Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Performance and enhancement for HD videoconference environment

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTOR: Daniel Turull Torrents

DIRECTOR: Jesús Alcober Segura

DATA: 25 de juny de 2008

Títol: Performance and enhancement for HD videoconference environment

Autor: Daniel Turull Torrents

Director: Jesús Alcober Segura

Data: 25 de juny de 2008

Resum

El treball de final de carrera que aquí es proposa s'emmarca dins del projecte de recerca V3 (Vídeo, Videoconferència, i Visualització) de la Fundació i2CAT, que té per objectiu final el disseny i desenvolupament d'una plataforma de vídeo, videoconferència i visualització independent de resolució en alta i súper alta definició sobre xarxes IP de nova generació. La Fundació i2CAT utilitza programari lliure per aconseguir els seus objectius. Utilitza UltraGrid per la transmissió de vídeo HD i SAGE per la seva visualització distribuïda entre múltiples monitors.

L'equipament emprat per a la gestió (captura, processat, enviament, visualització, etc.) dels fluxos multimèdia d'alta definició en l'entorn de treball ha d'estar optimitzat de manera que es puguin aprofitar tots els recursos disponibles, a fi i efecte de millorar la qualitat i estabilitat de la plataforma. Estem parlant del tractament de fluxos de dades de més d'1Gbps amb formats raw, de manera que l'optimització de l'ús dels recursos disponibles d'un sistema es torna una necessitat.

En aquest projecte s'avaluen el requeriments de l'alta definició sense comprimir i es realitza un estudi de la plataforma actual, a fi i efecte d'extreure'n els requeriments funcionals que ha de tenir un sistema òptim per a treballar en les millors condicions. A partir d'aquesta informació extreta, es realitza tot un seguit de proves en el sistema per tal de millorar el rendiment, des de nivell de xarxa fins a nivell d'aplicació.

S'han provat diferents distribucions del sistema operatiu Linux per tal d'avaluar el seu rendiment. Aquestes són Debian 4 i openSUSE 10.3. També s'ha provat la creació d'un sistema Linux partint únicament de les fonts de programari per tal d'optimitzar el seu codi en la compilació. S'ha realitzat amb l'ajuda del projecte Linux From Scratch. També s'ha provat d'utilitzar sistemes de temps real (RT) amb les distribucions utilitzades oferint una major estabilitat en la taxa aconseguida.

Un cop testejats els sistemes operatius, s'ha procedit a provar diferents compiladors per tal d'avaluar la seva eficiència. S'han provat el GCC i el Intel C++ Compiler, aquest segon amb resultats més satisfactoris.

Finalment s'ha realitzat un Live CD per tal d'incloure totes les possibles millores en un sistema de fàcil distribució.

Title: Performance and enhancement for HD videoconference environment

Author: Daniel Turull Torrents

Director: Jesús Alcober Segura

Date: June, 25th 2008

Overview

In this work proposed here is framed in the project of research V3 (Video, Videoconference, and Visualization) of the Foundation i2CAT, that has for final goal to design and development of a platform of video, videoconference and independent visualization of resolution in high and super though inside new generation IP networks. i2CAT Foundation uses free software for achieving its goals. UltraGrid for the transmission of HD video is used and SAGE is used for distributed visualization among multiple monitors.

The equipment used for management (capturing, sending, visualization, etc) of the high definition stream of work environment it has to be optimized so that all the disposable resources can be used, in order to improve the quality and stability of the platform. We are speaking about the treatment of datum flows of more of 1 Gbps with raw formats, so that the optimization of the use of the disposable resources of a system is given back a need.

In this project it is evaluated the requirements for the high definition streams without compressing and a study of the current platform is carried out, in order to extract the functional requirements that an optimum system has to have to work in the best conditions. From this extracted information, a series of systems tests are carried out in order to improve the performance, from level of network until level of application.

Different distributions of the Linux operating system have been proved in order to evaluate their performance. These are Debian 4 and openSUSE 10.3. The creation of a system from sources of software has also been proved in order to optimize its code in the compilation. It has been carried out with the help of Linux From Scratch project. It has also been tried to use systems Real Time (RT) with the distributions used. It offers more stability in the stream frame rate.

Once operating systems has been test, it has proved different compilers in order to evaluate their efficiency. The GCC and the Intel C++ Compilers have proved, this second with more satisfactory results.

Finally a Live CD has been carried out in order to include all the possible improvements in a system of easy distribution.

AGRAIMENTS

A la meva mare, per la seva paciència

A Xavier Miguélez, professor de la UPC, pel seus comentaris i correccions aportades en aquest treball i per la seva crítica constructiva.

A Francisco Iglesias, investigador de la Fundació i2CAT, per la seva ajuda en la comprensió de temes d'alta definició i les seves correccions i puntualitzacions.

A Davide Vega, pels seus comentaris sobre la memòria i el seu suport.

A Pedro Lorente, investigador de la Fundació i2CAT, pels seus coneixements de Linux i a les seves idees.

A Xavier Calvo i Javi Lopez pels seus suggeriments.

I a totes les persones que m'han ajudat. Gràcies.

INDEX

INDEX OF FIGURES

INTRODUCTION

Nowadays, the current trend in the scientific community is to work in collaborative environments, not only in the technological or engineering research field, but also in many others as medicine, science, education, art or cinema. A High Definition videoconference can help reducing the distance between geographic spread teams. Recently, these systems have been developed enabling a real time communication and interactive services, as e-learning, remote videoconferences or telemedicine.

i2CAT Foundation is working towards a solution of videoconference and visualization of High Definition (HD) and super High Definition (SHD) based on open source software. This project is called V3 (Video, Videoconference and Visualization). One of the main goals is to reduce the price of the equipment necessary for transmitting and displaying uncompressed HD video streams. Also, this group is working on a system for displaying super high definition video by a grid of screens (tiled display), called SAGE. In this environment, the performance improvement of all the machines involved is needed.

In this final degree project it is studied and implemented some techniques for increasing the system performance, from the optimization of the compilers for the target platform to the modification of the core system (Linux kernel). Also, has been implemented an autonomous live CD which incorporates the platform and all the improvements made.

In the first chapter the basic concepts of the videoconference service are explained. There are an introduction of high definition video and then explained the V3 project. With purpose of understanding better a Linux System there are a brief introduction their and some aspect of tuning performance.

In the second chapter, the current platform system is studied. First, it is described the objective of the project, and then the scenario is analyzed. In the middle part it is described the tools used for evaluating the system. Finally, there is a study to determinate the bottleneck of the system.

In the third chapter it is explained the actions tried for improving the system, and for solving the bottleneck of the system. We try different tips in the system, from kernel modification to network tunes. Also is explained the analysis of Linux From Scratch (LFS). Linux from scratch explains the way of making a Linux system compiling all the components of the system, for adjusting to the hardware. Moreover, has been tried a 64 bits Operation Systems and studied the differences of performance between 32 and 64 bits. Furthermore, it has been tried to modify some graphical aspects to improve the overall system. Finally, have been joined all the performance in a live CD for the easy movement of the system.

Finally, in the fourth chapter is explained the tests made in the application environment for trying to improve its performance. It has been tried different compilers, GCC and Intel C++ Compiler with their respective flags. Also, it has

been tested tools for profiling with the purpose of improving the code of the application.

CHAPTER 1. BASIC CONCEPTS

In this chapter is explained the basic concepts, concretely a brief explanation of videoconference system, high definition systems, V3 project and a brief introduction of performance tuning.

1.1 Video conference introduction

Whereas the communication is a natural ability and a really necessity of the humanity, the technology enables to do it in a couple of cases. In this scope, telecommunications offer a lot of possibilities for this objective. One of this is videoconference system, which brings the possibility to communicate people around the world without moving around.

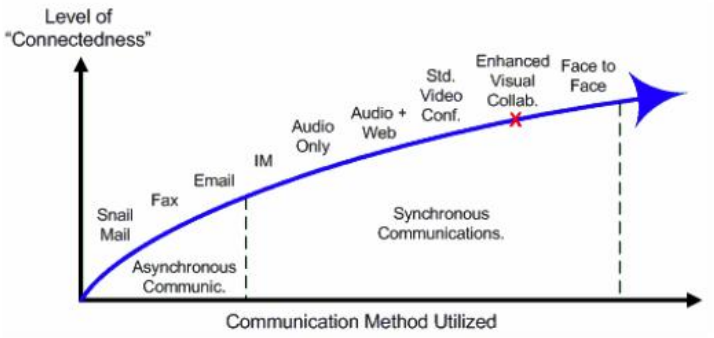


Fig 1.1 Communication level connectedness. Source [1]

1.1.1 Videoconference systems

A videoconference is enabled by a system that is able to transmit and receive simultaneous and synchronous video and audio in real time between points that are far from each other, through the network.

The videoconference systems and, in general, interactive systems, have strict requirements on time. One of the most important is the delay, which could break the interactivity between users. If we want an interactive conversation, we should have a delay of less than 150 ms (or 300 ms of RTT).

The goal of these systems is to enable an interactive communication with each other, like people do in a face-to-face way.

1.2 Introduction to High Definition Television (HDTV)

When we talk about High Definition Television, we have the goal to have a better image than traditional analogue television. The quality and the clarity of an image depend on the resolution. High definition signal is digital and its relation aspect is 16:9 rather than 4:3 in traditional television that offers a wide display of pictures.

1.2.1 HD resolution

High Definition standard describes four different models depending on the resolution of each image and the number of frames per second (fps). These are the 1280x720 HDTV (Standard 720p) or HD Ready, the 1920x1080 HDTV (Standards 1080p and 1080i) or Full HD and the 3840x2160 SHD. Their progression are quadratic, in form that the format SHD (Super High Definition) and its equivalent in U.S 4k (2048x1080), represents for times the format Full HD or its equivalent 2k (2048x1080), that at the same time represents almost 4 times the size of the traditional television PAL or NTSC.

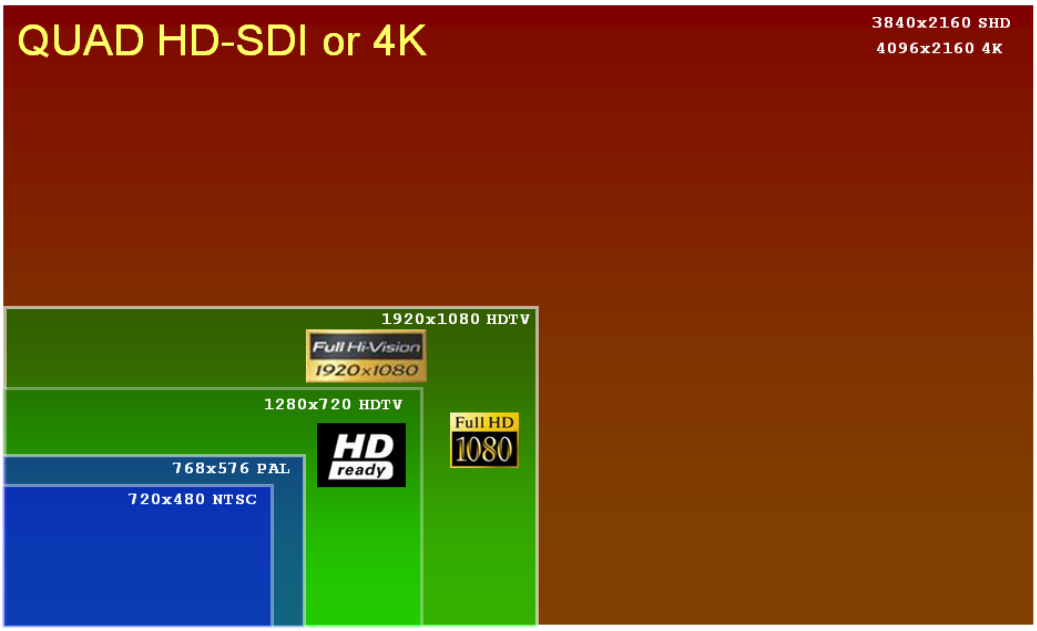


Fig 1.2 Display resolutions

1.2.2 HD frame rate

Another important thing in video is the frame rate; the responsible of the receiver brain perceives motion with static images. Frame rate is the number of

images displayed per second. Experimental studies prove that our visual perception starts when viewing a movement displayed around 20 fps or more.

In video there are two ways of displaying images:

- **Progressive scanning (p)**: where each scan displays every line in the image raster sequentially from top to bottom. Possible frame rate are 23.98 / 24 / 25 / 29.97 / 30 / 60 depending of the zone.

- **Interlaced scanning (i):** where each scan displays alternate lines (even or odd per field) in the image raster, and two complete scans are therefore required to display the entire image. Each scan is called field. Possible field rate are 50 / 59,94 / 60.

1.2.3 Color encoding

A color encoding is necessary for process the image and then it will be able to reproduce in other equipment. There are some theories about color encoding, which we only will explain a few aspects.

Usually RGB [2] (Red, Green, and Blue) encoding is used in electronic systems. This system assigns an equal weight for the primary colors (red, green and blue). With this encoding, it is possible to represent almost all visual colors in a black background.

Another color space is YUV [3] . YUV is a way of encoding RGB that offers some advantages. Firstly was thought for black and white compatibility, because YUV differs between luma component (Y), the brightness (1.1), and the chrominance (U and V), the color. U is the difference between blue and luma (1.2). V is the difference between red and luma (1.3).

$$Y = 0{,}299\ R + 0{,}587\ G + 0{,}114\ B \qquad (1.1)$$

$$U = -0{,}147\ R - 0{,}289\ G + 0{,}436\ B = 0{,}492\ (B - Y) \qquad (1.2)$$

$$V = 0{,}614\ R - 0{,}515\ G - 0{,}100\ B = 0{,}877\ (R - Y) \qquad (1.3)$$

Since the human visual system is much more sensitive to variations in brightness than color, a video system can be optimized by devoting more bandwidth to luma than to the color difference components. In that sense, appears the 4:2:2 scheme which requires two-thirds of the RGB bandwidth. This reduction results in almost no visual difference as perceived by the viewer. 4:2:2 are sometimes used in High Definition video.

1.2.4 HDTV and interactivity

Multimedia applications usually use video compression systems in order to reduce the size of the file reducing memory space and network bandwidth requirements. This method has some disadvantages, which are accepted in some cases, but not in HD videoconference.

First of all, digital compression implies a delay due the time to get the information and running the compression algorithms. Another disadvantage is the extra computational process in the receptor and the transmitter due to the images compression that consume CPU time. In small video stream, such standard DV, this time is minimally, but working with High Definition this process would provoke a considerable delay and it is not predictable. Other possibility is a hardware encoding system that encodes with JPEG 2000 without sensitive losses but its cost is very high. This delay is one of the causes of the loss of real time interactivity between the participants of the videoconference.

1.2.5 Advantages and disadvantages of High Definition in videoconference

Like other technologies, the use of high definition entails advantages and disadvantages.

The main advantage is the clarity of the image, which can be used in different fields, where high precision and resolution is needed, and also using uncompressed data will avoid possible compression losses and helps to the interactivity.

Otherwise, problems appear normally related with economic aspects, like:

- It is necessary a huge bandwidth for High Definition Streams transmissions.
- The necessity of raising the capacity of the equipment involved in the streams processing due to the high volume of data to be transmitted.
- High cost of the entire device involved in transmission and reception.

1.2.6 Bandwidth

The minimum bandwidth needed to assure that all the information can be transmitted at the same speed that is generated with propose of assuring that there will not be any kind of delay (no information stored). It can be calculated using the following formula:

$$BW = height \times width \times bits/pixel \times fps \times (Y+U+V)/4 \quad (1.4)$$

1.3 V3 Project

i2CAT Foundation [4] has a research project called V3, which its goal is to develop a super high-quality system experimental platform regardless the display resolution, video and videoconference over advanced Internet networks. Nowadays there is not any platform of video of 4k resolution for the user, and any platform of videoconference of high definition without compression implemented with free code or open source.

1.3.1 Final scenario

The final objective is to enable a videoconference with high definition resolution, with the smallest delay as possible, between four participants. This will be possible in an ultra-high speed network, HD cameras and 4k displays. These displays could be an individual 4k display, a 4k projector or a distributed visualization system. Each node should have a HD camera and 4k displays.

The distributed visualization system is the main way of research because can reduce dramatically the cost of the display and it offers scalability. Nowadays, there are few 4k projectors on sale and they are very expensive (around 1 million dollars). 4k screen is not possible because there are not these types of display in the market, and several of less resolution forming a 4k resolution display is very expensive. In fact, solutions like that, related to distributed displays, have an architecture based on connecting several displays in serial and the final resolution is the same as one screen but for all screens due the serial connection.

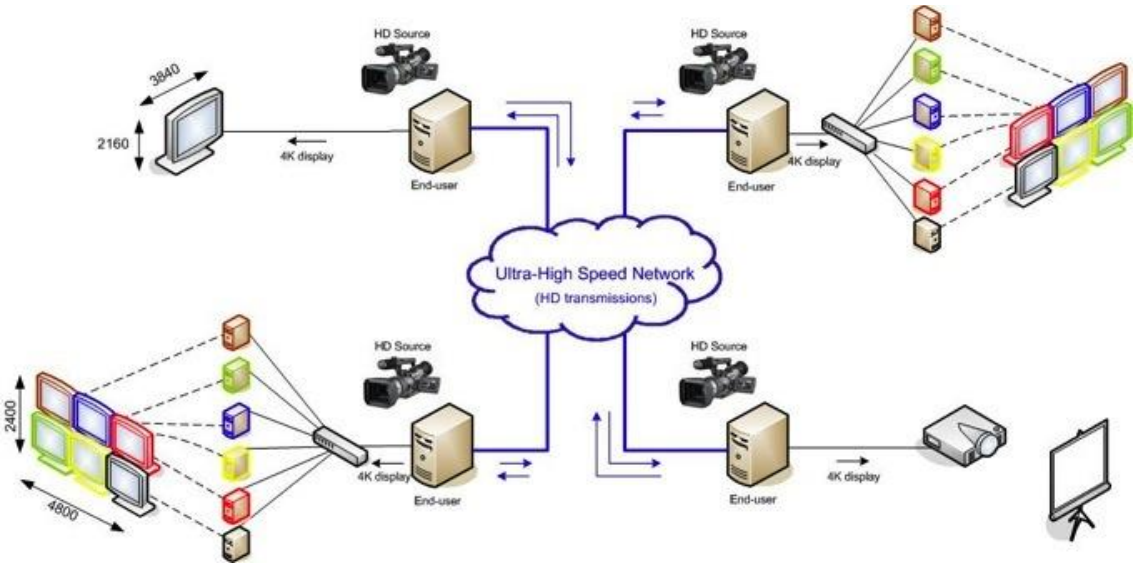


Fig 1.3 V3 Final scenario. Source [5]

1.3.2 Software involved

i2CAT Foundation uses Open source software for reducing cost of licenses and strength the collaboration witch other institutions of research.

1.3.2.1 *SAGE*

SAGE [6] is a UNIX application developed by Electronic Visualization Laboratory of Illinois University. Its function is to provide the needed architecture for the creation and management of a graphic display environment in a grid of screens. Although, it is able to receive different streams from more than one source and change properties at execution time, such position and size.

SAGE is able to get an image or video bigger than one screen and put it on in different screens cutting in the appropriate point. In this way it is possible to make a composite video wall with screens smaller than the video. Its main application, called SAIL, is able to decode the stream and send it to each receiver screen. The information goes uncompressed because it deletes delay and the receiver does not have to process the video, only paint it on the screen.

SAGE provides easy interface for migrate new multimedia applications to be capable to use sage as visualization mode.

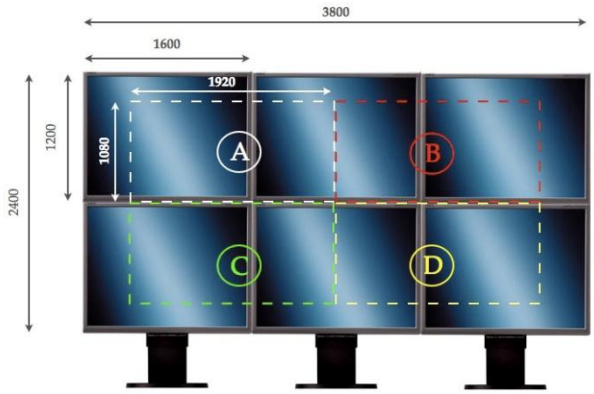


Fig 1.4 V3 distributed display for 4k manages by SAGE

1.3.2.2 *UltraGrid*

UltraGrid [7] is a high definition (HD) video conferencing and distribution system. It is also considered the first system capable of supporting uncompressed gigabit rate high definition video over IP. In fact, an UltraGrid node converts SMPTE 292M high-definition video signals into RTP/UDP/IP

packets, which can be distributed across an IP network reaching transmission rates until 1.5 Gbps. This application is very useful for videoconference in high definition because only introduce transmission delay, doing null the encoding delay. In this way, we obtain a quality almost perfect and a resolution (like digital cinema) higher than present videoconference systems.

UltraGrid is a software Open Source initially developed by the ISI EAST with the main goal to stress network launching a flow of video of high quality (HD) without compressing, in order to provoke congestion. This software has double function capture / display and transmitter / receiver, correspondingly. Therefore, it takes video on real-time and packs it in IP datagram to be sent over network. In reception, it made opposite work, receiving, unpacking and retrieving HD-SDI signal for any device that can understand it. It can work like a tunnel SMPTE input and SMPTE output though the IP network.

