CGL Architecture Specification

Mika Karlstedt

Helsinki 19th February 2003
Seminar paper for Seminar on High Availability and Timeliness in Linux
University of Helsinki
Department of Computer science

Abstract

The CGL architecture specification gives a more detailed picture of the requirements specified in the Requirements specification. The architecture specification covers every requirement in the Requirements specification V1.1 and gives examples of why these requirements are important and how these requirements can be achieved.

In this seminar paper I cover the most important aspects of the requirements specification, covering both why and how when applicable. For a person who is familiar with Linux and High availability environments, most of the material in this paper is already familiar and in fact covered in the previous paper about Requirements specification.

Contents

1 Introduction

The CGL Requirements Definition V1.1 describes the functionality which is required of any platform to be CGL compliant. The requirements can be divided into two separate groups, mandatory requirements and optional requirements. However The Requirements Definition does not tell how these requirements can be met.

The CGL Architecture Specification gives a more detailed picture of the requirements along with some examples of how these requirements can be met. The examples are meant more to give a usable idea of the functionality required, but the actual implementation can be different than the example given with the Architecture Specification.

The Architecture Specification is divided into five chapters (High Availability; Serviceability; Performance and Scalability; Tools and Standards). The next five chapters of this paper covers those chapters in the same order. The last chapter is a conclusion about the topics covered in the Architecture Specification.

1.1 Acronyms

There are numerous acronyms used in the literature, some of which may be unfamiliar to someone who has not studied the area. Therefore some of the most commonly used acronyms are collected here for quick reference.

- HA High Availability
- NIC Network Interface Card
- OS Operating System
- RAID a method of replicating disks (not quite accurate definition but suffices in this context)
- NAS Network Attached Storage, a way to share disk storage with other computers (once again not quite accurate)

2 High Availability

High Availability means that a system should be working properly seven days a week, 24 hours a day and 356 days a year. However no machine is perfect and cannot therefore provide this kind of service. Failures are caused by malfunctioning or broken hardware, errors with OS, bugs in the applications etc. These are all failures to which one have to be prepared for all the time. There is also the problem of future changes. When a system is used long enough it is feasible to assume that some time sooner or later there is a need update either the software or the hardware or both. These updates shouldn't cause any disruptions to services provided.

There are different ways of providing the high availability and this covers the methods chosen by the OSDL CGL for providing HA.

2.1 Hot Swap

One way of masking hardware failure (in fact the only way) is to use redundant hardware. However it is not enough that we have a spare disk in case the original breaks. After the original has broken we don't have a spare disk anymore. It is vital to have a means to detach the broken hardware and to replace it with a new working hardware while the backup hardware (which has now become the operational hardware) provides service. With disks this can be achieved by using RAID but similar requirements hold for any other hardware too e.g. CPU cards, NICs etc. Hot Swap is a means for providing this service. It makes it possible to detach a piece of hardware from the system and later attach either the same hardware or some other similar hardware back. This requires a capability to gracefully remove the drivers of the hardware so that the system wouldn't try to use the now removed hardware anymore as well as a capability of inserting new hardware and configuring it "on the fly". Currently this kind of service is provided for USB devices but there is need to provide same kind of service to the devices attached to the PCI bus.

Another fact closely related is the identity of the newly inserted hardware. If a NIC breaks there is a need to remove it and replace it with a new NIC. Since the system has a spare NIC the system can use it while the broken NIC is changed. But what happens to the IP address of the broken NIC. The backup has its own

IP and the system starts to use it instead of the IP of the broken NIC (unless Ethernet link aggregation is used). This means that the original IP is available when a new NIC is inserted. However it might not be obvious whether the new card should get IP of the old broken NIC or not. This is especially true when the old NIC wasn't broken and we are just moving it from one slot to another while at the same time we are adding a third NIC to the system. This kind of problem requires a way to solve Hot Device Identity. Another point is that kernel doesn't try to keep the old identity of the devices across system boots. It is possible that some applications require that a hardware keeps it identity after reboot even if the attaching point (e.g. PCI bus slot) is changed. This requires that the the device identity information has to be stored to a non-volatile memory and when a system is booted or new hardware inserted, the system first checks whether there is already an existing identity to the device. The approach chosen was to have a list of (name, value) pairs stored somewhere where they can be checked after a reboot or after a new device is attached.

