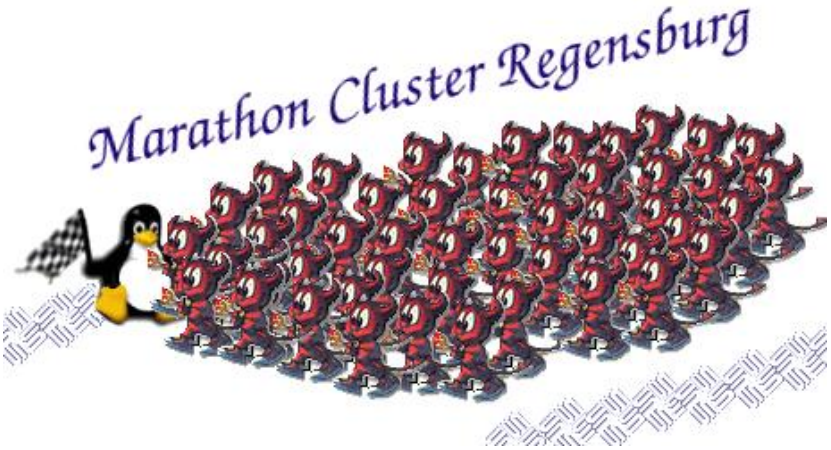


Reaching the Goal with the Regensburg Marathon-Cluster

– A NetBSD Cluster Project –

Hubert Feyrer <hubert@feyrer.de>

October 7, 2002

Abstract

This paper discusses the technical setup and execution of a video rendering cluster based on Open Source software. First, deployment of the cluster clients is discussed, followed by computing steps performed by the cluster - splitting a big video stream into single images and rendering individual videos from the images. Details are given on the hardware and software used as well as optimizations made. Various experiences gained by the cluster project are listed before showing some performance graphs displaying the cluster under load. Lists of facts, numbers and links sum up the whole project.

Contents

1 Introduction **3**

2 Setup of the cluster client machines **3**

3 Computation performed by the cluster **5**

3.1 Preparations 5

3.2 Step 1: Splitting the sequences 5

3.3 Intermediate step: Enter image data into image database 6

3.4 Step 2: Creating the videos 6

4 Experiences gained from the Marathon-Cluster **8**

5 Various images & graphs **10**

6 Facts **15**

6.1 Subclusters 15

6.2 Cluster Control 15

6.3 Numbers 15

6.4 Software 16

6.5 Participants 17

7 Links **17**

1 Introduction

Last summer, R-KOM and the University of Applied Science (Fachhochschule, FH) Regensburg, Germany, took their share of the Regensburg city marathon by putting an individual video and image of each runner reaching the goal on the Internet. A cluster of 45 machines rendered the more than five thousand videos.

Being a sponsor of the Sports Experts Marathon, R-KOM contributed by putting the video and images of each runner completing the course up on the Internet. Due to increased demands on quality, the company's machines were not fast enough to render all the images and videos in a reasonable amount of time. With contacts to the Fachhochschule Regensburg, the department of computer science offered a sufficient number of machines as well as support for setting them up, and a collaboration was made .

The university of applied sciences (Fachhochschule) Regensburg provided 45 machines, including installation and management of the nodes, data storage, and also helped optimizing and tuning the Unix-based software created by employees of R-KOM based on freely available components.

The software consisted of components to split video streams into single pictures as well, as rendering video sequences from single pictures based on the time of arrival of each runner. A fully automated, scriptable processing was important here for the 5.500 runners reaching the goal.

Preparations of the Regensburg Marathon Cluster started on early saturday evening by installing the cluster machines under supervision of Hubert Feyrer of the computer science department of the Fachhochschule Regensburg as well as several students, who helped deploying the machines. Adjusting and tuning the cluster software took the rest of the pre-marathon evening.

After the last runner reached the goal on Sunday afternoon, processing of the video material provided by the local TV company TVA started under the supervision of Jürgen Mayerhofer, R-KOM. The material was used to render video sequences and pictures of each runner's completion of the course.

Using NetBSD, a Unix/Linux-like Open Source operating system, on the cluster clients and other Open Source software to control and calculate the MPEG-animations of the marathon videos and pictures gave a solid base for a software project of this size.

2 Setup of the cluster client machines

The computer science department of the FH Regensburg provided three public rooms full of machines for the cluster, with a total of 57 computers. 15 of the machines were running Solaris, which we were not allowed to change. The remaining machines were available for re-installing, and we chose NetBSD as client operating system, as all the software we needed was easily available through the NetBSD 3rd party software collection, the system was simple