This software is very modular and it is possible to add new feature only adding the appropriate modules. Francisco Iglesias and Xavier Miguélez developed a module that do an interface between UltraGrid and SAGE for visualizing high definition streams on a title display. This interface will be used in tests made in this TFC.

Another feature added recently is DV (Digital Video) Standard transmission and reception. With this feature we can do a videoconference with a format more lightly for the network (around 30 Mbps)

1.4 General aspects of Linux

The entire environment is based on Linux and some previous aspects must be known in advance.

The global framework in Linux, that is not transparent to the user, is similar to the others UNIX Systems. Almost all the components of the system can be seen as file shape. This paradigm is applied to all the system, from the interaction with the hard system to the kernel tuning.

One important concept in Linux is /proc interface. Linux follows the concept of file system of UNIX and have a *filesystem* root /, and various points and directories that come from it. The /proc directory is one of these and it is a pseudo file system that does not reside in no disk, only in memory. In it, the *kernel* shows its interiorities and state by *filesystem* interface. Inside these directories there are data about several kernel subsystems. Some of these data are read only, and usually show information about present kernel status. However, with this interface we can view and modify kernel behavior without stop the system, by doing an echo to the appropriate entry when we are root. Must be noted that, the default values are balanced for almost all applications. This interface will be used for tuning some parameters.

1.4.1 Linux kernel Introduction

Linux Kernel is the core of Linux system. Technically, Linux is the kernel, and the rest of the system is a set of applications that it has not relation with the kernel. Because of it, also it is called GNU/Linux. The kernel offers an interface for the hardware and it is the responsible of the control of all processes that are running on the machine. In the kernel are loaded all the drivers of the system, the network stack, and parameters related to the processor, memory, buses, etc.

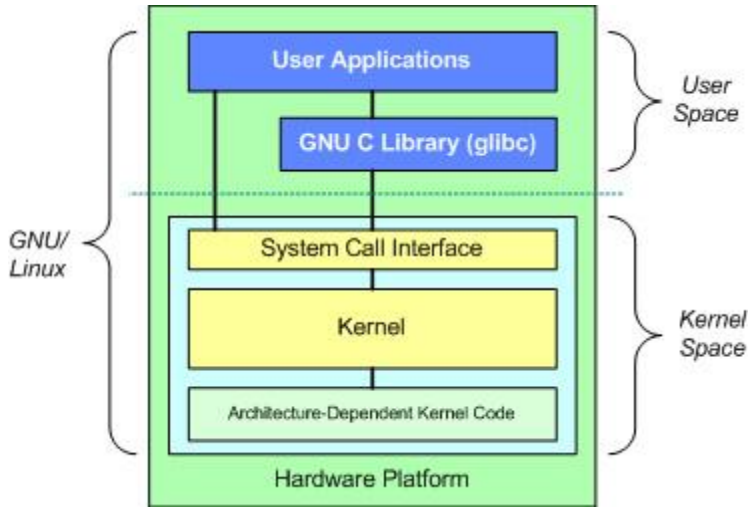


Fig 1.5 The fundamental architecture of the GNU/Linux operating system. Source [8]

At the top there is the user, or application, layer. This is where the user applications are executed. Below the user space there is the kernel space. Here, the Linux kernel works.

There is also the GNU C Library (glibc). This provides the system call interface that connects to the kernel and provides the mechanism to transition between the user-space application and the kernel. This is important because the kernel and user application occupy different protected address spaces. And while each user-space process occupies its own virtual address space, the kernel occupies a single address space.

Kernel features and drivers can stay on the kernel as built-in or as a module. When features are built-in always are loaded in the kernel and consume memory and increase the size of the kernel. Otherwise if features and drivers are loaded as modules they should be load a runtime for doing its purpose. Both options are good because the first one gives speed and robustness and the second one give more flexibility to the system because drivers loaded only when are needed or used.

1.4.2 Processes

A process is an instance of code that is executed for the processor, or in other words, it is a program that runs on the processor. The process uses any resources that the Linux kernel can handle to complete its task.

1.4.3 Context switching

In multitask systems, such as Linux, there are a lot of processes running at the same time. During the process execution, the information of these processes is stored in registers on the processors, called context. When the processor change the task that is executing, with the intention of running another task, this information it has to be saved for the next time of execution. These changes are necessary for a multitask system. These changes are called context switching, and they waste some time on the processor. For this reason, a high context switching value goes against global performance. Otherwise, context switching is absolutely necessary for a multitask system.

1.4.4 Interrupt handling

An interrupt is a signal that is usually generated by I/O devices, such as network interfaces. When an interrupt is launched, CPU stops and switch to a function called interrupt handler that notifies to the Linux Kernel about an event has occurred. This is critical for the system stability and a huge volume of interrupts will cause a rise of CPU use.

1.4.5 CPU scheduling and priority

The scheduler makes possible to execute multiple programs at the same time, thus sharing the CPU with users of varying needs. A scheduler can temporarily allocate a task (in quantities called *slices* of time) in the CPU.

Each CPU has a runqueue made up of 140 priority lists that are serviced in FIFO order. Tasks that are scheduled to execute are added to the end of their respective runqueue's priority list. Each task has a time slice that determines how much time it is permitted to execute.

In addition to the CPU's runqueue, which is called the active runqueue, there is also an expired runqueue. When a task on the active runqueue uses all of its time slice, it is moved to the expired runqueue. During the move, its time slice is recalculated (and so is its priority; more on this later). If no tasks exist on the active runqueue for a given priority, the pointers for the active and expired runqueues are swapped, thus making the expired priority list the active one.

The job of the scheduler is simple: choose the task on the highest priority list to execute.

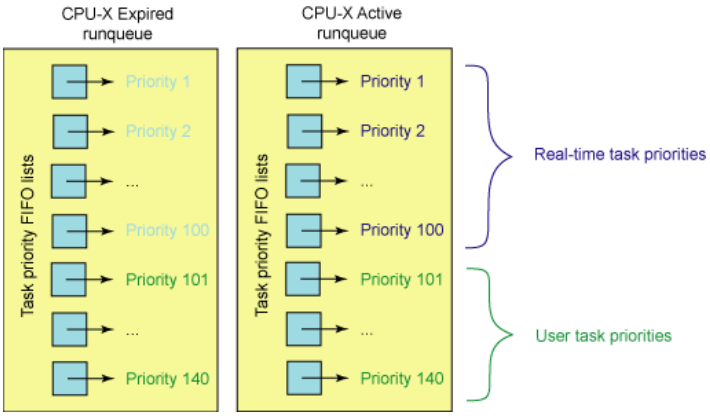


Fig 1.6 Linux scheduler structure. Source [9]

1.4.6 System load average queue

System load queue is the number of tasks that are ready and waiting for being run by the processor. The load average measures these values within intervals of 5, 10 and 15 minutes. An optimal value is less than one and means that there are never tasks waiting for being run. When the average is more than one, means that always there is some element of the system which is delayed.

1.4.7 Resource saturation and starvation

Saturation and starvation are related concepts, but have differences. Saturation could be seen as a subsystem that suffers an overload. Starvation is a problem that appears when there are waiting processes during long periods of time, when they want to access the resources that they need. In most cases, starvation is a consequence of saturation, but not always. Resource starvation in some cases can be an example of broken code where a part of the code has the resource and it does not release, and another part of the code want to access it.

1.5 General aspects of performance tuning

In the following points are explained what is performance tuning, what can be adjusted in a Linux system, tuning methodology and a consequence of taking samples in a system.

1.5.1 What is performance tuning?

In a simple sense, performance tuning is related to obtain something work more efficiently. But the efficiently always has a simple rule: “Depends on…”

Really, performance tuning is a process that should contemplate in its totality. There are two types of performance tuning:

- Proactive: planning and control carefully the system continuously.
- Reactive: solving the problems while they appear.

1.5.2 What can be adjusted?

Simply, all can be adjusted. Almost all the tuning in Linux Systems is related to the kernel, and some aspect of X Window.

The kernel is divided in elements, known as subsystems. In almost all cases, the subsystems are based in input and output (I/O). The subsystems that have been adjusted are:

1. CPU
2. Disk
3. Memory
4. Network

Something very important is how these subsystems interact and affect with each other. For example, one problem may disguise other.

1.5.3 Performance methodology

There are some rules that should be following for doing the performance in a correct way [10] [11] :

1. Assess the problem and establish numeric values that categorize acceptable behavior.
2. Measure the performance of the system before to perform any modification
3. Identify the part of the system that is critical for improving the performance. This is called the bottleneck.
4. Modify that part of the system to remove the bottleneck.
5. Measure the performance of the system after modification.
6. Depending on the results obtained in 5, return to 4.

Then we should start again the cycle identify another bottleneck.

1.5.4 Sample precision

To obtain samples (that will be used to perform statistics), it is necessary to interact with the system in execution. In the interaction it is altered the system. It is possible that only adding a task, but the values obtained there not will be absolutely reliable. Nevertheless, these values could give us an approximately global vision of the utilization of the system.

CHAPTER 2. CURRENT PLATFORM STUDY

In this chapter is described the test scenario, the techniques applied for analyzing it and the results interpretation. Also is explained the tools that are used and the parameter observed.

2.1 Objectives

One of the objectives of this project is to implement a scalable 4k display with the small price as possible. For this reason the use of a dummy client behind each screen of the tiled display is a good choice, reducing the total cost of the system and raising up the scalability. In our scenario, these dummy clients are barebones (chose previously as a good option) with a Debian Linux [12] operation System and with the SAGE. This equipment on raw is not enough for displaying correct HD videos and for this reason, they should tune up for improving its performance.

Before to modify parameters of the system and the application must be known what can be tuned up. So, the global system is analyzed and then where the bottleneck is figured out. In consequence, solutions for the bottleneck are designed, implemented and tested for improving the system.

2.2 Test Scenario

The scenario is composed by four dummy clients with its own 20 inch TFT display and the sender and receiver. We transmit a HD video stream from UltraGrid TX to UltraGrid RX. The size of the video is 1920 x 1080 pixels and its frame rate is 30 fps. The color encoding is YUV 4:2:2 and it is transmitted with 8 bit color depth. The result of this configuration is a stream of video around 1 Gbps of network bandwidth. The receiver receives the stream and retransmits the corresponding piece of the video to each dummy client with the application SAGE. The version used in SAGE is 3 and UltraGrid is 3.9.3 i2CAT version. All the dummy clients have a Debian 4 etch of 32 bits.

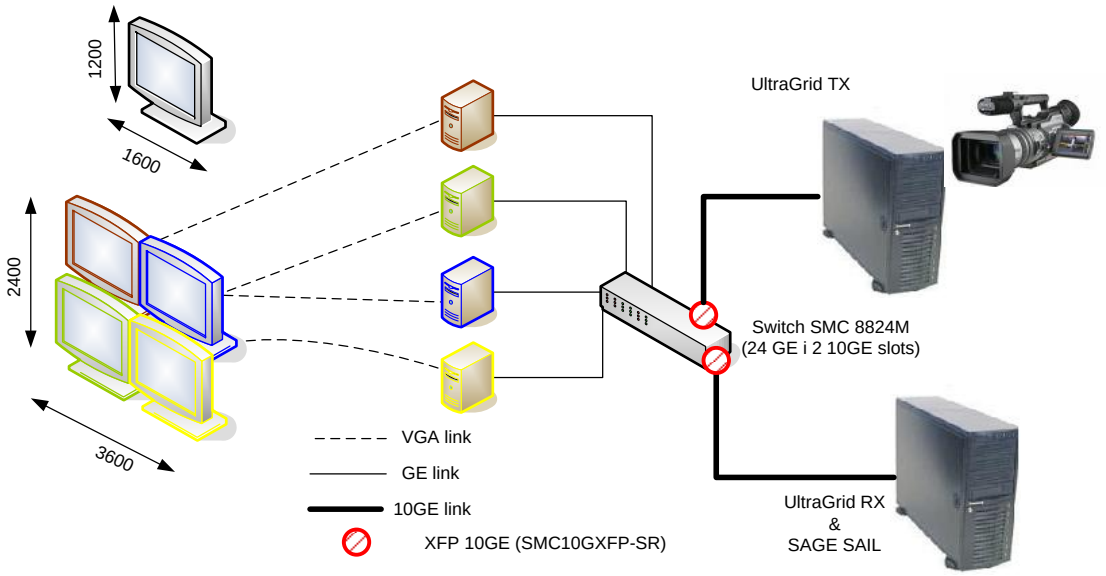


Fig 2.1 Global Test Scenario

The video is captured from a SONY HD camera with a color component output. The 3 component signals from the camera are converted to a HD-SDI stream using an AJA Converter and then introduced to the PC with a DVS Centaurus or AJA XENA cards. This middle step is necessary because the camera does not have an HD-SDI output directly. Cameras, which have this output are very expensive. When the information has already processed by UltraGrid Tx, the transmission to other videoconference nodes is done by an Intel 10 Gbps Ethernet Network Device.

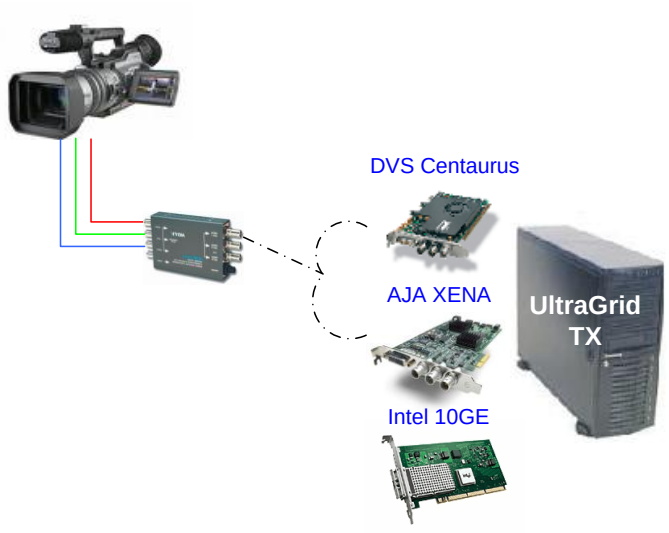


Fig 2.2 Capture system of V3

The dummy clients are Shuttle Barebones, model SN68SG2[1], with an integrated 1Gbps network device and its graphic card it is also integrate with the motherboard. The graphic card is NVIDIA Geforce. This configuration will bring us some problems with the internal busses because these devices use the same controller chipset. The monitors used are NEC models of 20 inch with a resolution of 1600 x 1200 and a color depth of 24 bits. For more detail about the specifications see the annex.

Fig 2.3 Dummy client used on SAGE. Source [13]

2.3 Monitoring tools

The monitoring tools are used to monitor the environment for knowing the state of the system depending on the actions taken in the tuning process. Have been used different applications for this proposes. There are generic tools, such as iperf, and specific tools, such as SAGE UI.

2.3.1 SAGE UI

SAGE UI is the controller of the overall system. One of its features is related to monitoring, showing information about the render and the data displayed. This information includes the current and average frame rate and bandwidth, very useful for detect bottleneck whether it be on the render side or the displaying one.

[1] Available: http://eu.shuttle.com/es/desktopdefault.aspx/searchcall-12/searchcategory-256/noblendout-1/tabid-72/170_read-14215/

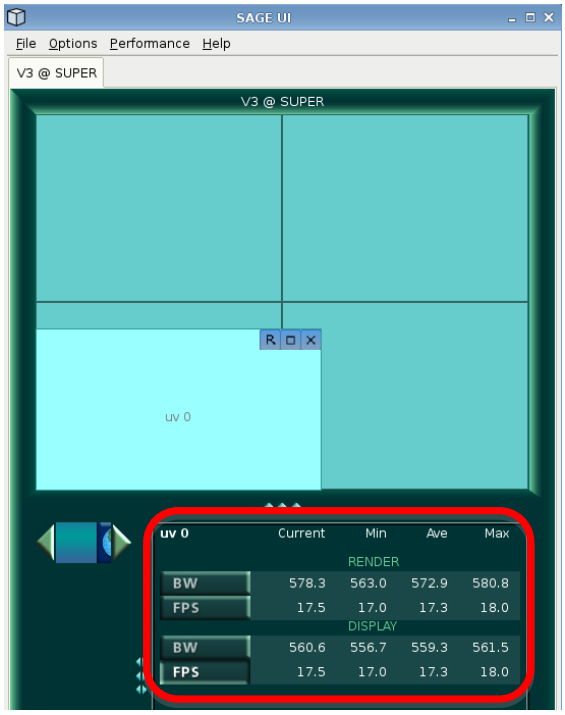


Fig 2.4 SAGE UI

2.3.2 Iperf

Iperf [14] is the classical Linux tool that enables to test a network connection between two nodes. It can operate with TCP and UDP and returns the network bandwidth of a given link. It is important that there is not any other connection on the link while the tests are done because it will interfere over the results. Iperf follows traditional client-server architecture. A server is listening for incoming connection and the client starts the connection and measures the network bandwidth in a unidirectional way. It's possible to measure bidirectional way.

2.3.3 SYSSTAT& KSar

SYSSTAT [15] is a group of utilities that permit monitoring an UNIX system. We use mpstat and sar.

Mpstat shows statistics of the processor. Most useful statistics are:

- User: percentage of CPU utilization for the user (applications).
- sys: percentage of CPU utilization for the system (kernel)
- iowait: percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request
- irq: percentage of time spent by the CPU or CPUs to service interrupts.

- Idle: percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.

Sar is another useful application that collects information for all system, from processor to network. The output of this tool is redirect to a file for posterior analyzing. Sar is used for monitoring network parameters, the memory utilization, the process load, and for determining the bottleneck of the system. A script has been made for collecting data from multiple nodes at the same time.

Ksar [16] is a java application that graph sar output in a simple and fast way.

2.4 Benchmarking tool

Benchmarking means measuring the speed with which a computer system will execute a computing task, in a way that will allow comparison between different hard/software combinations. [17]

Have been tried different benchmarking tools to evaluate our system, but only some of them have been finally used in this thesis. In spite of testing the programs explained in the next points we have not extracted significant information about it, because our application don not carry out with the specifications for these tests.

2.4.1 Lmbench

Lmbench [18] is a program that does bandwidth benchmarks. Have been measured the memory read and write speed and we observe that the system provides enough speed for the bandwidth required.

2.4.2 LTP

Linux Test Project [19] is an open source project managed by IBM. LTP has thousand of test. Have been tried the general test with the purpose of obtaining a number that assesses our system but these test only show system information and non summary results.

2.4.3 Customized tool

According to the working behavior of SAGE, it has been created a personalized benchmark with iperf and x11perf. It has been simulated at the same time a TCP stream and graphic utilization. Iperf is used for network traffic and x11perf

for evaluating X server latency. x11perf is used with the test of putting an image of 500 x 500 pixels to the screen. With this test we will obtain a value that in theory is the maximum that we can reach with SAGE in the current machine.

2.5 Identifying the bottle neck

First of all, the scenario has been analyzed and then the bottleneck pointed. If we send the video at centre of the display (each dummy client receives a forth part of the stream) we observe with the SAGE UI that frame rate is 30 fps and the total bandwidth is around 1000 Mbps. In this screen position, load is balance between the four clients and each one receives approximately 250 Mbps.

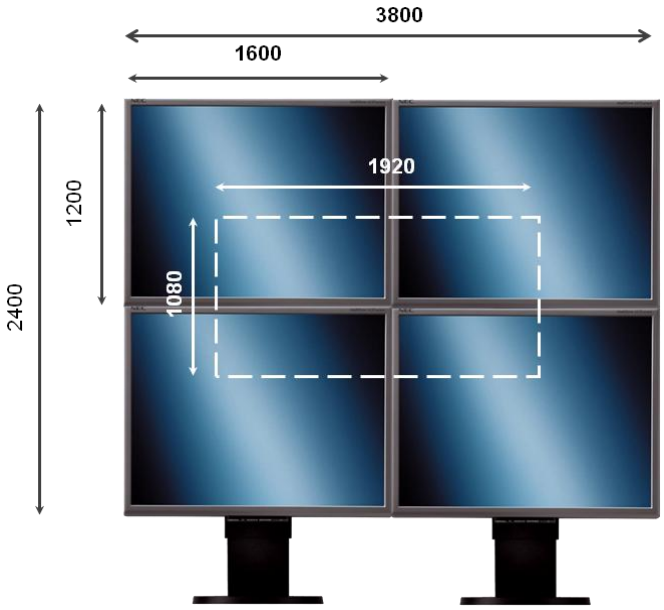


Fig 2.5 Scenario with center video

When we move up the video and it is only showed in two screens the frame rate and bandwidth is reduced. And finally, if we display as much as possible on one screen we can show the worst case. After we analyzed the scenario, we guess that the bottleneck is on the dummy client, so first of all we have tried to improve its performance.

