IT-Technology for High Available Solutions in the Telco Environment

RTP - Reliant Telco Platform

Konrad Wiesneth and Stefan Arntzen

Fujitsu Siemens Computers

Abstract: This paper gives an overview about a software approach to use standard computer technique in the telecommunication environment. The most important requirements in this sector are highest availability together with short response time. Servers which fulfill these requirements are often called "Carrier Grade Servers". A special implementation for such a Carrier Grade Server is the Reliant Telco Platform (RTP) based on UNIX - or NT-Cluster- Technology and an appropriate middleware layer.

Keywords Reliable service platform, Cluster-system, Availability, Scalability

1. BACKGROUND AND PARADIGM SHIFT

Back in the 80s and early 90s most countries had a state-owned or semi-state telecommunications company which, in conjunction with the national government, dictated prices and kept them constant for relatively long periods. This monopoly resulted in a high price policy for telecommunication services and large profits for the network providers. Thus in turn they could afford proprietary and expensive network infrastructures. Technically, this period was characterized by the dominance of fixed networks and proprietary equipment. Also telecommunication networks and the upcoming data networks of the IT-world were considered as entirely separate worlds.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35522-1_37](https://doi.org/10.1007/978-0-387-35522-1_37)

But then happened two things which changed the whole telecommunication world: Firstly technically: People began to look at data and voice networks as one infrastructure; the market for mobile networks exploded; there was a dramatic need for so-called "intelligent services" like freephone, follow-me or prepaid; VoIP became more and more important. Secondly financially: Within a very short time deregulation broke down monopolies and new competitors arose.

At the same time the triumphant march of the Open Systems, e.g. UNIX- or NT-Systems, has begun in the IT-world. This resulted also in two big outcomes: Firstly technically: It becomes very easy for customers to opt for a different server manufacturer and to use heterogeneous components and software products in virtually any combination.. Secondly financially: This competition lead to considerable price drops and a development dynamic never dreamt of before.

In addition to these points Internet has become one of the most important communication networks. Therefore exchange of data get extremely cheap and the request to have the same features with the same price advantages grows. Now the time has come where telecommunications and information techniques will grow together even faster.

2. REQUIREMENTS FOR A TELECOMMUNICATION PLATFORM

The complex business environment of telecommunications imposes a lot of requirements on the systems used in the telecommunication area:

- High availability,
- Scalable performance,
- Short response time,
- Connectivity to different networks,
- Disaster recovery,
- Integration into different network management systems.

Another very important item is the relationship between price, availability and performance (Fig. 1). To solve these requirements in an economic way an open system architecture can be the solution. Since several years clusters are the response of the IT-world to the requirements availability and scalability and have in the meantime a high maturity.

Figure 1. Fundamental Requirements

3. CLUSTER-TECHNIQUE

Although some hardware and software measures for increasing the availability of individual single systems has been implemented, only an availability of standard servers between 99.9 and 99.99% could be reached. This is not enough in the Telco-environment. In this area, an availability of greater than 99.999 % - that means an outage of less than five minutes per year - is required. There are two approaches to improve the availability above 99.999 %. Firstly, Fault Tolerant (FT) servers with a special non-standard HW-technology or secondly, clusters of standard servers combined with a special middleware layer.

Figure 2. Price, scalability, and availability

FT-servers are normally proprietary high-price solutions and often restricted with regard to scalability and protection against context-specific software errors. Clusters are a bundle of individual servers (nodes) connected via

via a very fast cluster interconnecting and working with a common set of peripherals, especially disks.

A cluster offers two dimensions for improving performance and availability. Firstly, more processors and more availability functions (e.g. hot plug functions) in the individual servers based on a symmetrical multiprocessor technology (SMP-systems) and secondly more nodes within the cluster. Fig. 2 illustrates scalable system-configurations with increasing performance and availability with an optimal cost/performance ratio.

Because several instances of operating systems (one per node) and applications are involved, protection against context-specific software errors by changing the environment and continuing the application on a different node is possible and functions like "rolling upgrade to new versions of the operating system or the application" can be realized.

However programming clusters, especially parallel clusters, is normally a complicated and error-prone process. One has to think about load distribution, nonstop programming, inter-cluster communication, data locking, upgrade mechanism, changes of the configuration (e.g. adding or removing nodes) and so on. And last but not least one has to control the nodes states and in case of node-problems one has to switch the application to the surviving node.

