Database Tuning

Hardware Tuning

Nikolaus Augsten

University of Salzburg
Department of Computer Science
Database Group

Unit 11 – WS 2014/2015

Adapted from "Database Tuning" by Dennis Shasha and Philippe Bonnet.

Outline

Overview

- Tuning the storage subsystem involves configuring:
  - disk allocation
  - disk array (RAID level)
  - controller cache

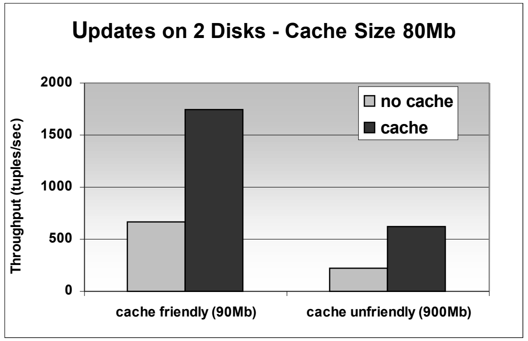
Disk Allocation – Raw vs. Cooked Files

- Cooked file:
  - regular operating system file
  - buffered through the operating system
  - logically contiguous blocks might not be physically contiguous
  - possibly indirection to access a particular page (inodes in Unix/Linux)
- Raw file: also "character special device"
  - block device (hard disk) configured as raw device (using `raw` command) e.g., `/dev/rsd0f` is the raw device of block device `/dev/sd0f`
  - not buffered by the operating system
  - logically contiguous blocks are physically contiguous
  - more efficient than cooked files

The Controller Cache

- Read Cache: performs read-ahead
  - after read request, controller continues to read and store in cache
  - database can do better read-ahead since it knows the access patterns
  - in general it is better to turn read cache off!
- Write-back mode: request terminates when data is written to cache
  - data is written from cache to disk later
  - writes become faster since they do not have to wait for the disk
  - if cache contents get lost (power failure, no battery), then data is lost
- Write-through mode: request terminates when data is written to disk
  - if cache has not battery, this mode is safer
  - if cache is overloaded, write-through might be more efficient (depends on the efficiency of the replacement policy algorithm)

Write-Back Mode – Experiment



- Cache controller in write-back mode vs. no cache.
- Cache friendly load: volume of update slightly larger than cache.
- Cache unfriendly: volume of update much larger than cache.
- Write-back gives similar benefit in both cases.
- The controller implements an efficient replacement policy.

SQL Server 7 on Windows 2000

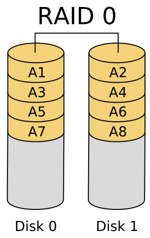
RAID – Redundant Arrays of Inexpensive Disks

- RAID array
  - multiple hard disks
  - one logical disk (operating system sees only one disk)
  - disks divide and replicate data
  - RAID controller is the interface
- Benefits:
  - fault tolerance by introducing redundancy
  - increased throughput due to parallel disk access

RAID Levels

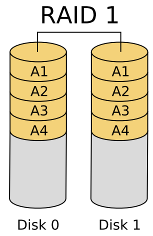
- RAID configurations are numbered ("levels"):
  - RAID 0: striping
  - RAID 1: mirroring
  - RAID 5: rotated parity striping
  - RAID 10: striped mirroring
- other (less important) RAID levels exist

RAID 0 – Striping



- RAID 0 – striping:
  - data is split into stripes of the same size
  - consecutive stripes are written onto consecutive disks
- Example: stripe size 1kB, 2 disks
  - write datablock A of $8kB$
  - block is split into $A = A_1 + A_2 + \ldots + A_8$
  - disk 1: A_1, A_3, A_5, A_7, disk 2: A_2, A_4, A_6, A_8
- Read/Write: RAID 0 with n disks, stripe size s
  - small data $\leq s$: read/write results in access to one physical disk
  - large data $\geq s \times n$: read/write results in parallel access to n disks
- Benefits/drawbacks:
  - \+ fast sequential read/write and concurrent seeks
  - \+ 100% utilization of disk space (=cheap)
  - \- no fault-tolerance (array inaccessible if single disk fails)
- Database use: temporary files

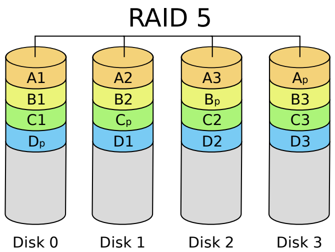
RAID 1 – Mirroring



- RAID 1 – mirroring: 2 disks
  - the same data is written to both disks ("mirrored")
  - no striping
- Example: 2 disks
  - write datablocks A_1, A_2, A_3, A_4
  - disk 1: writes A_{1-4}, disk 2: writes A_{1-4}
- Write one data block:
  - physical write two both disks
  - operation terminates when slower disk is done (!)
- Read of one data block: physical read from single (least busy) disks
- Benefits/drawbacks:
  - \+ fault tolerant (no interruption if one disk fails)
  - \+ concurrent seeks (faster random read access)
  - \- only 50% utilization of disk space (=expensive)
  - \- write speed not increased from single-disk solution
- Database use: log file (fault tolerance, sequential writes)

