
GUIDE TO CHOOSE BEST REPLICATION STRATEGY FOR YOUR APPLICATION

Introduction	2
Benefits	2
Ease of use	2
Performance and Consistency	2
Scalability	3
Feature Summary	3
Replication Granularity	3
Update Propagation Modes	3
High Availability	4
Load Balancing	4
Recovery	4
Data Consistency	4
Platform	5
Deployment Options	5
Active – Standby Replication	5
Active - Active Replication	6
Load Balancing Cluster for Read	7
Load Balancing Cluster for Read/Write	8
Load balancing cluster with Distributor	9
Load balancing cluster with distributed data	10

Introduction

Application that provide service 24/7 cannot tolerate service downtime which impacts the revenue and credibility with customers. Two important aspects for highly available systems are down time and data consistency during fail over.

Web and real time applications are read intensive and employing cluster shall reduce load on the backend databases resulting in high throughput.

High Availability and Load balancing is achieved by keeping multiple copies of same data in multiple database instances through data replication. Though these replication strategies are well known techniques and are supported by leading commercial databases for long time, open source vendors like MySQL and Postgres are catching up with this in recent years.

Data stored in CSQL main memory database shall be replicated using CSQL Replication product, which provides real time update anywhere 'N' way replication model which suites for both high availability and load balancing.

CSQL Replication uses transaction log based replication scheme using TCP/IP, to deliver high performance and consistent data replication. These transactions logs are applied to peer sites either synchronously or asynchronously.

Group of CSQL instances taking part in the replication process is referred as 'quorum' and each instance is referred as 'site' in this document. Site where transaction originates are called source sites and sites where these transaction logs are propagated are called destination site.

In case of synchronous mode, source site transaction commit return only after the transaction is committed at source instance and other destination instances in the quorum, whereas in case of asynchronous mode, source site transaction commit return immediately after the transaction is committed at the source site. Replication process takes care of applying these transactions at destination sites in the quorum.

CSQL Replication provides synchronous and asynchronous transaction propagation modes enabling applications to choose between high throughput and high consistency.

Benefits

Ease of use

CSQL Replication is simple to use and requires minimal setup to get the data replicated. Requires no changes to application and minimal administration as CSQL has inbuilt self-healing mechanisms to tolerate faults of individual sites in the quorum.

Performance and Consistency

Load balancing database operations on multiple servers improves throughput by multifold. Most of the real time applications are read intensive and by employing cluster with multiple CSQL instances, load can be distributed among all of them to improve the overall application throughput.

Transactions originating at source site can be applied on destination sites either synchronously or asynchronously. The former provides high throughput and the later provides high data consistency. In case of asynchronous mode, the data will be available at the destination sites after few seconds.

Scalability

CSQL supports clusters up to 12 updateable CSQL instances, each propagating updates either in synchronous and asynchronous modes to other sites. There is no limit on the total number of read only CSQL instances in the quorum.

Feature Summary

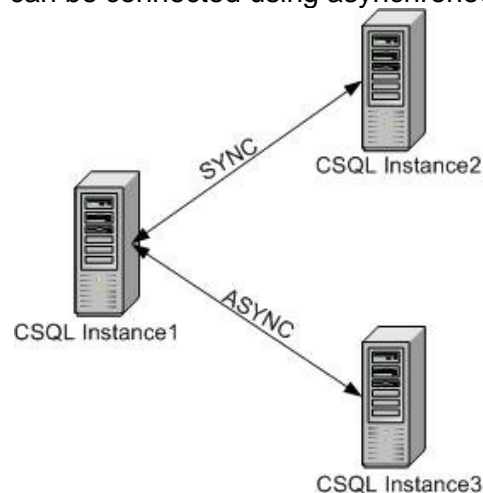
Replication Granularity

CSQL supports replication at multiple granular levels. Database level replication, replicates all the tables in the database whereas table level replication allows applications to choose what tables to replicate in the quorum. It also supports partial table replication in which only subset of the records are replicated between two sites. Refer "Load Balancing Cluster with distributed data" for more information.

Transaction Propagation Modes

Transactions are applied at destination sites in two modes namely, Synchronous and Asynchronous. The mode between two replicating sites need not be same. For example for two sites (site1 and site2 in replication quorum), site1 can propagate transactions to site2 in synchronous mode and site2 can propagate transactions to site1 in asynchronous mode or not to propagate transactions to site1.

