System availability is determined by the level of redundancy designed and built into each part of a software system. The redundancy used to increase the fault-tolerance of a system is distinct from the redundancy that enables horizontal scalability. In some cases, such as a load balanced web server farm, the processing power of a system is increased by adding more servers to the server farm (horizontal scalability) while redundancy used for fault-tolerance is provided by adding *extra* servers to the farm. In this case, the same load-balanced server farm provides both scalability and fault-tolerance at the same time.

Data storage in a typical distributed computing system is provided by standard database clusters. These types of clusters work well within a single location, but generally struggle to replicate data between two geographically separate locations. However, DGP has its own omnidirectional data replication that has been specifically designed to work between multiple locations, and which can also add a degree of horizontal storage scalability using database shards.

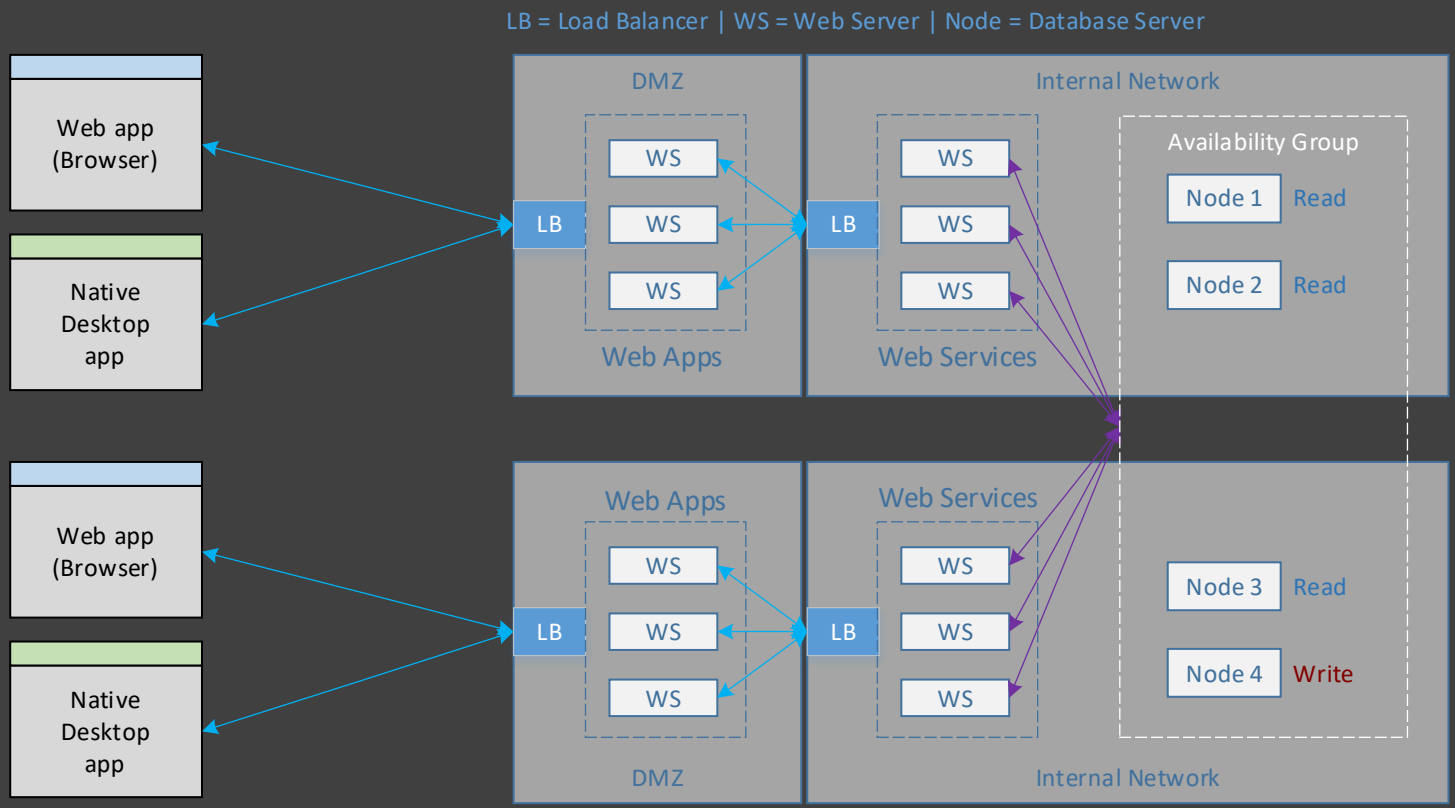
CAP Theorem

CAP Theorem is a mathematical proof used for distributed software systems which states that only 2 of 3 desired capabilities can be provided by a distributed system at one time: strong data Consistency, high system Availability, or Partition tolerance (CAP).

