DATABASE PERFORMANCE IMPROVEMENTS USING TRANSPARENT PROXYING OF RELATIONAL DATABASES

Tonny Gregersen, Lars Chr. Hausmann, Peter Korsgaard, and Claus Thomsen

Group 727

Supervisor: Steen Andreassen

INTRODUCTION

It is common knowledge that large centralized database systems suffers from performance limits imposed by hardware costs and that distributed systems provide superior scalability and cost-efficiency [1]. However, the step from a centralized to a distributed system comes at the cost of higher development and management costs [2]. As a consequence this project deals with the development of a system which benefits from distribution, but without the added development and management costs. More specifically the project evaluates the following hypothesis:

> *Can methods used to ensure high availability, high performance, and high reliability in distributed systems be used to increase performance and flexibility of an existing centralized database system, while not changing any elements of the existing system.*

METHODS AND MATERIAL

In order to test the hypothesis a prototype system has been developed. This system acts as a proxy distributing queries from a number of clients to a number of database servers. An overview of the system can be seen in figure 1.

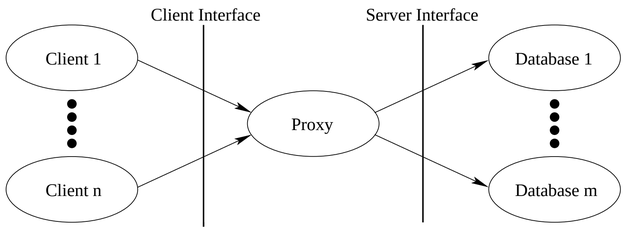

Figure 1: An overview of the developed system

During the design of the system the following issues have been considered:

- Transparency towards clients and servers
- Performance of search queries
- Data consistency during updates and failures

The system has to be transparent to both the clients and database servers in order of avoid changing any existing components of a system. Transparency in the proxy has been accomplished by emulating key parts of the database protocol in the interface of the proxy. In the prototype the database protocol emulated is that of the PostgreSQL opensource database [4].

The system is designed for optimal performance regarding search queries. Because of this full replication of all data among the database servers is used. This makes it possible to perform search queries in parallel.
In order to further improve the performance a load-balancing scheme using status information retrieved from each database server is designed. In the prototype status information is retrieved using the UC-Davis implementation of the Simple Network Management Protocol [3].
The overhead concerning creation of new connections to database servers is minimized by using a pool of persistent connections to each of the database servers.
The system has to retain consistency of the data in the database. This is done by locking the database during updates. Database failures are handled by removing the affected database server from the list of active database servers.
The performance of the prototype has been evaluated through a performance test. The test has been conducted using 10 clients concurrently sending queries to the proxy. The test is repeated using 1 to 3 database servers while varying the scale between search and update queries. The outcome of the test is the number of performed queries per second. This is compared to the values found during a reference test using a single database server.

RESULTS

The test results are shown in figure 2.

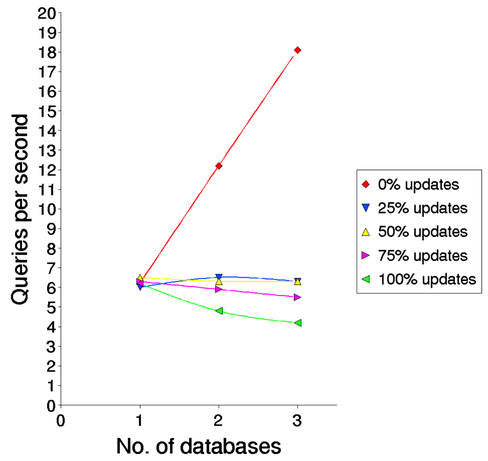

Figure 2: Test results

In the case of 100% searches performance increased linearly with the number of servers in the cluster. At 50% updates the system performs equally using 1,2 or 3 database servers. At higher ratios of updates the system experienced a performance decrease compared to a single site database.

DISCUSSION

The test of the developed system showed that the system was able to increase performance in search intensive environments, hereby confirming the hypothesis.
The test also showed that the performance increase regarding searches comes at the cost of a performance degradation regarding updates. This is due to the overhead imposed by the need to keep all database replicas consistent. The method used for this in the developed system is not the optimal solution and as a consequence update performance might be improved by using a more efficient update strategy, hereby improving the overall performance of the developed system.

REFERENCES

1. **M. D. Schroeder**. A State-of-the-Art Distributed System: Computing with BOB. In S. Mullender, editor, *Distributed Systems*, chapter 1. Addison Wesley, 1994.
2. **A. Silberschatz et al**. *Database Design Concepts*. McGraw-Hill, 3. edition, 1997: 543-564.
3. **The NET-SNMP Project**. http://net-snmp.sourceforge.net.
4. **PostgreSQL, Org**. http://www.postgresql.org.