In case of multi site cluster, application can choose the transaction propagation mode between each and every site in the quorum. Lets say you have three sites namely site1, site2 and site3 in the quorum. Site1 and site2 can be connected using synchronous mode and site1 and site3 can be connected using asynchronous mode.



High Availability

CSQL Provides sub second fail over as all transactions are applied at all the sites in the quorum. CSQL Replication provides continuous availability during planned maintenance operations such as software or hardware upgrade with no data loss.

Load Balancing

CSQL Replication can be deployed as load balancing cluster with many sites to improve the throughput of the application. Transaction logs can be propagated by mediator components called 'distributors' whose job is to propagate transaction on behalf of the source site to the other entire destination site, reducing the load on the source site.

CSQL Replication supports intelligent load balancing techniques where subset of records is replicated rather than the complete table at every site. This mode is called as distributed replication, where each site contains records belonging to their zone and a centralized server, which contains the complete table for all zones. For more information refer "Load Balancing Cluster with distributed data" deployment option.

Recovery

When a site go down for any reason (planned maintenance, crash, or disaster), all the other sites in the quorum retains all the transactions at their local site until the site which went down resumes its operation. When the site comes back, it receives transaction updates from all the sites (transactions which happened when this site was down) and gets sync with all the other sites in quorum. This recovery is automatic and does not require any manual intervention.

When a site go down connected with its destination site using synchronous mode transaction propagation, the mode is automatically switched to asynchronous thereby avoiding transactions to fail. (Otherwise all transaction fails due to transaction execution failure at destination site, which is down). This makes sure that no transaction fail because of site failure. When the down site resumes its operations, the mode shall be switched back to synchronous.

Data Consistency

CSQL Replication employs transaction consistent replication, if transaction fails at any destination site connected using synchronous propagation mode fails, the transactions fails at the source site also. This ensures that two parallel conflicting transactions (For example deducting balance from account) running in two different sites to fail and ensure consistency of data across sites in the quorum.

For asynchronous mode propagation, transaction conflicts at destination site are detected and logged into file so that the administrator can resolve conflicts. Conflict resolution could be additive, subtractive or replacing based on the data on which conflict occurred. For Example in case of conflicting deposit, conflict resolution is additive whereas for withdrawal, it is subtractive.

CSQL Replication provides tools to check for consistency between sites connected in asynchronous mode to assist application for maintaining data consistency across quorum.

Platform Supported

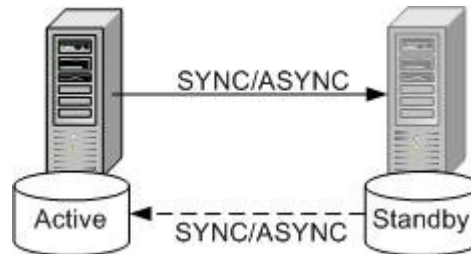
Linux – x86

Linux – x86_64 and Solaris are under development

Deployment Options

Active – Standby Replication

In active –standby configuration, application performs database operations only on active instance. If the active CSQL instance goes down for any reason, then application shall access the standby CSQL instance for performing database operations. This mechanism is called as fail over. CSQL keeps data at standby instance same as active database instance so that application can fail over to standby instance in sub seconds.

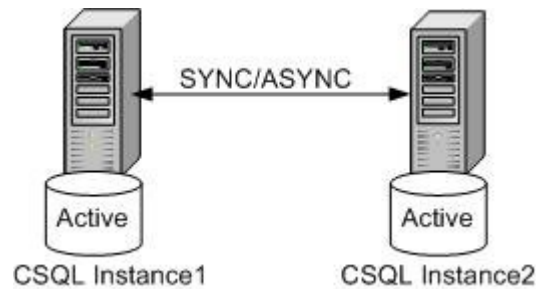


All the database operations such as INSERT, UPDATE, and DELETE on tables at active instance are applied transparently on standby database tables. CSQL can be configured either synchronously or asynchronously to apply these database operations on standby instance. For real time application, which require high throughput for modification operations, it makes sense to use asynchronous configuration. In asynchronous mode, execution returns immediately after it is performed at the active instance providing high throughput. Later CSQL replicator process makes sure that database operations at active instance are applied at standby instance without affecting the transactional behavior.

When the active sites resumes its operations, application can choose between either of the instances to make it active or standby based on its requirement.

Active - Active Replication

In active –active configuration, application performs database operations on both the active instances. If the active CSQL instance goes down for any reason, then application accesses the other CSQL instance for performing database operations. CSQL keeps data at both active instances same to provide sub seconds fail over.

