Availability

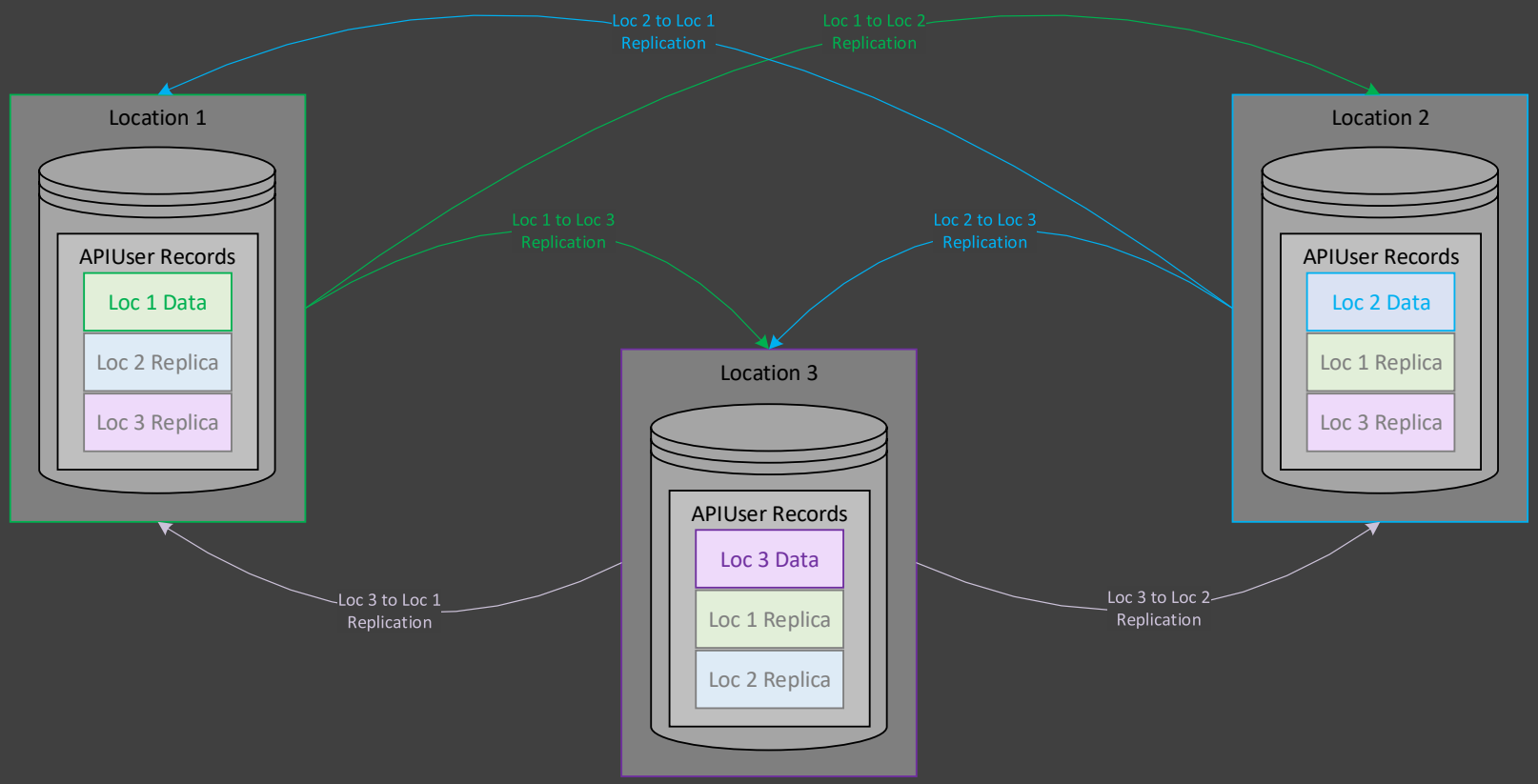
100% availability for a software system means that it is up and running, 7 x 24 x 365, with no planned or unplanned downtime (maintenance is done while the system remains in use). Vendors of various products imply that they have this level of availability, but are not able to truly deliver 100% system uptime. The only real examples that do are the large Internet-scale systems from Amazon, Google, Facebook, etc. They are able to deliver true 100% availability due to their expertise in large-scale distributed grid systems. However, their grid systems are proprietary, closely guarded secrets that are not available to other businesses.