Figure 3. Fault Tolerant Systems versus Cluster Systems

Our approach to hide these problems from the application is to provide a new middleware-layer, called RTP (Reliant Telco Platform). This platform provides a single-system-image to the telco-specific application and hides the complex cluster environment completely. In Fig. 3, a comparison is given between a single fault-tolerant system and a cluster solution based on RTP.

4. SINGLE-SYSTEM-IMAGE

To protect the application developer from all these nasty effects of a cluster-environment a single-system-image API (Application Programming Interface) is necessary. The view of the application of a cluster should look like a single high-available computer, no matter of how many nodes the cluster consists of (Fig. 4). For example the application should not be aware of the location of its resources and devices. The application sees only a logical cluster-wide addressing scheme; the transition to the physical addresses of the resources is task of the RTP

Figure 4. Single System Image

5. DESIGN PRINCIPLES OF THE RTP

For better understanding of the following some basic design principles for the RTP-cluster approach will be explained:

DOUBLE ERRORS:

One of the main basic principles is the provision of a fault-tolerant system which protects against single errors. Double errors are also tolerated in certain components (e.g. failure of two cluster nodes), but they may result in data and call losses. A concept that protects against multiple errors is extremely complicated and error-prone, as procedures for handling complex processes can themselves result in new error sources. Hardware expenditure would increase dramatically with negative impacts on performance. Such a concept has therefore not been developed.

SOFTWARE ERRORS:

Software cannot be written error-free but fault-tolerant. Many software problems are transient; that is, the problem is caused by an unusual environment state typically resulting from a transient hardware problem, a resource limit is exceeded, or a race condition occurs. In such cases, reinitializing the program state to an earlier point and resuming execution often works successfully because the environment is different. Nevertheless it is essential that applications are tested in a non-stop environment as far as possible.

ERROR DETECTION:

Error detection methods represent the main difference between special fault-tolerant systems and the cluster concept. The cluster concept cannot protect against processor errors. Although plausibility checks are used to try to detect and correct error states during program execution and periodic audit processes, there can be no 100% protection against processor errors. Special-purpose systems attempt to protect themselves against such errors by having several processors working in clock-controlled synchronism execute the same commands and detect errors by comparing the results. Experience has shown that this type of processor error is extremely rare and (once it occurs) can be very easily detected by the above-mentioned plausibility checks.

LIMITING THE SPREAD AND EFFECTS OF ERRORS:

Errors must be detected as early as possible and prevented from spreading out to other components, and in particular throughout the entire cluster. The design principle of combining local operating systems into a cluster by means of a narrow cluster package offers effective protection against the spread of errors. The cluster concept enables any component to be replaced online; the

appropriate node is shut down, the component is replaced, and then the node is reintegrated into the cluster. With this procedure, however, errors have a relatively strong impact on performance and availability. Therefore, methods such as hot pluggable devices and I/O boards are intended to minimize the effects of errors.

ROLLING UPGRADE:

In order to meet availability requirements, it must be possible to carry out all administrative activities online, i.e. without having to shut down the entire system. The following sections describe important techniques used for this purpose, e.g. online data backup. In particular, they include installing patches and updating to new versions of the products affected. Once again, the cluster concept offers the best solution here. It enables a cluster node to be shut down, the new or corrected product version to be loaded, the node to be reintegrated back into the cluster, etc.. However, it must be ensured that all affected products are capable of interacting with partner products of a different version on other nodes, at least for a brief transition period.

NON-STOP PROGRAMMING:

An application – particularly a client/server application – can be regarded as a continuous sequence of consistent states. The consistent states may be defined as contexts describing the corresponding process environment or global states. In the event of an error, it is possible to resume processing at the last consistent state, provided the corresponding context is still available.

REENTRANT PROGRAMMING:

Cluster nodes are normally SMP (Symmetrical Multi-Processor) systems. The performance and scalability of an SMP system essentially depends on whether or not there are sufficient executable processes available at a particular point of time. This is the only requirement an SMP system requests from the application; in all other points a SMP system looks like a simple mono-processor system. On the other hand, for performance reasons, it is not advisable to generate a separate process for each request. The generation and deletion of processes results in considerable overheads. A process pool represents a compromise between these two requirements. Working with process-pools or thread-pools requests a reentrant programming technique. That means one process can handle several dialog steps (call or call-segments) in parallel.

