CloudTran

TRANSACTIONAL DATA GRID REPLICATION

A Technology Whitepaper

CloudTran, Inc.

4000 Pimlico Drive, Suite 114

Pleasanton, CA 94588

www.CloudTran.com

CloudTran is the industry's only solution for managing the complete data layer in distributed computing environments. We integrate in-memory data grids with back-end data stores asynchronously and provide ACID transactionality across the entire architecture.

Introduction

This paper describes a new product from CloudTran called CloudTran Replicator Service™, which backs up data grid information between data centers.

As well as providing high availability and disaster recovery between data centers, CloudTran Replicator Service accomplishes the following:

- preserves transactionality during replication
- synchronizes data to a remote grid and also to databases and/or NoSQL stores if present
- provides a mechanism to do consistent reporting – such as daily reports for partners and suppliers
- makes it easy to implement schema and code upgrades without application downtime.

Two very important characteristics of CloudTran Replication Service are that it does not lose any messages, and that it maintains transactionality during replication. Losing data is a key risk in data management; CloudTran removes that risk. Maintaining end-to-end transactionality reduces costs in initial development, maintenance, and on-going data analysis and correction.

Background

CloudTran enables a data grid architecture that maintains consistency between the grid and persistent stores (e.g., databases) with a fast, write-behind transactional approach.

A common requirement is to provide backup to a remote data center, to support Disaster Recovery (DR) and High Availability (HA) for the complete application. With more than one source of data, consistency and synchronization between the data centers are important issues — but even more important is the requirement to avoid data loss during replication.

The Starting Point

A common approach with NoSQL products is to have multiple data centers all creating transactions. The "data center backup" approach for such products is to reconcile transactional inconsistencies using a distributed voting procedure, which leads to transaction commit times of the order of 100ms. This level of performance is adequate for low-end applications.

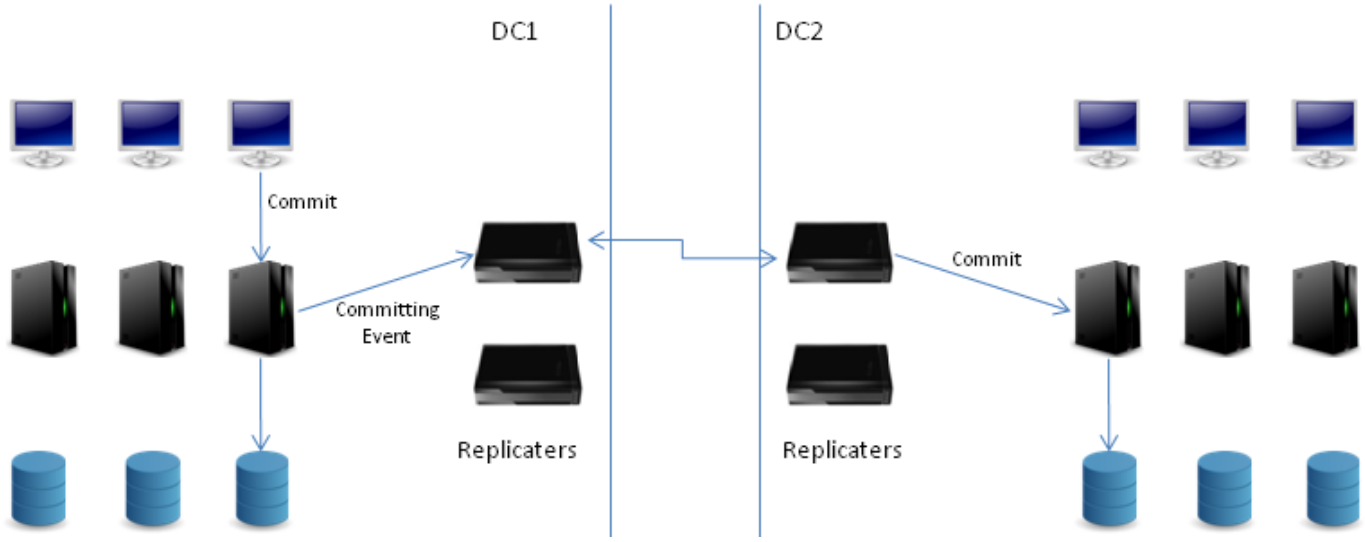
The preferred approach with CloudTran is to use an active-passive configuration: one data center is primary, the other is backup. It may be possible to organize the application into separate domains (where transactions only involve data from one domain), which allows an active-active configuration: one data center is primary for half of the data set and backup for the other half.

The primary/backup approach makes transactions much faster – 10 to 100 times. This clearly impacts the response time to the user. Less obvious is impact on server capacity: reducing the length of time for a transaction reduces the amount of information that must be tracked in the server. The end result is less cost to buy and run servers in the data centers.

With the primary/backup scheme, the starting point for replication is a client committing a transaction in the primary data center. The transaction manager instance that is handling the transaction publishes a "committing" event. CloudTran Replicator Service listens for that event and starts the replication process.

The Replicator Service

CloudTran Replicator Service ("CRS") logically sits beside the clients, grid nodes, and databases, as shown in the following diagram. The physical deployment is likely to run the service on the same hardware as other services.



