availability and Disaster Recovery
  - availability
  - backup and log management

Approach

- LOTS of ideas & prototypes underway!
- Leverage existing infrastructure in DB2
  - Optimizer’s detailed model of run-time environment
  - Monitoring tools
  - Workload captured for DB2 Index Advisor
  - DB2 Control Center GUIs, Data Management Tools
- Exploit IBM’s strength in software research
- Get something out there, & improve it over time!
  - Where the need is greatest
  - Where we have ideas/skills
- Earn the DBA’s trust
  - Create tools that speed/simplify/improve DBA's job
  - “Free the DBA!” -- DBA retains ultimate decision power
  - Longer-term goal is complete automation
Agenda

- Introduction & Motivation
- DB2 Autonomic Computing Project
- Existing DB2 Autonomic Features
  - Index Advisor
  - Configuration Advisor
  - Health Advisor
- New in “Stinger”
  - Design Advisor
  - Automated Statistics Collection
- Ad. Tech. & Research Projects
  - Progressive Optimization
- Conclusions

Index Selection: The Problem

- Huge number of possible indexes
- Dependent upon workload (queries) anticipated
- For each query, user has to trade off:
  - Benefits:
    - Apply predicates efficiently (save reading entire table)
    - Provide a row ordering needed by query for certain operations
    - Index-only access (avoid fetching data pages)
    - Enforce uniqueness (e.g., primary keys)
  - Costs:
    - Storage space
    - Updating
    - More plans for the optimizer to evaluate
- Time-consuming trial & error process to choose the best set of indexes
  - Create index (system sorts entire table on key of the index)
  - Collect statistics on it (system scans entire table AND all indexes)
  - Re-optimize all queries in all apps that might benefit
  - See if
    - Index was used
    - Performance improves
  - Iterate!
Solution(1): DB2 Index Advisor (V6, 1999)

Exploits Optimizer to:
- Suggest good candidates, per query
- Evaluate combinations, for entire workload

SQL Workload

Constraints on resources
- Disk Space Allowed
- Time/Complexity

Indexes Designed by DB2 for Your Environment & Workload

Index Advisor (DB2 V6) – The Math

- Variant of well-known "Knapsack" Problem
- Greedy "bang-for-buck" solution is optimal, when integrality of objects (indexes) is relaxed

- For each query Q:
  - Baseline: Explain each query w/ existing indexes, to get cost E(Q)
  - Unconstrained: Explain each query in RECOMMEND INDEXES mode, to get cost U(Q)
  - Improvement ("benefit") B(Q) = E(Q) - U(Q)

- For each index I used by one or more queries:
  - If query Q used index I, assign "benefit" B(I) to index I:
    - B(I) = B(I) + B(Q)
  - Assign "cost" C(I) = size of index in bytes
  - Order indexes by decreasing B(I) / C(I) ("bang for buck")
  - Cut off where cumulative C(I) exceeds disk budget
  - Iterative improvement: exchange handfuls of "winners" with "losers"

Configuration Parameters

**The Problem:**
- Almost 150 configuration parameters in DB2 UDB
- Users didn’t know:
  - How to choose the right values
  - Possible interactions between them
- Had to stop and restart DB2 to have them go into effect
  - Bad for availability, too!

**Solution(1):**
- Make many configuration parameters dynamic!
- No need to stop and restart DB2 to change them
- Not easy to implement, e.g. shrinking buffer pool
- Prerequisite to automatically tuning them

**Solution(2): Configuration Advisor (V8.1, 2002)**

- **What is it?**
  - Sets ~36 configuration parameters key to performance, including:
    - Memory heaps (buffer pool, sort heap, statement cache)
    - Connections (max and average, remote/local)
  - Based upon answers to 7 high-level questions
  - Equations from performance experts relate parameters

- **Enhanced in V8.1:**
  - Available in V7 as "Performance Configuration Wizard"
  - More sophisticated model in V8.1
  - Easier to invoke via:
    - CREATE DATABASE command extension
    - AUTOCONFIGURE command
  - Better decisions for OLTP and DSS workloads
  - Surprising benchmark results
    - (well-known, industry-standard OLTP workload)
Configuration Advisor: The Questions