Loosely related to above problems is the device driver hardening. Most device drivers are not designed to handle gracefully the situation where the hardware malfunctions. But when the system has redundant hardware a failure of a single hardware component doesn't mean that the system has to be shutdown. Instead the operation can continue normally with maybe a small decrease in the service since there is only one functioning piece of hardware left. However if the device driver isn't capable of handling malfunctioning hardware it is possible that the whole system shuts down because the kernel panics after the device drivers behaves "insanely". The purpose behind Device driver hardening is to make sure that no matter what happens, the device driver works properly. It cannot provide any services if the hardware it is handling is broken but at least it won't cause the kernel to panic. An existing device driver should be checked to make sure that it has timeout on all waits and that it checks every input and output parameter for validity.

2.2 Watchdogs and Heartbeat

In spite of every precautions some times the kernel just stops working. For a situation like this a special watchdog cards have been devised. These cards have a timer that is ticking down. If the timer gets to zero the card generates an interrupt. When the system is operating normally the OS will periodically reset the

watchdog timer thus preventing it from reaching zero. What happens when the interrupt is received depends on how the system has been configured. A simple approach is to make the system reboot whenever watchdog timer expires and generates an interrupt.

For applications this same service is provided with heartbeat. It is small message that the application sends periodically to the monitoring software. If the monitoring software doesn't get a message within certain time it starts to suspect that the application is not working properly anymore. Once again the operation of the monitoring system depends on the configuration of the system. A simple approach is to have the monitoring system restart the monitored application.

2.3 Ethernet Link Aggregation

Link Aggregation is a means to create a single logical Ethernet link using many different Ethernet cards as a bundle. All the cards share a single IP address, but are used to send and receive messages together. This provides two advantages; increased bandwidth and increased availability. The availability is increased because failure of one NIC decreases the available bandwidth but does not bring the link down. And the bandwidth is increased simply because all the NICs are transmitting simultaneously. Obviously the increase in bandwidth can only be reached when the simultaneous transmission from different NICs don't interfere with each other (e.g. old thin cable Ethernet cards wouldn't get increase).

2.4 RAID

RAID is a well known method of increasing both the availability and bandwidth of the disk subsystems. The OSDL CGL requires the support for RAID 1 which is disk mirroring. Linux kernel provides as is, device drivers for many existing RAID driver cards as well as support for software RAID. One major advantage of the RAID systems is the ability to do hot swap on the malfunctioning disk drives without reducing the serviceability of the RAID system.

2.5 Resilient disk management

It is not enough that the disk subsystem works well if there is a change that a system crash could corrupt the disk subsystem. Therefore the OSDL requires support for journaling file systems. Journaling file systems keep a log of all file system changes that have not been stored to the disk (Note that the log is stored to the disk). If a system crashes one only needs to finish the transactions in the journaling log to get the file system into a consistent state. Without journaling the whole file system would have to be checked which can take hours with big file systems.

Closely related to Journaling file systems is Logical Volume Management (LVM). Here the idea is to create a single "logical" volume out of many different disks. The volume has a single file system, but if there is need to increase the size of the volume this can be done by using the extra space left into the volume when it was created. Often it is impossible to know how much disk space various users (groups etc.) need. With LVM it is possible to allocate little disk space for every user. Later when some users start to run out of disk space, the extra space left unallocated can be allocated to these users. It is also possible to add new physical drives to LVM and increase the available disk space although adding new physical disks can require rebooting the system e.g with IDE disks.

2.6 Fast reboot

Sooner or later something will go wrong and there is need to reboot the system. Rebooting can however take quite a long time. OSDL requires that the rebooting cycle should last less than 1 minute.

3 Serviceability

This section considers the issues related to the serviceability of the system. While methods considered in the previous chapter makes the system more available they don't necessarily guarantee that a system provides the kind of services required. Issues concerned here are related to creating an embedded installations of Linux, gathering data of the operation of the system etc.

3.1 Boot Cycle Detection

It is possible that the hardware or software is not working properly. This could lead to a situation where the system crashes often and thus reboots often. This is very undesirable situation from the customer's point of view since he/she will get many broken connections within a short time. It would be better for the machine to power down and let another node to take over. Therefore there is a need to create a means for detecting boot cycles. If a node keeps rebooting very often it is vital that the administrator gets a notify of the situation and that the malfunctioning node powers down.

3.2 Support for embedded installations

Many embedded systems differ quite considerably from the way normal desktops are constructed. For one there may not be any disk in the system. This means that the system has to support booting from the floppy, CD-ROM, Flash memory card or from the network.

Many embedded systems are made of systems boards installed to a rack. It is advisable to create a way for these systems to communicate with outside world. Obviously these machines are attached to a LAN but if the NIC breaks it is impossible to contact them. Therefore some systems have a serial link which can be used to communicate with the individual systems in the rack. There is a switch attached to the rack so that a single keyboard and a single monitor can be shared by every machine in the rack. OSDL requires this kind of functionality from the CGL compliant systems.