RAID 5 – Rotated Parity Striping

- RAID 5 – rotated parity striping: n disks
  - fault tolerance by error correction (instead of full redundancy)
  - striped as in RAID 0, but $n - 1$ stripes have an additional parity stripe
  - parity stripes are evenly distributed over disks
- Example: stripe size 1kB, 4 disks
  - write datablock AB of $6kB$
  - datablock is split into $AB = A_1 + A_2 + A_3 + A_p + B_1 + B_2 + B_3 + B_p$
  - disk 1: A_1, B_1, disk 2: A_2, B_2, disk 3: A_3, B_p, disk 4: A_p, B_3



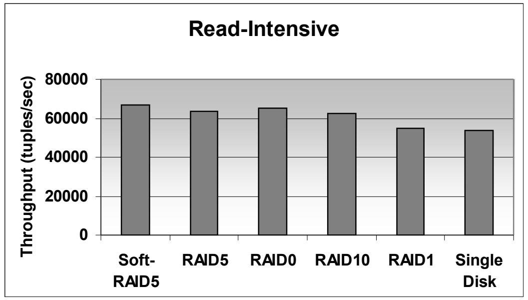
RAID 5 – Rotated Parity Striping

- Read: like RAID 0 with $n - 1$ disks, parity stripe is not read
- Writing 1 stripe requires 2 physical reads and writes
- Write: Update data stripe S with S'
  1. read old data stripe S and old parity stripe P
  2. new parity stripe $P' = S \text{ xor } S' \text{ xor } P$
     (for each bit flipped between S and S' flip the corresponding bit in P)
  3. write S' and P' (substituting S and P, respectively)
- Recovery: n disks, failure on disk x
  - S_i is the stripe on disk i (either parity or data)
  - lost stripe $S_x = S_1 \text{ xor} \ldots \text{xor } S_{x-1} \text{ xor } S_{x+1} \text{ xor} \ldots \text{xor } S_n$
- Benefits/drawbacks:
  - \+ fault tolerant (slowdown, but no interruption if one disk fails)
  - \+ fast sequential read and concurrent seeks
  - \+ $(100 - 100/n)$% utilization of disk space (=cheap)
  - \- write is slower than single-disk solution
  - \- recovery after failure much more difficult than with RAID 1
- Database use: data and index files (if reads predominate writes)

RAID 10 – Mirroring + Striping

- RAID 1+0: mirrored RAID 0
  - stripe data on first $n/2$ disks (as in RAID 0)
  - use the other disks to mirror these disks
- Benefits/drawbacks:
  + best performance of all RAID levels
  + fault tolerant (no interruption if one disk fails)
  + fast sequential read/write and concurrent seeks
  - 50% utilization of disk space (=expensive)
- Database use:
  - log file if RAID 1 is too slow
  - data and index files if writes are too slow on RAID 5

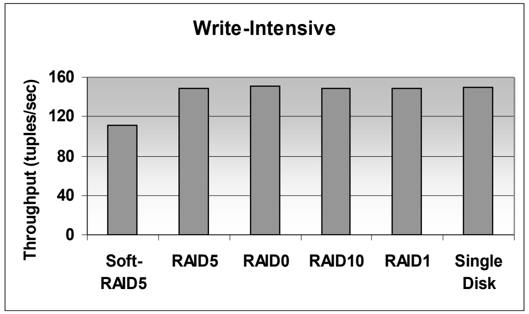
RAID with Read-Intensive Application – Experiment



- RAID 1 slightly improves on single disk solution.
- Striped RAID levels (0,5,10) significantly improve read performance.

SQL Server 7 on Windows 2000

RAID with Write-Intensive Application – Experiment



- Software RAID 5: negative impact of the additional read/write operations clearly visible.
- Hardware RAID 5: controller cache hides negative impact of additional read/write operations.

SQL Server 7 on Windows 2000

Outline

Exam

- Oral exam, around 15-20 minutes per student
- Concurrency:
  - while student A is being interviewed
  - student B is preparing for the interview
- Relevant documents:
  - slides of lecture notes
  - "Database Tuning" by Dennis Shasha and Philippe Bonnet
- Relevant chapters in the book:
  - Chapters 1-3 (except 2.4)
  - Chapter 4.6
  - Appendix B.1–B.4
- Do the exercises in the book!

Types of Exam Questions

Theory question Example: Explain write-ahead logging and how the logging mechanism can be tuned.

- illustrate situation (database buffer, log buffer, log file, data files)
- use correct terminology and give precise definitions (e.g., what is a checkpoint?)
- structure your answer (how does WAL work? list tuning opportunities, then discuss each of them)
- discussion (advantage/disadvantage)
- be prepared for the questions "why?" and "what if?"

Types of Exam Questions

Questions with practical part Example: What is transaction chopping and how does it work? Show the algorithm on the following transactions: T_1: $R(a), R(b), W(b), R(e)$, T_2: $R(b), R(e)$, ...

- answer theory question
- give an overview of how you are going to solve the example
- before you execute a step in the solution, explain the step
- again, be prepared for the questions "why?" and "what if?"

Follow-up questions

- detailed questions on the same topic to test understanding
- relation to other topics

Outline

Summary

- Tuning the Storage Subsystem
  - raw vs. cooked file
  - setting the controller cache mode
  - choosing the RAID level