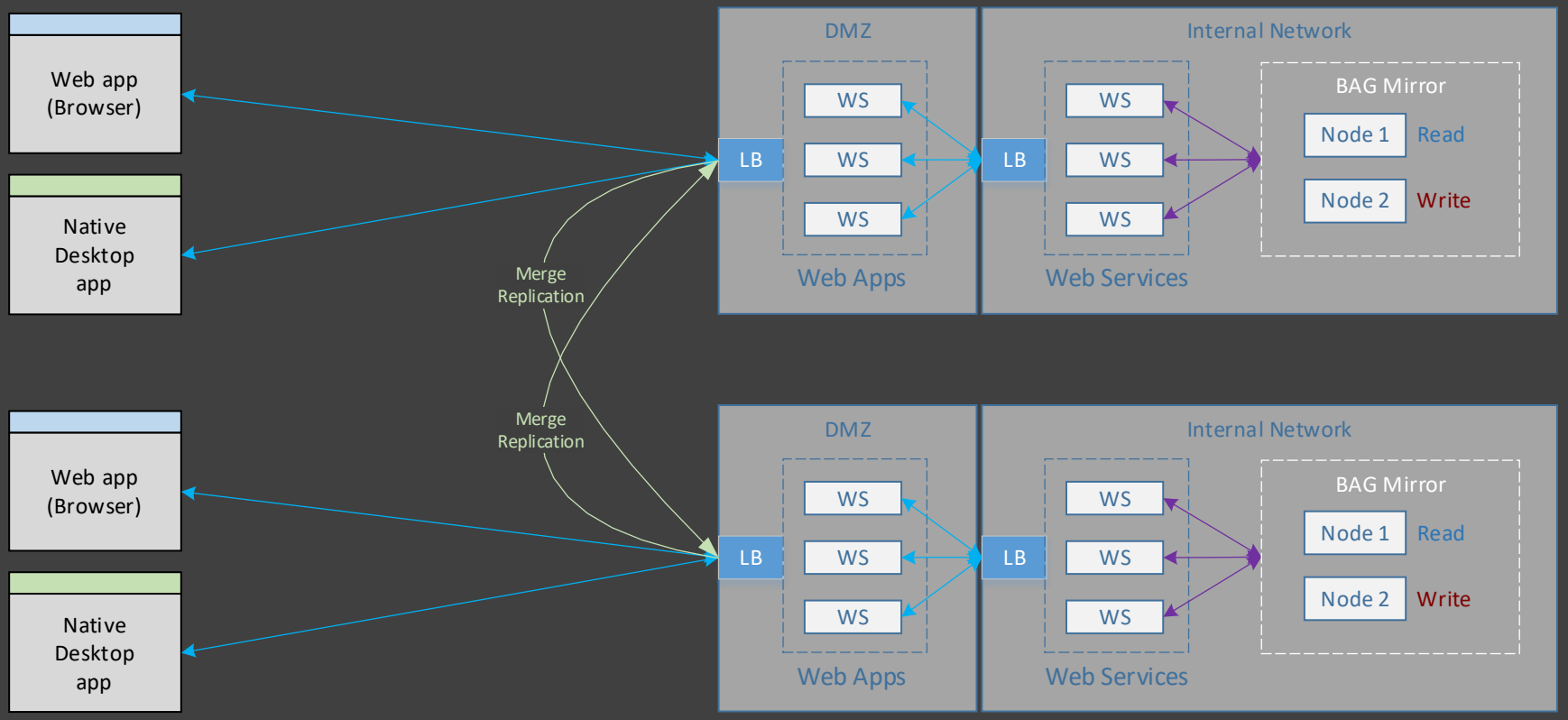
DGP's omnidirectional data replication is an AP system that provides high availability and partition tolerance at the expense of eventual data consistency for the data replicated between locations. However, if users only connect to and use a single primary location at a time, this results in strong sequential read-after-write data consistency. Data created in the primary is replicated to the backup location in the background. The backup location is active and writable, but is not used unless the primary goes offline.

