CS590D Presentation:

Crash Recovery in QuickStore

Presented by
Changgui Shi

Nov. 19, 1996

References

1. **Seth J. White and David J. DeWitt**, *QuickStore: A High Performance Mapped Object Store*, **CS Dept., University of Wisconsin at Madison.**

2. **Seth J. White and David J. DeWitt**, *Implementing Crash Recovery in QuickStore: A Performance Study*, **CS Dept., University of Wisconsin at Madison.**

Overview

Four recovery methods are used in QuickStore.

1. Page differencing
2. Sub-page differencing
3. Whole-page logging, and
4. Redo-at-server

Related work including EXODUS, ARIES, ObjectStore, Texas and Dr. Hosking's work

Challenges for Recovery

- Memory-mapped architecture

  It makes detecting the portions of objects that have been updated more difficult.

- Non-traditional database applications

  Applications read objects into memory and then work on the objects intensively.

- Client/server architecture

  It raises the issues of cache consistency, client and server cooperations and network traffic control.

Recovery in ESM

ESM recovery is transaction based:

- Both client and server manage their own local buffer pools
- A client asks a page from a server, the server sends the page to the client, at the same time, it keeps a copy of the page and locks this page
- The client updates the page and generates log records
- Both dirty page and log records are sent back to the server at transaction commit time
- The server keeps the log records and updates the page

Page Differencing Approach

- A client keeps a buffer pool, a recovery buffer and a page descriptor
- Buffer pool stores working pages
- Recovery buffer stores clean pages
- Page descriptor is a table containing clean pages to dirty pages mapping as see in Figure 1
- At transaction commit time, log records are generated by comparing clean pages with dirty pages, as see in Figure 2
- A disadvantage of PD is CPU cost at client

Sub-Page Differencing Approach

- Each page is divided into equal size blocks
- An array is used to mark dirty and clean blocks, see Figure 3
- Block is the unit of sub-page diffing
- Each time a client updates blocks, it will also mark the corresponding entries of the array if necessary
- Log records can be generated by diffing clean blocks with dirty blocks as in PD
- One could avoid the cost of diffing by simply logging the entire array, e.g., SL-ESM

Whole-Page Logging Approach

- In WPL, an entire dirty page is sent back to the server as a log, no log record is generated.
- When the server receives a dirty page, it appends the page to the log and caches a copy in its own buffer pool.
- The server keeps enough redo and undo data before committing the transaction.
- For recoverying from crash, the server periodically takes checkpoints
- After a crash, the restarted server can get data and instructions from the log to redo/undo the last uncommitted transaction
- The advantages of WPL: reducing client CPU and memory costs
- The disadvantage of WPL: logging all dirty pages

Redo-at-Server Approach

- **A client only sends log records to a server**
- **When a log record is received at the server, the redo information in the log record is used to update the server's copy of the page**
- **An advantage of REDO: dirty pages are not sent to servers**
- **The disadvantage: A server repeats its clients' computations**

Performance Results

- **Small database, unconstraint cache:**

  PD-REDO > SD, PD, SL-ESM > WPL

- **Small database, constraint cache:**

  T2A: SD-ESM > PD-ESM > WPL

  T2B: SD-ESM > WPL > PD-ESM, PD-REDO

- **Big database:**

  client# less than four: WPL > SD-ESM-1/2

  client# greater than four: SD-ESM-1/2 > WPL

  In general, PD-ESM-4 closes to SD-EMS-4, both are worse than WPL and SD-ESM-1/2

(where '>' means better performance)

Conclusions

1. **Memory size and memory divisions affect recovery performance greatly**
2. **Differencing methods are better than whole-page logging**
3. **REDO has poor scalabilty**
4. **SD-ESM performance closes to PD-ESM's**
5. **To achieve good performance, one should balance the recovery work load between client and server**