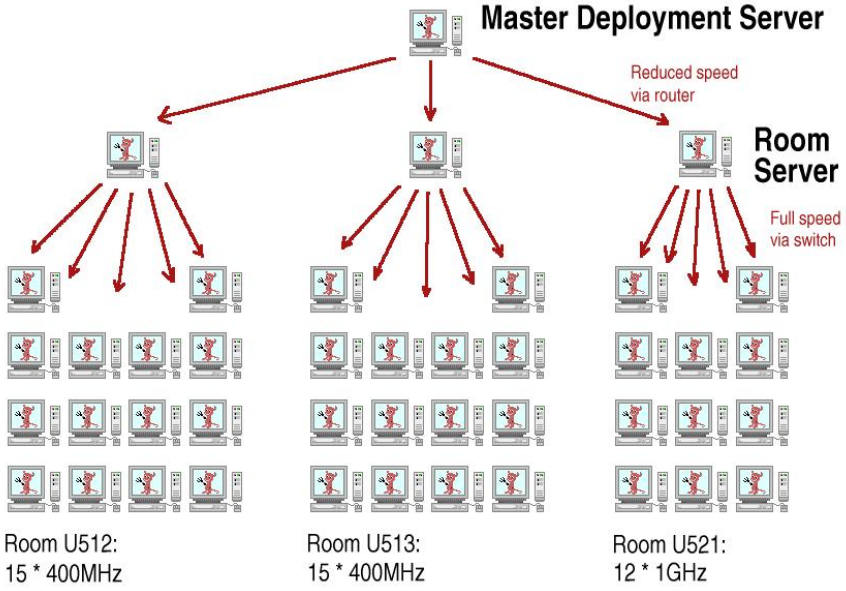
to install on the client machines and we had a lot of know-how for NetBSD available. The hardware consisted of Dell OptiPlex PCs with various configurations of RAM and CPU. In two of the rooms (U512, U513) were 15 machines each with PII-400MHz, 64 MB RAM and 4GB HD each, the other one (U521) had 12 machines with PIII-1GHz, 256MB RAM and 10GB HD each.

For the cluster setup, we installed one of the PII-400 machines with NetBSD and added all the necessary programs (dumpmpeg, mpeg_encode) from the NetBSD Packages Collection. We had a single account for the whole cluster project, which had it's home directory mounted from a NFS server and we also used NFS for temporary diskspace used during the calculations. The NFS server used was the Unix server of the computer science department, a Sun Ultra 10 with a 300MHz CPU, 1024MB RAM and 120GB harddisk (Arena hardware IDE-to-SCSI RAID).

For monitoring purposes we also installed an SNMP agent, the rstat service as well as the 'tload' program, which gives a console-based overview of the machine's load. Various programs needed by dumpmpeg and mpeg_encode completed the client installation.

After the cluster client was configured and running properly, the harddisk-image of the client machine was stored on the master deployment server of the FH Regensburg using the "g4u" harddisk image cloning software. The 4GB harddisk image was compressed to about 650MB in that process, deploying that 4GB image to both the 4GB as well as 10GB disks of the clients wasn't a problem with g4u.

For the deployment of the client, the image was first installed on one machine in each of the available rooms (using g4u again). After rebooting these machines and starting up NetBSD, the just-installed client machines were setup themselves as "room-server". For that, the client's harddisk image was put into a prepared FTP area, and after that, the remaining clients in that room were able to retrieve the image from their nearest room-server. As deployment from the master-server was via a router, deployment within the various rooms was much faster as the machines operated in a pure 100MBps switched network, with no slow router in between.

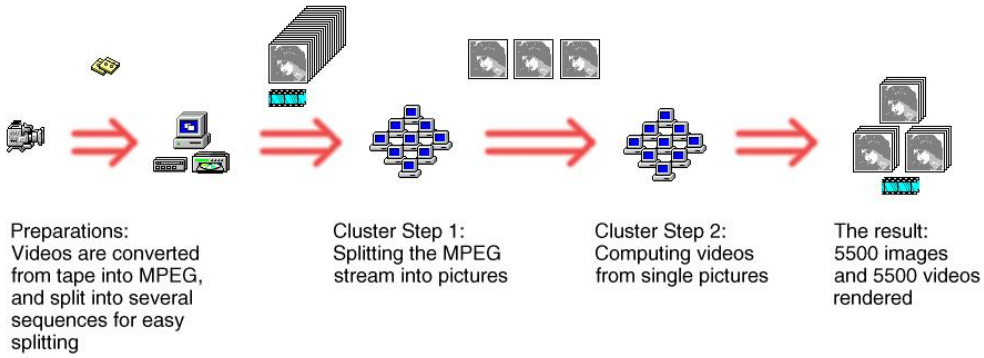


3 Computation performed by the cluster

The computation performed by the Regensburg Marathon Cluster consisted of two steps:

1. splitting the video sequences into single images
2. merging single images into individual video sequences

The following image gives an overview of the whole process:



3.1 Preparations

At the goal of the marathon, two video cameras placed by the local TV station TVA recorded the whole event of the runners reaching the goal. When the video tape of one camera reached the end after about 90 min, the second camera was switched on, with an overlap of about five minutes. That way, we got about five hours of video material, in Sony Betacam format.

Next, the four videotapes were converted into MPEG sequences using a Sony Betacam VCR unit as well as a PC running Windows 98 that had a Hauppage PVR card with hardware MPEG encoder. Each MPEG sequence consisted of 11 minutes of film material with 25 frames per second, a resolution of 352x288 pixel and a color depth of 24 bit, which led to about 110 MB per sequence. The sequences were then FTP'd to a PC that was connected via a crossed TP cable and that ran RedHat Linux 7.1. There, the sequences were archived to CD using mkisofs and cdrecord.

3.2 Step 1: Splitting the sequences

The fast 1GHz machines from the FH were used for splitting the MPEG sequences. After inserting a CD that contained a sequence, the Script "create_dir.pl" split it into single images and wrote them to the NFS server. Each of the 11 minutes long MPEG video sequences were split into about 16.500 single images in JPEG format by the program "dumpmpeg" in about 45 minutes. Each JPEG image was about 24KB in size, which lead to about 400MB of JPEG image data for each MPEG sequence.

