A Scalable and Highly Available Networked Database Architecture

Roger Bamford, Rafiul Ahad, Angelo Pruscino

Oracle Corporation
500 Oracle Parkway
Redwood Shores, California
U.S.A
rbamford,rahad,apruscin@us.oracle.com

Abstract

The explosive growth of the Internet and information devices has driven database systems to be more scaleable, available, and able to support online, mobile, and disconnected clients while keeping the cost of operations low. This paper presents the concept of Scalable Server that has the above characteristics and that can directly serve applications and data.

1. Introduction

The explosive growth of Internet commerce, combined with the increasing capability of cell phones and handheld information devices has imposed some challenges on the application system architectures. The system must provide near-linear scalability and high availability, it must support online and offline mobile applications, and the cost of operating the system must be low.

For the traditional application deployment, the three-tier client-server application architecture, with thin clients in the first tier, an application server in the middle tier, and a database server in the third tier solved the scalability problem to some degree. By replicating the application server, these systems were able to support more users. There are two basic models of the three-tier architecture. The first model uses a single database server. The second model uses many database servers with one designated as a master and the rest as replicas.

The single database server model creates a scalability and availability bottleneck in the database server. A typical approach to solve these problems is to use redundant high-end hardware for the database server. Such a solution is very costly and still has a low threshold for the number of concurrent users it can support. On the other hand, using database replicas create system management overhead as many databases have to be maintained and replicating and reconciling changes in them with the master database is non trivial.

This paper presents an application architecture that is optimized for scalability, availability, and cost of operation for the new breed of online and mobile applications. The architecture is based on the following observations:

1. The price of mid-size server hardware is low and continues to fall.
2. The cost of operations of an application system is directly proportional to the number of independent application and database servers used in the system.
3. The bandwidth of the "last mile" of wireless traffic of the Internet has not increased significantly but the number of wireless/mobile clients of Internet commerce is growing rapidly, resulting in traffic congestion in the last mile.

The architecture consists of a large number of *logically thin clients* served by a single *Scalable Server* (Scalable

Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

Server) either directly (two-tier) or via an application server (three-tier). The clients can operate in, and switch between, both connected and disconnected mode seamlessly. The clients carry a satellite database that contains a subset of data from the server. This database is managed by an ultra-slim DBMS. Both applications and data are transparently downloaded into the client a priori so that it uses less of the available bandwidth during operation. A good deployment strategy will permit clients to operate in a disconnected mode for an extended period of time.

The Scalable Server is a network of inexpensive physical servers that share the same disk storage system. A high speed interconnect among the physical servers is used to keep their states in sync. The client sees the Scalable Server as a single server with a single IP address. By using a Redundant Array of Inexpensive Disks (RAID) for the disk storage and redundant physical servers, the Scalable Server can support high availability. The Scalable Server is operational so long as at least one of the physical servers is running. To support more users without degrading the response time, more physical servers can be added while the system is operating. Thus the Scalable Server supports scalability without downtime and with virtually no administrative overhead. Finally, the Scalable Server presents a single-system view to its users, which simplifies manageability by reducing the complexity of managing a server cluster. Due to this feature, the administrative overhead, and thus the cost of operation, is very low.

Section 2 of this paper presents the Scalable Server architecture in more detail. Section 3 provides an overview of the client technology. Section 4 contains the concluding remarks.

2. Architecture of the Scalable Server

The Scalable Server is responsible for providing highly-available and scalable access to data and applications. The hardware platform of the Scalable Server is a scalable cluster of mid-sized commodity servers sharing storage over a scalable interconnect. On top of this platform the Scalable Server software provides a single-system view for data and applications. At the core of the Scalable Server is an extensible and scalable shared-disk database server running on the platform's commodity single-node operating system. Cluster coordination code in the database server talks across the interconnect to provide lock management and a globally-coherent distributed data cache. Experiments show that over current interconnects this architecture scales linearly with only a slight response time degradation when compared to single-node database. Although it is well-known that single-node database servers have the best base-line performance, the exponential price-performance curve of the hardware and the vulnerability to single component failures makes single-node database servers unsuitable as the core for the Scalable Server.

On top of the database server, the Scalable Server provides scalability, availability, and accessibility services such as event management, single IP appearance, and cluster volume management. Since the application and the database reside on the same computer, special attention is paid to minimizing the cost of the context switch between conventional applications and the database. Our studies show that many of the performance gains of in memory databases can be attributed to their low-overhead (linked-in) connection to the applications. As commodity operating systems such as Windows NT and Linux mature support is expected for light-weight subsystem calls that can provide similar performance gains without loss of fault isolation.

