Reaching the Goal with
the Regensburg Marathon Cluster
- A NetBSD Cluster Project -

Hubert Feyrer <hubert@feyrer.de>
Introduction

• 5,500 runners
• Cooperation between FH Regensburg and R-KOM
• 45 machines
• Video rendering
• 100% Open Source based
Cluster Client Setup: Hardware

• Four public rooms with 15 machines
• 15 machines with Solaris preinstalled
• Remaining machines available for reinstall
• Hardware: Dell OptiPlex PCs
  - PII-500MHz, 64MB RAM, 4GB harddisk
  - PIII-1GHz, 256MB RAM, 10GB harddisk
Cluster Client Setup: Software

- Chosen node OS: NetBSD
  - Supports the hardware
  - Easy to install
  - Know-how available in-house
  - Software available in 3rd party software collection

- Cluster software:
  - dumpmpeg, mpeg_encode
  - tload, ucd_snmp, statd

- Image cloning: g4u
Cluster Client Setup: Deployment
Tasks of the Cluster

Preparations: Videos are converted from tape into MPEG, and split into several sequences for easy splitting

Cluster Step 1: Splitting the MPEG stream into pictures

Cluster Step 2: Computing videos from single pictures

The result: 5500 images and 5500 videos rendered
Cluster Task #1: Splitting MPEG Sequences

• Splitting sequences of the input video into single images
• 11 minutes per sequence
• 16,500 resulting images
• 45 minutes on 1GHz machines
• Software: dumpmpeg
Cluster Task #1: Optimisations (I)

- dumpmpeg writes BMP per default
  - we needed JPG for the 2nd step
  - sizeof(BMP) >> sizeof(JPG)
- No JPEG-writing routines in SDL and smpeg
- Source code changed to use NetPBM tools
- After 250 BMPs written to disk,
  batch conversion to JPG in one run
Cluster Task #1: Optimisations (II)

- Replacing external calls (fork/exec are expensive) with NetPBM and jpeg lib functions not done (ENOTIME)
- Improving access times by placing 250 images each in their own directory
Intermediate Step

• For each sequence, record exact time of first and last image into a MySQL database
• Calculate actual framerate for this sequence
• Framerate is not always 25 frames/sec due to thermal effects and resulting mechanical inaccuracies
• A small difference could add up to unusable results over 5 hours of video material
Cluster Task #2: rendering videos (I)

• Render videos for each runner reaching the goal
• 5,500 runners (reaching the goal; >7,000 starters)
• Three disciplines:
  - Marathon (42km)
  - Half-marathon (21km)
  - Speed skating (21km)
• Separate lists of results for women and men
Cluster Task #2: rendering videos (II)

- Image selection:

![Diagram showing time of goal and video frames](image-url)

- Images were copied to a working directory
Cluster Task #2: rendering videos (III)

• Credit frames include data for the runner, written into a template:
Cluster Task #2: rendering videos (IV)

• Image of the runner reaching the goal:
Cluster Task #2: rendering videos (V)

• Software: mpeg_encode
• First send a few images to each machine, to estimate machine speed
• Distribute remaining images accordingly
• Images are read from NFS storage by the nodes
• Resulting video-parts are written back to NFS storage
• The master mpeg_encode process then collects and merges the video-parts at the end
* Cluster Task #2: rendering videos (VI)

- The available machines were split into four subclusters:

- Separate mpeg_encode config file for each subcluster
Cluster Task #2: rendering videos (VII)

• List of results was available as CSV file, containing name, place and time

• For each runner:
  - Prepare working dir with images
  - Render video
  - Store video
  - Store image of runner reaching the goal
Cluster Task #2: rendering videos (VIII)

- `mpeg_encode` used `rsh` (not `ssh`) for accessing the cluster nodes to prevent authentication overhead:
  - rendering MPEG: 3-8 s
  - ssh authentication: 2 s
Experiences

• Deployment took longer than expected
• dumpmpeg has problems on Solaris
• dumpmpeg ran longer than expected
• mpeg_encode doesn‘t scale infinitely
• mpeg_encode sometimes hangs
Experiences: Deployment

- Image size: 650MB
- Deployment of one image took about 30min (for setup of room server)
- Deployment of 11 / 14 machines from one room server took rather long (>2h) due to many machines fighting over network bandwidth and disk IO
- All client nodes were connected to the same switch, possible improvement: one switch per room
Experiences: dumpmpeg & Solaris (I)

- dumpmpeg worked fine on NetBSD and Linux
- dumpmpeg sporadically dumped core on Solaris
- some poking in gdb shows crashes in malloc(3)
- probably overwritten memory
- Guess: Solaris takes overwritten buffers more serious than NetBSD and Linux
- No quick fix was available, so we lost 15 machines!
- In retrospect, linking with libbsdsmalloc would probably have helped
Experiences: dumpmpeg & Solaris (II)

• With more time and testing on the real target platform, this could have been avoided.
• Not all the world is Linux!
Experiences: dumpmpeg too slow

• 18min test sequence took 60min to split w/ 1GHz
• For 12 machines running through 5 hrs of video input, we estimated 5 hours.
• In reality, the machines took 8 hours.
• Possible reasons here are related to disk IO on the local disk and NFS storage, network load etc.
Experiences: mpeg_encode & # of nodes

- A sequence of 156 images cannot be computed on more than about 15 machines
- As a result, we did split the available machines into several subclusters
- Minor adjustments of config files and handling scripts was needed
- Scheduling of which lists to run on which subcluster was done manually.
Experiences: mpeg_encode hangs

- After printing „Wrote 160 frames“, mpeg_encode
- sometimes hangs
- After some quick code inspection, there‘s no obvious
  reason what‘s happening.
- Workaround was to
  - ^C the program
  - edit the list of runners to process,
    removing the ones already done
  - restart the subcluster in question
Some stats

• Disk utilisation of the NFS server (\textit{write=blue, read=green}): 

![Disk utilisation graph]

• Network traffic between the cluster machines and the control machine (\textit{blue=client read, green=client write}): 

![Network traffic graph]
More stats (I)

- System load (load average) while splitting sequences:
More stats (II)

• The cluster running at full steam on all eng^Wnodes:
Some numbers

• Participants: 5,501
• Available computers: 57
• Running time of video tapes: 5 h
• Number of images after step #1: 669,936
• Diskspace of images after step #1: 17.5 GB
• Average size of image (JPEG): 27 kB
• Average size of video (MPEG): 987 kB
• Overall data images: 150 MB
• Overall data video: 5.4 GB
Software

- dumpmpeg: splitting MPEG into JPEGs
- mpeg_encode: rendering MPEGs from JPEGs
- SDL, smpeg, NetPBM: for dumpmpeg
- perl, gimp, ImageMagick: misc utilities
- tload, xmeter: node monitoring
- g4u: image deployment
- NetBSD: OS of the cluster client machines
The Marathon Cluster Team

- Hubert Feyrer
- Jürgen Mayerhofer
- Oliver Melzer
- Daniel Ettle
- Christian Krauss
- Tino Hirschmann
- Fabian Abke
- Udo Steinegger
Thanks!

Questions?

- Hubert Feyrer
  <hubert@feyrer.de>