All the tests described below are made with this display position:

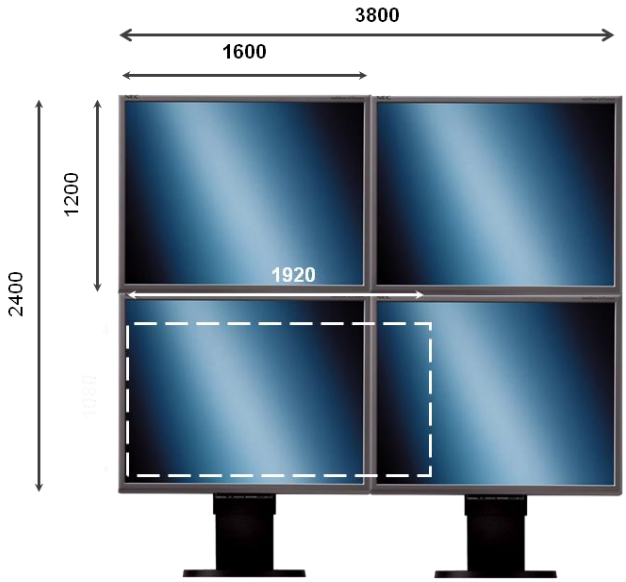


Fig 2.6 Scenario with the most loads to one client.

After of localizing the equipment bottleneck, we have to determinate which component is the main responsible of this low performance.

In the next points we will analyze CPU, memory and networking when SAGE works. We monitor the loaded dummy client for fifteen minutes. The main function of SAGE clients is received the data uncompressed from the network and painting to the screen. Also there is synchronization between screens. The disk is not used for this program.

2.5.1 Analyzing CPU

First of all, we focus on CPUs. Whereas the dummy clients have two CPUs we will analyze both. Before SAGE starts, used processor in both is around zero. When SAGE starts first CPU reaches around 95 % of user used and around five in system used. In the other hand, the other CPU reaches 50 % of system and around 10 of user. This assignment is made by kernel cpu dispatcher and usually is the best way to execute the tasks.

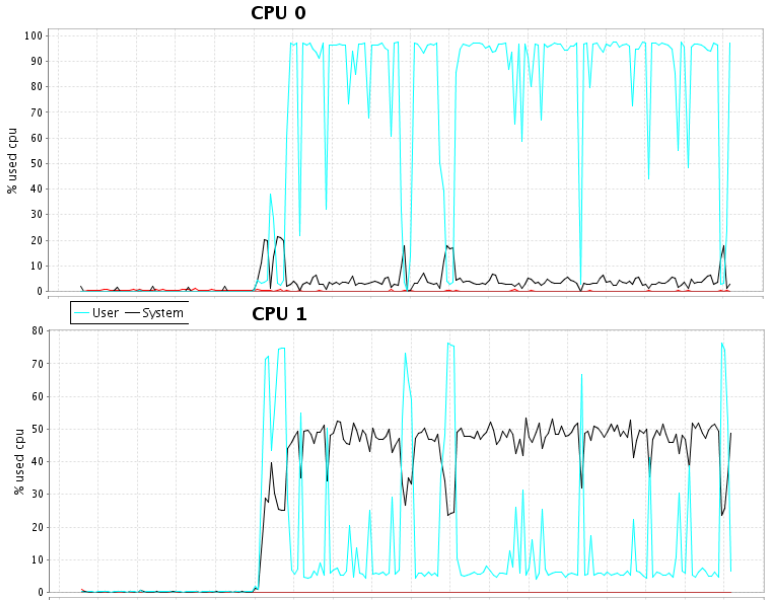


Fig 2.7 Percentage of used CPU with SAGE running

System activity is kernel activity related to networking activity. Its load is balanced to one CPU. User activity is overload because reach the maximum capacity of the CPU. One of the possible problems of poor performance of our system is the high CPU utilization. We have to determinate if our code is correct executed.

If we observe mpstat we observe a huge amount of interruptions (almost all made by network device). These interruptions consume time of CPU. Due there are lot of interruptions there are a lot of context switching, in the way of wasting cycle of processor in these changes. We should find a way of reducing these interruptions.

In the next chapter, we will study some techniques for rice the performance, reducing the use of the CPU of the system, compiling with more efficient compilers and more specific instructions for the processor and some other techniques.

2.5.2 Analyzing memory

The system has 1 GB of RAM memory. This size of RAM is enough for our system because before starting SAGE its value is around 180 MB used and when we use SAGE is value reach 480 MB and it is constant. We do not have problems with RAM memory

2.5.3 Analyzing network

The receptors have a gigabit Ethernet card and the transmitter a 10 gigabit Ethernet card. With this configuration, in theory we can reach around one gigabit for one dummy client. But this is not true because, network and graphic card is controlled for the same chip. This situation will reduce our performance.

2.5.3.1 *Iperf*

First, we test network card without SAGE and not load and we obtain 933 Mbps of network bandwidth. Then we try with graphic and CPU load made by X11perf. X11perf is a tool of X server that measures the maximum speed of different kind of graphic tests. We try putting an image of 500x500 pixels to the screen. We note that network bandwidth is reduced around a 17 % and reach to 767 Mbps of network bandwidth. This denotes that network activity performance is directed related of graphic and CPU utilization.

2.5.3.2 *SAGE*

When network traffic is monitored, we observe with SAGE UI and sar that not all the bandwidth offered for the link is used (more or less 50 % used). This is a good signal because give us a leeway of rising. In the other hand, we have some error on frame (around 15 errors/second). This is a symptom of overloading. One reason for this fact is 1500bytes MTU is used due network devices not support higher MTUs and the network device receive a lot of packets per second (around 42000 packets / second)

2.5.4 Conclusion of current bottle neck

With the analysis made at before points we arrive to the conclusion that the first problem of the system is the load made for the CPU. First of all, we have to reduce this load. Then the network can improve its utilization. Hardware architecture it's a possible bottle neck due the chipset control of network and graphic device, both the most used by SAGE.

CHAPTER 3. TUNING THE SYSTEM

This chapter describe the techniques that are tested for improve performance of the system and its results.

3.1 Simple actions for reducing unnecessary load

There some actions that can reduce the load of the system with only few changes, such as disabling daemons, desktop manager and so on.

3.1.1 Disabling daemons

Daemons are programs that reside on memory and usually offer services that are probably not needed. Disabling these daemons frees memory, decreases startup time, and decreases the number of processes that the CPU has to handle and in some time use network bandwidth.

3.1.2 Disable desktop manager

Usually Linux has a Desktop manger that is in charge of managing all the action related on desktop, such as windows, visual programs, etc. There are different desktop manager, but all of them has one thing common: consume a lot of resources. SAGE only needs the X server, not a complete desktop and, of course, no a desktop manager. For this reason, we can execute SAGE at runtime level 3, and then execute only the X server, nothing else. In this way, we avoid loading multiple tasks.

3.1.3 Delete unnecessary terminals

Linux has multiple terminals which it is possible to access with the combination of keys: Ctrl + Alt + Fn. Each terminal is managed by one task. Usually all these terminals are never use and we can safety reduce the number of terminal to one or two. This is possible editing the file /etc/inittab. Also in this file we can select the run level which one we want to run Linux on start up.

3.1.4 Performance improvement with simple actions

The differences between before applying the techniques described below, and the results are very similar. With SAGE UI we cannot view significant

differences. In spite of this fact, we have reduced the number of tasks in execution, and therefore, we have freed the processor activity and possible network unwanted utilization, and stabilize the system.

3.2 Linux Kernel

Linux Kernel [20] controls the operations between software and hardware and can be optimized. First of all, we can reduce the drivers loaded in it, with propose of reduce its size and the amount of memory that occupies. Also we can build it for a specific processor rather than generic processor, which option is default in most distribution. And finally, we can apply patches or modifications in its code for obtain other features.

3.2.1 Real-time patch

3.2.1.1 *What is real-time?*

Real-time system (RT) [21] gives a guaranteed maximum time of what can happen.

Real-time applications need to be scheduled before a deadline of an event is reached because in another case the task realized will not valid, because its information will be expire. This definition is related with multimedia programs that need to display or reproduce a determinate data in a determinate time. If the time is passed, the data will lost its value and should have to be dropped for a correct viewing. Real-time tasks are designed to use kernel resources in managed ways then delays can be eliminated or reduced

We try to adapt a Linux System to a preemptive mode for trying if the performance is increased. Sometimes real-time might actually slow down the system slightly.

3.2.1.2 *Real-time in Linux*

Linux not support totally a real-time system; only support a partial real-time system. The option of the kernel for active real-time compatibility is preemption mode. The newer general kernels could be a *No Forced Preemption (server)*, *Voluntary Kernel Preemption (Desktop)* and *Preemptive Kernel (Low-Latency Desktop)*.

There are some projects for include all the features on it. One of these is Real-time patch developed in kernel.org. This project offer a patch for a kernel which modify a lot of source code for adapt to it. With this patch appears another

option on preemption mode, *Complete Preemption (Real-time).* Also include support for timer of high resolution, for reducing the amount of time between each interrupt.

3.2.1.3 *Conclusions of real-time tests*

All the combinations of real-time options in the kernel (with patch and without it) of dummy clients (see ANNEX) are tried in this project and it is observed that the best performance is produce with real-time turn off. This is one of the possible results because there is a tradeoff between throughput --- the overall efficiency of the system --- and latency. Real-time add some overhead, and for this reason, the throughput is reduced.

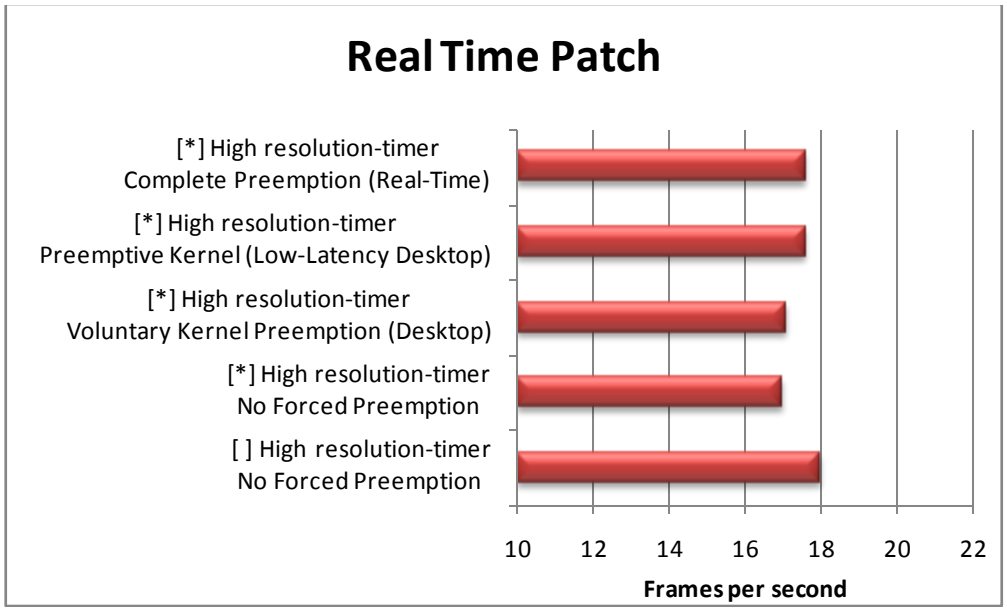


Fig 3.1 Real Time results

Also we experiment some unstable behavior on the system. Furthermore, if maximum priority to SAGE process is set, the system is completely freeze and it is necessary to restart. This is because the system, not have sufficient resources for attend SAGE, and because it has the maximum priority, starves all other task and the system cannot do anything else.

When the system has enough resources real-time give stability. Also real-time patch on UltraGrid Rx (Supermicro) is tried and it is obtained a more stable rate to barebones. Without real-time the results give variations around 5 frames per second (±25 %) and when it is enable only 0,5 frames per second (± 2,5 %).

With these results we recommend real-time patch when the system has enough capacity of manage all the process but when the system does not have sufficient resources is not recommended because produce starvation.

3.2.2 Reducing kernel size

A generic Linux kernel has thousands of drivers and features that are never used and it make compatible with almost all devices. Adjusting the correct drivers and removing drivers and features that never are used can reduce the size of the memory. It should rice the speed of the kernel, due consume less resources. Also if we compile the used modules as built-in in kernel we should gain some performance because the communication between drivers and features is more efficient and faster.

openSUSE [22] Linux is tested and we observe an increase speed of 5%, or in other way, one frame per second faster in the worst case scenario. But, for the other hand, compatibility with some extra system equipment is reduced.

Kernel has an option that reduces the size of the kernel but obtains a less optimized kernel. For default kernel is enable but if it is unselect we gain a little performance.

3.2.3 Processor options

Kernel permits some actions to manipulate behavior of CPU and task on it.

3.2.3.1 *Shielded CPU*

Shielded CPU is a technique that permits assign a process in an exclusive way to one or more processor. This technique has been tested with SAGE and the dummy clients but in the way that SAGE uses more than one processor, this action only reduce the performance of the application.

3.2.3.2 *CPU affinity for interrupt handling*

Another option to reduce the load of one processor (on system with multiple processors) is configuring the kernel for dispatching all interruptions launched by one device (determinate for IRQ) to a specific CPU. This is called CPU affinity. This action was tested with no visible results.

3.2.3.3 *Nice and renice*

Other option is change priority of the process that execute the program which we want improve, in our case SAGE. With this action we can ensure that the process has more time on the processor. This action it is possible to do by program nice and renice. This is useful when there are a lot of task running.

We try to assign the higher priority for the SAGE in a typical system, and the performance did not increase. Sometime the system hangs up because it cannot attend critical functions on the kernel.

3.3 Linux from scratch (LFS)

Linux from scratch (LFS) [23] is a book where it is explained the way of making a Linux system from source; construct all the system, compiling all the operation system tools for learning how it works, and how to customize your system. In this project has been build following the recommendation of this book for trying to improve performance.

3.3.1 Brief description how to build it

The LFS system does not have an installer on a CD or DVD and need to use a previously installed Linux distribution (such Debian or openSUSE) for built it. The existing Linux system (the host) is used as a starting point to provide necessary programs to build the new system.

First it is necessary to create the partition and its file system where LFS system will be compiled and installed. Then it is necessary to download the packages and patches for building the LFS system. The book tells which are, why and what do them.

When necessary packages are downloaded, the next step is setup an appropriate working environment for making a cleaning installation. Then we have to create the tool chain. Tool chain is a temporary system for making a true cleaning install. There are two steps in building this minimal system. The first step is to build a new and host-independent tool chain (compiler, assembler, linker, libraries, and a few useful utilities). The second step uses this tool chain to build the other essential tools.

When minimal system is built, we complete the building of the full LFS system. The chroot (change root) program is used to enter a virtual environment and it starts a new shell whose root directory is the new system partition. Chroot permits working in another root directory with the kernel of the host. The major advantage is that “chrooting” allows the builder to continue using the host while LFS is being built.

To finish the installation, boots scripts are set up. After this, kernel is compiled and boot loader is configured. Finally, we can reboot the system and enter to our Linux from Scratch system.

LFS book only install a minimal tools on the system. For installing SAGE, we need a set of tools that Beyond Linux From Scratch (BLFS) tells how install it and explains theirs dependencies. Basically, we install net tools (for configuring network devices and access to the system remotely), graphical tools (X server and graphic libraries) and python tools, needed by SAGE.

3.3.2 Results

When the system is correctly running, LFS was tried with the same kernel as Debian tests and the values has obtained are more or less the same values. With these results we discard the use of a LFS system because its implementation cost. Also, in our context, it does not have visible advantages than other pre-build Linux distributions.

Otherwise, building a LFS system gives us a major perspective and knowledge of how Linux work inside it. Also it give us a way of how configure a Linux system.

3.4 64 bit Operation System

In his point is explained the test made in 64 bits operation system.

3.4.1 Why 64 bits?

Almost all of processors which are sold our days are 64 bits architecture. This is our case but the current Operation System is thought and compiled for 32 bits architecture. If we install a capable 64 bits operation system will should notice a rising of performance and speed. This improvement is due to more efficient use of processors. Also 64 bits architecture uses bigger registers and is optimized for manipulate huge amount of data.

3.4.2 Comparison among 32 and 64 bits with SAGE

Linux system has ported to 64 bits architecture since first commercial 64 bits processors appear. Nowadays, almost all distribution has its flavor in 64 bits. First it is tried a Debian amd64 because is the best choice for comparing with

Debian 32 bits. In SAGE, tests denote that performance is increase in 12 % between Debian 32 bits and Debian 64 bits.

Next step was tried SAGE with another distribution. We try OpenSUSE to its focus on multimedia. We obtain that OpenSUSE is around 11 per cent faster than Debian 64 bits and 20 per cent faster than Debian of 32 bits. When Debian kernel is tested in openSUSE distribution we obtain the same results as Debian. We can conclude that performance is very dependent of Linux kernel. OpenSuse community applies a lot of patches for improving the performance. Behind openSUSE are companies such IBM, Novell and AMD.

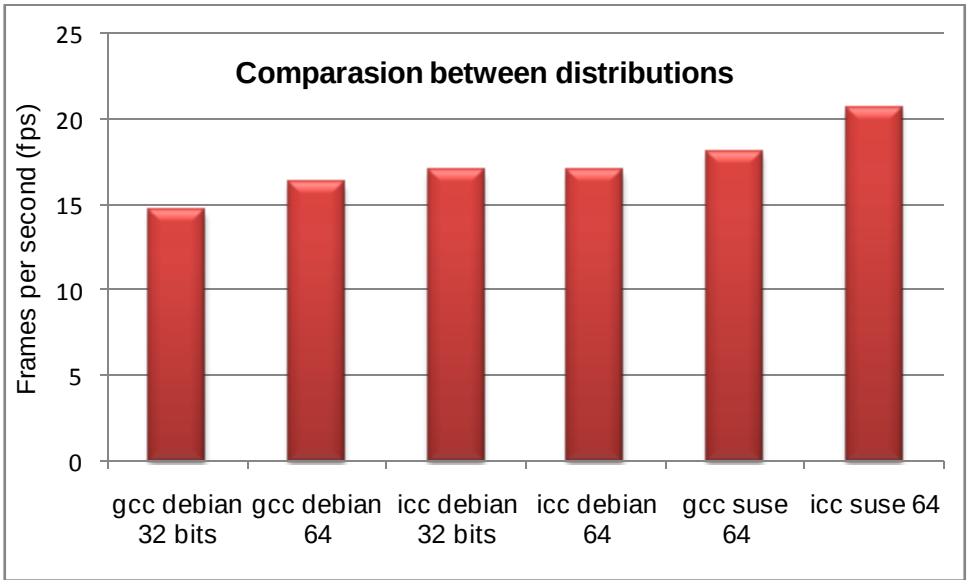


Fig 3.2 Comparison between distributions

In viewing these results we decide to change the operation system to OpenSUSE 10.3 64 bits for amd64 architecture.

3.5 Network

SAGE and UltraGrid use network in intensive way because a lot of data is necessary to transmit. We can modify some network parameters for trying to improve system behavior.

3.5.1 MTU

The Maximum Transfer Unit (MTU) is the maximum size of the packet. The network devices integrated with the barebones has a maximum of 1500 bytes,

the standard size of Ethernet network. Otherwise, in gigabit Ethernet it is possible to use a higher MTU. It exist jumbo frame packets, with a size of 9000 bytes. These packets, in the way it can transport more information, the number of interruptions for packets are reduced.

At present, SAGE use a 1500 bytes of MTU and this fact cause a lot of interruptions due the huge traffic involved. We try with iperf an external device but its performance is worst than onboard card. If only compare with this card that accepts higher MTU values we observe a gain of network bandwidth.

3.5.2 NAPI

NAPI ("New API") [24] is a modification to the device driver packet processing framework, which is designed to improve the performance of high-speed networking. NAPI works through:

- **Interrupt mitigation**: High-speed networking can create thousands of interrupts per second, all of which tell the system something it already knew: it has lots of packets to process. NAPI allows drivers to run with (some) interrupts disabled during times of high traffic, with a corresponding decrease in system load. This reduce system load.

- **Packet throttling**: When the system is overwhelmed and must drop packets, it is better if those packets are disposed of before much effort goes into processing them. NAPI-compliant drivers can often cause packets to be dropped in the network adaptor itself, before the kernel sees them at all.

3.5.2.1 *Test Results*

NAPI was tested in barebones and UltraGrid RX. The amount of interruptions is reduced from around a 30 %, but in SAGE there is not a visible improvement.

3.5.3 Offload

Offload is a hardware technique for reduce consume of CPU made by the network. If the network adapter supports it, the kernel can delegate its task to the adapter and it can reduce CPU utilization. It is possible to do checksum of IP/TCP/UDP and TCP segmentation (TSO) with this technique.

3.5.3.1 *Test Results*

We try offload on onboard network device on dummy clients with bad results. The performance is reduced when iperf is used. Also we try with an extra network card that support Jumbo Frame but it performance is worst than integrated device.

3.5.4 Socket buffers

UltraGrid uses a huge network bandwidth and the default buffers reserved by the kernel not are enough. It is necessary to rice its values. For making this, we use the proc interface. The size of the new reception and transmission buffers are a full frame. If this path is not applied, there are some green lines on screen due the insufficient reserved memory.

3.6 Reducing screen resolution

Other option is reduce the resolution of dummy clients screen. In one hand, the total network traffic and the image to be display is reduced. In the other hand, we will need more monitors and dummy client for displaying the same pixels.