For the new world of Java applications, the Scalable Server provides an integrated Java platform. Thin clients and legacy applications communicate with the applications and data server on the Scalable Server via standard protocols and content representations such as HTTP, FTP, LDAP, IMAP4, IIOP, HTML, and XML. Proprietary protocols are used for SQL data access, queue propagation, replication, and publish/subscribe.

Since all services and libraries required to run the applications are provided by Scalable Server software, and since all inter-node protocols are controlled by the Scalable Server, it is possible to mix-and-match the hardware and operating system when expanding a Scalable Server cluster. In addition, Scalable Server management services provide single point of administration for all components. Combined with the linear hardware price-performance curve, the effect is an economy of scale for total cost of ownership as the workload and Scalable Server grow.

3. Client Technology

As information devices such as cell phones and hand-held devices grow in popularity, businesses are looking for ways to support them as terminals in Internet commerce applications. The problem, however, is the relatively narrow bandwidth of the wireless communication medium used by these devices in the first leg of their communication with the servers. The CPU speed and the memory capacity of these devices are growing at a faster rate than the communication bandwidth. This has prompted a new thinking in client technology to reduce the use of the communication medium and support disconnected clients. This goes beyond the traditional technique of caching.

A popular approach to improving the response time on the client for slow networks is to cache the data on the client side. Caching does improve the performance of the client. However, it has the following limitations:

1. Caching is reactive. The information must be retrieved from the server when the client needs it for the first time. Subsequent access to the same information may not need to access the server.
2. Updating the cached information requires access to the server to update the corresponding information (write-through cache). This is due to the fact that keeping the cached information consistent with the version on the server is a difficult problem to solve when updates to the cache are not immediately written to the server.
3. The validity period of cached information is the minimum of the validity of information items in it. For example, if a browser is caching a Web page that contains several information items in it and one of them changes every minute, then the validity period of the Web page is at most one minute. Even if this information item constitutes a tiny percentage of the total Web page size, the whole Web page must be refreshed if it is used after the validity period of the information item has expired.

In addition to the preceding limitations, caching is only used for data. The application that processes the data can either reside on the server or on the client. Server-resident applications are easier to maintain but require the clients to have continuous connectivity to the server. Thus they cannot be used to support disconnected clients. Permanent installation of applications on the client can result in high cost of application maintenance.

Our approach is to use a small-footprint database management system on the client side and automatically install and update required applications on the client. By selecting the deployment configuration in terms of data and applications needed by each client, this approach can optimally support wireless and disconnected clients without incurring the high cost of maintenance associated with a "fat" client. Changes made by the client application to the client database are synchronized with the Scalable Server. Any conflicts which result from the client and server (or other client) making changes to the same data item are resolved by either built-in conflict resolution methods or user-defined methods on the Scalable Server. Different policies can be easily implemented to resolve conflicts that cannot be resolved by these methods.

1. Concluding Remarks

This paper presents a high-level application architecture that supports scalable and highly available servers and mobile and disconnected clients. Oracle has shipped these types of products for over two years. Oracle8*i* with the Parallel Server option is the latest example of the Scalable Server. It supports a shared-disk cluster of machines that can be extended as needed. A highly scaleable Java virtual machine (VM) and PL/SQL VM are included in the data server for executing Java and PL/SQL stored procedures and triggers. Standard protocols including HTTP and IIOP are supported to access applications running on the data server platform. Release 8.2 improves scalability of the cluster cache and extends scalability and availability services of release 8*i*.

The Oracle8*i* Lite product is designed for mobile and disconnected clients. It consists of two versions of a small footprint DBMS, a data synchronization module, and an application development and deployment environment. One version of the DBMS is a full-featured object-relational DBMS that supports up to 32 concurrent applications. It provides ODBC, JDBC, C, C++, and Java fast path interfaces. The other version of DBMS is a 50KB version for cell phones and information devices. Both versions use the same database format and can replicate to the same server.

The data synchronization module supports both transaction log-based and state-based data synchronization. It can use wireless, dial-up, or a LAN network to communicate with the server, and supports both synchronous and asynchronous replication models.

The application development and deployment environment currently supports easy management of Web-based application development and deployment. It supports a runtime environment that permits one-click installation and update of Web applications and data. It employs a small footprint Web server on the client side that permits both online and offline modes of operation for the Web applications.