Management as major challenge

- Much effort into building high-performance, scalable database systems
- DB systems and workloads are complex
- Ease of management as evolutionary step
  - difficult to tune systems to get their full potential
  - large number of (expensive) administrators required

Performance tuning and policy questions

- Important subset of management problems
- Some nightmare scenarios for a DB admin
  - clients complain about the performance they are getting from the database
    - admin needs to track system bottlenecks: CPU, memory, network, disks
    - at upgrade time, which resources should be upgraded?
      - can the admin quantify the effect of buying more RAM for each of the storage bricks?
  - Ideally the system itself would answer these

Common approaches

- DB admins as external expert systems
  - paid to act on their rules of thumb
  - difficult to quantify the effect of admin decisions
  - DB help: 400+ performance counters
Common approaches (cont…)

• DB modules as internal expert systems
  • use if...then… rules of thumb
  • thresholds and rules may be incorrect or outdated
  • difficult to resolve conflict among different rules
  • no root-cause analysis of problem
• Overprovision
  • expensive for multi-million dollar systems
  • tuning problems do exist, even in an over-provisioned system

• Self-predictability is not built-in from the start

What we propose: self-prediction

• DB should incorporate ability to answer
  What...If... questions about own performance

Example What...If... question

• What would be the throughput/latency of new-order transactions if the buffer pool size of the DB is doubled?
• What would be the throughput/latency of new-order transactions if I buy faster disks?

• If the DB disk queues > 8, then double the buffer pool size

What we propose: self-prediction

• DB should incorporate ability to answer
  What...If... questions about own performance
  What...If... modules built-in from the start
  What...If... interfaces convert complex tuning decisions into search-based approaches

• Main contribution:
  • prototype implementation and evaluation of a Resource Advisor for a commercial DBMS
### Resource Advisor goals

- Built inside SQL Server 2005 branch
- Answers *What*…*If*… questions about resource upgrades (CPU, buffer pool, network, disks)
  - point predictions and trend forecasting
- Provides visualization of system performance

### Outline

- Background and motivation
- Design and implementation of a Resource Advisor
- Evaluation
- Future work
- Summary

### Architectural view of Resource Advisor

- Detailed, yet general, system instrumentation
  - away from performance counters
  - towards end-to-end activity tracking
  - enables resource accounting and critical path info
  - built-in at system development time
- Specific *What*…*If*… modules
  - query system instrumentation framework to answer specific *What*…*If*…questions
  - some *What*…*If*…modules built-in from the start, others can be added as needed
  - utilize operational laws and simulation

### General system instrumentation

- Per-transaction, per-resource monitoring
  - stored procedure calls
  - buffer cache accesses + prefetching
  - SQL server task scheduler activity
  - NT threads scheduling
  - I/O activity
- Magpie used for trace collection [OSDI2004]
- Virtual performance counters & visualization
Models for answering \textit{What...If...} questions

- Two fundamental system assumptions
  - past is good indicator of future
  - DB performance is some function of system resources and workload
    - goal of prediction is to approximate the above function
- What we are and are not targeting
  - no help for initial purchase decision
  - no predictions during drastic changes
  - can predict when not to predict

What-if modules

- Goal: predict client performance as buffer pool size changes
- Performance metrics of interest
  - closed-loop throughput with a given think time
  - system saturation throughput
  - open-loop request response time

Throughput prediction

- Analytical model for throughput predictions
  - bottleneck analysis on CPU/RAM/disks/(client)
  - resource with highest demand determines the overall throughput of the system’s workload
  - query demands and predict bottleneck
  - \( T = \min \{ T_{\text{max/cpu}}, T_{\text{max/io}}, T_{\text{max/workload}} \} \)
  - \( T_{\text{max/cpu}} = t_{\text{cpu}}^{-1} \)
  - \( T_{\text{max/io}} = (n_{\text{io}}/t_{\text{io}})^{-1} \)
  - \( T_{\text{max/workload}} = N_{\text{users}}/t_{\text{think}} \) (for closed-loop)

\( n_{\text{io}}, t_{\text{io}} \): need to be predicted
Response time prediction

• Response time depends on critical path

• $t_X = t_{CPU} + t_{storage}$

• $t_{CPU}$ is measured

• $t_{storage}$ depends on number of blocking I/Os per transaction and service time per I/O

System integration

• System resources are augmented with analytical or simulation modules
  
  • DB cache manager: simulation module
  
    - predicts blocking and non-blocking I/Os as function of cache size

  • DB storage subsystem: analytical module
   
    - predicts I/O service time as function of storage parameters (seek, rotational latency, queue size, etc)

• A self-predicting system must integrate these models in and treat them as first-class citizens

System integration

Outline

• Background and motivation

• Design and implementation of a Resource Advisor

• Evaluation

• Future work

• Summary
Experimental setup

- 2 variants of TPC-C (open, closed)
- 100 warehouses (~10GB)
- 200 clients accessing DB concurrently
- Xeon 2.66GHz machines
- 64-128-256-512-1024 MB of RAM
- DB server runs SQL Server 2005

Buffer pool model evaluation

- How accurate is the buffer pool model in predicting I/O sequence?

Storage model evaluation

- How accurate is the storage model in predicting service times?
Storage model evaluation

Closed-loop throughput prediction

- For a closed-loop workload, can the Resource Advisor predict throughput at different buffer pool sizes?

Closed-loop throughput prediction

Predicting saturation from open-loop

- Given a non-saturation workload, can it predict saturation throughput at different buffer pool sizes?
Predicting saturation from open-loop

![Graph](image)

Predicting open-loop response time

- Given a non-saturation workload, can the Resource Advisor predict response time at different buffer pool sizes?
- Response time prediction is per-transaction type

Predicting open-loop response time

![Graph](image)

Resource Advisor overheads

- 6.2% CPU overhead
- 1.2% for tracing and logging
- 0.44MB/s of trace data

- Instrumentation burden
  - 189 lines of code in 6 source files
- What-if modules burden
  - 1150 lines of code
Future work

• In SQL Server
  • predict locking effects as more CPUs are added

• Apply similar techniques to a distributed storage system
  • Ursa Minor developed at CMU [FAST05]
  • opportunity to solve real admin’s What…If… questions

Summary

• Common tuning approaches inadequate
  • system offers little help to external admins

• Self-predictability should be built-in
  • need general instrumentation
  • need specific What…If…modules

• Main contribution: prototype Resource Advisor inside a commercial DBMS