The main advantage of a distributed grid compared to a standard distributed system is the ability to have multiple separate locations of the system be active and writable at the same time. Having multiple active writable locations running simultaneously eliminates the need for Business Continuity or Disaster Recovery mechanisms. Users that experience any problems when connecting to the primary location simply switch to the backup location, with no interruption in service and little or no loss of work. As long as one of the locations of a system remains online, the entire system is still effectively online.

DGP provides two ways to achieve 100% uptime. The first is to build two datacenters with a high-bandwidth, low latency WAN connection between them, and then configure a SQL Server Availability Group that spans across both datacenters as if it were a single LAN (but with redundant listeners in each datacenter). This option works very well, and has been used by a payment gateway company in production since 2014 with excellent results. 100% system uptime requires all maintenance to be performed while the system remains in use, which is a bit tricky for the nodes of the availability group, but has proven to be perfectly viable in practice.



The second option is DGP's omnidirectional merge replication. The diagram below shows DGP's merge replication between two large-scale locations. The replication processes are run in the DMZ for added security. In this example, SQL Server availability groups are used to provide fault tolerance within each location in addition to the redundancy of the data replication between the locations. Even though a large-scale system is shown in this diagram, the replication processes work exactly the same for small-scale systems that run all of the tiers of DGP on a single computer in each location. To achieve strong data consistency for DGP's AP system (from CAP theorem), one location is designated as the primary for all reads and writes, while the other location is used as a live backup.



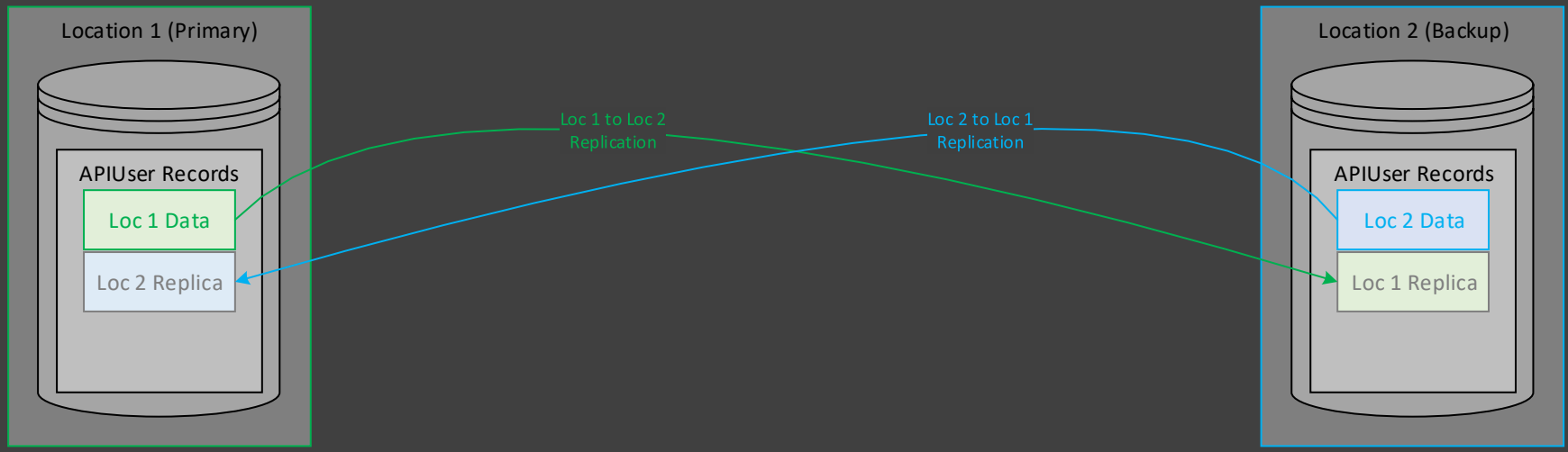
DGP Replication

DGP's omnidirectional merge replication incorporates a number of core concepts:

1. Each record in a replica table is considered to be immutable. In practice, this rule is relaxed for a single column storing the state of the record, which can be edited. The result is that no data is ever lost or overwritten – all versions of all records are preserved in each replica table. This also provides a full audit trail of all changes made to the data in a system.
2. Each replica table acts as a collection of immutable records that were created in a specific location.
3. Each replica table also acts as a union of the collections of records created in all of the other locations, incrementally merged together by the replication processes running constantly in the background in order to achieve eventual consistency.
4. The most important rule of the replication process is that every record from a source, regardless of its state value, must be duplicated in each of the destination locations, with no exceptions. The data verification processes depend on this.
5. The Last Write Wins (LWW) logic is used to flag which of the record versions is currently active. Any errors in this process can be corrected afterward as needed, since all versions of the database records are preserved.
6. An automated data verification process which scans replicated data in each destination for errors (duplicates, gaps, inconsistencies, etc.) is also run constantly in the background, repeatedly crawling through all the data looking for any issues.

Under normal conditions, the limitations of eventual consistency of the data synchronized between locations will only (potentially) be experienced when a user switches connections from one location to another, such as when a problem occurs with the primary location A, and the user then switches to backup location B to continue their work. If replication is interrupted or falls behind its workload, some data from location A could be missing from location B.

The solution for users that experience this problem is to redo the work that did not have a chance to be replicated, and then continue where they left off – even though this deliberately creates duplicate records of the work that is redone. When the problems with location A are resolved, merge replication will correct those duplicates by applying the "Last Write Wins" (LWW) rule. The previous versions of the work are merged from location A into location B with a state of "edited", leaving the newer records as "active". All versions of all records are still preserved (even for duplicates). The only challenge is to ensure that the latest records are flagged as active at any given point in time.



When using the DGP Replica or FileShard schema types, 1) each database is duplicated in multiple geographically distributed locations, and 2) each table in the database is treated as an append-only collection of immutable records. The asynchronous replication process runs constantly in the background to quickly merge the records created at each location into the corresponding tables of the other locations, creating an eventually consistent union of the records from all the locations (similar in concept to CRDT's). Replication from each source to each destination is configured and managed independently.

Each location functions autonomously, and does not attempt to verify the consistency of the data between other locations prior to writing new immutable records into its own local database. Instead, merge conflicts are corrected as they are detected during the replication and verification processes, and the state value of each record is adjusted accordingly.

In terms of CAP theorem, DGP is an AP system with eventual consistency of the data. However, using one location as the Primary for all the work of users (reads and writes) results in a single writable node, which provides strong data consistency. DGP replication then becomes nothing more than a very fast, incremental backup process running continuously in the background.

The other important aspect of the merge replication process is that when dealing with any merge conflicts, the last write always "wins" (LWW). The record with the latest time value will be flagged as "active" (using the editable rec_state field), taking the place of any older records that were created before it. The LWW convention requires that the servers in each location use UTC times and are

synchronized to a common network clock. For most LOB systems, the standard synchronization to a net time server will be sufficient. However, Windows Server 2016 and later can be configured to be synchronized within 1 MS of a GPS equipped source clock [Accurate Time for Windows Server 2016] if that level of clock synchronization ever becomes necessary.

The replication process has some built-in safeguards but cannot be considered to be perfectly reliable all by itself, since it will be interrupted periodically as the result of server problems, network problems, etc. That is why a separate verification process is used to constantly crawl through the data of the destination tables, comparing them to the matching source tables, searching for (and in most cases fixing) any errors in replication. One benefit of this approach is that it is able to periodically prove the correct synchronization of the replicated data in each location.

DGP System Maintenance

The objective for redundant DGP systems is to provide 100% uptime of the system as a whole, which means all patching, code deployments, upgrades, etc. must be performed while the system as a whole remains in constant use. This is accomplished by performing maintenance on each location of a DGP system sequentially, while users are connected to the other location(s) so that the system as a whole remains online. In this sense, the backup location taken offline effectively has its own private maintenance window, while all users are connected to and using the primary location.

In addition, all locations can be kept online during maintenance if it is possible to perform maintenance on individual servers in the stateless web server farms and database clusters, etc. This is easily done for the web servers, and can also be done for the nodes in SQL Server Availability Groups, but other types of storage clusters generally do not have this capability and would require a period of planned downtime (maintenance window).

Refer to the Maintenance documentation for more detailed information.