Transactions

Lei Yan

25-11-21

Slides partially adopted from Rishabh Iyer

ACID

- Atomicity
- Consistency
- Isolation
- Durability

Guaranteeing Atomicity upon Crash

- How to ensure atomicity of transactions when the machine crashes?
- Log all updates of a transaction to persistent storage
  - Upon recovery:
    - If transaction committed: Replay the log
    - If transaction not committed: Discard the log
  - Write a "committed" bit to the log immediately before committing the transaction
    - Assumption: This is a atomic operation.
  - Redo vs. undo logs

Guaranteeing Isolation

Different ways of guaranteeing isolation

- Pessimistic
  - E.g., lock the transaction with a big lock
- Optimistic
  - E.g., abort transaction upon read-write conflicts

Design exercise

Consider a new low-cost CPU that has multiple cores, each core with its own cache, but no coherence/consistency guarantees (i.e., you may write to one cache on one core, and that value may never propagate on its own to the caches on other cores). A cache can be flushed in its entirety by privileged-mode code (i.e., by the operating system kernel). Say you want to run on this CPU a multi-threaded application in which data is shared between threads.

(a) Is it possible for the application to run correctly without any assistance from the OS (i.e., the OS does nothing to help compensate for the lack of cache coherence)?

(b) Is it possible for the OS to transparently provide to the application the illusion of a sequentially consistent memory (i.e., the memory hierarchy appears to execute all reads and writes to a single memory location in a total order that respects the program order of each thread)? In other words, could an application written under the assumption of sequentially consistent memory run on this OS and CPU correctly without any modification?

(c) What is the right balance of functionality between the application and the OS kernel when it comes to operating efficiently on this CPU? Propose a concrete system call API offered by the OS.

Transactional Memory (not discussed)

- How would you implement it?
  - In hardware
  - In software

CAP Theorem - Motivation

- Goal: Make system designers aware of a wide-range of trade-offs
- Terms
  - Consistency
  - Availability
  - Partition Tolerance
- All three properties are continuous NOT binary

CAP Theorem

- 2 out of 3 is misleading
  - Pick from **C**onsistency and **A**vailability in presence of partitions
- PACELC
  - In the presence of **P**artitions choose between **A**vailability and **C**onsistency, else choose between **L**atency and **C**onsistency

Dealing with Partitions

- During partitions you give up at least one of perfect consistency and availability
- You want both once communication is restored
- Points in spectrum to achieve this?
  - ❖ Give up availability during partition. Maintain perfect consistency
  - ❖ Give up consistency. Maintain perfect availability
  - ❖ Give up a little of both
    - Limit operations
    - Explicit post-partition recovery to restore invariants (e.g., ATM)

Consistency (ACID vs CAP)

- ACID - Maintain invariants
- CAP – **Client** sees a single copy
  - Strong consistency (from Baseball paper)

Consistency is an interface concept

Consistency Models (Replicated Data)

- Consistency is not binary, is a continuous spectrum
- Different users/applications may have different requirements

Strong Consistency
Eventual Consistency
Consistent Prefix
Bounded Staleness
Monotonic Reads
Read My Writes

Consistency Models (Replicated Data)

- Consistency is not binary, is a continuous spectrum
- Different users/applications may have different requirements

Strong Consistency	See all previous writes.
Eventual Consistency	See subset of previous writes.
Consistent Prefix	See initial sequence of writes.
Bounded Staleness	See all "old" writes.
Monotonic Reads	See increasing subset of writes.
Read My Writes	See all writes performed by reader.

Consistency Models (Replicated Data)

- Within an application, different clients may want different consistency guarantees
- Should be able to occupy individual sweet-spots on the tradeoff curve

Official scorekeeper
Umpire
Radio reporter
Sportswriter
Statistician
Stat watcher

Consistency Models (Replicated Data)

- Within an application, different clients may want different consistency guarantees
- Should be able to occupy individual sweet-spots on the tradeoff curve

Official scorekeeper	Read My Writes
Umpire	Strong Consistency
Radio reporter	Consistent Prefix & Monotonic Reads
Sportswriter	Bounded Staleness
Statistician	Strong Consistency, Read My Writes
Stat watcher	Eventual Consistency