The program “dumpmpeg” that was invoked by the “create_dir.pl” script was modified for the Marathon Cluster, as it was only able to write BMP images initially, which would have led to too big images and thus wasted storage space, plus the software used in the second step of the cluster required JPEG as input. Because of this, the source code of the open source program dumpmpeg was modified so that after saving 250 images (10 seconds of video) they were converted from BMP to JPEG using the netpbm tools. Conversion with netpbm tools is rather slow, but there were no alternatives available in the SDL- and smpeg-libraries used by dumpmpeg that allowed direct saving in JPEG format. Optimizing the many fork(2)- and exec(2)-calls away by using routines from the netpbm-tools was not possible due to a tight timeframe for the project.

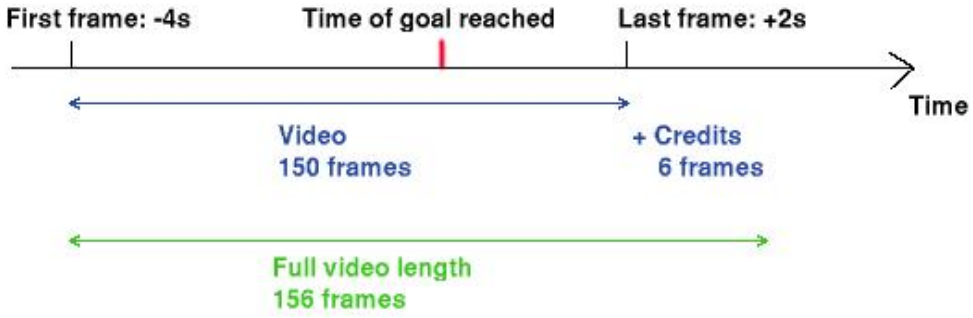
The second change that was made to dumpmpeg was the directory, in which the program created the images. Instead of placing all images into a single directory, 250 images were each placed in their own directory, improving access time.

3.3 Intermediate step: Enter image data into image database

For the further processing in step 2, exact start- and end-time of each MPEG sequence was needed, as well as the corresponding filenames of the first and last JPEG image. These data were stored in a MySQL database on the job control machine. Furthermore the actual frame rate was calculated, as the VCR device didn’t always provide a constant 25 images per second, e.g. due to heat and resulting mechanical inaccuracies. A tiny difference could increase massively over five hours of video material, leading to useless results when the exact images were needed for a certain time.

3.4 Step 2: Creating the videos

After the MPEG sequences were split into single images and stored on the NFS server in the first step, they were merged back into little videos in the second step. The goal was that each runner can watch a picture of themself reaching the goal giving their start number, and that each runner gets his individual MPEG video, rendered by the Marathon Cluster. Of the 7.000 runners who started the marathon, about 5.500 reached the goal. There were three disciplines: marathon (42km), half-marathon (21km) as well as speedskating (21km). For each discipline, separate lists of results for men and women existed, recording the time when they passed the goal. Based on this time and the corresponding image, we advance two seconds to the last image of the video, and from there we went back 150 frames that were copied into a work directory. These 150 frames are equal to six seconds of animation with 25 frames each.



Finally, six more frames were copied into the working directory that all had the same contents: Name of the runner, his start number, time, place as well as discipline were painted into a prepared mask displaying logo of the sponsors of the videos, using the program “convert” (part of ImageMagick). These six frames are then displayed at the end of the video:

Besides the images for the video showing the runner reaching the goal, the image of the exact time of him reaching the goal is also copied, and name of the runner, time, place and discipline are added via “convert”:

Rendering of the video from the single images in the temporary working directory was done with the program “mpeg_encode”. A config file described location of the working directory, which images to use for the video, and which client machines to use for rendering the video.

mpeg_encode first starts by calculating a few images on each of the client machines, to get an estimation on how fast the machines are. After that, the remaining images are distributed among the cluster nodes according to this speed estimate. The nodes read the images via NFS, and write rendered parts of the videos back via NFS, where the main process will pick them up and assemble them into the resulting MPEG file. All this is performed automatically by the freely available program without any need for manual interaction or tuning, which saves a lot of time and prevents possible errors.

The config file used by mpeg_encode was available for each of the four subclusters, so that we were able to render videos for each distinct discipline on a dedicated subcluster - depending on which one hat CPU time available. The job scheduling and distribution of discipline lists among the subclusters was done manually. There were six disciplines altogether:

Marathon women:	148 Finishers
Marathon men:	1268 Finishers
Halfmarathon women:	893 Finishers
Halfmarathon men:	2469 Finishers
Speedskating women:	202 Finisher
Speedskating men:	521 Finisher

The results was available in a separate list for each discipline in the form of a CSV file, which were split into ASCII files using a perl script. This was used as an input for another perl script “mpeg250_pic.pl”. For each runner, the input file contained the name, place as well as the time of them reaching the goal. The image database described above (see 3.3) was used to calculate the sequence in which the runner was, and from that the corresponding exact image of the runner reaching the goal was calculated. As described above, this served as a reference point for the 150 frames used for the individual videos. After the images including the ones for the credits images were copied to the temporary working directory, the client machines were used to render the individual MPEG for one runner. This individual MPEG as well as the exact image of the runner reaching the goal were then archived to a separate directory.

The program mpeg_encode which is started via “mpeg250_pic.pl” then started the job on the various (sub)clusters via rsh. rsh is used to prevent expensive authentication as used by ssh. As the rendering of a single MPEG was between 3 and 8 seconds, the overhead of SSH authentication - which is about 2 seconds - would have been to big, without any gain at all.