DGP is able to deliver true 100% availability just like the large Internet-scale systems due to its immutable append-only storage and multi-master merge replication. Unlike the conventional redundant systems, each DGP location is autonomous and writeable at the same time, and depends on the merge replication running constantly in the background to synchronize the data between locations in real time. As long as one location for a system is online, then the system as a whole is online. The synchronization of data between locations will automatically repair itself once downtime or connectivity problems have been corrected. It is important that the mechanisms that enable redundancy must not interfere with the mechanisms used for scalability in each location of a system.

DGP servers in each location are divided into processing nodes (web servers) and storage nodes (database servers). Each type of node has different mechanisms for redundancy, failover and recovery. For small systems, none of the nodes in a location have any redundancy, and failover between locations is the only option.

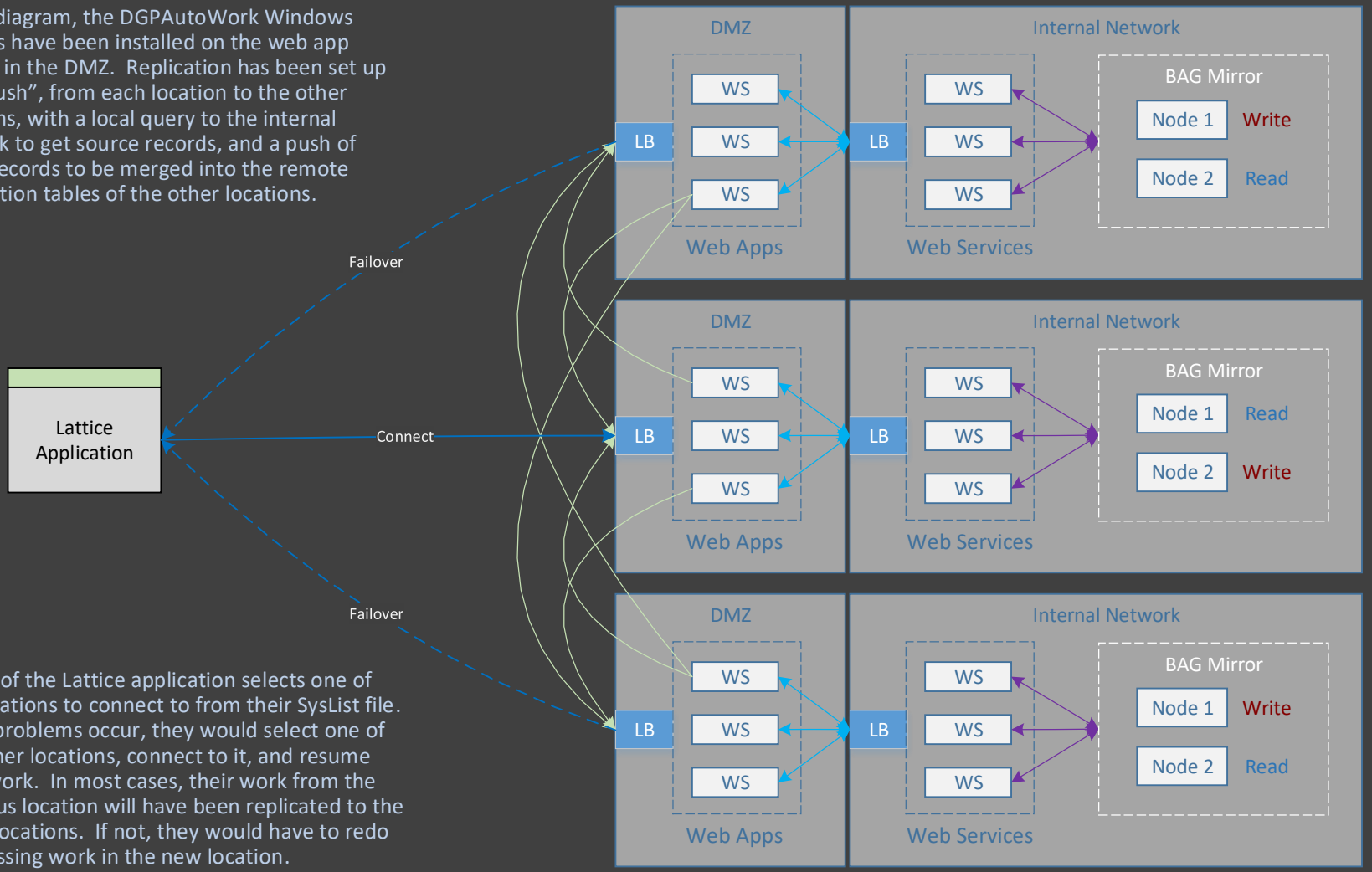
In larger systems, redundancy and fault-tolerance for the processing nodes is added to a location by introducing load-balanced web server farms. The load balancer handles the failover and recovery of individual processing nodes in the server farm. Adding some extra servers to the farm provides fault-tolerance, so if individual servers in the farm are down, the system as a whole is not affected. The load-balanced web server farm is the best and simplest physical topology that provides excellent redundancy, failover and recovery, and is one of the main reasons that so much of the logic and work in a DGP system is consolidated in this tier.

