The new NCLT Cluster
An Introduction

Joachim Wagner
November 2007

National Centre for Language Technology
School of Computing, Dublin City University

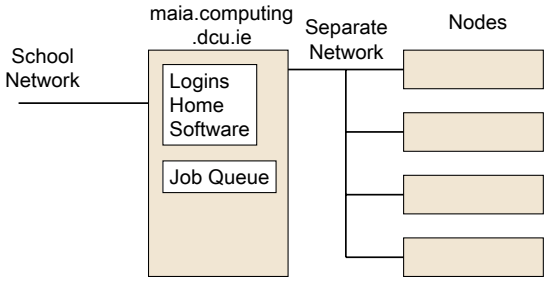
Talk Outline

- Why do we need a cluster?
- Architecture
  - Machines
  - Logins
  - Job management
- Running jobs
  - Commands
  - PBS Job discriptions
  - Taskfarming
- Plans

Why do we need a cluster?

- Resource conflicts
  - Waiting for colleague's job to finish
  - Trouble, e.g. disk full
- Medium-size jobs
  - Too big for desktop PC
  - Too small for ICHEC
- Preparation of ICHEC runs

Cluster Architecture



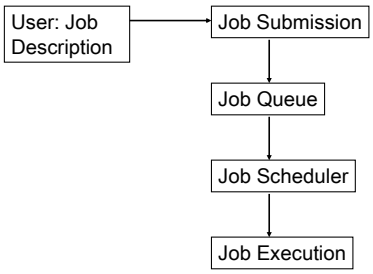
Installed Software

- OpenMPI
- SRILM
- MaTrEx, Moses, GIZA++
- XLE, Sicstus
- Johnson & Charniak's reranking parser
- In progress:
  - LFG AA, incl. function labeller

Machines

- Frontend (maia): 4 GB RAM
  - 2,500 GB disk space (home, database, backup)
  - RAID 5, occasional backup of /home
- 2 nodes with 8 GB RAM
- 2 nodes with 4 GB RAM
- All nodes and maia
  - 4 CPUs (2 x Dual Core Xeon 5110), x86_64
  - 24 GB Swap space
  - 30 GB /tmp space

Job Management



Nodes are allocated job-exclusive for the duration of the job

7

PBS Job Description

```
#! /bin/sh

# Common PBS variables
#PBS -l nodes=3:ppn=4
#PBS -N testjob
#PBS -M jwagner@computing.dcu.ie
#PBS -m eba
#PBS -l walltime=00:01:00

source ${HOME}/.bashrc

exp_dir=/home/jwagner/
cd ${exp_dir}

# run multiple copies
mpiexec -n 12 /home/jwagner/demo.py

# run single process
/home/jwagner/hello.py
```

Number of nodes CPUs/node

Notification: end, begin and abort

Maximum runtime

8

Job Management Commands

- qsub myjob.pbs
- – submit a job
  - PBS description: shell script with #PBS commands (ignored by shell)
- qstat, qstat –f jobnumber
- qdel jobnumber
- pbsnodes –a
  - list all nodes with status and properties

9

Example: Memory-Intensive Job

```
#! /bin/sh

# Common PBS variables
#PBS -l nodes=1:ppn=4:min8GB
#PBS -M jwagner@computing.dcu.ie
#PBS -m eba
#PBS -l walltime=00:01:00

source ${HOME}/.bashrc

exp_dir=/home/jwagner/
cd ${exp_dir}

# run single process
/home/jwagner/hello.py
```

10

Node Properties

- min4GB, min8GB: at least this much
- mem4GB, mem8GB: exactly this much
- x32, x64: 32 vs. 64-bit CPU
- Plans:
  - min3GB, mem3GB
  - CPU type and speed (per core)
  - Disk space (zeus, hera, etc.)