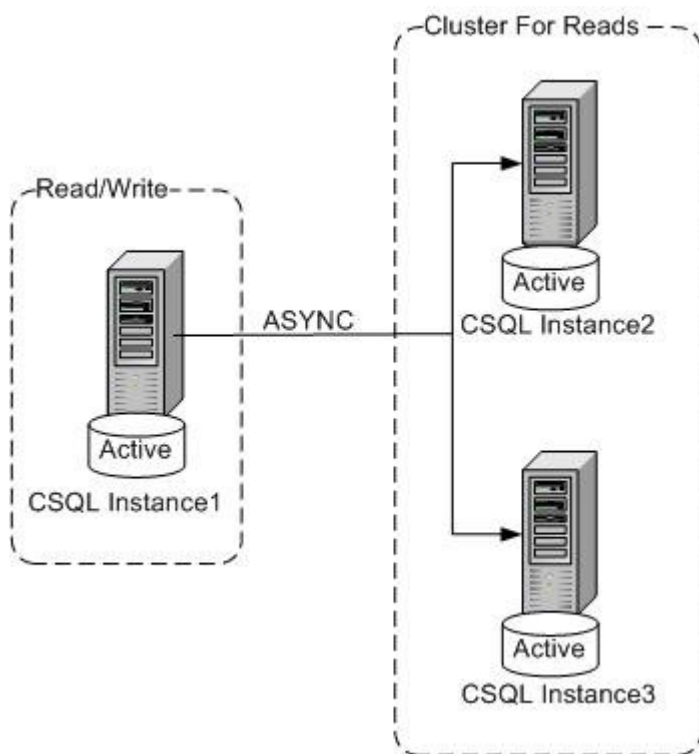


All the database operations such as INSERT, UPDATE, and DELETE on tables at CSQL instance1 are applied transparently on database tables of CSQL instance2.

If any CSQL instance goes down for any reason, the other instance preserves all the database operations that need to be applied (for updates which happen when the peer instance is down) and applies it when the instance which went down resume its operation.

If CSQL instances are connected using synchronous mode, when one instance go down for some reason, it automatically changes to asynchronous configuration to preserve update logs for the other instance. Once this transition happens, application can reconnect and resume operations on it. The configuration can be changed to synchronous after the instance, which went down resumes its operation.

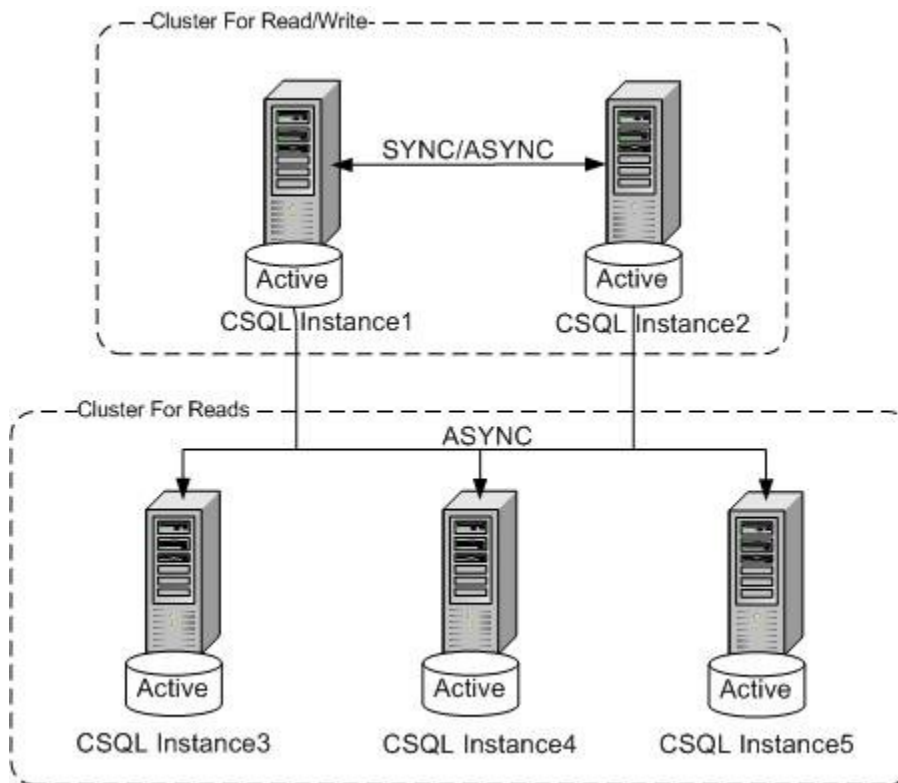
Load Balancing Cluster for Read



Most of the real time applications and high traffic web sites are read intensive. These applications shall distribute the query load to multiple instances to improve the throughput. In the above configuration, one instance is used for performing both read and write operations and other instances (CSQL Instance2 and CSQL Instance3) are used to distribute the load for read operations.

