Databases Form Core of Reliable Network Design

Though often given short shrift, sound real-time databases are the foundation of high-availability networks, but which architecture?

By Tomas Ulin

The shift to an all-Internet Protocol (IP) network, and IP-based services in general, demands communications systems that achieve traditional telecommunication-grade levels of service continuity and responsiveness, while meeting ever-shorter development cycles. What’s more, communications infrastructure is becoming more complex as demand rises for converging voice, data and multimedia services. Adding to the mix is the fact that information, once largely static in nature, is becoming more dynamic, with the increasing use of personalized subscriber profiles and the growth of content-rich, prepaid and on-demand services.

All of this drives up the sheer quantity of data that must be stored and accessed quickly, reliably and accurately each and every time a service request is made.

In response to these pressures, system vendors are adopting an approach based on open, standard building blocks to shorten time-to-market and reduce the costs of developing and maintaining communications infrastructure. For instance, the Service Availability Forum (www.saforum.org) is addressing this issue by creating, and promoting the industry-wide adoption of, open-standard specifications that facilitate the development and deployment of highly available infrastructure and applications.

High-speed, real-time databases are a central building block in this equipment and provide the data-handling performance required by next-generation networks. These databases also ensure that the challenges of service availability and reliability are met.

Such databases have traditionally been based on proprietary solutions, entailing high development costs and longer time-to-market. Today, infrastructure vendors can benefit from off-the-shelf real-time database products for delivering competitive communications platforms faster and at lower cost.

Storing data in fixed memory is not reliable and may leave the database open to system crashes.

Traditional approaches

Conventional disk-based databases can be used with disk-replication techniques to ensure data availability. To improve performance, stored procedures can be used to move processing routines from the application into the database itself. Access time can be improved by caching data in RAM. This approach is still comparatively slow, especially for write-intensive applications where disk I/O remains a significant bottleneck.

For higher performance, special-purpose real-time data-management systems have been designed to handle data entirely in RAM. Higher performance is achieved by storing data in fixed memory locations, instead of moving it around as in the case of a cache. This solution is also optimized for simple data structures. However, real-time databases of this type don’t offer high reliability levels, and they tend to be vulnerable to system crashes.
REAL-TIME DATABASE DESIGN

Dynamic reconfiguration can also be used to support online rolling upgrades. Such a design is capable of reaching the high-availability requirements of a maximum of 30 seconds downtime per year, which translates to 99.9999 percent uptime.

To enable a wide range of application developers to quickly use these features, it is essential to make them available under a standard application programming interface. At the same time, flexibility needs to be maintained to meet the requirements of a wide range of applications. This means supporting both shared-memory and shared-nothing architectures transparently, together with a very flexible redundancy model. A so-called N*Active/M*Standby redundancy model has therefore been proposed for real-time distributed databases and is being standardized within the Service Availability Forum.

In an N*Active/M*Standby redundancy model, the database uses N active nodes on which the database server (or servers) runs transactions on behalf of the applications. Redundancy in such a model is achieved in two ways: through replication of data across the active nodes and replication to the standby nodes.

The M standby database nodes can be activated during reconfigurations of the database. These may be planned reconfigurations, ordered by a database manager, or unplanned reconfigurations due to node failures. At failure of an active database, applications reconnect to a standby database.

In full replication, all data is replicated across all active nodes. For most purposes, however, it has been found that a partitioned replication with two replicas meets most availability requirements. In such a setup, the data is fragmented (distributed) over the N active nodes through a distributed linear hashing scheme, transparent to the application developer. The active nodes are then grouped into node groups of two, and the data is replicated among nodes within a node group, creating a primary and a secondary replica of the data (see Fig. 1).

The system is operational as long as all node groups have at least one active node that is functioning.

The distributed database can handle multiple node failures. When a node fails, the other nodes in the group will still contain the relevant pieces of information and can continue operating. When the node is restarted, the primary replica will assist the restarting node in regaining a current version of all the fragments. This is performed without any system downtime. Even writes to the database are permitted in this period.

Commit is reported after all replicas have performed the update and written the log to main memory. Transactions are (optionally) committed asynchronously to disk in a consistent manner. This allows the database to be restored to a transaction-consistent state if a system failure has occurred.

For disaster recovery, asynchronous replication is used between geographically dispersed databases. Parallel log channels between the global replicas supply a scalable mechanism that provides for network redundancy.

Benchmark tests have shown that real-time distributed databases based on a shared-nothing cluster design provide better performance than nondistributed
REAL-TIME DATABASE DESIGN

For instance, NDB Cluster, the real-time distributed database developed by Ericsson, has been benchmarked on commercial Unix systems to handle more than 1.5 million transactions per second on a 32-node database cluster configuration. The load was generated by 23 remote processors connected to the database cluster by a high-bandwidth interconnect.

Real-world databases

Distributed real-time databases are very common in core telecom network nodes, such as mobile switching centers, media gateways and media gateway controllers. The database is used for storing session-oriented data that reflects the state of ongoing connections and communication services. Preserving session data during partial system failures is crucial in achieving high availability. The database is also used for resource management, as shown in the example below.

For instance, in a media gateway, the database is used to manage a virtual pool of objects that represent the physical resources of the node (see Fig. 2). The task of the media gateway is to set up and tear down calls by handling the actual data flow (user plane).

The media gateway application implements the user plane control functionality. It starts by allocating the actual resources for the call the media gateway controller is trying to set up. The application sets up the devices for the respective protocols involved in the call to complete the connection and handle the subsequent data flow. At the end of the call, it disconnects and frees up the devices.

In case of the failure of an application process, another process may take over by retrieving the information from the database. For calls that cannot be saved, resources are freed up by using the information in the database. Using a database significantly simplifies resource management in the media gateway.

Factors to take into consideration for the database include:

• high-availability characteristics;
• fail-over below 50 ms in case of database processor failure;
• availability of data during live software upgrades;
• performance and scalability;
• 10,000 transactions/second on real-memory databases, together with better reliability and scalability.

For more on network reliability, see:

“For more on network reliability, see:


Tomas Ulin is general manager of Alzato, an Ericsson Business Innovation venture that develops and markets distributed real-time database technology to the telecommunications industry. Ulin holds a PhD in computer science from the Royal Institute of Technology in Stockholm, Sweden, an MS in electrical engineering and applied physics from Case Western Reserve University (Cleveland) and an MS in engineering physics from Sweden’s Uppsala University.