Managing Consistency with Berkeley DB-HA

Data management applications often need to balance conflicting requirements. One of the most common types of requirements is performance. Performance requirements concern such things as the amount of data processed in a given time (throughput) or the amount of time an operation can take (latency). Applications also have other types of requirements, such as availability, durability or consistency guarantees. These other requirements can require tradeoffs with performance requirements.

Availability means that the data is accessible when it is needed, and could be defined in terms such as promptness of service or maximum allowable downtime. Some applications improve availability by distributing copies of their data to multiple nodes, which also can improve latency. Durability means that the data survives a system failure or restart, and this requirement is extended to all nodes in a distributed application.

Consistency guarantees can mean different things depending on whether an application is distributed. For a single-node application, consistency generally means that a transaction can change data only in defined ways. For example, a transaction cannot commit a change that violates a constraint. In a distributed application, consistency also concerns the correctness and completeness of data across the different nodes.

A single-node Berkeley DB (BDB) application has many tuning knobs to control availability, durability or consistency, to the extent that it has full control of what are traditionally referred to as the "ACID" properties. Berkeley DB High Availability (BDB-HA) helps meet availability requirements by replicating copies of the data to additional nodes. This can provide geographical distribution of the data and improve read scalability for applications with a high volume of read operations, thus improving promptness of service. BDB-HA also extends the concept of consistency because it maintains multiple copies of the data.

The purpose of this paper is to explain how to manage consistency requirements for BDB-HA applications. It covers the ways in which BDB-HA can meet consistency requirements and some of the tradeoffs involved. After explaining the relevant BDB-HA features, this paper describes ideas for several typical applications and the ways in which they can use BDB-HA features to meet their consistency requirements.

BDB-HA Features

BDB-HA default replication

BDB-HA provides single-writer multiple-reader replication such that a single node (the electable master) performs all write operations and the other nodes (the replicas) can perform read operations. A

replication group is the set of all of these nodes. If the master node fails, another node in the replication group automatically elects a new master, keeping the replication group available for write operations. Replicas are kept up-to-date via log shipping, meaning that the log records for each master operation are sent to each replica.

A BDB-HA replication group has certain behaviors inherent to distributed systems. Changes committed to the master node take some finite amount of time to propagate to the replicas. If the master node fails, there is a (usually brief) period of time while electing the new master when the replication group is unavailable for write operations.

These behaviors affect read operations on the different nodes in different ways. Read operations on a replica node may be stale if they occur before the most recently committed transactions are available on that node. Read operations on the master node generally reflect the most recently committed transactions. However, there are rare cases where the most recently committed transactions are rolled back on the master node. This can occur after a network device failure separates the master from the replicas, resulting in a change of master. BDB-HA has additional features that provide stronger consistency guarantees with these distributed behaviors.

BDB-HA read-your-writes

The read-your-writes feature increases read consistency for a specific replica read operation. This feature is used by an application that requires specific replica read operations to be consistent with data that was previously updated at the master. It provides a way for a replica read operation to wait for a particular master transaction to arrive at the replica node. The application must send information about the master transaction to the replica node. This feature may increase latency for the replica read operations that use it, but ensures that these replica read operations reflect the most current changes.

BDB-HA master leases

If a BDB-HA application has strong read consistency requirements in the event of a failover, it can use master leases and perform its read operations on the master node. Master leases guarantee authoritative reads on the master node by verifying that a majority of nodes in the replication group have the most up-to-date version of the data. Master leases also exert more control over when a new master can be elected and prevent the rare cases where a master read operation can return data that is later rolled back.

It is important to understand that master lease guarantees only apply to read operations on the master node. Read operations on replica nodes can still return stale data even when master leases are in use. There are some costs associated with the use of master leases. Master leases add processing time to master read operations and increase the amount of time it takes to elect a new master.

Typical BDB-HA Applications

Product catalog

The product catalog contains information about various products available for sale through an on-line shopping site. In addition to listing the products, it might contain detailed information about each product and its price. The product catalog is frequently accessed by read operations, with infrequent write operations to add, update or remove products.

The product catalog has low latency requirements so that customers can conveniently browse product information. It does not have strong read consistency requirements because it is acceptable if a customer browsing the catalog does not see the very latest product catalog changes.

The product catalog can use default BDB-HA replication with reads performed either on the master or replicas. Performing reads on replica nodes can help meet its low latency requirements.

Stock ticker

The stock ticker application stores a list of stock names and prices and is primarily used to display this information to the user. This application is most frequently accessed by read operations to display stock prices. Stock prices change frequently, so there are also a significant number of write operations.

This application has a read consistency requirement to display the most current stock price data. For example, a user might base decisions about stock transactions on this application's output. While low latency is also important, the need for the most current stock data has a higher priority.

The stock ticker application can provide lower latency by performing replica read operations. It should also use read-your-writes on these replica read operations to meet its read consistency requirement. Use of read-your-writes reduces some of the latency benefit from using replica read operations, but this is justified by this application's need for the most current data.

Password authentication/authorization

This application must securely store a list of users and passwords. It is most frequently accessed by read operations when a user is signing into the system. It has occasional write operations to add or remove a user or when a user updates his or her password.

Due to the password security needs, this application has very strong read consistency and durability requirements. For example, consider a case where a user's password is stolen, the user is notified, and the user changes the password. It would be a major security violation if the person who stole the password is able to sign in with the old password due to a stale read. While it is also important to have reasonably low latency for password lookups and updates, the security requirements have a much higher priority.

The password application should use master leases to meet its strong read consistency requirements and perform all reads on the master node. This will guarantee that reads return only up-to-date, durable password values. The additional processing for master leases is justified by this application's security needs.

Summary

The table below summarizes the typical application requirements and the BDB-HA solutions that can be used to meet them.

Table 1: Typical Application Requirements and Solutions

	Product Catalog	**Stock Ticker**	**Password Authentication/ Authorization**
Latency	Low	Low	Medium
Durability	Low	Low	High
Consistency	Low	High	High
Workload	Mostly reads	Reads and writes	Mostly reads
BDB-HA Solution	Default BDB-HA replication, master or replica reads	Default BDB-HA replication, master or replica reads, read-your-writes for replica reads	BDB-HA replication with master leases, master reads only

It is important to consider consistency requirements when designing a BDB-HA application. BDB-HA offers a variety of solutions to help distributed applications meet their consistency requirements.