Table 3.1 Screen resolution vs theory network bandwidth

Width	Height	Frame size (Mbits)	Network bandwidth per screen (Mpbs)
1024	768	12,6	377
1280	1024	21,0	629
1600	1080	27,6	829
1600	1200	30,7	922

With the current equipment, satisfactory results will be obtained reducing the resolution of the screens. Although theory calculation, a full rate video (30 fps) is obtained when 1024x768 resolution is set. When we set the next resolution the frame rate reach at 25 fps.

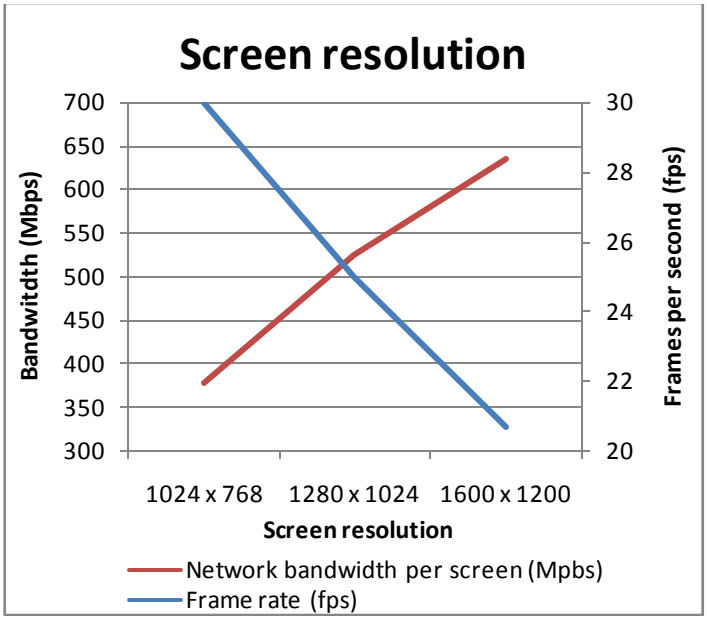


Fig 3.3 Experimental Frame rate and Network bandwidth related to screen resolution

3.7 Changing graphics card

Other possible improvement is a hardware improvement. We try a NVIDIA Geforce graphic card on one barebone and we raise the average frame rate to 25 frames per second in one display. If we go from 20 fps, we gain around 25 per cent. It is necessary to say that graphic card tested it is not designed for the barebone box and we have to dismantle the box and left it open.

3.8 Make all in common. Making a customized live CD

Once we try different tips for trying to improve the performance of the system, now is time to pack it in a useful live CD. A live CD is a complete operation system, usually Linux, which can be boot without installation. This is useful because you know that the system always is how you think. Also you can move to another compatible system without any installation. More over it is useful for trying new systems without installing and you have your system ready in a few minutes.

We made two live CD. The first one is for the UltraGrid Receiver and the second one for the dummy clients of SAGE. UltraGrid Receiver live CD is based on Debian system and the second based on OpenSUSE. Both CD are made in different way with propose of testing different shape of done the same thing. The Debian live CD was made with Debian-live project and the OpenSUSE live CD with Linux-live scripts.

3.8.1 Debian-live

Debian has a project for developing a Debian Live CD, called Debian Live [25] . It consists on a set of scripts that can automate the building of a customized live CD based on Debian Distribution. Basically, the operation of building the live CD consists in three parts. First, it is necessary to initialize the program with *lh_config* for making the directories for configuring the live CD and the scripts. Second, personalize the content of the live CD listing packages desired to install. Also is possible to access to the new live CD with chroot for making a finest personalization. Finally, the program creates an ISO CD image or an usb image with *lh_build or lh_binary*.

Before starting the live CD is necessary to make some steps. First of all we have to configure our kernel for being capable to be executed in a live CD. A live CD is a normal Linux system but with some differences. The most important is that the medium on resides is read only. This tool include the default Debian kernel on the CD but we want to use our personalized kernel with real-time. The main reason is that default Debian kernel is 2.6.18 and don't support real-time. Also with our personalized kernel we can add some performance features. Because we use a personalized live CD, it is necessary to add some modules to the kernel for permit to work properly. These modules are squashfs and unionfs, which they are filesystems. First is for compressing the system in a simple file and the second is a useful filesystem for a live CD system.

For making the task to integrate the kernel to the live CD easy, we create a deb file with the kernel. This is done by *make-pkg*, a tool provide by the Debian package *kernel-package*. When this file is created is insert at the directory of included packages (*chroot_local-packages*) and reference in the config file (*chroot*) of the Debian-live.

Also we have to personalize the packages include on the CD because UltraGrid and Sage has their own dependencies. We copy the packages to the chroot directory

When all are done, we insert create live CD files with the command chroot. We access to the future live CD and add scripts are made for improve software performance. Also we compile Ultragrid and SAGE

Finally, we create the live CD with *lh_binary.* More detailed explanation is the ANNEX.

3.8.2 Linux-live

Linux Live [26] is a set of scripts that permit to create a live CD from a current installed system. It can download from www.linux-live.org. Only is necessary to

apply two patches to the kernels for making bootable from cdrom drive. These patches are for Squashfs and Aufs filesystems.

Squashfs is intended for general read-only file system use, for archival use and in constrained block device/memory systems where low overhead is needed. A union mount is a mount that allows several filesystems to be mounted at one time, appearing to be one filesystem. Both filesystems are used for providing a read-write environment for live Linux distributions. This takes advantage of both the SquashFS's high speed compression abilities with the ability to drastically alter the distribution while running it off of a cd.

When the kernel is patched, you have to personalize the live CD name and start the script. It takes some time, and when it finished it is necessary to run another script for making the iso file.

3.8.2.1 *Problems during Linux-live creation*

We encounter some problems with parching the openSUSE kernel due it has a lot of patches. All main distribution applies their patches to kernel for trying to improve. One of these patches affects file systems and the code provided by aufs repository is not compatible with this source and it has errors during the compilation step. In spite of this fact, we change the kernel used for standard kernel and we do not have more problem.

3.8.3 Qemu

Qemu [27] is software which permits to try the live CD made in the above point without burning the CD. We use Qemu for trying all the version of our live CD. It is a very functional tool for virtualizate CDs and permit to try in different platform, such 32 bits or 64 bits architecture.

3.8.4 Live CD Results

A useful live CD is obtained with Debian Live. This live CD has the minimal programs necessary to run UltraGrid and SAGE. Also has a script that ask some question to the user for configure in a fast way the equipment for being capable to run V3 software.

Real-Time kernel for making the stream transmit more stable is included. It is made in 64 bit architecture. This kernel is correctly patched with unionfs and squashfs. If these patches are not applied, the live CD will not run.

The behavior of UltraGrid in Live CD is the same than a normal system – installed on hard disk. This is because, there are not access to CD-ROM due all

the programs used are loaded into RAM memory. The only difference is the time of booting the system is a little slower.

Also it is tested in dummy clients of SAGE display with good results.

CHAPTER 4. TUNING THE APLICATION

This chapter describes the actions done with the application SAGE for improving its performance. Basically, we modify some parameters at compiling time, such compiler or some flags. Also we try profiling tools that helps to know where the application spends their execution time.

4.1 Compilers

A compiler is a program that transforms a code, which is easy to understand for humans, such as c or c++ language, to a machine code, that can understand a binary processor, such as 0s and 1s.

We study the viability of compiling SAGE – the application bottle neck – with different compilers.

4.1.1 GCC

The default compiler on Linux systems is gcc [28] . This compiler is generic and is not focus on performance. GCC compiler is open source and a lot of people and companies collaborate to develop it. It has lot of flags for improving performance.

First, we try the gcc that come with Debian distribution and second we try with OpenSUSE. OpenSUSE gcc version has a newer version and we obtain better results with it.

4.1.2 Intel® C++ Compiler (ICC)

Intel also has its own compiler that it is optimized for its processors, called Intel C++ Compiler [29] . Although is designed for Intel processors, compatible CPU such AMD processor can work with this compilers and improve its speed and performance.

We try the non-commercial version that can download freely from Intel website. If the results of this compilation will sell this license will not be valid and it should have to purchase a commercial version.

4.1.3 Test conclusions

We compile SAGE on the dummy clients with gcc and Intel® C++ Compiler and compare the values of frame rate and bandwidth displayed by SAGE UI. All tests are explained at *Annex C TESTS RESULTS*. Here there are the conclusions.

Both compilers have general optimization options [30] [31] . These options are generic for all the processors and make faster optimize the binary. If we not enable these options the performance of the binary is reduce.

We compare both compilers on openSUSE 10.3 AMD64 and Debian 4 etch i386. First tests are made with Debian because it was the first operation system used.

In Debian, when both compilers are tested with the same flags, Intel C++ Compiler makes binary faster, more or less, a 9%. When its best options for optimization are tested icc gain around a 16 %. The rice of performance that we observe depends on the code of application and the hardware architecture but is really significant in our case.

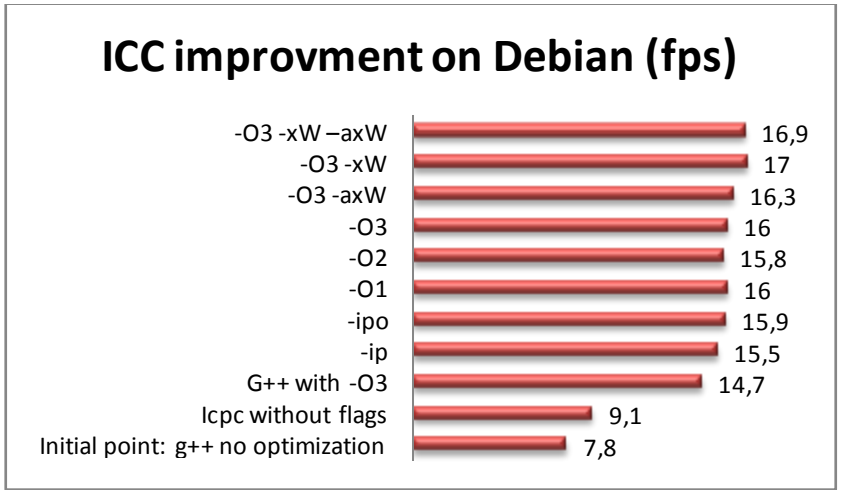


Fig 4.1 ICC Flags improvements

In Debian, with the equipment of the laboratory, the most successful gain is for specific instructions for processor flag. This it is because without specificity the processor family, the compiler creates a binary compatible with processors of current architecture. Compiler makes the binary compatible with older processors. However, if we reduce the compatibility to the current processor and more advancing, instructions used can be more specifically and more efficient.

In OpenSUSE, when the maximum flags of optimizations are applied, the difference between both is the gain of one frame, in other way gains around a

five percent. In this case, we obtain the maximum gain with O3 flag, which one is for automatic optimizations options. The difference between both compilers is not too much and don't justify the investment of buying the commercial license of Intel C++ Compiler.

Better results are obtainted in OpenSUSE because a newer version of gcc is used and we use a 64 bits based operation system.

4.1.4 Other compilers

According to AMD website [32] , there are two more compilers compatible with AMD processors. These are PGI Compiler [33] and Pathscale Compiler [34] . These compilers is supposed to be faster and have better optimizations for AMD processors than Intel C++ Compilers

We try to test both compilers in our laboratory but we have some problems with testing. First problem is these compilers are trial version that expired. PGI expires in 15 days. Pathscale expired in one month but it is necessary to be approving the trial license by Pathscale people. In the case of PGI, when we had the trial version installed, we stay some days with other issues and when we finally compile SAGE, trial period had expired. In the case of Pathscale, we weren't had the opportunity of obtain a working trial version. Pathscale people don't respond to our request.

Other point is that this software is not cheap, and it is not open source. This point is very important because V3 project has the goal of implement an open source platform for video conference system with the loss cost as possible. For these reason we don't insist more.

4.2 Profiling

Profiling [35] is a technique that allows views where a program spends its time and which functions are called while it was executing. This information can show where the bottle neck of the program is and modify the code to improve its performance.

There are different program that allow profiling in a Linux system, but we try OProfile for its easy use and complete information displayed.

4.2.1 OProfile

OProfile [36] consists of a loadable kernel module and a system daemon process that collects sample data from a running system. OProfile uses hardware counters to collect profiling information. It uses a set of CPU registers

that can count events such instructions executed or cache misses. It also has several post-profiling tools for taking the data and turning it into useful information about the system.

One of the most important advantages of OProfile is that is unobtrusive. It means that it is not necessary to recompile the source with specific flags. Also all the code running on the system is profiled. It offers different tools for show the results. Other important advantage, which decide us to use it, is application does not have to end for taking samples. Other alternatives, such gprof, need a correct execution of the binary for collect data.

All the detail of test and captures are on the ANNEX.

4.2.2 Gprof

Gprof [37] is the profiling tool that comes with gcc.Gprof is the standard UNIX profiling tool. The program to be analyzed requires a special compilation for making the profiling. The compiler inserts timing probes in every function for knowing how many times are executed. This information is save when the program finished its execution to a file. This file is analyzed for gprof and, with the probes inserted bellow and with results, it establishes the relation between functions, called call graph. Gprof also calculates the amount of time spent in each function.

4.2.3 Vtune

Vtune [38] is the profile tool developed by Intel. It is a commercial product. The way it work is similar to OProfile because insert a kernel modules which take samples and it is not necessary to recompile the program. But Vtune installation is more heavy installation and it consumes a lot of resources. Vtune has a graphical user interface that make easier to observe the samples and the results.

4.2.4 AMD CodeAnalyst

AMD website describes AMD CodeAnalyst [39] as:

AMD CodeAnalyst Performance Analyzer for Linux is an open source, front-end graphical user interface to Oprofile. The graphical user interface simplifies the profile configuration and enhances the post profile data analysis. Four performance counters can be configured from a large set of processor performance events for simultaneous profiling. The profiling data is provided in both tabular and chart formats through hierarchical drill down tabs.

4.2.5 Profile Guided Optimization

Some compilers can make an *automatic profiling* improvement. First of all the program is compiled with a specific flag (depends on compiler). Then the program is running as usual. During this execution some profiling file are created. Finally the program is compiled again with a different flag. Compiler make optimizations according with the information collected at second step. We try profile guided optimization with icc and gcc but the improvement results are not visible.

4.2.6 Results of profiling

SAGE was profiled when receive an UltraGrid stream, and we found that around 60 % of the time in function for converting to RBG to YUV, in other way, in mathematical operations.

There are some alternatives to try to improve this bottle neck.

- Translate this operation to the graphic card between with OpenGL functions. OpenGL is the library that controls graphic cards. If we translate this operation to the library, the graphic operation should be more optimized and therefore will be faster.

- Try to parallelize the operation between both dummy clients processors. We observe that SAGE don't use the totally of processor. This conversion operation is done a lot of times. If we parallel simultaneous conversions, due it is done for every pixel, we will gain time and speed. This parallelization can be done with openMP [40] .

- Make this conversion at UltraGrid RX, due that is a more power system. Although the conversion will be faster, we will required more network bandwidth and the current display size will not be viable in gigabit Ethernet due it will be necessary 1,5 Gbps.

4.3 CUDA

CUDA ("Compute Unified Device Architecture") [41] is a technology that allows a programmer to use the C programming language to code algorithms for execution on the graphics processing unit (GPU). This is developed by NVIDIA and it is include since GeForce 8 series graphics cards from NVIDIA. GPU is the processor of graphic cards and is optimized for mathematical and graphical operations. Usually GPU was only accessible from graphics card drivers and the code was specific for each card. With CUDA, a generalist code is used – C language – and GPU can be used as desired, for not specific graphic operations, especially for mathematical calculations.

CUDA can be a future step for improving SAGE in systems that have newer NVIDIA graphic cards.

CHAPTER 5. CONCLUSIONS AND NEXT STEPS

In this chapter is described the conclusions of the current TFC, the environment impact of activities done in it and future steps for improving the current state of V3 project.

5.1 Achieved objectives

We study platform performance and the way of how evaluate the system. We determinate that bottle neck are dummy clients, basically in their processor, network interface and graphical interface. We have some problems with network and graphic because they use the same chipset and when both work their performance is reduced.

We try different compilations for making the application faster. Also we try different flags. More specific compilation makes binary faster. Also flags for optimization make huge rice of speed. We can archive a gain of more than 110% between no flags of optimization and a good combination of these flags. Intel C++ Compilers make faster binaries but in 64 bits architecture the gains are not too important for making the investment of buying its commercial license. Also profiling tools, such Oprofile, can help to determinate in the code where the bottle neck is.

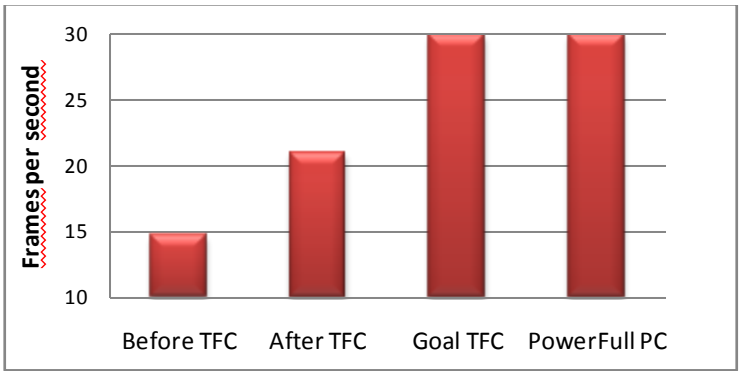


Fig 5.1 Frame rate improvements

Related of the operation system, we obtain better results with openSUSE of 64 bits, rather than Debian of 64 bits or 32 bits. Performance is very depended of Linux kernel. OpenSuse kernel has lot of patched developed by companies behind it, such IBM, Novell or AMD. Due this, it has better performance. The performance in global system is increased around 40 % since this TFC is started.

Also the test of Linux from Scratch (LFS) is not successful because has the same performance as Debian 4 of 32 bits. The results of LFS are not the desired but, in other way, it determinate that the most important components that use SAGE is kernel and their code.

Also we evaluate real-time (RT) in Linux. RT gives a more stable frame rate to displays. But when the system is overloaded by real-time application, RT can produce negative effects such reducing performance or making the system uneatable. In dummy clients do not give a visible gain. But in the system that sends the stream to the display, RT gives more frame rate stream stability due its constant rate.

The current V3 hardware combination is not enough for displaying at desired resolution in distributed display. It is necessary to reduce expectation or buy powerful equipment. If we buy some equipment with the same price that we have, we will have enough power. It is important that graphic card has an own controller chip. Also if network Ethernet device has the ability of manage jumbo frames packets it will be better. We have tested a newer PC with the expected results: a frame rate of 30 frames per second.

Also a Live CD is made. Live CD system is similar to a normal system but some differences. The most important is that use a non-writable medium and need some other filesystems for manipulate system data. Also there are scripts that can help of making a live CD. Creating a bootable CD often is not an easy task. We have to go through some trial and error before we get things set up just the way we want. Finally, we obtain a functionally live CD. This live CD permit to run the software of V3 project with the improvements proposed in this TFC. It can be used by inexpert user. Also, it can be used as plug and play operation system.

5.2 Environment impact

Nowadays, in a globalized world, working groups are distributed around the world. Several companies are geographical disperse and managers travel for meeting on headquarters. Also research group work in a cooperative way with other international institutions and need to make face to face meeting. Also in telemedicine, doctors can save to travel huge distance for evaluate their patients, especially in medical specialists. These are few examples where high definition video conference can be useful to save time, money and reduce pollution related on people transport.

Negative impact may be needs of high amount of bandwidth that are needed for making done the videoconference, needing the construction of new lines of optical fiber, satellite communication and aerial constructions.

In distributed display, working with an ordinary PCs, it’s a positive environment impact because these equipment can be used in other task when the distributed display is no required.

5.3 Future steps

There are some steps than can be interesting for improving the system but that are not done because lack of time.

First possible action is trying other Linux system that is made based on compiling all its components. Its name is Gentoo and has a big community behind it. It is ported to different architectures and is based on optimization the system. It has a lot of documentation for installing it. But in the other hand is slow to install due to its flexibility and shaped to the platform.

Second possible action is trying to parallel SAGE execution. When we monitor SAGE execution we observe that not all the processors are used at full capacity when system are stressed and overloaded. Although the frame rate does not give the maximum expected about it. This can be done with openMP in one system or openMPI in a distributed system.

Third possible action is trying to adapt SAGE to CUDA technology. CUDA is a technology developed by NVIDIA that is used for use GPU (Graphic card's processor) for general purpose programs, especially programs that need a lot of CPU and can be parallel. This technology is applied since 8000 series of NVIDIA graphic cards.

Finally, and obvious, buying new equipment with higher features can help to improve the performance.

REFERENCES

[1] . **Vega D'Aurelio, Davide.** *Programació i avaluació de la capa de transport per al sistema de visualització distribuïda SAGE.* Vilanova i la Geltrú : Treball Final de Carrera UPC, 2008. Vol. Treball Final de Carrera.

[2] . **Wikipedia.** RGB. [Online] [Cited: March 23, 2008.] http://en.wikipedia.org/wiki/RGB.

[3] . **Wikipedia.** YUV. [Online] [Cited: March 23, 2008.] http://en.wikipedia.org/wiki/YUV.