COMBINING REENTRANT AND NON-STOP PROGRAMMING:

The RTP attempts to fulfil both requirements, reentrant and non-stop programming, with the same programming model. The consistency points of the application are the beginning and end of a dialog-step (or call-segment). The

associated context at these points of time is stored in a so-called context data pool, which is mirrored for availability reasons on a backup system within the cluster. All the programmer needs to do is to retrieve the context data at the beginning of a dialog step and to save it again at the end; the RTP takes care of the rest. This technique makes the occasionally complex reentrant and non-stop programming transparent to the applications programmer. In case of a node error, the dialog can be continued on the node where the context data are mirrored by the previous dialog-step.

STORING DATA

The simplest way to store data in a cluster is to save them in a cluster wide available database. This is the simplest but not the most performant way. Depending on the purposes, the RTP uses therefore different techniques for storing data

- shared memory which allows the access to the data from several processes within a node,
- mirroring of the data to a second node which makes it possible to address the data from the original and the buddy-node, and
- a cluster-wide available and accessible database.

6. IMPLEMENTATION OF DETAILS OF THE RTP

The RTP consists of the following units:

- Connectivity-Functional-Unit: Responsible for the integration into the network;
- Snmp/Corba-Functional-Unit: Responsible for the integration into the network management;
- Data Base-Functional-Unit: Used for RTP- and User-Data;
- Scalability-Functional-Unit: Responsible for load distribution and dispatching;
- Availability-Functional-Unit: Responsible for Node- and Process-Management and to allow Reentrant & Nonstop-Programming
- Data- and Utilities- Functional-Unit: Responsible for Trace, Ticket-Handling, Object-Management, Statistics, etc.;
- Manageability-Functional-Unit: Platform-Management, especially "Rolling Upgrade" with zero call-interruption time.

Fig. 5 shows the architecture of a RTP-based cluster-system indicating the position of the diverse functional units, particularly for the requirements scalability, availability, data and utilities, and manageability.

Figure 5. Architecture of a RTP-based Cluster System

Such an architecture can be highlighted by the following properties:

- High availability thru cluster-technology
- Single system image
- Nonstop programming
- Load distribution
- Connectivity to Telco-networks
- Mated pair configuration (Disaster recovery)
- Event and alarm management
- Integration into different network management systems
- Scalability; predictable performance, and availability

In the following some components are give in more detail:

6.1 Super Node Manager

The Super Node Manager is responsible for starting, supervising and re-starting the functional units mentioned above. It is also responsible for node management and control. For this purpose, a health check mechanism is implemented and the node state distributed within the cluster

6.2 Signalling System No.7 (SS7)

In the voice-oriented telco-industry, the Signalling System No.7 (SS7) plays an important role. A cluster as possible destination of a SS7-message (signaling message) must be seen from the outside world as a whole and requires therefore a special implementation of the SS7 protocol-stack. This implementation is related on the mirroring of the addressing information and protocol states within the cluster to guarantee a common addressing scheme (one point code) or a suitable load distribution and recovery procedures in case of node errors. By a time out mechanism and by repetition of messages call losses can be avoided.

Figure 6. Signalling System No.7 (SS7)

Fig.6 illustrates the necessary protocol stack for interfacing with SS7 which uses at the lower three protocol layers the Message Transfer Part modules MTP1 to MTP3. The module SCCP (Signalling Connection Control Part) performs connection management and the module TCAP (Transaction Capability Application Part) handles packet communications. The two units SSP (Service Switching Point) shown in the lower part of the figure are the interface modules located at the switching nodes of the network.

6.3 Distribution of processes

One of the main objectives of the Reliant Telco Platform is to provide the applications programmer with a single system image. Because the underlying operating systems work locally, i.e. they are not aware of the existence of the cluster, whereby the Reliant Telco Platform is responsible for building the cluster accordingly. Some of the most important objects of the application toolkit are the processes which actually carry out the work. These processes are distributed throughout the cluster, and can thus reside on different nodes.

The Node Manager is responsible for the startup and shutdown, restart and monitoring of processes on one particular node. The monitoring is done by issuing health check messages in regular time frames.