Example videos can be found at

- http://www.feyrer.de/marathon-cluster/video-50.mpg
- http://www.feyrer.de/marathon-cluster/video-2.mpg

4 Experiences gained from the Marathon-Cluster

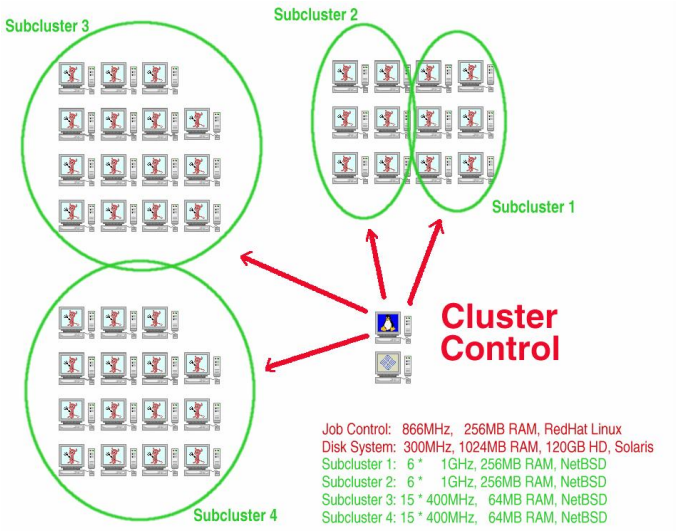
Deployment of the client machines took longer than expected. Installing a single client machine via g4u from the master server took about 30 minutes, copying the ca. 650MB

big image again after that (to prepare it as room server) took the same time. The following deployment to all machines from a single room took rather long, as 11 / 14 machines were fighting for bandwidth to the room server, plus all three rooms were connected via a single 100MBit 3Com switch, not one per room! A network design based on separate switches for each room would probably improved deployment times here.

dumpmpeg works on NetBSD and Linux, but not on Solaris/x86 using identical hardware. The cluster software was tested by R-KOM on Linux only, and it worked without problems there. On the Solaris based FH machines, dumpmpeg - which is based on SDL and smpeg - dumped core sporadically. Debugging with gdb showed that blocks of memory were overwritten, as the crashes happened in malloc(3). We guessed that the malloc(3) implementation by NetBSD and Linux are different than Solaris', so that possible overwritten memory blocks are treated less gracefully in the later case. In that case, the problem would have been located and fixed in the SDL- or smpeg-sources, but this would have taken more time than we had. The problem here was not Solaris itself, but the result was that we were not able to use the Solaris machines for the cluster, and we lost 15 valuable machines. With a bit more time for preparation and better tests, such bad effects can be avoided.

dumpmpeg ran longer then expected. The test sequence of 18 minutes length that was used for tuning and configuring the cluster took about 60 minutes to split on one of the 1GHz machines. Running the dumpmpeg process on several machines in parallel achieved big speed improvement, but there were also some shortages for network and disk resources. Especially reading and splitting the MPEG sequences was with eight hours about three hours longer than estimated.

mpeg_encode cannot render on an unlimited number of machines. A sequence of 156 images cannot be computed on more than about 15 machines. When trying to use more machines, errors are printed and the program hangs. To achieve maximum parallelism, the scripts that created the temporary working directories, filled them with images and ran mpeg_encode were changed to do so for different subclusters. The config files for mpeg_encode had to be adjusted to give work to several subclusters:

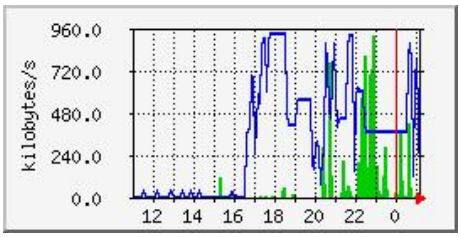


mpeg_encode sometimes stops after printing "Wrote 160 frames". There's no obvious

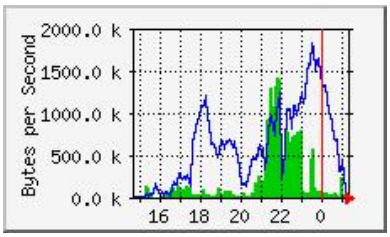
reason, and a quick view into the source code didn't lead to any enlightenment. Aborting the script was the fastest way of recovery here. The input list of runners to process just had to be edited so noone's processed multiple times.

5 Various images & graphs

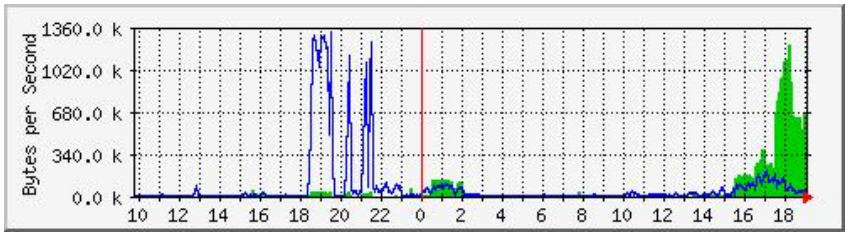
- Disk utilization of the NFS server shows that splitting of the sequences (blue) began at about 4:30pm, at 1:15am all sequences were split. Writes (green) show when the videos for the disciplines who went through the goal first - speedskaters - were rendered starting 10pm:



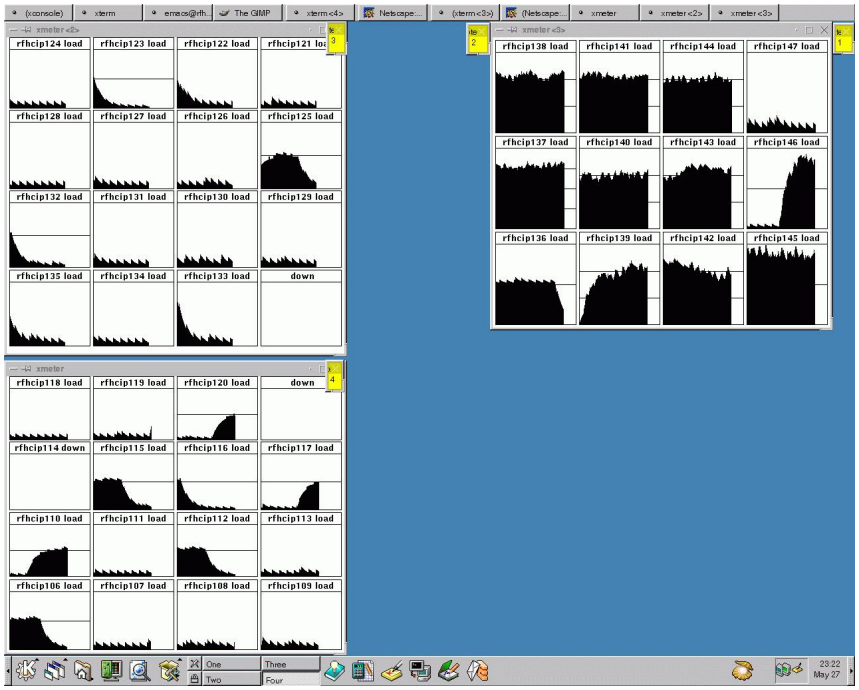
- Looking at the network traffic of the NFS server, it's very clear at which times the split images were written to disk (blue), and when they were read again for processing them for the speed skaters (green).



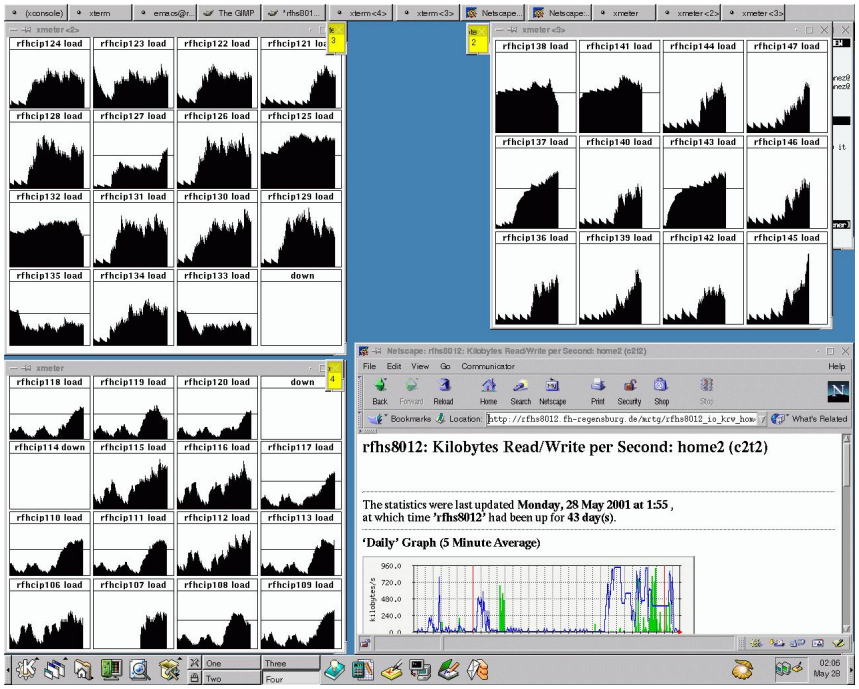
- Network traffic between the cluster machines and the control machine: while there were mostly read operations (blue) for the disk image during deployment of the clients on saturday between 18h and 22h, the cluster itself started producing data (green) on sunday at about 15h:



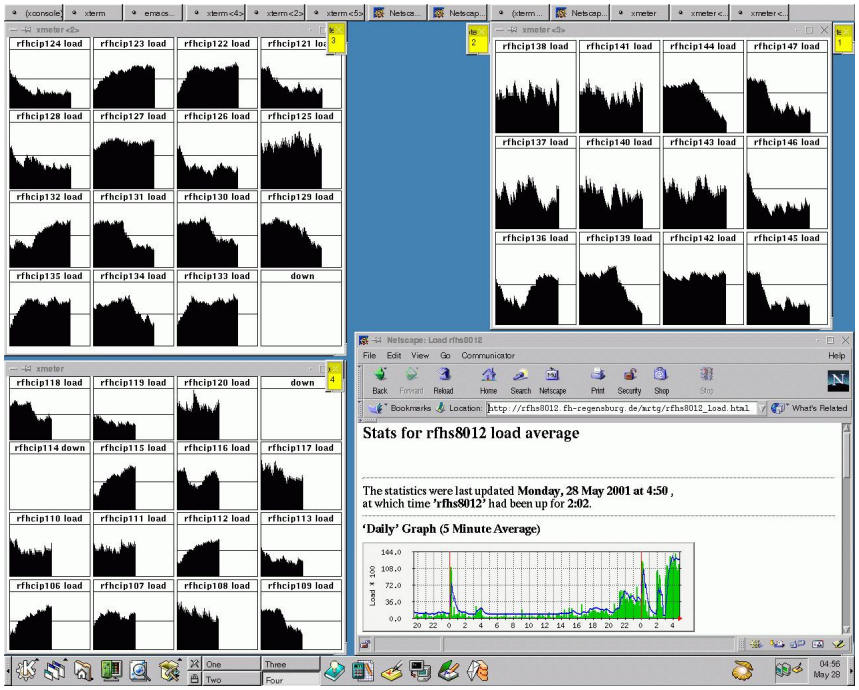
- Sunday, early afternoon: while the subcluster 3 and 4 (left side, yellow numbers) are still waiting for data, the GHz machines in subclusters 1 and 2 (upper right) are already splitting MPEG sequences into single images.



