Replication

The process of creating and managing duplicate versions of a database. Replication not only copies a database but also synchronizes a set of replicas so that changes made to one replica are reflected in all the others. The beauty of replication is that it enables many users to work with their own local copy of a database but have the database updated as if they were working on a single, centralized database. For database applications where users are geographically widely distributed, replication is often the most efficient method of database access.

Source: www.webopedia.com

Replication is the process of sharing information so as to ensure consistency between redundant resources, such as software or hardware components, to improve reliability, fault-tolerance, or accessibility. It could be data replication if the same data is stored on multiple storage devices, or computation replication if the same computing task is executed many times. A computational task is typically replicated in space, i.e. executed on separate devices, or it could be replicated in time, if it is executed repeatedly on a single device.

The access to a replicated entity is typically uniform with access to a single, non-replicated entity. The replication itself should be transparent to an external user. Also, in a failure scenario, a failover of replicas is hidden as much as possible.

It is common to talk about active and passive replication in systems that replicate data or services. Active replication is performed by processing the same request at every replica. In passive replication, each single request is processed on a single replica and then its state is transferred to the other replicas. If at any time one master replica is designated to process all the requests, then we are talking about the primary-backup scheme (master-slave scheme) predominant in high-availability clusters. On the other side, if any replica processes a request and then distributes a new state, then this is a multi-primary scheme (called multi-master in the database field). In the multi-primary scheme, some form of distributed concurrency control must be used, such as distributed lock manager.

Load balancing is different from task replication, since it distributes a load of different (not the same) computations across machines, and allows a single computation to be dropped in case of failure. Load balancing, however, sometimes uses data replication (esp. multi-master) internally, to distribute its data among machines.

Backup is different from replication, since it saves a copy of data unchanged for a long period of time. Replicas on the other hand are frequently updated and quickly lose any historical state.

Replication in distributed systems

Replication is one of the oldest and most important topics in the overall area of distributed systems. Whether one replicates data or computation, the objective is to have some group of processes that handle incoming events. If we replicate data, these processes are passive and operate only to maintain the stored data, reply to read requests, and apply updates. When we replicate computation, the usual goal is to provide fault-tolerance. For example, a replicated service might be used to control

a telephone switch, with the objective of ensuring that even if the primary controller fails, the backup can take over its functions. But the underlying needs are the same in both cases: by ensuring that the replicas see the same events in equivalent orders, they stay in consistent states and hence any replica can respond to queries.

Database replication

Database replication can be used on many database management systems, usually with a master/slave relationship between the original and the copies. The master logs the updates, which then ripple through to the slaves. The slave outputs a message stating that it has received the update successfully, thus allowing the sending (and potentially re-sending until successfully applied) of subsequent updates.

Multi-master replication, where updates can be submitted to any database node, and then ripple through to other servers, is often desired, but introduces substantially increased costs and complexity which may make it impractical in some situations. The most common challenge that exists in multi-master replication is transactional conflict prevention or resolution. Most synchronous or eager replication solutions do conflict prevention, while asynchronous solutions have to do conflict resolution. For instance, if a record is changed on two nodes simultaneously, an eager replication system would detect the conflict before confirming the commit and abort one of the transactions. A lazy replication system would allow both transactions to commit and run a conflict resolution during resynchronization. The resolution of such a conflict may be based on a timestamp of the transaction, on the hierarchy of the origin nodes or on much more complex logic, which decides consistently on all nodes.

Database replication becomes difficult when it scales up. Usually, the scale up goes with two dimensions, horizontal and vertical: horizontal scale up has more data replicas, vertical scale up has data replicas located further away in distance. Problems raised by horizontal scale up can be alleviated by a multi-layer multi-view access protocol. Vertical scale up is running into less trouble since internet reliability and performance are improving.

Source: https://en.wikipedia.org