The new NCLT Cluster
An Introduction

Joachim Wagner
November 2007

Talk Outline

• Why do we need a cluster?
• Architecture
  – Machines
  – Logins
  – Job management
• Running jobs
  – Commands
  – PBS Job descriptions
  – 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

- User: Job Description
- Job Queue
- Job Submission
- Job Scheduler
- Job Execution

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

PBS Job Description

```bash
#!/bin/sh

# Common PBS variables
#PBS -l nodes=2:ppn=4
#PBS -N test_job
#PBS -M jwagner@computing.dcu.ie
#PBS -m a
#PBS -l walltime=00:10:00

source $HOME/.bashrc

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

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

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

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

Example: Memory-Intensive Job

```bash
#!/bin/sh

# Common PBS variables
#PBS -l nodes=2:ppn=4
#PBS -N jwagner_CPU_0000
#PBS -M jwagner@computing.dcu.ie
#PBS -m a
#PBS -l walltime=00:10:00

source $HOME/.bashrc

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

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

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.)

CPU-Intensive Jobs

- Parallelisable, for example
  - Sentences independently processed
  - Cross-validation runs
  - Parameter search
- Split into parts
  - Run each part on a different CPU
Example: CPU-Intensive Job (1)

```bash
#!/bin/bash

# Common PBS commands
#PBS -N testnodes
#PBS -M jwagner@comp.iol.ie
#PBS -q seab
#PBS -l walltime=01:30:00
#PBS -V

source $HOME/.bashrc

cmp_dir=/home/jwagner/xle_parsing/
cd $cmp_dir

mpirun -np 5 /home/jwagner/taskfarm.py \ 
/home/jwagner/xle_parsing/xle.tfm
```

Example: CPU-Intensive Job (2)

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

```plaintext
time
  000 CPU 1
  001
  002
  003
  004
  005
  006
  007
  008
  009
  010
  011

idle
  000
  001
  002
  003
  004
  005
  006
  007
  008
  009
  010
  011
```

Example: CPU-Intensive Job (3)

```bash
#!/bin/bash

# Taskfarm
#PBS -N testnodes
#PBS -M jwagner@comp.iol.ie
#PBS -q seab
#PBS -l walltime=01:30:00
#PBS -V

source $HOME/.bashrc

cmp_dir=/home/jwagner/xle_parsing/
cd $cmp_dir

mpirun -np 5 /home/jwagner/taskfarm.py \ 
/home/jwagner/xle_parsing/xle.tfm
```

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

```
run-package.sh
```

Task distribution

<table>
<thead>
<tr>
<th>CPU 1</th>
<th>CPU 2</th>
<th>CPU 3</th>
<th>CPU 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>001</td>
<td>002</td>
<td>Master reads .tfm and distributes tasks</td>
</tr>
<tr>
<td>003</td>
<td>005</td>
<td>004</td>
<td></td>
</tr>
<tr>
<td>006</td>
<td>008</td>
<td>007</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>009</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>012</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Killed at Walltime Limit
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

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

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

Questions?