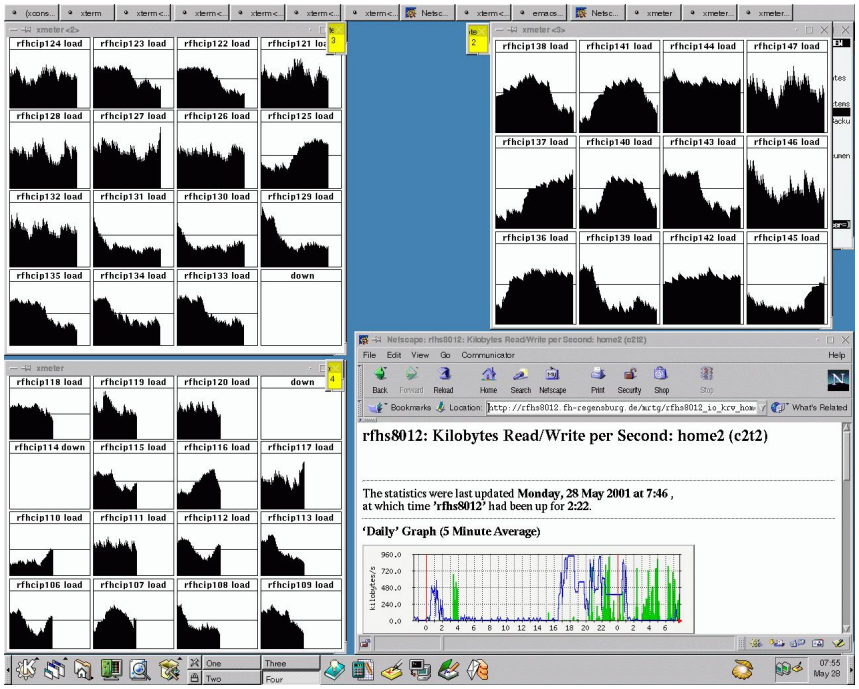
- Monday 2am: The last sequence is split, from now the second step can be run on all machines in all subclusters in parallel:



- Monday 5am: The NFS server running Solaris 2.6 had to be rebooted at 3am due to mysterious NFS/RPC/NIS problems. After that, the whole cluster is running with full steam again. Three hours from now the machines have to be available for students again!

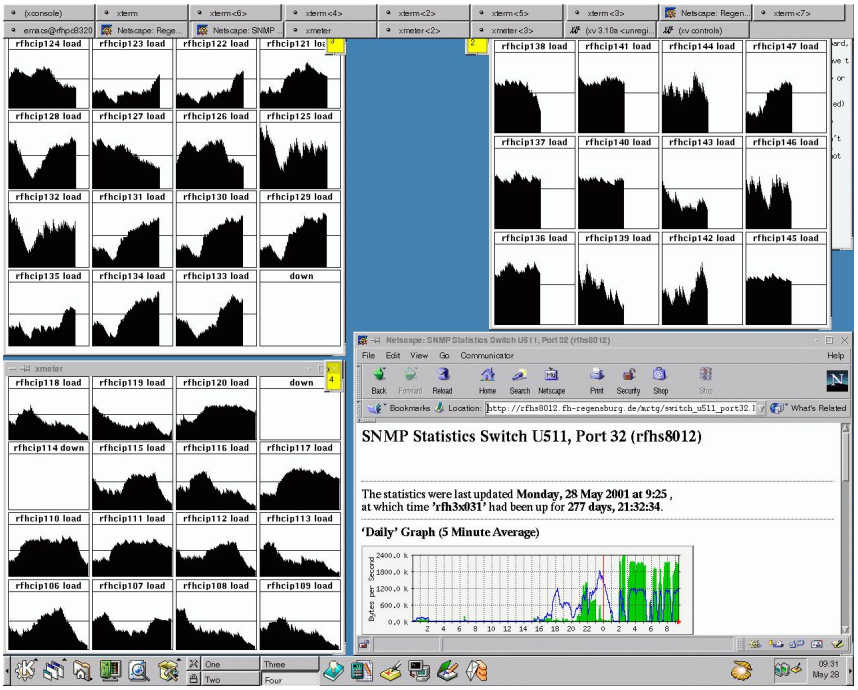


- Monday 5:30am: We just lost all the scripts by a lazily typed "`rm tempfiles *`" (note the second space!). A backup made the previous evening saved the project.