3.3 Kernel Dump Enhancements

It is not enough that the system get rebooted after it has crashed. It is also vital to have enough information about why the system crashed so that the problem can be isolated and later corrected if there is need to it. Therefore it is vital to get the core dump created after the kernel panics. There is couple of problems which require a solution. One is where to store the core dump. The dump might be big and there is no guarantee that the system is equipped with disk. Therefore there must be a way to store the dump to somewhere where it can later be retrieved whether it is a NAS or flash card or some other network location.

Another problem is the size of the dump. If everything is written to it the dump might get too big for the environment. Therefore a way must be devised that the administrator can decide which parts of the memory is stored into the dump and which parts are omitted. The three choices presented are

- Don't dump any memory content
- Dump the entire contents of the memory
- Dump kernel memory used for data

The next problem is that there need to be a way to analyze the contents of the dump. This problem is closely related to the tools parts of the OSDL CGL specification.

The last problem is that at the moment the kernel dump includes only the state of the executing thread if the application is a multi-threaded application. This is not enough, so the dumping mechanism need to be improved to include the data of the other threads too.

3.4 Kernel Messages and Event Logs

At the moment the kernel messages are logged through syslog facility provided by the kernel. This method has couple of problems. One of them is that it is difficult and time consuming to parse the messages and another problem is that some key values like severity is ignored. Also there is no way to attach an operation to a given message.

OSDL CGL provides a way to use POSIX compliant event logging facility and to attach an operation to any given message so that if the message is received from the kernel a proper application can be informed about the situation and a policy based action can be performed.

When a hardware error is detected it is the job of the signal handler to create a properly structured kernel message and to deliver the message to the event log facility.

The final point is that some times it is impossible or time consuming to go to the place where the node is located just to get the event logs. Therefore CGL requires the administrator can use SNMP to retrieve the event logs from the remote location.

3.5 Probes

Probes are a small piece of code that can be attached to just about everywhere in the code. The probes have full access to all the memory and hardware registers available to the application where the probe was attached. This can be used to monitor the behavior of the application and especially to debug those hard to find bugs which occur infrequently when the system is running but whose reason is difficult to find.

4 Performance and Scalability

It is not enough to have a highly available system if the system is not scalable. When a system is used long enough, especially in the telecom area, it will eventually be either underpowered or overpowered. To tackle this problem it is vital that there are ways to monitor the performance provided by the system and also to have a way to either increase or decrease the overall performance of the system when need raises.

4.1 Resource Monitoring

In an embedded environment it is vital to have a way for monitoring the resources. It can never be guaranteed that the system would not malfunction so it is vital to have a way to monitor for such occurrences. If resources are depleted for some reason it is better to know it in advance so that something can be done to prevent the system from crashing. OSDL provides a resource monitoring framework which includes different kinds of monitors which can be used to monitor the use of the resources and to inform the administrator if there is beginning to be a problem.

4.2 Real-Time enhancements

Linux is not a real-time operating system nor is it possible to make it one. There is a need to provide at least soft real-time capabilities to systems used in the environment where the CGL is targeted. Therefore the real-time capabilities of the Linux kernel needs to be improved. Fortunately there exists some projects which

provide just these kind of improvements. These include:

- pre-emptible patch
- low-latency patch
- high resolution timer
- application pre-loading and locking

Pre-emptibility means that a process running in the kernel space can be interrupted and the processor given to some other higher priority process at any given time. This can be done when the process does not hold any kernel data structure locks. Low latency patch aims at shorten the time when the process needs to hold a kernel lock and can be used with pre-emptibility patch. However most important parts of this patch have already been included in the 2.4 series kernel and there is therefore no need to use it. High resolution timer gives Linux kernel the ability to create timer interrupts more often than 100 times a second (once every 10 ms).

And the application pre-loading and locking means that the process is loaded completely to the main memory and locked in there. Normally kernel only loads those parts of the process memory space that it requires at any given time (a technique called demand-paging). When a new page is needed it is brought from the disk. This could however lead to unpredictable delays in the process execution. Similarly by locking the process in to the memory the system can be sure that no part of the process memory is swapped to the disk at any given time.

4.3 SMP and Lock Contention Scaling

Most CGL nodes are expected to be equipped with from 1 to 4 CPUs. Traditionally 2.4 kernel has used these quite well but there is still room for improvement which could improve the performance of the CGL.

The problem with SMP machines is that when one processor is manipulating certain kernel data structure no other process even in another processor can have an access to that data structure. Otherwise important kernel data structures could be left to inconsistent state which would cause the system to malfunction or crash. There are ways to minimize the time when a given data structure need

to be locked and these techniques are what SMP and lock contention scaling is all about.

