Self-tuning DB Technology & Info Services: from Wishful Thinking to Viable Engineering

Gerhard Weikum, Axel Moenkeberg, Christof Hasse, Peter Zabback

Teamwork is essential. It allows you to blame someone else.

Acknowledgements to collaborators:
Outline

- Auto-Tuning: What and Why?

- The COMFORT Experience
  - The Feedback-Control Approach
  - Example 1: Load Control
  - Example 2: Workflow System Configuration
  - Lessons Learned

- Where Do We Stand Today?
  - Myths and Facts

- Where Do We Go From Here?
  - Dreams and Directions
Auto-Tuning: What and Why?

DBA manual 10 years ago:

- Tuning experts are expensive
- System cost dominated and growth limited by human care & feed
  \( \Rightarrow \) automate sys admin and tuning!

<table>
<thead>
<tr>
<th></th>
<th>Default value</th>
<th>OK to change?</th>
<th>Range of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESSES</td>
<td>25</td>
<td>yes</td>
<td>5 to O/S dependent</td>
</tr>
<tr>
<td>ROW_CACHE_BUFFER_SIZE</td>
<td>200</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The size in bytes of the row cache circular buffer.</td>
</tr>
<tr>
<td>ROW_CACHE_ENQUEUES</td>
<td>100</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The number of &quot;concurrent row cache managed enqueues&quot;. In CREATE TABLE, each NOT NULL constraint entry requires 1.</td>
</tr>
<tr>
<td>ROW_CACHE_INSTANCE_LOCKS</td>
<td>100</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
Auto-Tuning: What and Why?

DBA manual today:
Intriguing and Treacherous Approaches

**Instant tuning:** rules of thumb
+ ok for page size, striping unit, min cache size
− insufficient for max cache size, MPL limit, etc.

**KIWI principle:** kill it with iron
+ ok if applied with care
− waste of money otherwise

**Columbus / Sisyphus approach:** trial and error
+ ok with simulation tools
− risky with production system

**DBA joystick method:** feedback control loop
+ ok when it converges under stationary workload
− susceptible to instability

An engineer is someone who can do for a dime what any fool can do for a dollar.
Outline

Auto-tuning: What and Why?

- The COMFORT Experience
  - The Feedback-Control Approach
  - Example 1: Load Control
  - Example 2: Workflow System Configuration
  - Lessons Learned

Where Do We Stand Today?
- Myths and Facts -

Where Do We Go From Here?
- Dreams and Directions -
Feedback Control Loop for Automatic Tuning

- Observe
  Need a quantitative model!
- Predict
- React

Current Load and Performance Indicators

Data and Workload Statistics

Adaptive Tuning

Control Parameters

Load Control
- Multi-programming Level
- Transaction Cancellation
  - On/Off

File Manager
- Striping Unit
- Disk Cooling
  - On/Off

Buffer Manager
- Page Pool Size
- Replacement Policy
  - LRU/MRU
Performance Predictability is Key

”Our ability to analyze and predict the performance of the enormously complex software systems ... are painfully inadequate”

(Report of the US President’s Technology Advisory Committee 1998)

ability to predict

\[
\text{workload} \times \text{knobs} \rightarrow \text{performance}
\]

!!!  !!!!  ???

is prerequisite for finding the right knob settings

\[
\text{workload} \times \text{knobs} \rightarrow \text{performance goal}
\]

!!!  ???  !!!
Level, Scope, and Time Horizon of Tuning Issues

- (workflow) system configuration
  - (EDBT’00, Sigmod’02)

- query opt. & db stats mgt.
  - (VLDB’99, EDBT’02)

- caching
  - (Sigmod’93, ..., ICDE’99)

- load control
  - (ICDE’91, VLDB’92, InfoSys’94)

- index selection

- data placement
  - (Sigmod’91, VLDB J. 98)
Level, Scope, and Time Horizon of Tuning Issues

(workflow) system configuration
(EDBT’00, Sigmod’02)

query opt. & db stats mgt.
(VLDB’99, EDBT’02)

caching
(Sigmod’93, ..., ICDE’99)

load control
(ICDE’91, VLDB’92, InfoSys’94)

index selection

data placement
(Sigmod’91, VLDB J. 98)
uncontrolled memory or lock contention can lead to performance catastrophe
How Difficult Can This Be?

typical Sisyphus problem

arriving transactions

trans. queue

active trans.

response time [s]

DBS

MPL

typical Sisyphus problem
Adaptive Load Control

\[ \text{conflict ratio} = \frac{\# \text{locks held by all trans}}{\# \text{locks held by running trans}}. \]

Critical conflict ratio \( \approx 1.3 \)

Transaction admission

Transaction cancellation

Backed up by math (Tay, Thomasian)
Performance Evaluation: It Works!

avg. response time [s]

Creative redefinition of problem: replace one tuning knob (MPL) by another – less sensitive – knob (CCR)

Robust solution requires
- math for prediction and
- great care for reaction
WFMS Architecture for E-Services

Clients

Comm server

WF server type 1

WF server type 2

App server type 1

App server type n
Workflow System Configuration Tool

Workflow Repository

Operational Workflow System Config.

