Cloud Storage and Computation

Storage and computation distributed across a multiple datacenters. Want to provide support for transactions that execute general operations not just reads and writes.

- Database queries
- File-system operations
- Generic computation

Desired Properties

Large Scale Scale across a large number of data partitions.
Fault Tolerance Data should not be lost, should be accessible when needed.
High Performance Low access latency, minimal forced-writes, high throughput.
Consistency Clients should be provided the abstraction of one-copy serializability.

Why is Consistency Hard?

Data must exist as multiple copies in multiple locations for fault tolerance and low latency.
Data spread across multiple machines for scalability.
Want to give the illusion of a single copy of data on a single machine.

Status Quo

Ignore the problem, i.e., eventual consistency:
Require clients to sort out inconsistencies. Complicates application development, can lead to errors, limits scope of applications built atop infrastructure.

Two-phase commit:
Lock copies of data while changes are made. Higher latency, locking requirements, potential for abort.

Granola

Generic platform for distributed transaction coordination. Provides single-copy serializability. Support file systems, databases, object stores, etc.

Data stored at repositories. Transactions coordinated by masters. Read-only operations serviced by caches.

Masters and repositories replicated for availability.
Masters coordinate transactions across descendants without compulsory locking or aborts due to write conflicts.

Transaction Types

Single Repository (1R) Execute on a single repository, e.g., increment a value.
Multi-Repository (MR) Execute atomically across multiple repositories, e.g., atomic increment, snapshot read.
Predicated Multi-Repository (MP) Execute atomically after evaluating an application-defined predicate, e.g., transfer funds between accounts on different machines, commit a long-running database transaction.

Coordination Protocol

Single forced-write per transaction vs. at least two in 2PC. Forced-writes handled by replication protocol rather than costly disk writes.

Optimizations

Transaction Reordering Push 1R operations ahead of MR/MP operations that are awaiting votes. Track serial dependencies to maintain consistency.
Batched Agreement Run single replication phase for multiple 1R transactions.
Transaction Batching Combine multiple MR/MP transactions at master into a single batched transaction to increase parallelism.

Evaluation

2-repository throughput with 5% MP load
Single-repository request latency