CRS is configured as a service running on multiple nodes. There is one active node at any given point in time, with the other node(s) being available as backups. The committing event listener in the transaction manager sends the data for the transaction to the active CRS node. This is a single node, which raises the question of whether one node can handle all the traffic. Our experience is that the bandwidth of the LAN adapter determines performance. At 20,000 2Kb transactions per second, this requires 40MBytes/sec, or 320 Mbits/sec, which should be easily handled by a 10Gbit/sec network.

CRS stores the transaction information in a local store. Because this is part of the critical path for commit, SSD is best here – it avoids any delay to the overall transaction time.

Note that this step locks in the transaction for replication – the transaction is committed to persistent store on behalf of the remote site before the transaction returns to the caller. This interlock is very similar to the transaction log, which protects data in the persistence path (i.e., en route to the database) and is written before returning to the caller.

For full HA, the CRS can be configured to write to two SSDs. To enable seamless access by the backup, the SSDs should be dual-ported.

The WAN protocol used to talk between data centers is pluggable; out of the box, TCP/IP and RabbitMQ are supported. It may be possible to use third-party messaging services to implement the WAN protocol, and possibly even provide the persistent store for CRS. However, careful attention will need to be paid to response time and throughput: CRS is designed to meet the most demanding response time and

throughput requirements.

CRS is implemented using grid features (Oracle Coherence in the current version), so compared to a third-party solution it is easier to deploy and operate because it uses the same management and monitoring controls as the main grid.

The Remote Replicator Service

At the remote data center, CRS receives the entries to put into (or remove from) the remote grid. The persistent format of the transaction (e.g., the rows and operations on a database) is not sent to the remote CRS – the format conversion can easily be done at the remote side, roughly doubling the achievable transaction rate on the WAN link.

Normally, the local network is processing transactions and the remote CRS can immediately commit the transaction into the remote grid. The remote CRS does this using the regular transaction management process: it sends the transaction to a transaction manager as though it were a client, so the data is committed to the grid and then to the relevant data stores. The remote CRS sends a "transaction committed remotely" message to the local CRS, which can then release its copy of the transaction on SSD.

Note that the transaction rate in the grid is decoupled from the response time of the WAN link. This means that the backup data center can be anywhere in the world.

In summary, CRS extends transactional guarantees – atomicity, isolation, durability – to the remote data center.

Comparison with Database Replication

There are two major advantages of using Cloudtran Replicator Service over database replication.

The first advantage is due to the use of the grid as the system of record – i.e., it is where we construct and stamp transactions. The commit into the grid is done before returning to the caller. In contrast, the transaction is sent to the database lazily and may be significantly delayed. This means that replication is completed quicker when it is done grid-to-grid.

The other advantage is that when there are multiple data stores involved, reconstructing the in-grid transaction at the remote data center is difficult or impossible. This situation arises when multiple SQL databases are used, or a mixed environment is used with some databases and some NoSQL data stores – an increasingly common situation as enterprises deploy NoSQL stores. By replicating the transaction to the remote grid, it is then easy to reproduce the persistence to the various stores.

Consistent Reporting

To do consistent reporting, CRS has a special "pause" request that holds transactions that were committed after a certain time; it only releases them to the remote CRS when the matching "continue" request is issued. So, for example, to get a daily report, the local CRS can be told to pause transactions committed after midnight. The transactions before midnight will be flushed out of the remote CRS some time later, at which point the report can be run on the remote database.

This approach gives a fully consistent report - there will be no later updates to the data - at a given point in time, with the application still providing a full service to users. It also means items that have been reported on do not need to be updated just to record that they were included in the report – the transaction timestamp will tell that.

During this pause, the local data center is providing the application service.

This technique extends easily to active-active configurations, where each data center is serving

half of the traffic. The administrator issues the pause command to CRSs at both data centers. This pauses the backup in both directions. The reports can be run against the database in each data center – but only for the backup dataset for that data center (because the primary data will be changing as transactions continue executing).

Upgrades

The main problem with upgrades is the length of time it takes to warm up the grid from the persistent stores. The pause at CRS can reduce this time, and so help with some types of upgrades – code upgrades, and data upgrades that can be deduced from the existing data.

The grid to be upgraded is isolated from the live grid during the CRS pause. As no requests are being served, the data in the grid is unchanging. This gives the opportunity to roll out data to its local file system, refreshed with a new code base and possibly data type changes, then reloaded with the original data or slightly changed by an ETL transform.

So, rather than the reload process having a bottleneck at the database – either in retrieving the data or being limited by a single network interface – each node can stream data from its own file system and then send data – so more network interfaces are used to distribute the data. Furthermore, if the grid data is de-normalized (multiple database rows in one cache entry) this avoids time at the database to gather rows.

The more nodes in the grid, the more the time savings with this approach - the time to bounce the grid will reduce at least by a factor of 2, and probably by much more.

Summary

CRS extends CloudTran transactionality from the LAN to the WAN with

- loss-less reliability
- atomicity, isolation and durability
- imperceptible overhead on the speed of transaction processing in the grid
- supporting heterogeneous, distributed persistence

The "pause" facility buffers replication safely, allowing time-related snapshot reporting and the easy installation of software upgrades, all without affecting the availability of transaction processing.

Grid to grid transfers, also allow CloudTran Replicator Service to maintain a speedy switchover of transaction processing from one data center to another, either as a planned move or as an emergency failover, without having to wait for grid warming.