Paper Summary for "Free Transactions with Rio Vista" by David E. Lowell and Peter M. Chen

Austin J. Heath
CS6210: Advanced Operating Systems

College of Computing, Georgia Institute of Technology

March 24, 2020

1 Introduction

Atomic transactions address the problem of corrupting files if an application is editing a file prior to a system crash. This mechanism allows programmers to manipulate files safely in memory, however, this comes at a cost of high overhead - requiring at least one synchronous disk I/O per transaction. The purpose of this paper is to introduce a system utilizing Rio and Vista that achieves a 5 microsecond overhead, reducing transaction overhead by a factor of 2000.

- **Rio** - a file cache hosted in main memory that survives operating system crashes by using an uninterruptible power supply. Applications will use Rio as a safe buffer for file system data.
- **Vista** - a user-level library tailored to run Rio, includes code for basic transaction routines, recovery code, and persistent heap management; written in C.

2 Related Works

Vista is most closely related to RVM, a Recoverable Virtual Memory system that supports persistent virtual memory in the face of system crashes. RVM transactions happen as such:

1. When a transaction begins, the process notifies RVM of the range of memory it intends to update - RVM copies this range into an in-memory region called the **undo log**.
2. The process writes the updates to memory.
3. If the transaction commits, RVM reclaims the space of the **undo log** and writes the new data to an on-disk region called the **redo log**.

4. Once the **redo log** fills up, new data is propagated to the disk and space is reclaimed in the **redo log**.

If a process or the entire system crashes, RVM recovers by replaying committed transactions from the **redo log**.

Some facts about RVM:

- RVM utilizes a common policy for transaction systems, the **no-force policy** - dirty database pages need not be forced synchronously to disk for every transaction, accelerating commit time.
- RVM performs up to **three** copy operations for each transaction.
  - A copy to the in-memory **undo log**.
  - A copy to the on-disk **redo log**.
  - Log truncation adds one additional copy for each modified datum in the log.

Differences between Vista and RVM:

- Leveraging Rio allows Vista to eliminate the redo log and its two copy operations per transaction.
- Vista doesn't use expensive system calls like **fsync** and **msync** that are used to flush data to disk.

3 The Rio File Cache

Rio utilizes a process called **warm reboot** in which, after a crash, Rio will write its file cache data to disk. Rio protects against power loss and system crashes by preventing the operating system from conducting wild stores that will corrupt its file cache. Rio improves reliability and performance over other recoverable memory systems by removing the need for synchronous I/O and using an infinite write-back delay - this allows for more data to be stored in the file cache and these updates will be preserved and written to disk when a system recovers.

There are **two** ways in the which applications can use Rio:

- **write** - system call to copy data from a user buffer to the Rio file cache. Data is permanent as soon as it is copied into the file cache.
- **mmap** - maps a portion of the file cache into process address space. No copying is needed to store data in the file cache after mapping

Differences between **write, mmap** and **fsync, msync**:

- Applications running on non-Rio systems must use **fsync** to force data to disk to guarantee immediate permanence.
- Applications using **mmap** on non-Rio systems must use **msync** to ensure data is stored to disk.

Pros and Cons of **write** with Rio:

- **Pros**
  - Isolates the file cache from application errors.
- **Cons**
  - Requires an extra memory-to-memory copy.

Pros and Cons of **mmap** with Rio:

- **Pros**
  - Significantly lower overheads; no system calls or copying for store instructions.
- **Cons**
  - Does not isolate the file cache from application errors.

4 The Vista Recoverable Memory

The central difference between Vista and RVM is that Vista eliminates the need for a **redo log**, two of the three copies described earlier, and all system calls. Vista's transactions are tailored for Rio - all operations are conducted within the Rio file cache.

When a transaction starts, Vista copies the original image of the data to be modified into an **undo log** inside of the Rio file cache - this memory is persistent unlike in the previous implementation where the **undo log** was hosted in volatile memory. The database is demand-paged into the file cache, mapped into the process's address space - all changes modify the file cache version of the database.

If a transaction is aborted, the original data from the **undo log** is copied back into the mapped database. When recovering from a system crash, Rio and Vista work together to reload the changes that were made in the file cache - Vista inspects any changes and determines if the changes were committed or not. If changes were not committed, Vista will discard the changes and copy the **undo log** for the modified memory region back into the database.