This deployment model is useful if your application has real time transaction requirement as well as near real time reporting requirements for different components. For example in SMS gateway application, the component which initiates the SMS sending process and stores the information in database requires real time performance and another component (reporting) which provides statistics and reports on the SMS application requires near real time performance. The former requires real time performance as it directly affects the total number of SMS sent / sec and the later require near real time performance where reporting web interface should display results within few seconds.

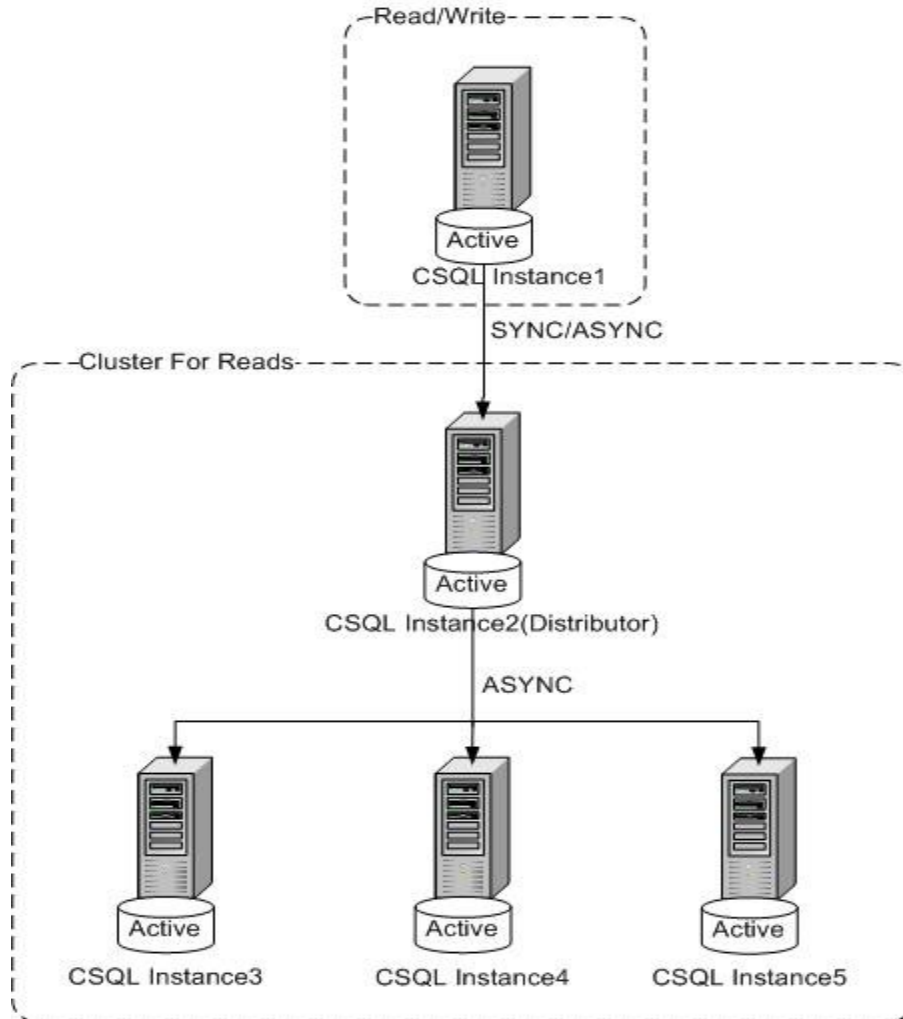
Load Balancing Cluster for Read/Write



This deployment option suits application which has mixed read /write load where performance improvement is required for both read and read/write transactions. This is achieved by employing two clusters; one, which allows both the operations and another, which allows, only read operations. Read/Write cluster can be configured either in synchronous or asynchronous mode based on the application requirement for freshness of the data.