- Percentage of Real Memory to dedicate to DBMS
- OLTP vs. Complex query vs. Mixed
- Length of Transaction (typical # of SQL queries per transaction)
- Relative priority of Recovery vs. Query speed
- Number of Local and Remote Connections
- Whether the database is populated or not
- Isolation Level

DB2 Configuration Advisor vs. Human Experts

- Speeds deployment
- Improves performance
- Frees up resource

DB2 Configuration Advisor Results
Health Monitoring

- **The Problem:**
  - How do you know if DB2 is running okay, performing well?
  - What do you do if you do manage to figure out it's "unhealthy"?
  - Too difficult to determine what to monitor and when to monitor it
  - Need to set up monitors, notification & resolution mechanisms

- **The Solution: Health Center**
  - DB2 monitors its own health right out of the box
  - Notifies user upon encountering unhealthy conditions
  - Advises on severity of condition, and suggests resolutions
  - Initiates corrective action if required, requested
  - Easy installation: just provide an e-mail or pager address
  - User can modify thresholds for notification

**Solution: Health Center (V8.1)**

```bash
(horman@healthy) /home/horman $
$ db2 get health snapshot for DBM

Database Manager Health Snapshot
Node type                                      = Database Server with local clients
Instance name                                  = horman
Snapshot timestamp                             = 03-27-2002 13:24:51.799180
Database Manager Health Indicators:
  Health Indicator ID                        = 2 (db2.sort_privmem_util)
  Value                                   = 86
  Alert state                             = warning
```
Health Monitor and Health Center

- Alerts sent by Health Monitor to Contacts on Contacts List
- Details in Notification Log can be viewed via Health Center, Web Health Center, CLP, or API

Health Center: "Drilling Down"

- If you need to do some digging/investigation before choosing an appropriate action, Health Center launches tools in context

**Use Memory Visualizer** to consider "competitors" of a constrained resource

Other investigative actions include:
- Storage Management
- Indoubt Transaction Manager
- Event Monitor

**NOTE:** for many corrective actions, DB/DBM cfg parms can be dynamically updated!!!
Agenda

- Introduction & Motivation
- DB2 Autonomic Computing Project
- Existing DB2 Autonomic Features
  - Index Advisor
  - Configuration Advisor
  - Health Advisor
- New in “Stinger”
  - Design Advisor
  - Automated Statistics Collection
- Ad. Tech. & Research Projects
  - Progressive Optimization
- Conclusions

Design Advisor (“Stinger”)

- An extension of existing Index Advisor (V6)
- Headquarters for all physical database design
- Recommends any combination of:
  - Indexes
  - Materialized Views (Materialized Query Tables (MQTs))
    - Called Automatic Summary Tables (ASTs) before V8.1
  - Partitioning of tables (in partitioned environment)
  - Multi-Dimensional Clustering (MDC) storage method (New in V8.1)
- Takes interactions of these into consideration
- Status:
  - Coming soon (“Stinger”)! 
  - Beta testing on customer databases now!

REFNS:

- "DB2 Design Advisor: Integrated Automatic Physical Database Design", VLDB 2004
- "Recommending Materialized Views and Indexes with IBM’s DB2 Design Advisor", IEEE Intl. Conf. on Autonomic Computing (ICAC 2004)
Multi-Dimensional Clustering (MDC) – V8.1

Cells are the portion of the table containing data having a unique set of dimension values; the intersection formed by taking a slice from each dimension. Blocks are the storage units that compose each cell.

Each cell contains one or more blocks.

Design Advisor Architecture (MQTs only)

db2advis utility

- Get Workload
- Get Candidate MQTs
- Determine Stats (optionally sample)
- Choose Solution
- Evaluate Solution

Workload

Candidate MQTs

Optimizer

DB2 Server

“RECOMMEND” mode

“EVALUATE” mode

Costs of Queries

© 2004 IBM Corporation

© 2004 IBM Corporation
Design Advisor: Partition Advisor

**Scope:**
- DB2 "partitioned environment" (was called EEE prior to V8.1)
- "Shared-nothing" parallelism
- Data stored horizontally partitioned
  - In a partition group, spread across specified partitions
  - Based upon hashing of partitioning column(s)
  - May be replicated across all partitions of partition group
- Need to co-locate similar values for joins, aggregation in queries
- Partitioning required for a given table may be different
  - Between queries
  - Even within a query (joined on different columns)!