Factors that contribute to the simplicity of Vista:

- **Redo logs** and **log truncation** are eliminated because all modifications to memory with Rio are persistent.
- No **check-pointing** and **recovery code** required because the **redo log** is eliminated.
- Vista issues no disk I/O.
- Vista only handles the transaction features of **atomicity** and **persistence**.

Vista also provides a mechanism for applications to dynamically allocate persistent memory within the file cache - a **persistent heap**.

Advantages of **persistent heaps**:

- More flexible than file systems, applications can manipulate permanent data structures in their native form.

Disadvantages of **persistent heaps**:

- Metadata describing a persistent heap is mapped into user address space, leaving it corruptible. Vista attempts to protect against this by mapping each segment's metatdata into an isolated range of addresses.

5 Performance Evaluation

The test setup for Rio Vista's performance evaluation is as follows:

- Systems tested:
  - Vista
  - RVM
  - RVM-Rio
- Performance benchmarks:
  - Synthetic benchmarks to quantify the overhead of transactions as a function of transaction size.
  - **TPC-B**, industry-standard benchmark that emulates banking transactions.
  - **TPC-C**, industry-standard benchmark that emulates the activities of a wholesale supplier.
- Definitions:
  - **overhead per transaction** - the time per transaction of a given system minus the time per transaction of a system that does not provide atomic durability.
  - **time per transaction** - the running time divided by the number of transactions.

6 Performance Evaluation Results

The performance evaluation results by benchmark are as follows:

- The **synthetic benchmark** compares transaction size to transaction overhead in microseconds.
  - **Vista** - fastest with a minimum of 5 microseconds of overhead for small transactions. This overhead represents the cost of beginning the transaction. As transaction size increases, overhead almost scales linearly. Overhead increases due to the need to copy more data into the **undo log**.
  - **RVM-Rio** - the second-fastest with a minimum overhead of 500 microseconds. The overhead is generated from **write, fsync** calls, copying to the **redo log** and the need to conduct **log truncation**.
  - **RVM** - the slowest with a minimum overhead of 10 milliseconds. Overhead is incurred from synchronous disk I/O with the **redo log** created per transaction, as well as some portion of synchronous disk I/O conducted for **log truncation**.
- The **TPC-B and TPC-C benchmarks** compare database size to transactions per second:
  - **Vista** - results are similar to the synthetic benchmark while the database fits into memory. Thrashing begins to occur when the database is larger than available memory.
  - **RVM-Rio** - results are similar to the synthetic benchmark. The **RVM-Rio** begins to thrash earlier than Vista and RVM due to **double buffering**. **Double buffering** results from frequent writes to the database file, copying the database file into the Rio file cache. Two copies of the database exist: one in the process's address space and another in the Rio file cache.
  - **RVM** - results are similar to the synthetic benchmark. Like Vista, thrashing begins to occur when the database is larger than available memory.

Evaluators implemented and tested a **group commit** mechanism for RVM, accumulating and then synchronously writing changes to disk in an attempt to increase performance. While the **group commit** mechanism does increase performance, it also suffers from some problems:

- Depending upon the number of transactions accumulated, **group commit** mechanism can lengthen the response time of an individual transaction.
- Throughput for large transactions can be limited by the speed of writing to the **redo log**.
- **Group commits** works only when there are many concurrent transactions.

Again, Vista wins because it eliminates the need for a redo log, all system calls, and all but one of the copy operations required to conduct transactions with RVM. RVM-Rio is slower than Vista because of the extra copies and system calls. RVM is primarily slower than Vista because of synchronous disk I/Os.

7 Conclusion

Rio and Vista provide free persistent memory and nearly free transactions, allowing operating system designers the ability to use Vista for even finer-grained tasks for which disk I/O is too expensive.

The persistent memory abstraction provided by the Rio file cache allows a recoverable memory library like Vista the ability to eliminate the standard **redo log** mechanism and its accompanying overhead, two out of three copy operations present in RVM, and all system calls - this achieves a performance improvement three orders of magnitude greater than RVM with a fraction of the code size.

8 Works Cited

Lowell, David E., and Peter M. Chen. Free Transactions with Rio Vista, Computer Science and Engineering Division, Department of Electrical Engineering and Computer Science, University of Michigan, Oct. 1997, `https://web.eecs.umich.edu/~pmchen/papers/lowell97.pdf`.