11

CPU-Intensive Jobs

- Parallelisable, for example
  - Sentences independently processed
  - Cross-validation runs
  - Parameter search
- Split into parts
  - Run each part on a different CPU

12

Example: CPU-Intensive Job (1)

```
#! /bin/bash

# Common PBS variables
#PBS -l nodes=2:ppn=4:min4GB
#PBS -N testxle5
#PBS -M jwagner@computing.dcu.ie
#PBS -m bea
#PBS -l walltime=01:30:00
#PBS -V

source ${HOME}/.bashrc

exp_dir=/home/jwagner/xle_parsing/
cd ${exp_dir}

mpiexec -n 8 /home/jwagner/taskfarm.py \
/home/jwagner/xle_parsing/xle.tfm
```

13

Example: CPU-Intensive Job (2)

- Taskfarming
  - .tfm: one task per line
  - taskfarm distributes tasks to nodes

```
/home/jwagner/xle_parsing/run-package.sh 000
/home/jwagner/xle_parsing/run-package.sh 001
/home/jwagner/xle_parsing/run-package.sh 002
/home/jwagner/xle_parsing/run-package.sh 003
/home/jwagner/xle_parsing/run-package.sh 004
/home/jwagner/xle_parsing/run-package.sh 005
/home/jwagner/xle_parsing/run-package.sh 006
/home/jwagner/xle_parsing/run-package.sh 007
/home/jwagner/xle_parsing/run-package.sh 008
/home/jwagner/xle_parsing/run-package.sh 009
/home/jwagner/xle_parsing/run-package.sh 010
/home/jwagner/xle_parsing/run-package.sh 011
/home/jwagner/xle_parsing/run-package.sh 012
```

14

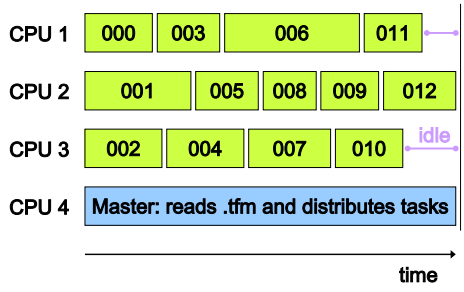
Example: CPU-Intensive Job (3)

run-package.sh

```
#!/bin/bash

PACKAGE=$1

# never use more than 995 MB
ulimit -v 995000

# keep time
touch /home/jwagner/xle_parsing/Files/$PACKAGE.start

# prepare data
bunzip2 /home/jwagner/xle_parsing/Files/$PACKAGE.bz2

# run parser
/usr/local/xle/bin/xle -noTk -e "create-parser /usr/local/sha
re/xle-grammars/english/english.lfg; parse-testfile /home/jwa
gner/xle_parsing/Files/$PACKAGE; exit" 2>/dev/null >/dev/null

# delete files not needed anymore
rm -f /home/jwagner/xle_parsing/Files/$PACKAGE
rm -f /home/jwagner/xle_parsing/Files/$PACKAGE.new

# compress results
bzip2 /home/jwagner/xle_parsing/Files/$PACKAGE.stats
```

15

Task distribution



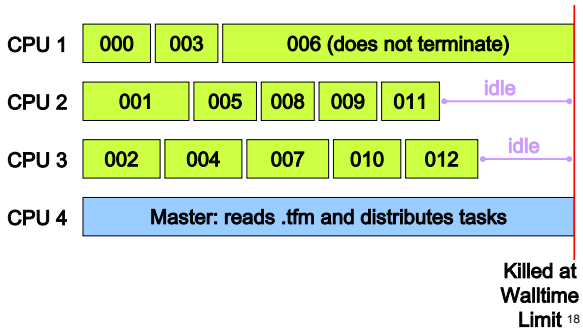
16

Effect of Task Size

- Job will wait for last task to finish (or be killed when walltime limit is reached)
- What if a task crashes?
  - Results are incomplete
  - Next tasks is executed
- What if a task does not terminate?
  - Results are incomplete
  - Fewer CPUs available for remaining tasks
- Overhead of starting tasks

17

Task distribution



18

Estimating Walltime

- Collect durations from test run
- Usually high variance of execution time
  - Long sentences
  - Parameters
- #packages x avg. time per package
  - High risk (~50 %) that more time is needed
- Random sampling with observed package durations: /home/jwagner/tools/walltime.py

19

Multiple CPUs per Node

- 8 GB node -> 2 GB per CPU (core)
- CPUs compete for RAM
  - Swapping of one task effects 3 other tasks
- Relatively slow CPUs in new nodes
- Optimise throughput of cluster / EUR
- Not: throughput of node or CPU
- Depends on application

20

Plans

- Fix sporadic errors of taskfarm.py
- XML-RPC-based taskfarming
  - Run master on maia
  - Run workers also outside the cluster
  - Set parameters at runtime
- Add new nodes
  - Next Generation Localisation project: up to 24
- Add old machines to cluster
  - Over the next 6 months
- Install additional software

21

Questions?

22