[4] . Fundació i2cat. [Online] www.i2cat.net.

[5] . **i2CAT.** HD Wiki. [Online] [Cited: June 2008, 1.] hdwiki.i2cat.net.

[6] . **EVL.** SAGE. [Online] [Cited: February 27, 2008.] http://www.evl.uic.edu/cavern/sage/index.php.

[7] . Ultragrid. [Online] [Cited: March 2, 2008.] http://ultragrid.east.isi.edu/.

[8] . **Jones, M. Tim.** Anatomy of the Linux kernel. [Online] IBM, June 6, 2007. [Cited: May 12, 2008.] http://www.ibm.com/developerworks/linux/library/l-linux-kernel/.

[9] . **Jones, Tim.** Inside the Linux scheduler. [Online] June 30, 2006. [Cited: May 3, 2008.] http://www.ibm.com/developerworks/linux/library/l-scheduler/.

[10] . **Wikipedia.** Performance Tuning. [Online] http://en.wikipedia.org/wiki/Performance_tuning.

[11] . **Fink, Jason and Sherer, Matthew.** *Ajuste del rendimiento y planificacion de la capacidad con Linux.* Madrid : Prentice Hall, 2002.

[12] . Debian. El sistema operativo universal. [Online] March 4, 2008. http://www.debian.org/.

[13] . **Shuttle.** Shuttle Model used in test. [Online] [Cited: April 3, 2008.] Shuttle Barebone used for SAGE tilde display. http://eu.shuttle.com/es/desktopdefault.aspx/searchcall-12/searchcategory-256/noblendout-1/tabid-72/170_read-14215/.

[14] . Iperf. [Online] [Cited: June 1, 2008.] http://sourceforge.net/projects/iperf.

[15] . **Godard, Sebastien.** SYSSTAT. [Online] [Cited: March 3, 2008.] http://pagesperso-orange.fr/sebastien.godard/.

[16] . kSar : a sar grapher. [Online] [Cited: March 2, 2008.] http://ksar.atomique.net/.

[17] . **Balsa, André D.** Linux Benchmarking HOWTO. [Online] August 1997. [Cited: May 2, 2008.] http://www.faqs.org/docs/Linux-HOWTO/Benchmarking-HOWTO.html.

[18] . LMbench - Tools for Performance Analysis. [Online] [Cited: May 3, 2008.] http://www.bitmover.com/lmbench/.

[19] . Linux Test Project. [Online] [Cited: April 4, 2008.] http://ltp.sourceforge.net/.

[20] . Kernel.org. [Online] [Cited: May 2008, 3.] kernel.org.

[21] . Real Time Project. [Online] [Cited: March 23, 2008.] rt.wiki.kernel.org.

[22] . **Novell.** openSUSE. [Online] www.opensuse.org.

[23] . Linux From Scratch. [Online] [Cited: March 29, 2008.] http://www.linuxfromscratch.org.

[24] . **Linux Foundation.** Net:NAPI. [Online] [Cited: April 24, 2008.] http://www.linux-foundation.org/en/Net:NAPI#NAPI_API.

[25] . Debian Live wiki. [Online] [Cited: April 30, 2008.] http://wiki.debian.org/DebianLive/.

[26] . **M, Tomas.** Linux Live for CD & USB. [Online] 2008. [Cited: June 1, 2008.] www.linux-live.org.

[27] . **Bellard, Fabrice.** QEMU - Processor Emulator. [Online] 2008. [Cited: June 10, 2008.] www.qemu.org.

[28] . GCC, the GNU Compiler Collection. [Online] [Cited: June 12, 2008.] gcc.gnu.org/.

[29] . **Intel Corporation.** *Intel ® C++ Compiler 10.1, Professional and Standard Editions, for Linux.*

[30] . **Intel Corporation.** *Intel® C++ Optimizing Applications.* 315890-002US.

[31] . **Intel Corporation.** *Quick-Reference Guide to Optimization with Intel® Compilers version 10.x For IA-32 processors, Intel® 64¹ processors, and IA-64² processors.*

[32] . **AMD.** Software Optimization Guide for AMD Athlon 64. [Online] [Cited: May 4, 2008.] http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/25112.PDF.

[33] . **STMicroelectronics.** http://www.pgroup.com/. [Online] [Cited: May 15, 2008.] http://www.pgroup.com/.

[34] . **PathScale.** PathScale Compiler Suite . [Online] [Cited: May 2, 2008.] http://www.pathscale.com/.

[35] . Profilers. [Online] AIT LINUX Support. [Cited: April 12, 2008.] http://ait.web.psi.ch/services/linux/hpc/hpc_user_cookbook/tools/profilers/index.html.

[36] . **R, Darren.** Profile -- Linux Profiling Tool. [Online] Novell Cool Solutions. [Cited: April 23, 2008.] http://www.novell.com/coolsolutions/feature/11789.html.

[37] . GNU gprof. [Online] http://www.cs.utah.edu/dept/old/texinfo/as/gprof_toc.html.

[38] . **Intel Corporation.** Intel Vtune Perfomance Analyser. [Online] [Cited: May 3, 2008.] http://www.intel.com/cd/software/products/asmo-na/eng/239144.htm.

[39] . **AMD.** AMD CodeAnalyst™ Performance Analyzer for Linux®. [Online] http://developer.amd.com/cpu/codeanalyst/codeanalystlinux/Pages/default.aspx.

[40] . OpenMP. [Online] [Cited: June 5, 2008.] http://openmp.org/wp/.

[41] . **NVIDIA Corporation.** Cuda Zone. [Online] 2008. [Cited: May 25, 2008.]

[42] . **Shakshober, John.** Choosing an I/O Scheduler for Red Hat® Enterprise Linux® 4 and the 2.6 Kernel. [Online] June 2005. [Cited: April 12, 2008.] http://www.redhat.com/magazine/008jun05/features/schedulers/.

[43] . **Ciliendo, Eduardo and Kunimasa, Takechika.** Linux Performance and Tuning Guidelines. [Online] IBM Red Books, 2007. [Cited: March 2, 2008.] http://www.redbooks.ibm.com/redpapers/pdfs/redp4285.pdf.

ANNEX A. V3 HARDWARE SPECIFICATIONS

This annex describes the hardware used during the tests.

In the first table there is the specification of the dummy clients. When they are bought (July 2007), they cost around 450 euros.

Table A.1. Dummy client specifications

Manufacturer	Shuttle
Processor	AMD Athlon(tm) 64 X2 Dual Core Processor 4000
Memory	DDR2
Video Card	NVIDIA Geforce 7025 (onboard)
Network device	Marvell 88E1116 Ethernet network controller (Gigabit PNY)
Hard Disk	SATA 80 GB
Monitor	NEC LCD2070VX (20 inch)

For getting complete information of CPU we use the following command:

cat /proc/cpuinfo

The result in our system is:
processor : 0
vendor_id : AuthenticAMD
cpu family : 15
model : 107
model name : AMD Athlon(tm) 64 X2 Dual Core Processor 4000+
stepping : 1
cpu MHz : 2100.330
cache size : 512 KB
(…)
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt rdtscp lm 3dnowext 3dnow pni cx16 lahf_lm cmp_legacy svm extapic cr8_legacy misalignsse ts fid vid ttp tm stc 100mhzsteps
bogomips : 4204.56
clflush size : 64.

We have to observe what kind of SSE have our cpu. In this case Athlon XP has SSE and SSE2.

Table A.1 UltraGrid transmitter and receiver.

Manufacturer	SUPERMICRO
Processor	Intel Xeon Dual Core Intel Xeon Dual Core
Memory	DDR2
Video Card	NVIDIA Geforce 7950 GT
Network device	Intel PRO 10Gb/SR
Hard Disk	SATA 500 GB

Table A.2 Other equipment

Camera	SONY HDR - FX1 (HD camera)
Switch	SMC 8824 (24 ports gigaEthernet 2 port 10 Gigabit Ethernet)

ANNEX B. COMPILING A LINUX KERNEL

Compiling a kernel is not an easy task for inexperience user but with some recommendations it can be simply.

Linux Kernel as said in Chapter one is the core of a Linux system. If it does not work properly, the system notices it.

First of all we need to download the sources. There are different ways of obtain the sources. Almost all Linux Distribution, such Debian or OpenSUSE, has its own patched kernel. Also, there is a main official kernel resource repository in *kernel.org*. We will use the official kernel of kernel.org and the explanation is made in a Debian system but can apply in all distributions.

When sources are downloaded from your repository, you should copy to /usr/src and untar in it.

cp linux-2.6.23.1.tar.gz /usr/src
cd /usr/src
tar xvf linux-2.6.23.1.tar.gz

The commands above create the file directory where there are all the sources of Linux kernel. In Debian, we will need *kernel-package* for getting the tools for create the kernel, such a compiler; also is necessary ncurses-dev for graphic configuration.

apt-get install kernel-package ncurses-dev

If we want to apply some patch, now is the moment, before continuing with kernel configuration. A patch is a modification of the source for give it more features or for improving the code. In our case we will apply real-time patch provided by rt.wiki.kernel.org. We download the patch at /usr/src. This part is not necessary if we do not want to apply a patch.

cd linux-2.6.23.1
patch –p1 < ../patch-rt-2.6.23.1

Now is time for configure the kernel. There are some methods but I recommend with graphical interface provided for ncurses (package libncurses5-dev). Before enter to the configuration, we can load the previous configuration with *make oldconfig* or a default configuration with *make defconfig.*

Make menuconfig

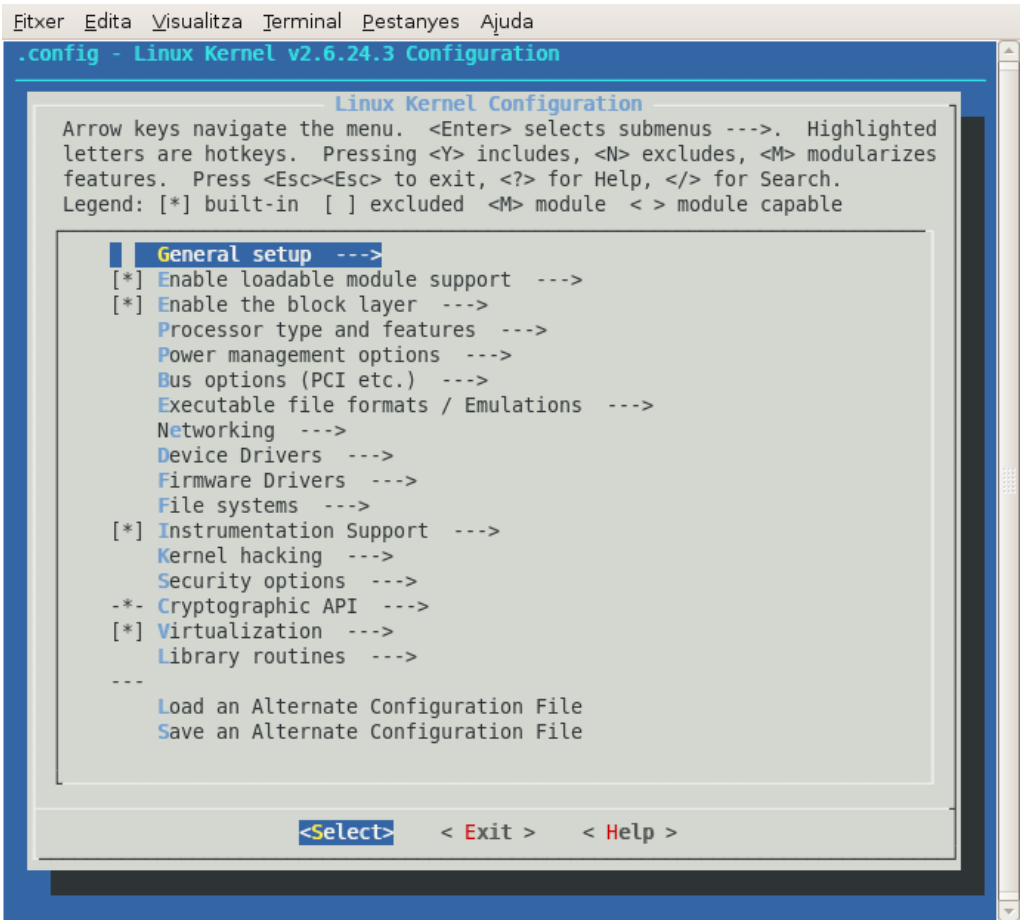


Figure B.1 Main menu of kernel configuration

Kernel is divided into different subsystems. Now we describe main of them:

- General setup: in this items it is selected the name of the kernel image, and some options that involves all the building. One of them is *optimized for size*. This should be unselected if you have sufficient space.
- Enable loadable module support: this item includes some options for kernel modules.
- Enable the block layer: this include options of block devices.
- Processor type and features: this item includes all the parameters related to processors, such family, real-time options and more.
- Power management options: this item include all options related of power management.
- Bus options: here you can configure the buses used for the system.

- Networking: in this item there are all the options related to network, such network protocols (IP, TCP), wireless options and more.
- Device Drivers: this item includes all the controllers and drivers needed for managing physical devices.

- File systems: this items include all the file systems supported by the system, such ext3 or fat32

- Kernel hacking: some options for debugging the kernel

- Cryptographic API: chippers algorithms include in the kernel

Once the kernel is properly configured you should compile with:

make
make modules_install
make install

Finally, edit the file /boot/grub/menu.lst and add the kernel entry with the name disposed in make install

Also it is possible to compile and install Linux Kernel in a Debian way with following commands:

make-kpkg –initrd –revision=1 binary modules

dpkg –i ../linux-image-2.6.23.1-rt11_1_amd64.deb

ANNEX C. MAKE A LFS SYSTEM

In this annex is explained the LFS book guided used for developing a Linux From Scratch System.

C.1 LFS Book

We will use a tool called jhalfs for create a LFS system in automatic way.

First of all we have to create a partition for desired size (10 GB) where install the system.

fdisk /dev/sda

Then we have to create a user with sudo privilege for execute jhalfs.

useradd user
visudo
copy the following line to visudo file.

user ALL=(ALL) ALL

Then execute make with this user inside the directory of jhalfs 2.3.1, and select the following setting. Be sure you have an Internet connection.

Book settings

- Release (Branch or stable book)
- Book version (6.3)
- [] Add custom tools support
- [*] Add blfs-tool support

General Settings

- [*] Change the default user/group and homedir for this build
- (user) User account
- [*] set group
- (user) GROUP account
- (/mnt/lfs1) Build Directory
- [*] Retrieve source files

Advanced Features

- [*] Use optimization

Answer yes to the entire questions displayed.

The compilation installation it take some hours, if all go well, you can do another thing.

When the main programs are installed you have to compile the desired kernel. For do this, access as chroot and compile and install it as explain as previous annex. In this version we try 2.6.23.1 from vanilla.
For have an access as chroot run the following commands.

```
export LFS=/mnt/pathtolfs
mount -v --bind /dev $LFS/dev
mount -vt devpts devpts $LFS/dev/pts
mount -vt tmpfs shm $LFS/dev/shm
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
chroot "$LFS" /usr/bin/env -i \
HOME=/root TERM="$TERM" PS1='\u:\w\$ ' \
   PATH=/bin:/usr/bin:/sbin:/usr/sbin \
   /bin/bash --login
```

Then we have to change the root password for a future access:

```
passwd
```

And type you password twice.

Then you have to edit the following file

- /etc/fstab: file where are the partitions which are mounted on start up
- /etc/hosts: file where are the name of the system
- /etc/sysconfig/network-device/ifconfig.eth0/ipv4: file where we can configure the IP address statically.

C.2 BLFS Book

LFS is only the base system. The system we mounted has other programs for make the system more useful. For this propose exists BLFS another book of LFS project where it is explained how configure a useful system. This book, also have an automatic scripts for compiling one per one the tools selected.

We install the BLFS package following the next order:

- Nettools: tools for controlling the network such ifconfig
- Dhcpd: tool for obtain an IP address from a DHCP server
- Xorg: the X system.
- Sdl: a useful graphical library needed for SAGE
- Freeglut: another graphical library for SAGE

- Openssh: A tool for secure access remote with SSH protocol. We have to enable root access modify configuration file /etc/ssh/sshd_config adding the following line or modifying it. PermitRootLogin yes
- Python: an interpreter of this language programmer code.
- Mesalib: graphical libraries

And finally, we install manually the following packages:

- readline
- Quanta
- sage installation without subprojects because only need sage core.

And because we use an NVIDIA graphics card we install the driver provided for NVIDIA.

C.3 LFS tests

We try the LFS system with gcc compiler and we obtain the following results.

Table C.1. LFS tests results.

Compiler Flags	Frame rate (fps)	Bandwidth (Mbps)
-03	14,6	471
-funroll-loops	14,6	472
-march=athlon64	15,6	504,6
-msse, -msse2, -msse3, -mmmx, -m3dnow	14,8	477
-msse, -msse2, -msse3, -mmmx, -m3dnow -march=athlon64	15,7	505.7

We obtain results very similar to Debian 4 etch of 32 bits.

ANNEX D. USER GUIDE FOR MAKING A LIVE CD

In this annex is explained how to create a Live CD with Debian Live and Linux Live scripts.

D.1 Debian based live CD

This point explains how create a Debina Live CD for HD platform of V·

D.1.1 Preparation

Creating a live CD with Debian-live in a Debian distribution needs some previous steps, like install necessary software.

Put this in your /etc/apt/sources.list:

```
deb http://www.backports.org/debian etch-backports main
```

Update the package indices...

```
apt-get update
```

...and install needed packages:

```
apt-get install -t etch-backports cdebootstrap
apt-get install -t unstable live-helper
```

You should have Internet connection to proceed the creation because is necessary to download a lot of packages.

D.1.2 Preparing customize kernel

First it is necessary to path the desired kernel with UnionFs and SquashFs.

Once it it patched, configure the kernel with the patched installed above.
Also it is necessary in General setup-- > Local version –apend to kernel release entry *–amd64* for the correct work of the DebianLive scripts.

Then generate deb file:

```
make-kpkg --initrd --revision=1 binary modules
```

This may take a while. When it is finished copy the file generate to config/chroot_local-packages of Live CD directory.

```
cp ../linux-image-2.6.23.1-rt11-amd64_amd64.deb [debian_live]/config/chroot_local-packages
cp ../linux-headers-2.6.23.1-rt11-amd64_amd64.deb [debian_live]/config/chroot_local-packages
cp ../linux-doc-2.6.23.1-rt11-amd64_all.deb [debian_live]/config/chroot_local-packages
cp ../linux-manual-2.6.23.1-rt11-amd64_all.deb [debian_live]/config/chroot_local-packages
cp ../linux-source-2.6.23.1-rt11-amd64_all.deb [debian_live]/config/chroot_local-packages
```

D.1.3 Initializing live CD

When the program is installed we initialize the tree directory of Debian live CD.

```
lh_config
```

Edit the file config/chroot and add the list of package to install as following:

```
LH_LINUX_PACKAGES="linux-image-2.6.23.1-rt11-amd64_amd64

LH_LANGUAGE="es"

LH_PACKAGES=libsdl1.2-dev g++ gcc make libmad0-dev libavcodec-dev libpostproc-dev libmpeg2-4-dev libwxgtk2.6-dev libavformat-dev libdvbpsi4-dev libreadline5-dev libfreetype6-dev libjpeg62-dev libpng12-dev freeglut3-dev libxmu-dev libxi-dev xserver-xorg xfce4 ffmpeg libavcodec-dev libmpeg2-4-dev automake1.9 libtool gettext cvs libpostproc-dev libavformat-dev libwxbase2.6-dev xfonts-100dpi libmagick9-dev libglew-dev libxv-dev python python-numarray python-wxgtk2.6 openssh-server openssh-client xfce4 psmisc
```

Also edit the file config/binary and add the following:

```
LH_USERNAME="root"

LH_HOSTNAME="ultragrid"
```

When the file is edited, type:

```
lh_bootstrap
lh_chroot
```

D.1.4 Personalize live CD

When the directories are created we have to enter as chroot into directory for personalize cd. If you do not want to personalize skip this point.

```
lh_chroot_hosts install
lh_chroot_resolv install
lh_chroot_proc install
```

Copy the needed files for installing SAGE and UltraGrid to live CD tree directory.

```
cp uv-0.3.9.3-tar.gz chroot/root/uv-0.3.9.3
cp sage2.tar.gz chroot/usr/share/sage2
cp vlc-1-2.tar.gz chroot/usr/share/
cp QUANTA-0.4.tar.gz chroot/usr/share/

chroot chroot
```

Now we are into the future live CD. Now is the time to customize cd.

Add a customized root password

```
passwd
```

And type twice the password desired.