Mapping

Monitoring

Admin

Modeling

Calibration

Evaluation

Recommendation

Hypothetical config

Max. Throughput

Avg. waiting time

Expected downtime
Workflow System Configuration Tool

Goals:
- min(throughput)
- max(waiting time)
- max(downtime)
+ constraints

Long-term feedback control
- aims at global, user-perceived metrics and
- uses more advanced math for prediction

Min-cost re-configure.
Outline

Auto-Tuning: What and Why?

- The COMFORT Experience
  - The Feedback-Control Approach
  - Example 1: Load Control
  - Example 2: Workflow System Configuration

- Lessons Learned

- Where Do We Stand Today?
  - Myths and Facts

- Where Do We Go From Here?
  - Dreams and Directions
COMFORT Lessons Learned: Good News

+ Observe – predict – react approach is the right one and applicable to both short-term and long-term feedback control; prediction step is crucial

+ Practically viable self-tuning, adaptive algorithms for individual system components

+ Automated comparison against performance goals and automatic analysis of bottlenecks
  + Early alerting about workload evolution and necessary hardware upgrades
    + minimizes period of degradation,
    + minimizes risk of performance disaster,
    + and thus benefits business
COMFORT Lessons Learned: Bad News

- Automatic system tuning based on few principles:
  Complex problems have simple, easy-to-understand, wrong answers

- Interactions across components and interference among different workload classes can make entire system unpredictable
Outline

The Problem – 10 Years Ago and Now

The COMFORT Experience

The Feedback-Control Approach

Example 1: Load Control

Example 2: Workflow System Configuration

Lessons Learned

Where Do We Stand Today?
- Myths and Facts -

Where Do We Go From Here?
- Dreams and Directions -
Where Do We Stand Today?- Good News

*Advances in Engineering:*
- Eliminate second-order knobs
- Robust rules of thumb for some knobs
- KIWI method where applicable

*Scientific Progress:*
+ Storage systems have become self-managing
+ Index selection wizards hard to beat
+ Materialized view wizards
+ Synopses selection and space allocation for DB statistics well understood
Where Do We Stand Today?  
– Myths and Facts –

- systems have adaptable mechanisms everywhere → they are self-managing
- query optimizers produce proper ranking of plans → QOs are mature
- many papers on caching → DBS memory mgt. solved
- OLTP and OLAP strictly separated
- concurrency control is least wanted subject for conf.

- adaptive systems need intelligent control strategies
- accurate estimates needed for scheduling, mediation etc.
- memory-intensive workloads, sophisticated caching options → very difficult problem
- mixed workloads require black art for MPL tuning etc.
- no theory for isolation levels other than serializability
Outline

The Problem – 10 Years Ago and Now

The COMFORT Experience
  The Feedback-Control Approach
  Example 1: Load Control
  Example 2: Workflow System Configuration
  Lessons Learned

Where Do We Stand Today?
  - Myths and Facts -

Where Do We Go From Here?
  - Dreams and Directions -
Autonomic Computing: Path to Nirvana?

Vision:
all computer systems must be self-managed, self-organizing, and self-healing

Motivation:
• ambient intelligence (sensors in every room, your body etc.)
• reducing complexity and improving manageability of very large systems

Role model:
biological, self-regulating systems (really ???)

My interpretation:
need component design for predictability:
self-inspection, self-analysis, self-tuning
aka. observation, prediction, reaction
Summary & Concluding Remarks

☆ Major advances towards automatic tuning during last decade:
  • workload-aware feedback control approach fruitful
  • math models and online stats are vital assets
  • „low-hanging fruit“ engineering successful
  • important contributions from research community
    (AutoRAID, AutoAdmin, LEO, Shasha/Bonnet book, etc.)

☆ Problem is long-standing but very difficult and requires good research stamina

☆ Success is a lousy teacher. (Bill Gates)

☆ Major challenges remain:
  path towards „autonomic“ systems requires
  rethinking & simplifying component architectures
  with design-for-predictability paradigm