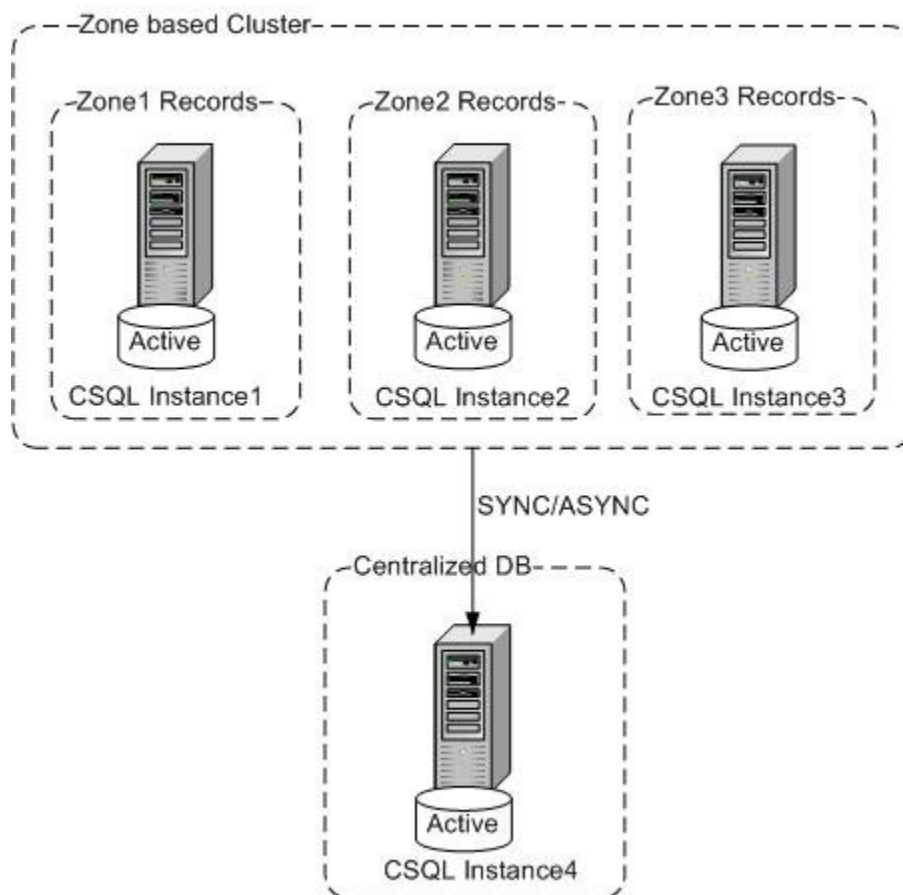
For example online charging (pre-paid billing) application does read/write transaction to retrieve and update the credit balance during the call for every minute (based on the pulse rate). This requires real time performance as the charging is done online during the call itself. These types of applications can employ the above deployment model for increasing their throughput.

Load balancing cluster with Distributor



This deployment model is well suited for application, which are Write intensive. Transaction logs when propagated to other sites, (CSQL Instance3 to CSQL Instance5) in quorum takes some CPU time, which may affect the throughput of the CSQL instance1 when there are large number of destination instances in the “Read cluster”. To overcome this and provide high throughput for application accessing CSQL Instance1, the application can use another instance (called Distributor), CSQL Instance2 to propagate transaction to all the sites in the “Read Cluster” on behalf of the source site. Multiple distributors can also be employed to scale the number of CSQL instances in the “Read cluster”.

Load balancing cluster with distributed data



This deployment model is well suited for application, which are distributed in nature but requires centralized management like banks and Visitor location Register (Telecom IN application). Instead of replicating the complete centralized database to all instances, application shall replicate data that belong to that particular zone or region it serves.

Telecom service providers divide the entire region they service into multiple zones (For example, state-wise) and keep records pertaining to that zone in the local data center rather than the remote centralized data center. This improves the performance of the application by multifold as it does not go over WAN for each database access.

In the above diagram, CSQL Instance1 stores only records for zone1 and replicates all the transactions that affect zone1 records to centralized database either in synchronous and asynchronous mode. CSQL Instance2 stores only records for zone2 and replicates all the transactions that affect zone2 records to centralized database.

The centralized database contains records up to date data for all the zones. This makes data management lot easier.

DBMS available in the market provide poor support for this configuration and puts more burdens on the application to replicate and manage data.

Summary

- Easy to use
- High performance and throughput
- Sub-second fail over
- Synchronous and Asynchronous transaction propagation
- Automatic recovery
- Load balancing with cluster