Customize network parameters:

```
cat > /etc/sysctl.conf  << "EOF"

net.core.rmem_max=8388608
net.core.wmem_max=8388608
net.core.rmem_default=8388608
net.core.wmem_default=8388608
net.core.optmem_max=8388608
net.core.netdev_max_backlog=300000
net.ipv4.tcp_congestion_control="highspeed"

EOF
```

Customize environment variables:

```
cat > /root/.bashrc  << "EOF"

export SAGE_DIRECTORY=/usr/share/sage2
export PATH=$SAGE_DIRECTORY/bin:$PATH
export
LD_LIBRARY_PATH=$SAGE_DIRECTORY/lib:/usr/local/lib64:$LD_LIBRARY_
PATH

EOF
```

Now we install QUANTA, a dependency of SAGE provided by EVL and necessary for synchronization.

```
cd /usr/share
mkdir quanta
mv quanta-0.4.tar.gz quanta/
cd quanta
tar xvf quanta-0.4.tar.gz
cd quanta-0.4
./configure CXXFLAGS=-fpermissive
make
make install
```

Now we install SAGE:

```
cd /usr/share
tar xvf sage2.tar.gz
cd sage2
make
make install
```

Generate ssh keys for accessing without password into SAGE:

```
mkdir ~/.ssh
chmod 700 ~/.ssh
ssh-keygen -q -f ~/.ssh/id_rsa -t rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

Also edit /etc/ssh/ssh_config and replace :

```
HashKnowHost yes
```

For:

```
HashKnowHost no
```

We can also install vlc with SAGE plug-in.

```
cd /usr/share
tar xvf vlc-1-2.tar.gz
cd vlc-0.8.5
./configure
make
make install
```

And finally, we install UltraGrid i2CAT version

Uncompress Ultragrid

```
cd /root
tarxvf uv-0.3.9.3.tar.gz
cd uv-0.3.9.3
```

Some modified UltraGrid dependencies also are needed to install:

```
cd /root/uv-0.3.9.3/tcl-8.0/unix
./configure
make
cd /root/uv-0.3.9.3/tk-8.0/unix
./configure
make

cd /root/uv-0.3.9.3/UltraGrid
./configure --enable-sdl
make
```

Personalize IP address:

```
cat > /etc/network/interfaces  << "EOF"

auto lo
iface lo inet loopback

iface eth1 net static
address 192.168.50.21
netmask 255.255.255.0
mtu 9180
auto eth0

EOF
```

Also we can personalize welcome message of live CD in /etc/issue:

Finally, we exit from chroot environment.

```
exit
```

If you have a NVIDIA graphic card, copy the module from host system to live CD system. Also it is necessary to copy NVDIA libraries. The host has to be run the same kernel as live CD. Download the NVIDIA driver from website (http://www.nvidia.com/object/unix.html) and install it to the system.

```
sh NVIDIA-Linux-x86_64-173.14.09-pkg2.run
```

```
cd /
tar cvf nvidia.tar  \
/usr/bin/nvidia-installer \
```

```
/usr/bin/nvidia-settings \
/usr/bin/nvidia-xconfig \
/usr/lib/directfb-0.9.25/gfxdrivers/libdirectfb_nvidia.a \
/usr/lib/directfb-0.9.25/gfxdrivers/libdirectfb_nvidia.la  \
/usr/lib/directfb-0.9.25/gfxdrivers/libdirectfb_nvidia.so  \
/usr/lib/libnvidia-cfg.so \
/usr/lib/libnvidia-cfg.so.1 \
/usr/lib/libnvidia-cfg.so.173.14.05 \
/usr/lib/libnvidia-tls.so.1 \
/usr/lib/libnvidia-tls.so.173.14.05 \
/usr/lib/tls/libnvidia-tls.so.1 \
/usr/lib/tls/libnvidia-tls.so.173.14.05 \
/usr/lib/xorg/modules/drivers/nvidia_drv.so \
/usr/lib/xorg/modules/libnvidia-wfb.so.1 \
/usr/lib/xorg/modules/libnvidia-wfb.so.173.14.05 \
/usr/src/linux-headers-2.6.23.1-rt11-amd64/drivers/video/nvidia \
/usr/src/linux-headers-2.6.23.1-rt11-amd64/drivers/video/nvidia/Makefile \
/usr/lib/libcuda.so.173.14.05 \
/usr/lib/libGLcore.so.173.14.05 \
/usr/lib/libGL.so.173.14.05 \
/usr/lib/libnvidia-cfg.so.173.14.05 \
/usr/lib/libnvidia-tls.so.173.14.05 \
/usr/lib/libXvMCNVIDIA.so.173.14.05 \
/usr/lib/tls/libnvidia-tls.so.173.14.05 \
/usr/lib/xorg/modules/extensions/libglx.so.173.14.05 \
/usr/lib/xorg/modules/libnvidia-wfb.so.173.14.05 \
/usr/lib/libcuda.so \
/usr/lib/libcuda.so.1 \
/usr/lib/libGL.la \
/usr/lib/libGL.so \
/usr/lib/libGL.so.1 \
/usr/lib/libGLcore.so.1 \
/usr/lib/libXvMCNVIDIA.a \
/usr/lib/libXvMCNVIDIA_dynamic.so \
/usr/lib/libXvMCNVIDIA_dynamic.so.1 \
/usr/lib/libXvMCNVIDIA.so.173.14.05 \
/usr/lib/xorg/modules/extensions/libglx.so \
```

Then go chroot/ and untar:

```
cp nvidia.tar  [debianlive]/chroot
cd chroot
tar xvf nvidia.tar
```

Also we can customize first image of live CD, also called splash screen:

Create your image with gimp, 640x400. Convert the image to 14 indexed colors (Image > Mode > Indexed). Save as ppm.

For LH_BOOTLOADER="syslinux"

```
$ ppmtolss16 '#d0d0d0=7' < splash.ppm > splash.rle
$ cp splash.rle config/binary_syslinux/splash.rle
```

Also set LH_SYSLINUX_SPLASH="config/binary_syslinux/splash.rle"

Remove links that permit work such independent chroot system.

```
lh_chroot_hosts remove
lh_chroot_resolv remove
lh_chroot_proc remove
```

D.1.5 Making the iso image and testing

And finally we create the iso image. This may take a while.

```
lh_binary
```

And the last step is burning the iso image to a CD or DVD and tries in the desired pc. Also we can try in QEMU with the command:

```
qemu-system-x86_64  -cdrom binary.iso
```

If we want to modify some step, we have to enter the following command

```
lh_clean binary
```

And reenter to chroot following the steps since preparing to enter to chroot.

D.2 OpenSUSE based live CD

Download linux-live scripts from www.linux-live.org

Untar into tmp directory

```
cp linux-live-6.2.4.tgz /tmp
cd /tmp
tar xvf linux-live-6.2.4.tgz
```

Apply Aufs patch. First you should download from cvs repository:

```
mkdir aufs.wcvs
```

```
cd aufs.wcvs
cvs -d:pserver:anonymous@aufs.cvs.sourceforge.net:/cvsroot/aufs login
(CVS password is empty)
cvs -z3 -d:pserver:anonymous@aufs.cvs.sourceforge.net:/cvsroot/aufs co aufs
```

In order to update files after the first checkout,

```
cd aufs.wcvs/aufs
cvs update

make -f local.mk kconfig

cp –r fs/aufs /usr/src/linux/fs/
cp include/linux/aufs_type.h /usr/src/linux/include/linux
```

Add the following line to linux/fs/Makefile. This file is in linux kernel source.

```
obj-$(CONFIG_AUFS) +=aufs/
```

Also add the following line at linux/fs/Kconfig file:

```
source “fs/aufs/Kconfig”
```

Apply SquashFs patc. Donwload from http://squashfs.sourceforge.net/

Untar and apply the patch acording to Linux kernel version.

```
tar squashfs3.3.gz

cd /usr/src/linux
patch –p1 < [path squashfs]/kernel-patches/2.6.23.1/squashfs-patch.diff
```

Finally we create the live CD

Edit .config file and change *ramdisk=555* by *ramdisk =512000*

Run:

```
./build
```

Live distribution's 'directory tree' will be created in /tmp/live_data_1234, where 1234 is a random number.

To make ISO image, run *make_iso.sh*

D.3 Auto configure SAGE scripts

The following file is created for making an auto configure execution of live CD for SAGE.

setupLive CD.sh

```
#/bin/bash
#
#Script that automatate SAGE configuration
#Author: Daniel Turull (i2CAT)
#
matrix (){
ROW=$1
COL=$2
j=0
COLS=" "
let "COLS=$COL+1"
DISPLAY=1
while [ $j -lt $ROW ]
do
        ROWS=" "
        i=1
        while [ $i -lt $COLS ]
        do
                ROWS="$ROWS | $DISPLAY "
                let "i+=1"
                let "DISPLAY+=1"
        done
        echo $ROWS
let "j+=1"
done
return 0
}

LOG="SAGEconfig.log"
KEY=" ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAv+/gqKP37M9x9NDypWxFdcbaJ/MsuIczDJ/n6sHl6Zfvm
fiGGLiY5ThQ9zYkiLnnJI5mRTGaayPSf7f3o1deCoo5hshVEVU+/E95MSFaPYGPxmP1aMLy0S
9TPTk4rWFmLLxs0TRxgyHTpmcF3WyLPJ3ueulnfQWKMlD7LuVvJft17H18otXMfO0aDxPjN6U
9qpMW+mOyLMj3vFi26llaolVfH/X6Z7lHeYeo3uQciiLWtUBwtUap1KlStywc8pAb51LusQ8TXnbh
U9KvUCjMsedaFpQgPwEWClHaNKyh1Cfsc4Ut/a0q9r/9RQ9uaBBzuOqwtAJDWdwWa8lKKiUh
kQ== "
echo "SAGE AUTOCONFIGURATION (i2CAT)"
echo ""
echo "Autoconfiguring NVIDIA card"
rm /etc/X11/xorg.conf
nvidia-xconfig > $LOG


echo "How many rows have your display? [default 2]"
read ROW
echo "How many columns have your display? [default 2]"
read COL


#Comprovation if are blank entries
if [ -f $ROW ]; then
```

```
ROW=2
fi
if [ -f $COL  ]; then
COL=2
fi
echo "You have a display of $ROW rows and $COL columns"
echo ""

echo "Entry net address for the display network. [default 192.168.50.0]"
read NET
if [ -f $NET ]; then
NET="192.168.50.0"
fi

echo "Entry net mask for the display network. [default 255.255.255.0]"
read MASK

if [ -f $MASK ]; then
MASK="255.255.255.0"
fi

echo "Your network configuration: "
echo "Network: $NET Mask: $MASK"
echo ""


PS3="Choose (1-2):"
echo "Choose the job of this computer"
select JOB in sail display
do
        break
done

echo "You are the $JOB"
echo ""

echo ""
echo "Interfaces on the system:"
ifconfig -a | grep eth
PS3="Choose interface:"
echo "Which interface want to configure as main?"
select ETH in `ifconfig -a | grep eth | awk '{print $1}'`
do
        break
done
echo "You will use $ETH interface"

################################
#Calculte IP_BASE & IP_START

i1=`echo $NET | sed 's/\./ /g' | awk '{print $1}'`
i2=`echo $NET | sed 's/\./ /g' | awk '{print $2}'`
i3=`echo $NET | sed 's/\./ /g' | awk '{print $3}'`
i4=`echo $NET | sed 's/\./ /g' | awk '{print $4}'`

BASE="$i4"
START="$i1.$i2.$i3."

################################
```

```
if [ $JOB = "sail" ]; then

echo "What is the horizontal resolution of displays? [default 1600]"
read rX
echo "What is the vertical resolution of displays? [default 1200]"
read rY
if [ -f $rX ]; then
rX=1600
fi
if [ -f $rY  ]; then
rY=1200
fi
echo "Which IP has Ultragrid transmitter? [default: 192.168.50.2]"
read IP2
if [ -f $IP2 ]; then
IP2="192.168.50.2"
fi

        #Asign IP
        let "ID=$BASE+1"
        IP="$START$ID"
        echo "Configuring IP address $IP"
        ifconfig $ETH $IP netmask $MASK

################################################

        #Generate SAIL configuration

################################################
        echo "Creating fsManager.conf"
        FSMANAGER="$SAGE_DIRECTORY/bin/fsManager.conf"

echo "fsManager       local $IP" > $FSMANAGER
echo "systemPort      20002 " >> $FSMANAGER
echo "uiPort          20001 " >> $FSMANAGER
echo "trackPort       20003 " >> $FSMANAGER
echo "conManager      206.220.241.46 15557 " >> $FSMANAGER
echo "" >> $FSMANAGER
echo "tileConfiguration stdtile-auto.conf " >> $FSMANAGER
echo "receiverSyncPort  12000 " >> $FSMANAGER
echo "receiverStreamPort   22000 " >> $FSMANAGER
echo "receiverBufSize    100 " >> $FSMANAGER
echo "fullScreen     0 " >> $FSMANAGER
echo "winTime         0 " >> $FSMANAGER
echo "winStep     1 " >> $FSMANAGER
echo "" >> $FSMANAGER
echo "rcvNwBufSize 65536 " >> $FSMANAGER
echo "sendNwBufSize 65536 " >> $FSMANAGER
echo "MTU 1450 " >> $FSMANAGER




################################################
#       Create stdtile configuration
#
########################################

echo "Creating stdtile-auto.conf"
```

```
STDTILE="$SAGE_DIRECTORY/bin/stdtile-auto.conf"

let "NUM=$ROW*$COL"
echo "TileDisplay" > $STDTILE
echo "   Dimensions $ROW $COL " >> $STDTILE
echo "   Mullions 0.625 0.625 0.625 0.625 " >> $STDTILE
echo "   Resolution $rX $rY " >> $STDTILE
echo "   PPI 90 " >> $STDTILE
echo "   Machines $NUM  " >> $STDTILE

SSH="/root/.ssh/known_hosts"
echo "127.0.0.1 $KEY" > $SSH
i=0
#get first ID for display
let "ID=$BASE+3"

while [ $i -lt $ROW ]
do
        ROWS=" "
        j=0
        while [ $j -lt $COL ]
        do
                echo "   DisplayNode    " >> $STDTILE
                echo "              Name auto$ID " >> $STDTILE
                IP_DIS="$START$ID"
                echo "              IP $IP_DIS " >> $STDTILE
                echo "   Monitors 1 ($i,$j) " >> $STDTILE
                echo "" >> $STDTILE

                echo "$IP_DIS $KEY" >> $SSH

                let "j+=1"
                let "ID+=1"
        done
        echo $ROWS
let "i+=1"
done

#end of configuring stdtile
###############################################
# Edit sage.conf
echo "Creating sage.conf"
APPLAU="$SAGE_DIRECTORY/bin/appLauncher/sage.conf"

#-----------------------------

echo "uv {" >> $APPLAU
echo "" >> $APPLAU
echo "configName uv.conf"  >> $APPLAU
echo "nodeNum 1  " >> $APPLAU
echo "Init 0 0 1920 1080  " >> $APPLAU
echo "exec 127.0.0.1 ./uv -d sage -b 8 $IP2" >> $APPLAU
echo "nwProtocol TCP " >> $APPLAU
echo "binDir /root/uv-0.3.9.3-2/ultragrid/bin " >> $APPLAU
echo "" >> $APPLAU
echo "}"  >> $APPLAU

########################################
#Edit uv.conf
echo "Creating uv.conf"
```

```
ULTRA="/root/uv-0.3.9.3-2/ultragrid/bin/uv.conf"


echo "bridgeOn false" > $ULTRA
echo "fsIP $IP " >> $ULTRA
echo "fsPort 20002 " >> $ULTRA
echo "masterIP $IP" >> $ULTRA
echo "nwID     1 " >> $ULTRA
echo "msgPort  23010 " >> $ULTRA
echo "syncPort 13010 " >> $ULTRA
echo "nodeNum  1 " >> $ULTRA
echo "appID    10 " >> $ULTRA
echo "pixelBlockSize 64 64 " >> $ULTRA
echo "blockThreshold 0 " >> $ULTRA
echo "winX 0 " >> $ULTRA
echo "winY 0 " >> $ULTRA
echo "winWidth 1920 " >> $ULTRA
echo "winHeight 1080 " >> $ULTRA
echo "streamType SAGE_BLOCK_HARD_SYNC " >> $ULTRA
echo "nwProtocol TCP " >> $ULTRA
echo "asyncUpdate true " >> $ULTRA
echo "" >> $ULTRA


#####################################################

        echo "SAIL configured"

        echo "Starting X server"
        startx &
        sleep 10
        export DISPLAY=":0.0"
        #start file server
        cd $SAGE_DIRECTORY/bin/file_server/
        python file_server.py &

        #start sage
        sage &


#######################
# Configuring display
#################
elif [ $JOB = "display" ]; then
        echo ""
        echo "Configuration display:"
        matrix $ROW $COL
        echo "Which position has the display? "
        read POS

        echo "You have $POS position"
        echo ""

        let "ID=$BASE+2+$POS"
        IP_DISPLAY="$START$ID"
        echo "Configuring IP address"
        ifconfig $ETH $IP_DISPLAY netmask $MASK
        echo "Asigned IP= $IP_DISPLAY NETMASK=$MASK"

        echo "Starting X server"
```

```
        X :0 &
fi
```

runUltragrid.sh

```
#bin/bash
echo "Ultragrid (i2CAT)"
echo ""
echo "Autoconfiguring NVIDIA card"
rm /etc/X11/xorg.conf
nvidia-xconfig

echo "Interfaces on the system:"
ifconfig -a | grep eth
PS3="Choose interface:"
echo "Which interface want to configure as main?"
select ETH in `ifconfig -a | grep eth | awk '{print $1}'`
do
        break
done
echo "You will use $ETH interface"

echo "Which IP has Ultragrid receiver (You) ? [default: 192.168.50.1]"
read IP1
if [ -f $IP1 ]; then
IP1="192.168.50.1"
fi

echo "Entry net mask for the network. [default 255.255.255.0]"
read MASK

if [ -f $MASK ]; then
MASK="255.255.255.0"
fi

echo "Which IP has Ultragrid transmitter? [default: 192.168.50.2]"
read IP2
if [ -f $IP2 ]; then
IP2="192.168.50.2"
fi
ifconfig $ETH $IP1 netmask $MASK mtu 9180

cd /root/uv-0.3.9.3-2/ultragrid/bin
startx &
sleep 10
export DISPLAY=":0.0"