- Monday, 8am: In the past hours, several interrupts stopped the work up to an hour. Reasons were again mysterious phenomenons in the NFS/automounter area. (Or maybe booting network components - who knows?). Monday 8am there are still 900+500 halfmarathon- and about 500 marathon-videos to render, distributed over all subclusters. There are no lectures in the computer rooms until 11:30am, so we have a bit more time.

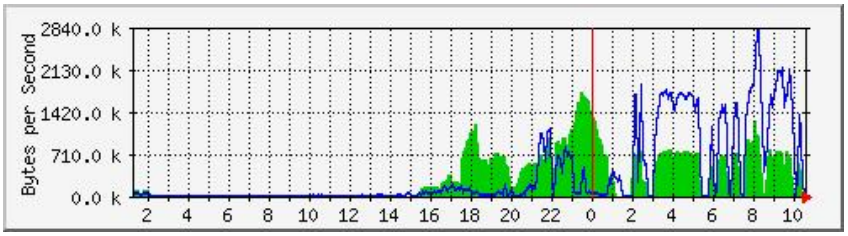
- Monday, 9:35am: After various NFS outages the clients run again with full load, the end is in sight!



- Monday, 10:40: Done! All result-lists are processed and spot checks on the resulting images and videos show good results.

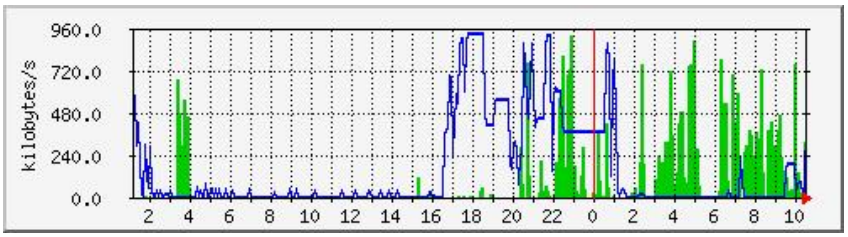
Here's an overview of all the performance graphs of the Regensburg Marathon Cluster:

- Traffic between control machine and cluster machines:



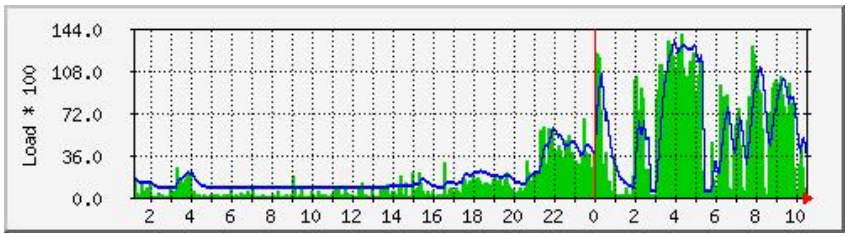
  The two steps can be easily distinguished here: writing the images from about 5pm to 1am, and both writing and reading from about 2am until the end at 10:40am.

- Disk utilization of the NFS server:



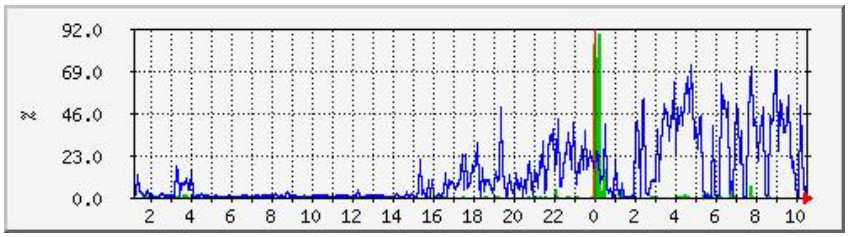
The difference between the first and second step is very obvious here. While there were only write operations while splitting the MPEGs into images (blue), they were read again for rendering the videos in the second step (green). While the first step was still running between 9pm and 11pm, the second step was running in parallel for rendering the speedskaters' videos.

- The system load (load average) of the NFS server:

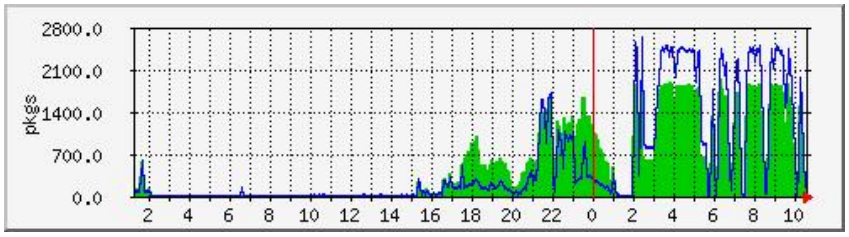


Step one (until about 2am) didn't put too much load on the NFS server, the high load at about midnight results from various cron job. Starting at about 2am the load increases due to parallel read- and write-jobs in the second computing-step of the cluster. Obvious are various interrupts at 3am, 5:30am and 8:30am.