**Problem:** What is optimal partitioning for each table, given:
- Workload of queries
- Schema, including set of partition groups & tablespaces
- Statistics on database


Performance Improvement on Customer Database (Partitioning only)

- 50 queries and 500 possible configurations
- Rank_best algorithm converges the fastest, 22% speedup
Automating Statistics Collection:

- **Problem:**
  - Optimizer requires that statistics on database be
    - Up to date (after updates)
    - Complete (multi-column)
  - User must invoke RUNSTATS

- **Solution:** Automate RUNSTATS
  - *Invocation* scheduled and prioritized
  - *Run silently* as a background daemon
    - Throttled based upon workload
  - LEO the LEarning Optimizer determines which *statistics needed*
    - Based upon learning from past queries
    - Groups of columns
      - Enables correlation detection
    - Communicated to RUNSTATS via statistical “profiles”

- Shipping in DB2 “Stinger”

**I can't believe I did that!**

Refn: "LEO -- DB2’s LEarning Optimizer", Intl. Conf. on Very Large Data Bases 2001 (Rome, Sept. 2001)
Cost depends heavily on number of rows processed (cardinality)

Optimizer’s model limited by simplifying assumptions
  - Especially due to statistical correlation between columns
  - EXAMPLE: WHERE Make = 'Honda' AND Model = 'Accord'
  - Impossible to know a priori which columns are correlated!

Why not use actual results from executed queries to
  - Validate statistics and assumptions
  - Advise when/how to run expensive statistics collection
  - Gather statistics that reflect the workload
  - Repair the model for optimizing "similar" future queries

Could achieve automatically
  + Better quality plans
  + Reduced customer tuning & administration time
  + Reduced IBM support time

Part of Automated RUNSTATS in “Stinger”

Query Optimization -- Today

SQL Compilation → Statistics

Optimizer

Best Plan

Plan Execution
EXPLAIN gives Optimizer's Estimates

1. Monitor

New: Capture Actual Number of Rows!

1. Monitor
Figure Out Where the Differences Are

1. Monitor

2. Analyze

3. Feedback

Augment Statistics with Adjustments

1. Monitor

2. Analyze

3. Feedback
Exploit: Learning in Query Optimization!

SQL Compilation

Optimizer

Best Plan

Plan Execution

Estimated Cardinalities

Actual Cardinalities

4. Exploit

3. Feedback

2. Analyze

Adjustments

1. Monitor

Agenda

- Introduction & Motivation
- DB2 Autonomic Computing Project
- Existing DB2 Autonomic Features
  - Index Advisor
  - Configuration Advisor
  - Health Advisor
- New in “Stinger”
  - Design Advisor
  - Automated Statistics Collection
- Ad. Tech. & Research Projects
  - Progressive Optimization
- Conclusions
Progressive Optimization (POP)

- CHECKpoints for cardinality estimates at TEMP tables
  - Pre-computed validity range for this plan
- When check fails,
  - Treat partial results as MQTs
  - Replace estimated cardinality with actual for the MQTs
  - Re-optimize the currently running query

Reuse results from partial execution

Ref: “Robust Query Processing through Progressive Optimization”. ACM SIGMOD 2004

Conclusions & Future Directions

- Autonomic features of DB2:
  - Key to lowering Total Cost of Ownership
  - A major DB2 differentiator
  - Now in DB2 are the "tip of the iceberg"!
  - Many more on the way in technology stream from
    - Development
    - Research
    - Universities
  - Rollout prioritized by Customers ("Free the DBAs"!)
  - Beginning to integrate IBM components autonomically
  - Ultimate goal is complete automation!
Autonomic computing systems are self-managing and always available, analogous to the human autonomic nervous system depicted abstractly on the cover. Papers in this issue describe a variety of research projects in which the concepts of autonomic computing are being developed.

http://www.ibm.com/autonomic

Finis