echo "./uv -d sdl -b 8 $IP2"
./uv -d sdl -b 8 $IP2
```

ANNEX E. TESTS RESULTS

This annex includes almost all test results are made during this TFC.

E.1 Network tests

We try network link with iperf program. Iperf send a stream of data from UltraGrid RX (client) to dummy client (server) simulating the maximum throughput.

In the server (dummy client) you should tape:

iperf –s

And in the client (UltraGrid RX)

iperf –c [ip]

[ip]: network IP address of the server

We try without load and then with load of all one CPU at full processor. We create load with:

x11perf –putimagexy500 –time 30 –repeat 1 –display localhost:0

Table E.1 Network bandwidth

	Sample 1 (Mbps)	**Sample 2 (Mbps)**	**Sample 3 (Mbps)**	**Sample 4 (Mbps)**	**Average (Mbps)**
Debian 32 bits without load	933	933	933	934	933,25
Debian 32 bits with load	742	783	778	766	767,25
openSUSE 64 bits without load	928	920	932	931	927,75
openSUSE 64 bits with load	786	782	763	814	786,25

We can observe a reduction of network speed when server has graphics load. This implies that when graphic card is working, network performance is reduced.

E.2 ICC test

In this point we will explain how to configure the dummy clients for compiling with ICC and its results.

E.2.1 Configuring dummy clients

ICC need the library for working properly: libsdtc++.5.2

First of all, we download Intel C++ Compiler from Intel website:
http://www.intel.com/cd/software/products/asmo-na/eng/219771.htm

After downloaded execute install.sh and follow the instructions. When compiler is installed you have to edit your path and added the next line:

source /opt/intel/cc/10.1.xxx/bin/iccvars.sh

xxx is the version of the compiler. This script initializes environment variables.

To compile c source, use command icc and to compile c++ source the command is icpc.

At our case, compiling SAGE we change the variable COMPILER form g++ to icpc at file config.mk and add the selected flags at MYFLAGS on file src/Makefile.

NOTE: When application is compiled and linked with Intel C++ Compiler, this application only work if the compiler is present. You must not uninstall icc if you want that your application will work.

E.2.2 Optimization flags

In the follow line we describe the most important flags involved in optimization with Intel compiler.

It is necessary to say that the optimization results depends on the application, the results of improvement are different from one application to other or one architecture to another.

E.2.2.1 General Optimization Options

Intel C++ Compiler has a set of options that permit automatic apply a set of action that improves the code speed that balance compile-time and run-time.

The options are: O0, fast, O1, O2 (default), O3. The best option for improve the code is O3. These options are generic for all processor. GCC also support these flags but the improvement of performance is less than Intel Compiler.

E.2.2.2 Parallel Performance

This option detects code that can execute in parallel and automatically generates multi-threaded code. This option improves the performance on SMP systems.

Option: -parallel

E.2.2.3 Processor-Specific Optimization Options

These options enable to archive the architecture for different family of processors.

- -x {S| T| P| O| N| W| K}. Generate specialized code for the indicated processor and enables vectorization.
- -ax {S| T| P| O| N| W| K} Automatic Processor Dispatch. Generates specialized code and enables vectorization for the indicated processors while also generating non-processor-specific code.

We utilize the W option in the dummy clients. This option is for Non-Intel processor witch which support SSE2 and SSE, instructions that support our processor.

E.2.2.4 Interprocedural Optimization (IPO)

Interprocedural optimization (IPO) is a compiler technique for improve performance in a frequently functions. This method analyzes all the code, not only a function and applies techniques of optimization in general aspects.

The options that we try are:

- -ip Single file optimization
- -ipo[value] Permits inlining and other interprocedural optimizations among multiple source files.

E.2.3 Test results

The results are taken from SAGE UI that has a monitor that display frame rate and total bandwidth send it.

The software versions of this first test are:

- Gcc 4.1.2
- Debian 4 etch 32 bits

Table E.2. Intel C++ Compiler measurements in Debian 32 bits.

Flags involved	Average Frame rate (fps)	Average Bandwidth (Mbps)	% of performance from gcc -O3	% of performance without optimization
Initial point: g++ no optimization	7,8	251	-47%	0%
Icpc without flags	9,1	293,6	-38%	17%
G++ with -O3	14,7	473	0%	88%
-ip	15,5	515	5%	99%
-ipo	15,9	513	8%	104%
-O1	16	517	9%	105%
-O2	15,8	509	7%	103%
-O3	16	514	9%	105%
-O3 -axW	16,3	526,6	11%	109%
-O3 -xW	17	550	16%	118%
-O3 -xW -axW	16,9	546	15%	117%

We try the flags –fast and –parallel and we have a compilation error with this flags. This is due of Intel C compiler is designed for Intel processor, not for AMD processor.

Table E.3 Intel C++ Compiler measurements in openSUSE 10.3 (64 bits)

Flag	Frame rate (fps)	Network bandwidth (Mbps)
-xW -axW	19,8	655
-xW	20,6	660
-O3 –xW	20,1	650
-O3	20,5	662
-O2 -xW	20,6	666

-O1 –xW	20,5	662
-O0 -xW	7,8	255
-03 –xW –fno-rtti	20,3	657

-fno-rtti. Using this switch instructs the C++ compiler to discard C++ run-time type information (RTTI). This may improve performance

E.3 GCC TESTS

We try different flags in gcc (version 4.2.1) compiler on openSUSE 10.3 (64 bits)

Table E.4 GCC measurement in openSUSE 10.3 (64 bits)

Flag	**Frame rate (fps)**	**Bandwidth (Mbps)**
-O2	14,7	475
-O3	19,2	619
-O3 –march=k8	18,8	609
-O3 –ffast-math	18,1	583
-O3 –march=k8 –ffast-math	18,8	605
-O3 –funroll-loops	18,8	606
-O3 –ftree-vectorize	18,2	590
-O3 –march=k8 -ffast-math -funroll-all-loops -ftree-vectorize	19,3	623

Using the *-ffast-math* switch allows the compiler to use a fast floating point model. The *-funroll-all-loops* causes all loops to be unrolled and makes code larger and could bring improvement in speed. The GCC 4.0 and later version compilers can perform loop vectorization by using the *-ftree-vectorize* flag.
Using -march=k8 flag specifies the processor family. AMDAthlon XP is inside k8 group family.

We observe the maximum speed when all the flags used at the same compilation.

E.4 Guided Profile Test

Intel compilers allow profile guided optimization. Use the following steps for profile guided optimization with Intel compilers.

1. Compile the program with the -prof_gen switch.
2. Run the executable produced in Step 1. Running this executable generates several files with profile information (*.dyn and *.dpi).
3. Recompile the program with the -prof_use switch.

Table E.5. ICC Guided Profile

Status	Frame rate (fps)	Bandwidth (Mbps)
Without profiling	20,4	678
During profiling (-prof_gen)	17,7	570
Profiling done (-prof_use)	20,3	657

The 64-bit GCC compiler allows profile guided optimization with the following steps:

1. Compile the program with *-fprofile-arcs.*
2. Run the executable produced in Step 1. Running the executable generates several files with profile information (*.da).
3. Recompile the program with *-fbranch-probabilities*.

Table E.6. GCC Guided Profile

Status	Frame rate (fps)	Bandwidth (Mbps)
Without profiling	19,3	623
During profiling (-prof_gen)	15	480
Profiling done (-prof_use)	19,3	620

We do not obtain improvements, even with GCC or ICC. We can observe in both compilers that when profiling is done, the speed is reduced due the binary has to sample code.

E.5 Lmbench test

Lnbench is a suite of simple, portable benchmarks created by Larry McVoy. They can make bandwidth benchmarks. We did a memory test for determinate the bandwidth of memory, with intention of knowing if the bus memory is the bottle neck of the system.

Table E.7 Lmbech results

Test	Bandwidth (Mb/s)
RAM Read	1220
RAM Write	1398
L2 read	2796
L2 write	2796

E.6 Real-time test

We made different test with kernel options.

E.6.1 General setup

Parameters selected:

- ( ) Optimized for size
- Without real-time (No Forced Preemption (Server))

Table E.8. Test 1

	Frame rate (fps)	**Bandwidth (Mbps)**
with	17	548
without	17	549

E.6.2 Real-time

We evaluate all the options involved in Real time Patch of kernel.org. First we monitor without apply real time patch with the following result:

Frame rate: 17,1 fps Bandwidth: 550 Mbps

The options are in kernel tree in *Processor type and features:*

[ ] High resolution-timer option
Preemption Mode -->

- *No Forced Preemption (Server)*
- *Voluntary Kernel Preemption (Desktop)*
- *Preemptive Kernel (Low-Latency Desktop)*
- *Complete Preemption (Real-Time)*

Table E.9. Real Time performance test

Options enable	Frame rate (fps)	Bandwidth (Mbps)
[ ] High resolution-timer option Preemption Mode --> No Forced Preemption	18	580
[*] High resolution-timer option Preemption Mode --> No Forced Preemption	17	549
[*] High resolution-timer option Preemption Mode --> Voluntary Kernel Preemption (Desktop)	17,1	550
[*] High resolution-timer option Preemption Mode --> Preemptive Kernel (Low-Latency Desktop)	17,6	566
[*] High resolution-timer option Preemption Mode --> Complete Preemption (Real-Time)	17,6	570

E.6.3 Test Default I/O scheduler

The Linux kernel, the core of the operating system, is responsible for controlling disk access by using kernel I/O scheduling. We try to change the scheduler due also affect at network scheduler (42).

These options are inside the kernel on:

Enable Block Layer
Default I/O scheduler -->

- Completely Fair Queuing
- Deadline
- NOOP
- Anticipatory

Included in Linux Kernel 2.6.x are four custom configured schedulers from which to choose. They each offer a different combination of optimizations.

The Completely Fair Queuing (CFQ) scheduler is the default. CFQ maintains a scalable per-process I/O queue and attempts to distribute the available I/O bandwidth equally among all I/O requests.

The Deadline elevator uses a deadline algorithm to minimize I/O latency for a given I/O request. The scheduler provides near real-time behavior and uses a round robin policy to attempt to be fair among multiple I/O requests and to avoid process starvation. Using five I/O queues, this scheduler will aggressively re-order requests to improve I/O performance.

The NOOP scheduler is a simple FIFO queue and uses the minimal amount of CPU/instructions per I/O to accomplish the basic merging and sorting functionality to complete the I/O.

The Anticipatory elevator introduces a controlled delay before dispatching the I/O to attempt to aggregate and/or re-order requests improving locality and reducing disk seek operations. This algorithm is intended to optimize systems with small or slow disk subsystems. One artifact of using the AS scheduler can be higher I/O latency.

Table E.10. I/O scheduler test

Default I/O scheduler	Frame rate	Bandwidth
Completely Fair Queuing	17,5	550
Deadline	17,3	558
NOOP	17,4	550
Anticipatory	17,0	500

There are not visible differences between schedulers when TEST is made with SAGE.

E.7 64 bits tests

Debian Etch 4 32 bits, Debian Etch 4 64 bits and openSUSE 10.3 64 bits were tested with both compilers analysed with the following results.

Table E.11 GCC 64 bits results

	Frame rate (fps)	Bandwidth (Mbps)	Difference with Debian 32	Difference with Debian 64
Debian 32 bits	14,7			
Debian 64	16,3	530	11%	
openSUSE 64	18,1	590	23%	11%

Table E.12 Intel C++ Compiler 64 bits results

	Frame rate (fps)	Bandwidth (Mbps)	Difference with Debian 32	Difference with Debian 64
Debian 32 bits	17			
Debian 64	17	550	0%	
openSUSE 64	20,3	630	22%	22%

We observe a huge rise of performance in openSUSE. The main reason is that Suse people apply a lot of parches for improving the official kernel source.

Finally we try Debian kernel in openSUSE. We obtain same results as Debian.

E.8 Mpstat captures

Mpstat monitor processor use. The next captures are made in barebone when SAGE stream is send directly to dummy client. Are made with the following command:

mpstat –P ALL 5 20

We can observe that scheduler distribute in different way the jobs in both Distributions. Suse balance all the user load to one CPU and system load to other CPU. Debian distributed the load as symmetric. We have better results in Suse way.

Table E.13 OpenSUSE 64 bits mpstat capture

CPU	%user	%nice	%sys	%iowait	%irq	%soft	%steal	%idle	intr/s
all	50,26	0	13,22	0	4,56	11,43	0	20,53	56595,58
0	95,06	0	3,22	0	0,18	0,7	0	0,84	224,12
1	5,48	0	23,22	0	8,91	22,18	0	40,21	56371,47

Table E.14 Debian 32 bits

CPU	%user	%nice	%sys	%iowait	%irq	%soft	%steal	%idle	intr/s
all	45,61	0	9,83	0	3,84	8,73	0	31,99	49003,88
0	38,36	0	11,23	0	4,02	8,83	0	37,56	25029,06
1	52,89	0	8,41	0	3,64	8,63	0	26,43	23974,82

E.9 Resolution test

Table E.15 Screen resolution vs theory network bandwidth

Width	Height	Frame size (Mbits)	Network bandwidth per screen (Mpbs)
1024	768	12,6	377
1280	1024	21,0	629
1600	1080	27,6	829
1600	1200	30,7	922

Table E.16 Screen resolution vs experimental network bandwidth

Width	Height	Frame rate (fps)	Network bandwidth per screen (Mpbs)
1024	768	30,0	377
1280	1024	25,0	524
1600	1200	20,7	636

If we reduce the resolution to 1024 x 768 we can reach the desired frame rate but in otherwise we will need more screens for displaying the same resolutions.

E.10 OProfile SAGE captures

OProfile is used for determinate where is the bottle neck.

For start collecting data you should tape:

opcontrol --vmlinux=/usr/src/linux/vmlinux
opcontrol --start

We can observe a reduction of 25 % of performance when data is collecting.

When we desided that we have enough data, we stop OProfile with the next command:

opcontrol --shutdown

Finally for displaying the information captured, we have some command:

Display source application with time used each line.

opannotate --source -D smart /usr/share/sage2/bin/sageDisplayManager

Display a summay of the data collected:

opreport –l

Display a summay of functions used in desired application:

opreport -l /usr/share/sage2/bin/sageDisplayManager

opreport –l

General:

```
CPU: AMD64 processors, speed 2100.2 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Cycles outside of halt state) with a unit mask of 0x00
(No unit mask) count 100000
CPU_CLK_UNHALT...|
  samples|      %|
------------------
  1046344 36.4491 vmlinux
   787630 27.4368 sageDisplayManager
   532325 18.5434 libGLcore.so.169.12
   354248 12.3401 zero
    89911  3.1320 libc-2.6.1.so
```

```
 27264  0.9497 libpthread-2.6.1.so
 12365  0.4307 oprofiled
  6156  0.2144 libGL.so.169.12
```

opreport -l /usr/share/sage2/bin/sageDisplayManager

```
Sumary:

CPU: AMD64 processors, speed 2100.2 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Cycles outside of halt state) with a unit mask of 0x00
(No unit mask) count 100000
samples  %        symbol name
744106   94.4741  sageMontage::loadPixelBlock(sagePixelBlock*)
4311     0.5473  sagePixelData::getFrameID()
3582     0.4548  sageBlockQueue::enque(sageBlock*)
3567     0.4529  sage::tokenSeek(char*, int)
3385     0.4298  sagePixelData::allocateBuffer(int)
2677     0.3399  sageBlockQueue::deque(int, bool, int)
2526     0.3207  sageBlockPartition::setViewPort(sageRect&)
2168     0.2753  _ZN14sagePixelBlockC9EP10sageMemoryi
1970     0.2501  pixelDownSync::fetchSageBlocks()
1940     0.2463  pixelDownloader::downloadPixelBlock(sagePixelBlock*, montagePair&)
1844     0.2341  sageRect::operator*(sageRect&)
1768     0.2245  sagePixelReceiver::readPixelData(int, sageBlockQueue*)
1544     0.1960  sageMemory::allocate(int)
```

opannotate --source -D smart /usr/share/sage2/bin/sageDisplayManager

```
/*
 * Command line: opannotate --source -D smart /usr/share/sage2/bin/sageDisplayManager
 *
 * Interpretation of command line:
 * Output annotated source file with samples
 * Output all files
 *
 * CPU: AMD64 processors, speed 2100.2 MHz (estimated)
 * Counted CPU_CLK_UNHALTED events (Cycles outside of halt state) with a unit mask of
0x00 (No unit mask) count 100000
 */
/*
 * Total samples for file : "/usr/share/sage2/src/sageDisplay.cpp"
 *
 * 666527 94.6917
 */


              :/*****************************************************************************
              : * SAGE - Scalable Adaptive Graphics Environment
              : *
              : * Module: sageDisplay.cpp - the display module in SAGE Receiver
              : * Author : Byungil Jeong, Rajvikram Singh
              : *
(...)
              :#define YUV444toRGB888(Y,U,V,R,G,B) \
              :   R = clip(( 298 * (Y-16)                 + 409 * (V-128) + 128) >> 8); \
              :     G = clip(( 298 * (Y-16) - 100 * (U-128) - 208 * (V-128) + 128) >> 8); \
```

```
           :     B = clip(( 298 * (Y-16) + 516 * (U-128)              + 128) >> 8);
           :
    8  0.0011 :   if (pixelType == PIXFMT_YUV) {
           :#if defined(GLSL_YUV)
           :     glDisable(GL_TEXTURE_2D);
           :     glEnable(GL_TEXTURE_RECTANGLE_ARB);
           :     glBindTexture(GL_TEXTURE_RECTANGLE_ARB, texHandle);
           :        glTexSubImage2D(GL_TEXTURE_RECTANGLE_ARB, 0, block->x, block->y,
block->width, block->height,
           :       pInfo.pixelFormat, pInfo.pixelDataType, block->getPixelBuffer());
           :     glDisable(GL_TEXTURE_RECTANGLE_ARB);
           :     glEnable(GL_TEXTURE_2D);
           :#else
           :     unsigned char *yuv = (unsigned char*)block->getPixelBuffer();
           :     unsigned char u,v,y1,y2;
           :     unsigned char r1,r2,g1,g2,b1,b2;
           :     int i, k;
           :     k = 0;
 69488  9.8720 :     for (i=0;i< (block->width*block->height)/2;i++) {
 13860  1.9691 :       u  = yuv[4*i+0];
 14290  2.0301 :       y1 = yuv[4*i+1];
 14169  2.0130 :       v  = yuv[4*i+2];
 17175  2.4400 :       y2 = yuv[4*i+3];
           :
279421 39.6966 :       YUV444toRGB888(y1, u, v, r1,g1,b1);
188110 26.7243 :       YUV444toRGB888(y2, u, v, r2,g2,b2);
 13810  1.9619 :       texture[k + 0] = r1;
   584  0.0830 :       texture[k + 1] = g1;
 25145  3.5723 :       texture[k + 2] = b1;
 13923  1.9780 :       texture[k + 3] = r2;
   584  0.0830 :       texture[k + 4] = g2;
 13869  1.9703 :       texture[k + 5] = b2;
   674  0.0958 :       k += 6;
           :     }
    35  0.0050 :     glTexSubImage2D(GL_TEXTURE_2D, 0, block->x, block->y, block->width,
block->height,
    32  0.0045 :     GL_RGB, GL_UNSIGNED_BYTE, texture);
           :#endif
           :   }
           :   else {
           :     //std::cerr << "block download " << block->x << " , " << block->y << " , " <<
           :     //   block->width << " , " << block->height << std::endl;
           :
           :     glTexSubImage2D(GL_TEXTURE_2D, 0, block->x, block->y, block->width, block-
>height,
           :       pInfo.pixelFormat, pInfo.pixelDataType, block->getPixelBuffer());
           :   }
           :#else
           :    glTexSubImage2D(GL_TEXTURE_2D, 0, block->x, block->y, block->width, block-
>height,
           :       pInfo.pixelFormat, pInfo.pixelDataType, block->getPixelBuffer());
           :#endif
           :
   421  0.0598 :   return 0;
(...)
```

E.11 Powerful PC Test

In the last days of this TFC, we can test a newer PC bought by i2CAT Foundation. We install an openSUSE 10.3 in it.

This PC has an Intel Quad Core CPU, 4 GB of RAM, a gigabit Ethernet and external NVIDIA Geforce 8600 GT graphic card. It cost is around 500 euros

We test with GCC compiler and Intel C++ Compilers. Both compilers has good results.

With this equipment we obtain the following results:

Measurement made by top program:
Summary
CPU0: 0 % used
CPU1: 0 % used
CPU2: 85% used 13%idle
CPU3: 10% system 56%idle
Memory used by SAGE 11,5 %

We obtain 30 fps, when HD stream is sent in the worst case scenario.

Iperf give an average network bandwidth of 911 Mbps

ANNEX F. MONITORING USER GUIDE

A script for collecting data of monitoring is made. This script initializes Sar, wait for the time specified and finally retrieve sar files to the controller computer. Computers to monitor are specified in a text file.

Script Code:

```
#!/bin/sh
LIST=`cat sage4.list`
ifconfig eth0 192.168.50.100
if  [ $# -ne 2 ]; then
        echo "Usage: ./initTest.sh <nameTest> <Time of sample>"
        exit;
fi
T=5
let "N = $2 / $T"

TEST="sar -A -o $1 $T $N > /dev/null"
echo $TEST

for node in $LIST
do
        echo "conection with $node"
        ssh root@$node "$TEST && exit" &
done

mkdir $1

let "TIME = $T * $N"
let "wait = $TIME / 4"
echo "waiting test: $TIME"
sleep $wait
echo "1/4 waited"
sleep $wait
echo "2/4 waited"
sleep $wait
echo "3/4 waited"
sleep $wait
echo "4/4 waited"
sleep $T
echo "test is finished."
z=1
for node1 in $LIST
do
        echo "copying $z and removing"
        scp root@$node1:$1 $1/$1"_"$z
        #remove file from remote system
        ssh root@$node1  "rm $1 && exit" &
        let "z += 1"
done
#log
echo "$TEST" > $1/log.txt
echo "end of monitor"

