High Availability of Commercial Applications

Kestutis Ivinskis
SAP AG

Email: ivinskis@sap-ag.de

Introduction

The increased performance capabilities of UNIX server systems have led to their acceptance as the server of choice for medium-sized and large organizations. But performance is just one facet. Another facet is the end user perception of the availability of an information system.

Traditional mainframe based IS shops have a long experience in supplying computing services to their commercial end users. The ultimate goal of the end user is to have no downtimes for his work at his PC or workstation terminal. Key issues related to system availability in client/server based information systems remain the same as in the mainframe based world, e.g. system responsiveness, maximum downtime per year and maximum number of system outages per year. But there are also new aspects, which have been introduced into the discussion.

In a multi-tiered client/server based information system the OLTP workload is distributed on different servers. Hence one can ask: Why should a failure of one server automatically imply downtime for the whole system ? Can't most of the system continue to operate ? Redistribution of the workload on the remaining active servers can be used to attack this problem. Workload balancing can be applied for replicated system services. Other techniques have to be applied for non-replicated system services.

This paper considers client/server based applications running in a local or wide area network of computers in a distributed system.

Availability of an Information System

The importance of availability becomes clear when various aspects are taken into consideration: loss of revenue, loss of staff productivity and motivation, circumvention costs. Relating the increasing business costs for downtime to the falling costs for high availability indicates an increasing future demand for highly available client/server based information systems.

A basic classification of system availability is to distinguish between the unavailability of a system caused by planned and unplanned downtimes. The term „high availability" usually refers to the property of minimal unplanned downtime, due to system failures. If system unavailability due to planned downtime is transparent to the end user, such a system has the property of „continuous operations". In this paper we will focus on highly available systems.

Traditionally, fault tolerance actions are performed at the hardware level. When one has to deal with availability strategies, one of the simple key insights is that high availability is a system concept, which should be measured in end user terms. High availability is not solved by one piece of hardware or software, but it is a system-wide consideration.

Analyzing Failures

In order to understand failures in an information system one can introduce basic architectural building blocks and classify their failures as proposed in [F]. This approach seems to be especially appropriate for client/server based information systems. It can be seen as a kind of hardware/software co-modelling, where one breaks down the information system in the following way: A „client" B depends on a service S delivered by „service provider" A, if the correctness of B's behaviour depends on the correctness of A's behaviour. If all end users are affected by the propagated failure of a service provider, we call this a *critical* component for the information system. An example of a critical component is the DBMS of an information system.

Design Goals for Highly Available Information Systems

The introduction of high availability concepts should take care of the following points (compare [BGHJ]):

- **No Loss of Committed Transactions:** There should be no loss of committed transactions. Uncommitted transactions should be rolled back. If no committed transactions are lost, only the last uncommitted transaction needs to be reentered by the end user. This request mainly addresses the corresponding property of the underlying DBMS and its interface to the middleware or application software.

- **Minimal Performance and Resource Impact:** In the fault-free case there should be a minimum impact on performance. Also the resource overhead should be minimal.
- **Reasonable Business Costs:** In order to keep business costs down the use of special purpose hardware should be avoided.
- **Minimal Loss of Volatile System Internal States:** The volatile internal states of the fault-free hardware servers (static and dynamic memory) should be preserved. If a hardware server fails, the rest of the servers should be able to keep their volatile data. This mainly addresses the different caching systems used on servers in a client/server based application. Otherwise, system performance may be affected after reconfiguration due to a failure.
- **Single Failure Resilience:** A highly available information system should at least be resilient to single failures.

Multi-tiered Client-Server based Applications

One of the advantages of a multi-tiered architecture is that it allows you to distribute the CPU load on servers of different types (scalability). In a three-tiered architecture the presentation (Screen I/O) is provided by a graphical user interface (GUI) program on PC's or „light-weighted" workstations. The application programs run on medium-sized servers and the database management is usually handled by a large server.

A three-tiered architecture allows you to configure many of its system services in a replicated way. Hence this architecture has an inherent advantage, when it is necessary to configure a highly available information system.

System Component Class
1. Presentation Software
2. Application Software
3. Middleware
3. Database Software
4. System & Network Software
5. Hardware Server Communication Network PC/Workstation

Table 1

Failure Classes

An information system consists of hardware and software components that are bound to fail eventually. In a three-tiered architecture all system components, which can fail, can be basically classified according to Table 1.

Examples of Failures

The failure of a GUI program (e.g. because the underlying PC fails) affects only one end user. The user may restart the GUI program (after changing to another PC, if necessary).

The failure of an application server affects all end users connected to this server. All these users have to restart their GUI program. Load balancing can be used to establish a connection to an active application server.

Failures of middleware critical components have to be handled by special fault-tolerant solutions.

The DBMS is a very critical component. The Shared-Disk Architecture with a standby server allows you to handle failures of the database server. The DBMS and the middleware should guarantee, that the switchover is completed within in a few minutes (depends on transaction load) and that it is transparent to the application.

SAP R/3 System

The SAP R/3 System is an example of a three-tiered architecture. The R/3 System is integrated standard software for client/server architectures, in which the applications can be distributed to hardware from different manufacturers. It has an extensive functionality and a high level of integration. R/3 was designed according to the concept of a „three-tier" architecture, in which the presentation, the application and the database level constitute a cooperative processing software configuration. It supports all popular relational database systems whose performance is adequate for the R/3 System.

[BGHJ] A. Bhide, A. Goyal, H-I. Hsiao, A. Jhingran, An Efficient Scheme for Providing High Availability, Proceedings of the ACM-SIGMOD International Conference on Management of Data, 1992

[F] F. Cristian, Understanding Fault-tolerant distributed systems, Communication of the ACM, Vol.34, No.2, pp.56-78, Feb 1991