4.4 Hyper-Threading

Hyper-threading is a technology where a single CPU will appear to be more than one CPU. This can lead to performance improvement when more than process is used. This naturally requires a super-scalar architecture from the processor and also support from the operating system. Newest Intel processors provide this feature and the kernel configuration of the OSDL CGL provides support for this feature.

5 Tools

In order to make it easier to create new applications and to debug existing applications and also to improve the performance of the Linux kernel, there is a need to create new tools to be usable by the system designers and implementers. At present two such tools are needed.

5.1 GNU Debugger

The GNU debugger is unable to access the data structures concerning threads inside the kernel at present. When debugging a threaded application it is however vital to be able to see and possibly modify what is the status of different threads at any given time. CGL will provide this functionality to the gdb.

5.2 Kernel debugger

Kernel Debugger is a piece of code that resides inside the kernel. It can be used to monitor and modify the behavior of the kernel while the kernel is running. This can be used to analyze the operation of the kernel and to find places where the kernel performs poorly or incorrectly. It is vital that the kernel debugger uses authentication since using the kernel debugger it is possible to sabotage the operation of the kernel or to violate the system security.

6 Standards

One aspect of open source software is that such software tend to follow open standards. Linux is no exceptions and a big part of Linuxes usability comes from the fact that it is compatible with numerous standards. There is still room for improvements and this chapter introduces the most important standards that need to be included to the OSDL CGL.

6.1 Linux Standard Base (LSB)

The Linux Standard Base is a project designed to create a common standards of how different Linux distributions should be installed. It includes stuff like file system hierarchy, location of system files in the file system, versions of different libraries etc. In short a way to make sure that when one writes an application that runs in LSB compliant distribution one can be sure that the application runs without modification in any other LSB compliant distributions, meaning that the application can be sure that system files are located in the same places and also that the required libraries are installed.

OSDL co-operates with LSB project and tries to conform with it. However the OSDL CGL requires some additional packets to be installed and some times this can lead to conflicts. The OSDL will make these conflicts explicit when such are found.

6.2 IPv6

New IP protocol IPv6 has been around for quite some time already. The standard Linux has a support for IPv6 but the performance of the present implementation is not yet well known. In preparing for the future the OSDL requires full support both for IPv6 and IPSECv6 and mobility support for IPv6. The list of actual RFCs that are required change with newer version of CGL and can be checked from the architecture specification.

6.3 SNMP

Since most machines are suppose to be spread geographically in a wide area, there is a need to have a remote access to these machines (nodes). This can be achieved using SNMP and OSDL requires that the CGL systems are compliant with SNMP and MIB-II.

6.4 POSIX

POSIX is a standard for a generic UNIX like operating system. The standard itself is huge and Linux is not fully compliant with it though it is compliant where it is necessary. There are some new standards under POSIX that are of interest to CGL and these are timers, signals, messages, semaphores, event logging and threads.

Signals, messages and semaphores are not currently included into Linux kernel but the same functionality can be achieved using system V IPC which is present with present kernel (2.4).

There is need to create more accurate timers for Linux, and there exists many different implementation about high-resolution timer and also at least one project which aims to provide a POSIX compliant high resolution timer.

Similarly OSDL will provide a POSIX compliant event log implementation which will be included to the *libposix*. And there is a project that aims at creating a new POSIX compliant thread implementation since the one used in the current kernel is not POSIX compliant. This functionality will be similarly provided by the *libposix*.

7 Conclusions

As a conclusion one could say that Requirements definition V1.1 contains requirements that for the most parts can be easily satisfied by using software available or by creating a small amount of extra software to cover the requirements that would otherwise be unsatisfied. In fact at least Montavista already provides a version of CGL Linux. The problem with V1.1 is that it doesn't cover clustering and getting high availability of six 9s is probably impossible without some kind of clustering. Montavista claims that five 9s can be reached with their product. Whether the CGL Linux is actually usable in the so called "user plane" i.e. the

communication network meant to deliver e.g. speech and data in the telecom networks, is still unsure. Linux systems can provide the required functionality when the system is lightly loaded but whether the CGL V1.1 is enough to provide the performance in the high load areas is uncertain.

It is likely that future releases make it possible to create Linux installations that can be used in the telecom industry even when high reliability and strict real-time capabilities are needed. But most likely these requirements require support for efficient clustering solutions and also efficient methods for monitoring the resource usage and operation of the applications. Without these the sixth 9 is probably unreachable goal.

8 References

- Requirements Definition V1.1
- Architecture Specification V1.1
- OSDL - *http://www.osdl.org/*