At a small scale, storage nodes in the system rely exclusively on DGP's merge replication between locations, plus failover between locations to enable 100% availability for the system. As locations grow larger, storage server redundancy can be added in a variety of ways. For SQL Server, Basic Availability Groups are one good choice that provides database mirroring between servers.



In terms of CAP theorem (Consistency, Availability, Partition tolerance), DGP is an "AP" system that focuses on 100% availability and automatic recovery from any server or location downtime, while providing eventual consistency of the data merged between locations. Users only connect to and work with a single location at a time and therefore are using a single database at a time, so they experience sequential read-after-write consistency for their own work. The effects of eventual consistency are not noticeable to them as they use the single location. Under normal circumstances, data is synchronized between locations within a few seconds.

In this diagram, the DGPAutoWork Windows services have been installed on the web app servers in the DMZ. Replication has been set up as a "push", from each location to the other locations, with a local query to the internal network to get source records, and a push of those records to be merged into the remote destination tables of the other locations.



A user of the Lattice application selects one of the locations to connect to from their SysList file. If any problems occur, they would select one of the other locations, connect to it, and resume their work. In most cases, their work from the previous location will have been replicated to the other locations. If not, they would have to redo the missing work in the new location.

Availability Verification

1. *Availability is determined by redundant resources, the failover mechanism between redundant nodes, and the recovery of the system from the effects of the failover. The N + 1 fault tolerance of load-balanced web server farms have been used successfully for decades, and do not need to prove their capabilities. The processing node availability of the system as a whole is maximized by consolidating as much logic as possible in the web services of the middle tier. Failover and recovery can be tested using the native capabilities of the load balancers during the monthly maintenance and patching of the web servers.*
2. *DGP has two levels of data redundancy: 1) DGP merge replication between locations, and 2) optional redundancy within locations. Small systems will only have merge replication, while larger systems will add the optional redundancy within locations (load-balanced web server farms, SQL Server availability groups, etc.). Failover and recovery can be tested using the native capabilities of the database server clusters during monthly maintenance and patching.*
3. *DGP merge replication can be tested on a single computer using the TestDB databases and various test files in the API Tester. SSMS is used to manually verify the results of replication by running queries for the specific source database name in a destination table.*
4. *DGP merge replication can be tested on multiple computers within a specific environment (Dev, Test, and QA) once the AutoWork records to manage replication in each direction have been configured. The AutoWork Tester can then run iterations of the merge replication between locations. An SSMS query for the specific source database name in a destination table can then verify the results of replication process iterations.*
5. *The CountCheck detection of gaps in destination data is tested by deliberately deleting some destination test records (after replication has occurred), and is corrected by running the Verify replication process between the source and destination tables.*
6. *When CountCheck finds fewer source records than the destination, it either indicates duplicate records in the destination, which can be detected by the DupeCheck process, or missing source records, which can be fixed by reverse replication using the Verify process from a destination table that has the missing source records back to the original source table.*