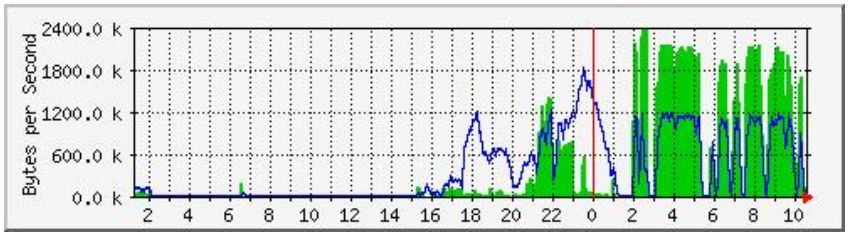
- The CPU utilization of the NFS server:



- Network utilization of the NFS server:



- Network traffic of the NFS server measured at the switch:



6 Facts

6.1 Subclusters

U512:	15	* 400MHz,	64MB RAM,	NetBSD 1.5.1_BETA2
U513:	15	* 400MHz,	64MB RAM,	NetBSD 1.5.1_BETA2
U521:	6	* 1000MHz,	256MB RAM,	NetBSD 1.5.1_BETA2
U521:	6	* 1000MHz,	256MB RAM,	NetBSD 1.5.1_BETA2

6.2 Cluster Control

noon:	1	* 866MHz,	256MB RAM,	RedHat 7.1
rfhs8012:	1	* 300MHz,	1024MB RAM,	Solaris 2.6, 120GB HD

6.3 Numbers

- Date of the Regensburg city marathon: Sunday May 27th 2001
- Female participants marathon: 148
- Male participants marathon: 1.268
- Female participants half-marathon: 893
- Male participants half-marathon: 2.469
- Female participants speedskating: 202
- Male participants speedskating: 521
- Participants overall: 5.501
- Available computers: 57, of which 15 were not usable (due to SDL/Solaris), and one had a broken floppy disk drive (=> no deployment possible)
- Computers participating in the cluster: 41
- Number of images after step #1: 669.936
- Number of reboots of the NFS server: 2
- Number of reboots per cluster client: 0
- Average size image of the runner reaching the goal (JPEG): 27 kB
- Average size video of the runner reaching the goal (MPEG): 987 kB
- Number of images of runners reaching the goal (JPEG): 5.501

- Number of videos of runners reaching the goal (MPEG): 5.501
- Time for deployment of the cluster: about 4 h (Sa 6pm to 10pm)
- Time for tuning and configuration of the cluster: about 4 h (Sat 10pm to Sun 2am)
- Time for setting up the VCR and other preparations: about 3.5 h (Sun 2pm to 5:30pm)
- Time for reading and splitting MPEG sequences (step 3): about 9 h (Sun 5:30pm to Mon 1am)
- Time for rendering videos and images: about 9 h (Mon 1am to 10:40am)
- Number of video tapes: 4
- Overall running time of video tapes: 5 h
- Number of MPEG sequences: 33
- Length per MPEG sequence: 11 min
- Uptime Hubert Feyrer: 22 h
- Uptime Jürgen Mayerhofer: 27 h

6.4 Software

- SDL 1.2.0, for dumpmpeg (http://www.libsdl.org/)
- smpeg 0.4.3, for dumpmpeg (http://www.lokigames.com/development/smjpeg.php3)
- dumpmpeg 0.6 (modified!): Converting MPEG->JPEGs (http://sourceforge.net/projects/dumpmpeg)
- netpbm 9.7, for dumpmpeg (http://netpbm.sourceforge.net/)
- mpeg_encode 1.5b: Converting JPEGs -> MPEG (http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_encode.html)
- perl 5.6.0: Scripting for job scheduling (http://www.perl.com/)
- gimp 1.2.1: Postprocessing of credits image, documentation (http://www.gimp.org/)
- Image Magick 5.2.7: Postprocessing of credits image and image of the runner reaching the goal (http://www.imagemagick.org/)
- tload 2.0.6 (modified): Monitoring cluster machines (ftp://people.redhat.com/johnsonm/procps/)
- xmeter 1.15: Monitoring cluster machines
- g4u 1.6: Client image deployment (http://www.feyrer.de/g4u/)
- NetBSD 1.5.1_BETA2/i386: Operating system of the cluster machines (http://www.netbsd.org/)

6.5 Participants

- Hubert Feyrer, Department of Computer Science , FH Regensburg:
  Design, implementation and execution of deployment of cluster clients, adaption of cluster software, documentation
- Jürgen Mayerhofer, R-KOM, Regensburg:
  Design and implementation of the cluster, coordination of videos and MPEG encoding
- Oliver Melzer, working student R-KOM & student FH Regensburg:
  Design and implementation of the cluster, coordination of videos and MPEG encoding
- Daniel Ettle, student of computer science (technical emphasis), FH Regensburg:
  Execution deployment
- Christian Krauss, student of computer science (technical emphasis), FH Regensburg:
  Execution deployment
- Tino Hirschmann, student of computer science (technical emphasis), FH Regensburg:
  Execution deployment
- Fabian Abke, student of computer science (technical emphasis), FH Regensburg:
  Execution deployment
- Udo Steinegger, Cable & Wireless, Munich:
  Team assistant

7 Links

- http://www.stadtmarathon-regensburg.de/:
  Web page of the Sports Experts Marathon
- http://www.stadtmarathon-regensburg.de/ergebnis_info.ph3:
  Results of the Sports Experts Marathon, including form for retrieving images and videos by starting number of the runner.
- http://www.r-kom.de/rkom-content-stadtmarathon2001.htm:
  Description of the intended project by R-KOM in the Marathon news paper
- http://www.feyrer.de/marathon-cluster/:
  Web site of this documentation about the actually performed project of the Regensburg Marathon Cluster.
- http://www.netbsd.org/:
  NetBSD, a free Unix/Linux-like operating system not only for PCs, which was used on the client machines of the marathon cluster.