The processes of the Reliant Telco Platform communicate with each other via the communication manager. Messages are the heart of the distributed environment of the RTP. Local systems like an operating system generally recognize two distinct worlds: the local environment and the network. The Reliant Telco Platform, however, sees only one world: the world of messages. In this environment, a process can send a message to another process on the same node just as easily as to a process on another node. Through corresponding addressing mechanisms, the actual location of the process is fully transparent to the caller.

Figure 7. Application/Communication Manager with backup processes

To achieve this, processes can be combined into alias groups, which are used for IPC (Inter Process Communication) addressing. This can be performed on the basis of m:n relations; a process can be a member of several groups. Each group has a cluster-wide logical name; its members are described in a list.

Two IPC addressing modes are supported for the alias groups: (1) (local) equal load distribution; (2) backup.

The mode “equal load distribution” means that processes in the group are addressed one after the other in a round robin procedure; so this mode guarantees an equal use of CPUs and nodes.

Extremely effective for non-stop programming is the use of backup alias groups. Backup alias groups are represented by a list of processes. One process in the list is known as the primary process. This may or may not be followed by several backup processes, which take over the responsibilities of the primary process should it fail. Fig. 7 illustrates that a failed process B in node 2 can immediately be taken over by the backup process B' in node 3.

Based on these backup alias groups, a concept of process pairs can be easily implemented. These process pairs communicate with each other internally and can be addressed externally as a single process.

6.4 Dispatcher/Load Balance

In a large cluster-environment not every node is connected to the network (e.g. to the SS7-network; at least two connections for redundancy reasons are necessary); therefore a dispatcher is necessary, to achieve reasonable scalability. This can be reached by distributing the load equally throughout all cluster-nodes. Fig. 8 illustrates load balancing with can be performed with different strategies.

Default-Strategy: Round-Robin

Figure 8. Load Balance

6.5 Context-Manager

The context manager plays a central role in implementing non-stop and reentrant programming models. The application contexts (checkpoints) must be stored safely to allow the continuation of the process in case of different errors, especially of process- and node-errors. In principle the space for storing the context can be seen as a cluster wide available shared memory. In principle such a shared memory could be realized via the data base; but this is would generate to much overhead. Therefore via the context-manager a technique is used which supports the context store as a mirrored shared memory. Fig. 9 shows an active and a backup process in each node with asynchronous mirroring of call contexts to the other node. In case of a node failure, a call in progress will automatically be continued by the backup process in the other node.

Figure 9. Context Manager

6.6 Rolling Upgrade

Rolling upgrade is only possible if the application, the middleware layers and the operating systems fulfill the compatibility requirement of versions n to $n+m$ (normally $m=1$). This requirement is hard to realize in a distributed environment. Not only the APIs must be compatible but also the communication mechanism and the contents of the messages. Furthermore, the upgrade for the hardware and software components require a computer element (CE) restart has to be done stepwise for each CE, always performing the whole cycle. In Fig. 10 the procedure for a rolling upgrade is given.

Figure 10. Rolling Upgrade

6.7 Disaster Recovery

The scalability of shared-disk and shared-nothing-database architectures highly depends on the latency of the messages sent to the different instances of the database-manager. The latency of the messages relates directly to the distance of the nodes of the cluster; the closer the node the better the latency. Therefore the distance of the node is restricted to several hundred meters. Disaster configurations which protect against global disasters like earth quakes require different technologies with asynchronous replication methods for the appropriate data.

Figure 11. Mated Pair / Disaster Recovery

The replication can be done by the logical disk subsystem or by the databases. Such configurations are often called Mated Pair Concepts. Updating

between to distributed nodes over wide area networks (WAN) is shown in Fig. 11.

7. SUMMARY

The Reliant Telco Platform RTP is not a telecommunication solution but a telecommunication "middleware" or "toolset" with the highlights: (1) Leading edge system based on a cluster technology; (2) Application programming interface (API) with a single system image; (3) Highly available (continuous operation, no planned or unplanned interruption of service); (3) Predictable performance, (4) Disaster recovery. This allows the application development to concentrate on their task to develop Services and not to think about handling a cluster especially in case of failure.

A main goal for the future is, to provide the RTP for the use in the JAIN environment. This will lead to a standardised application.

Figure 13. JAIN-Outlook

ACKNOWLEDGEMENT

Special thanks to Prof. Dr. Dieter Carl (Head of Siemens IN-development) and Dr. Ralf Rieken (Siemens Network Architecture), whose platform concept was the starting idea of our work.