Best Practices For High Availability

This document details best practices to maximize uptime for applications running on ClustrixDB. This covers a wide range of topics, from environmental requirements to change management procedures, all of which ultimately impact the availability of your application. Many of these are standard best practices or concepts with which you are likely already familiar.

Designing for high availability minimizes risk with the following strategies:

- Careful control of environment and application changes
- Elimination of single points of failure
- Provisioning of sufficient additional resources to allow for continued operation in the event of failure
- Regularly scheduled and tested backup and recovery plan

The following best practices maximize high availability on ClustrixDB:

Use Multiple Operation Environments

An operationally mature organization may have as many as four different environments:

1. Production
2. Disaster Recovery
3. Staging
4. Development

For highest availability, a **production environment** may contain a pair of identical clusters which replicate in a Master-Master configuration. Only one cluster should actively take writes, with the other available for immediate failover (active/passive); alternatively, if there are distinct applications or databases, these may be configured such that one cluster is active for a set of applications, while the other cluster is active for another. Having both clusters take writes (active/active) is possible but presents a number of operational challenges (see Master-Master Replication). Managing cut-over of application load from one cluster to the other can be handled through the use of an external load balancer, or by reconfiguration of application servers.

A **disaster recovery environment** is typically at a geographically distinct location and includes a cluster which replicates from the production environment, along with application servers that are able to provide site functionality (possibly with degraded performance) in case of site-wide failure of the production environment.

A **staging environment** will typically be a (scaled down) facsimile of the production environment, including comparable application servers and datasets. It is used to validate changes to the application software as well as upgrades to the ClustrixDB software. A crucial part of the staging environment is a test automation framework which allows for the exercise of the application and database with a load approximating peak load in the production environment.

A **development environment** allows for more ad hoc development, where the risk of developers interfering with each other's work is inconsequential.

To achieve optimum uptime goals Clustrix highly recommends the use of multiple clusters, which can provide the following:

- Isolation of application development from the production database
- Isolation of analytic workload from time-critical transactional workload
- Disaster recovery/business continuance failover for site-wide outages
- Pre-production validation of application changes and ClustrixDB software upgrades

Highly Available Production Environment

To take full advantage of ClustrixDB's fault tolerant architecture, the following environmental and provisioning requirements should be met:

- Each node's power supply should be connected to a separate power circuit
- Each production cluster should have two backend network switches
- Each backend network switch should be connected to a separate power circuit
- Each node should connect to the production network via two Ethernet ports (interfaces are bonded), to two separate Ethernet switches
- Cluster nodes should be provisioned to accommodate storage and concurrency requirements in the event of node failure. Zones should be configured to isolate faults.

Change Management

The vast majority of software failures arise from changes in application behavior, whether due to a bug in the application itself, or exposure of a bug in an underlying layer such as the database. Accordingly, the best practice is to first thoroughly validate such changes in non-production environment(s), and then carefully roll out into production, with roll back plans in place to undo changes in case of surprises.

Application Change Management

The existence of development and staging environments allows an organization to roll out new applications and application changes in a safe manner.

- New application development is restricted to the development environment where malformed queries and other abuse of the database can only impact other development work.
- Once the code has stabilized, it can be incorporated into the staging environment, where it is tested at load, and in conjunction with the existing production workload. (The ClustrixGUI Administration UI can provide helpful comparative information.)
- After validating proper functionality and performance in the staging environment, the changes can be rolled into production.

ClustrixDB Upgrade Best Practices

Upgrading the software on your ClustrixDB cluster can be treated in much the same way as application code changes. While ClustrixDB software releases are thoroughly tested in-house, changes such as new compiler optimizations can have an unanticipated impact on customer workloads; validation of the new release on a staging cluster running a simulated workload allows discovery of such issues prior to production rollout.

In an ideal operational environment, customers can participate in the beta program, obtaining early release candidates for use in their development cluster. Once a qualified release is available, they can test in their staging environment to eliminate problems evident under heavy workloads. When upgrading production, if a pair of clusters is available, one cluster should be upgraded first; the cluster can then undergo a full day or week of load before upgrading the second cluster, providing for failback to the second cluster running the prior, stable release.

Application Configuration for High Availability

While eliminating single points of failure in your application stack is beyond the scope of this document, the following guidelines pertain specifically to how your application interacts with the ClustrixDB database layer:

- Application servers should connect through a load balancer
- Application software should have retry logic for database transaction or connection failures
    - Initial retry timing should be aggressive as cluster typically handles component failures in a few seconds
    - Retry frequency and iterations should also accommodate possible longer duration recovery

Backup and Recovery

ClustrixDB parallel fast backup provides for rapid backup which allows for near-constant backup time as your cluster grows, as the work is split across the nodes. When planning your backup strategy, consider the following:

- The FTP server typically becomes the bottleneck during backup
    - It should be provisioned with 10Gb or several bonded 1Gb Ethernet interfaces
    - It should have fast enough disk I/O to synchronize parallel writes from all nodes in the cluster
    - It should have sufficient disk capacity to handle two or more full backups of the cluster (1/2 of total used capacity, as ClustrixDB backs up only one replica from each slice)
- Off-site archival of backups
- Regular tests of the ability to restore from backup
    - ClustrixDB parallel restore supports restoring subsets of a backup, as well as restoring to alternate locations (different database names), allowing for regular "spot checks" where a sufficiently large cluster for full restore is unavailable
- (Off-site) archival of binlogs using the repclient utility to support point-in-time restoration
    - This also requires the use of repserver to replay binlogs
    - The full process for point in time restoration is described in Point in Time Restoration