exit
```

For running the scripts you should tape:

./initTest [test name] [time in second]

The files are saved into a directory of test name.

The graphs are made by kSar. kSar is a java program. For running Ksar you should tape:

java –jar kSar-4.0.12.jar

For opening a captured file:

File → Run Local Command…

In the windows box that appears type:

sar –A –f [Test file]_[number of system to be analyzed]

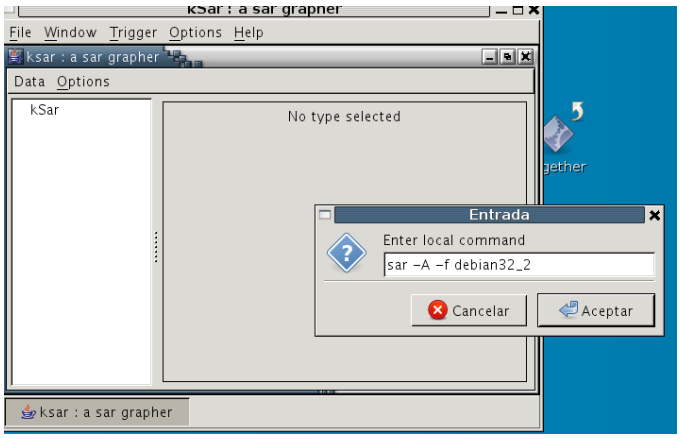


Fig F.1 Box where is selected file to be analyzed

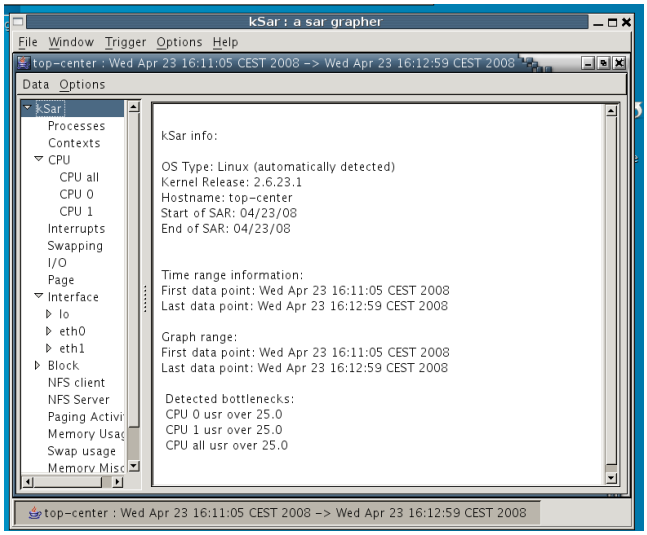
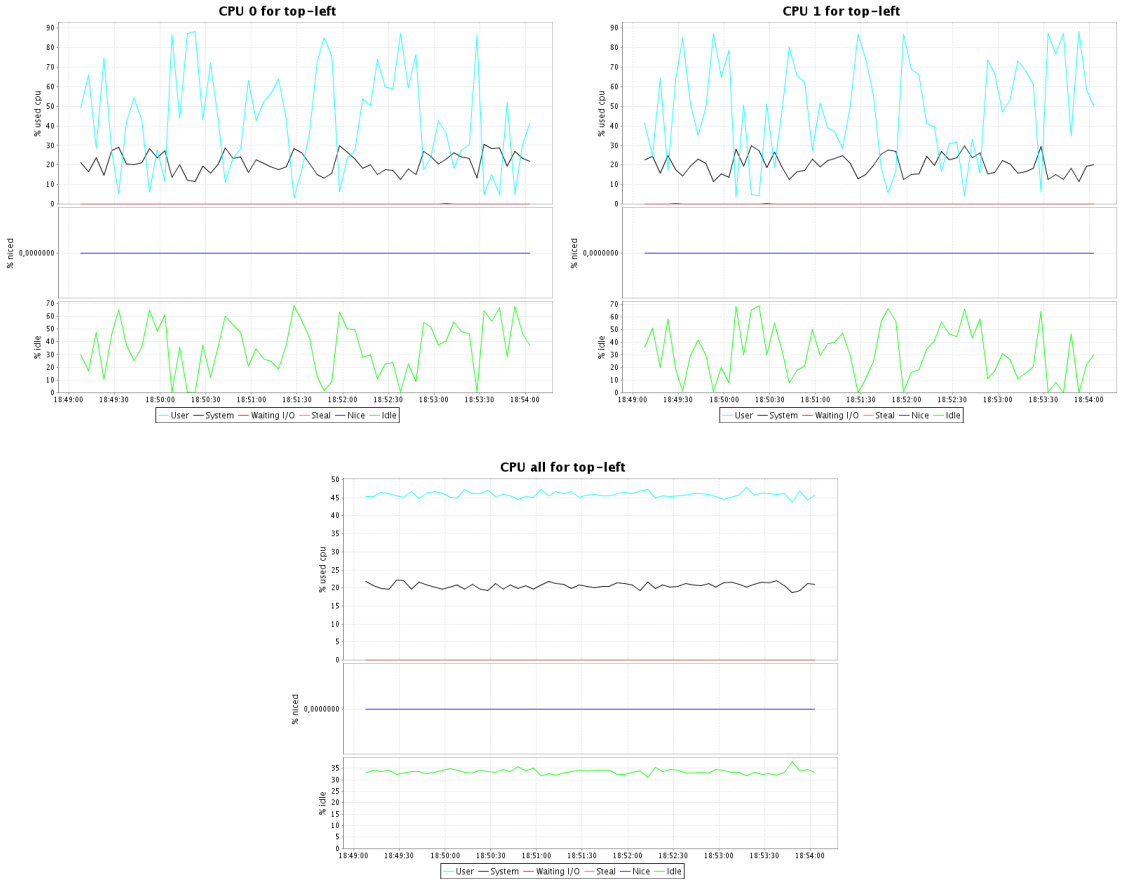


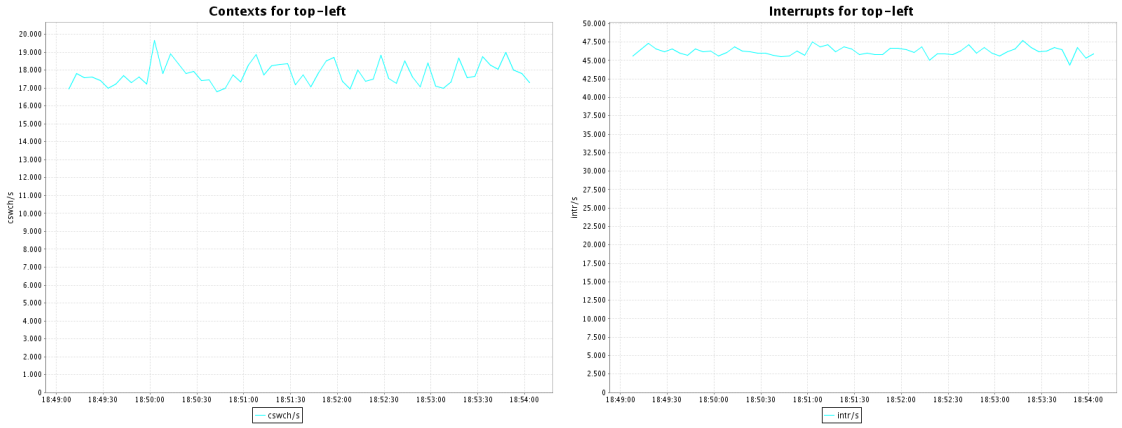
Fig F.2 Main window of Ksar application.

F.1 Debian capture

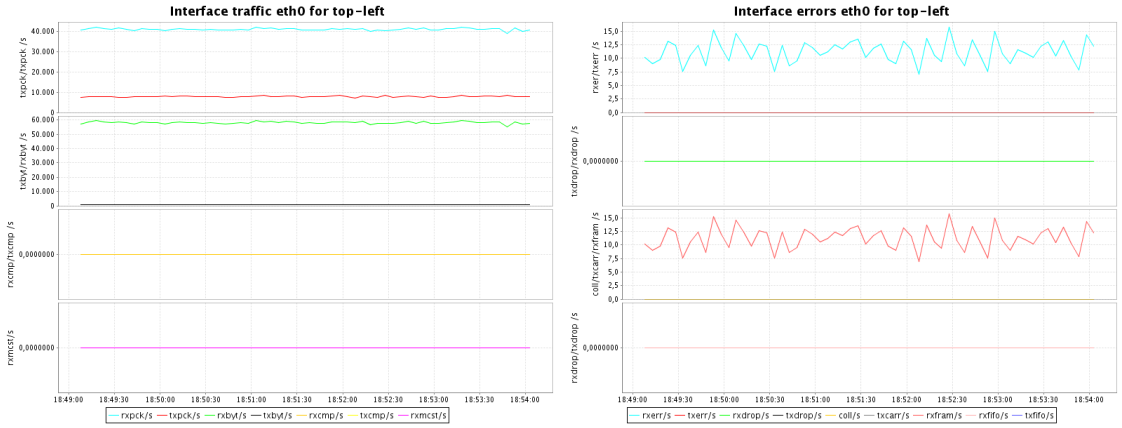
The following graphs are made with Debian etch 4 (32 bits) system. These captures are made when SAGE is running.



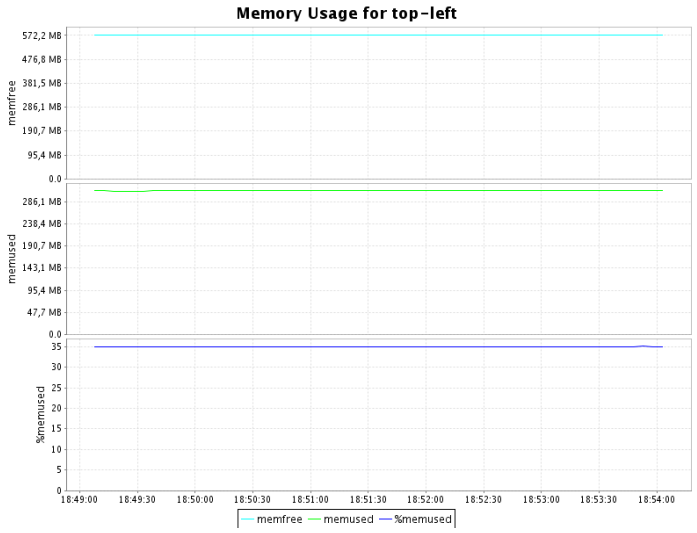
We can observe in CPU utilization that Linux dispatcher balance the load between both CPU but in all CPU user activity does not pass of 50 %. A process only can be executed in a CPU in the same time.



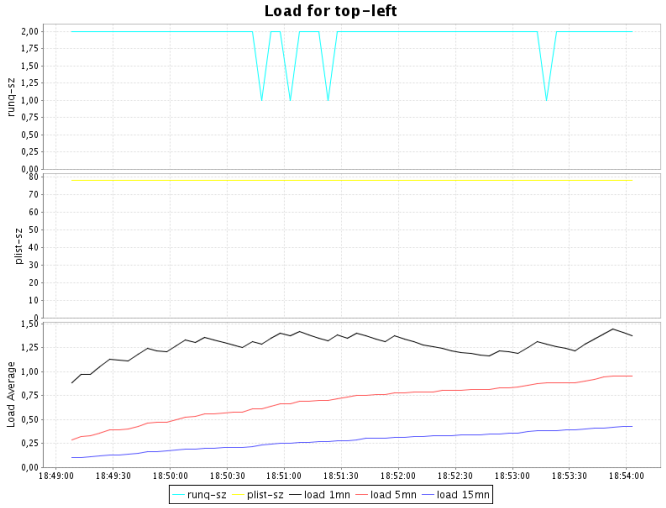
The high value of context switching and interrupts are casued because the system is overload in network interface and CPU usage.



We can observe that network traffic is constant. Frame error appears due huge amount of traffic with little packets. If we use Jumbo frames packets, the error will disappear.

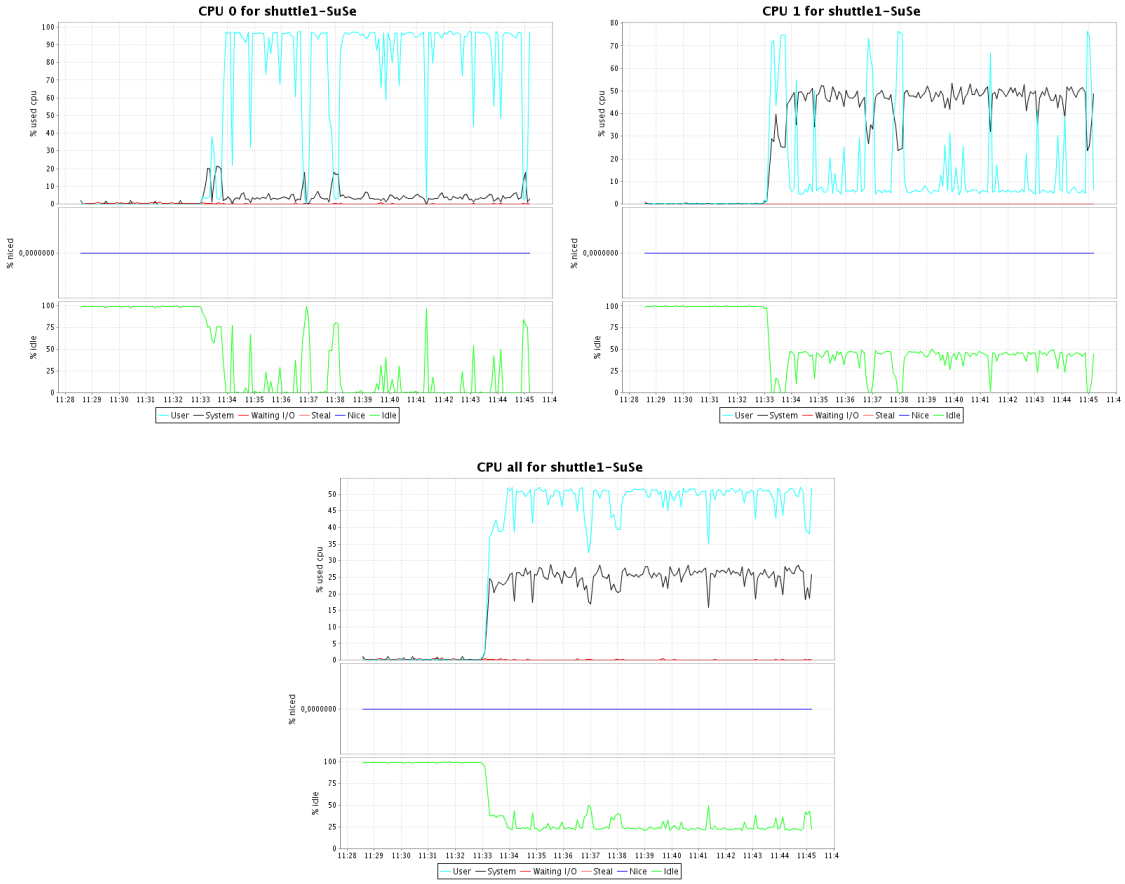


We can observe that memory size is not a problem due its uses is stable and only use a 35 % of total memory.

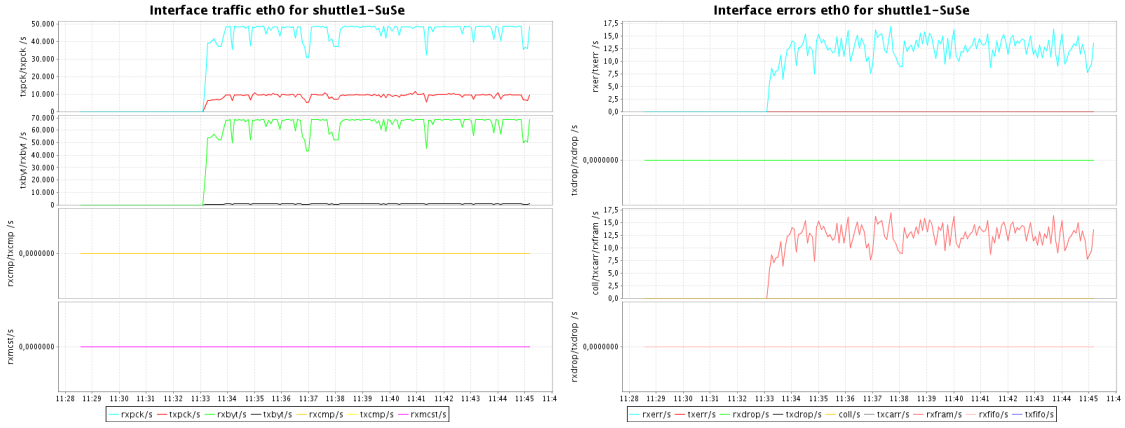


F.2 openSUSE capture

The following graphs are made with openSUSE 10.3 (64 bits) system. First there are all stopped and then SAGE is started.



We can observe that Linux scheduler balance the user activity to one CPU and system activity to the other CPU. System activity is related of network traffic. This policy give better results that switching the process to one CPU to other, like Debian do.



The behavior of above graphs is the same as Debian etch.

ANNEX G. GLOSSARY

AJA: Company that produce video capture devices

AMD: Advanced Micro Devices. Company that produce processors

AMD64. Computer architecture based on AMD processor of 64 bits.

Aufs: Another Union File System. See UnionFs.

Bandwidth: a rate of data transfer, measured in bits per second.

BLFS: Beyond Linux From Scratch (BLFS) is a project that continues where the LFS book finishes. It assists users in developing their systems according to their needs by providing a broad range of instructions for installing and configuring various packages on top of a base LFS system.

C: language of computer programming

C++: language of computer programming based in objects

Chroot: A chroot on Unix operating systems is an operation that changes the apparent disk root directory for the current running process and its children.

Compiler: A compiler is a computer program (or set of programs) that translates text written in a computer language (the source language) into another computer language (the target language). The original sequence is usually called the source code and the output called object code

CPU: A Central Processing Unit (CPU), or sometimes just called processor, is a description of a class of logic machines that can execute computer programs.

CUDA: Compute Unified Device Architecture is a technology that allows a programmer to use the C programming language to code algorithms for execution on the graphics processing unit (GPU). This is developed by NVIDIA

Debian: Debian is a computer operating system (OS) composed entirely of software which is both free and open source (FOSS). Its primary form, Debian GNU/Linux, is a popular and influential Linux distribution.

DV: Digital Video is a digital video format launched in 1996, and, in its smaller tape form factor MiniDV, has since become a standard for home and semi-professional video production;

E-learning: Electronic learning is a general term used to refer to a form of learning in which the instructor and student are separated by space or time where the gap between the two is bridged through the use of online technologies

Ethernet: It is a family of frame-based computer networking technologies for local area networks (LANs)

EVL: Electronic visualization laboratory is a graduate research laboratory specializing in virtual reality and real-time interactive computer graphics

FIFO: First In First Out. It is a queue policy where first element enter to the queue it is first element to out of it.

Fps: Frames per second

Gbps: Gigabit per second. 1 Gbps equals to 10^9 bits per second

GCC: Compiler of Linux systems.

Gentoo: that can be automatically optimized and customized for just about any application or need.

GNU: It is a computer operating system composed entirely of free software. Its name is a recursive acronym for GNU's Not Unix

GPU: A graphics processing unit is a dedicated graphics rendering device for a personal computer, workstation, or game console.

HD-SDI: is a digital video interface used for High Definition Videos with rate of 1,485 Gbps.

I/O devices: Input / Output Device

I386: Architecture based on x86 Intel Processors.

Intel: Company that manufactures CPUs

IP: Internet Protocol. Protocol used at Layer 3 in network communications.

ISI: The University of Southern California's Information Sciences Institute (ISI)

JPEG: compress image format.

Kernel: core of operation systems

LFS: Linux From Scratch is a project that provides you with step-by-step instructions for building your own customized Linux system entirely from source.

Linux is the name usually given to any Unix-like computer operating system that uses the Linux kernel. Linux is one of the most prominent examples of free software and open source development: typically all underlying source code can be freely modified, used, and redistributed by anyone.

Linux From Scratch: see LFS.

Live CD: is a computer operating system that is executed upon boot, without installation to a hard disk drive.

MB: Mega byte. 1MB= 1024 KBytes = 1024x1024 Bytes

Mbps: Megabits per second: 1 Mbps = 1000Kbps = 10^6bits per second

MTU: Maximum transmission unit, the size of the largest packet that a network protocol can transmit.

NAPI: New API is an interface to use interrupt mitigation techniques for networking devices in the Linux kernel. Such an approach is intended to reduce the overhead of packet receiving. The idea is to defer incoming message handling until there is a sufficient amount of them so that it is worth handling them all at once.

NTSC: National Television System Committee is the analog television system used in the United States, Canada, Japan, Mexico, the Philippines, South Korea, Taiwan, and some other countries. The standard called for 525 lines of picture information in each frame, and 30 frames per second.

Open source: Open source is a development methodology, which offers practical accessibility to a product's source.

OpenGL: Open Graphics Library is a standard specification defining a cross-language cross-platform API for writing applications that produce 2D and 3D computer graphics.

OpenMP: Open Multi-Processing is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C/C++ and Fortran on many architectures, including Unix and Microsoft Windows platforms. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.

OpenSUSE: It is a community project, sponsored by Novell and AMD, to develop and maintain a general purpose Linux distribution

PAL: Phase Alternating Line, is a color encoding system used in broadcast television systems in large parts of the world. Use 625 lines and 25 frame/s.

Python: It is a general-purpose, very high-level programming language.

RAM: Random access memory is a type of computer data storage. It is mostly associated with volatile types of memory

RGB: Red Green Blue is a color encoding for images in computer graphics.

RT: Real Time

RTP: Real-time Transport Protocol defines a standardized packet format for delivering audio and video over the Internet.

RTT: Round Trip Time. The elapsed time for transit of a signal over a closed circuit, or time elapsed for a message to a remote place and back again.

SAGE: Scalable Adaptive Graphics Environment is a graphics streaming architecture for supporting collaborative scientific visualization environments with potentially hundreds of megapixels of contiguous display resolution.

SAIL: SAGE program.

SMP: Computer with more than one CPU.

SMTPE: Society of Motion Picture and Television Engineers is the leading technical society for the motion imaging industry.

Squashfs: It is a compressed read-only filesystem for Linux. Squashfs is intended for general read-only filesystem use and in constrained block device/memory systems where low overhead is needed

SSE: Streaming SIMD Extensions (SSE) is a SIMD (Single Instruction, Multiple Data) instruction set extension to x86 architecture, designed by Intel and introduced in 1999 in their Pentium III series processors as a reply to AMD's 3DNow!. SSE contains 70 new instructions. By Wikipedia.

Supermicro: It is a manufacturer of servers.

TCP: Transport Control Protocol is

UDP: User Datagram Protocol is one of the core protocols of the Internet protocol suite. Using UDP, programs on networked computers can send short messages sometimes known as datagrams to one another.

Unionfs: It is a Linux filesystem service which implements a union mount for Linux file systems. It allows files and directories of separate file systems, known as branches, to be transparently overlaid, forming a single coherent file system

UNIX: It is a computer operating system originally developed in 1969 by a group of AT&T employees at Bell Lab

Vectorization: It is the process of converting a computer program from a scalar implementation, which does an operation on a pair of operands at a time, to a vectorized program where a single instruction can perform multiple operations or a pair of vector (series of adjacent values) operands. Vector processing is a major feature of both conventional and modern supercomputers. By Wikipedia

X Window System (commonly **X11** or **X**) is a windowing system which implements the X display protocol and provides windowing on bitmap displays.

It provides the standard toolkit and protocol with which to build graphical user interfaces (GUIs) on most Unix-like operating systems.

YUV model: It defines a color space in terms of one luma (Y') and two chrominance (UV) components