Common Pitfalls of Using the Cluster

Joachim Wagner 2009-07-01

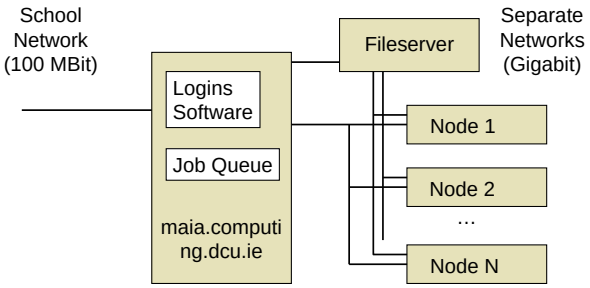
Outline

- Why do we need a cluster?
- Architecture: Machines and Properties
- Taskfarming
- Estimating Walltime Limits

Why do we need a cluster?

- More efficient and less costly (than high-end desktop PCs)
- Avoid 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
- Learning

Cluster Architecture



Node Properties (see command pbsnodes)

- min4GB, …, min32GB: at least this much
- mem4GB, …, mem32GB: exactly this much
- Partitions:
  - Switch1/2: which fileserver network + MPI communication switch
  - 2 groups of 8 and 4 groups of 4 (16 and 32 GB nodes)
  - Proposal: run short jobs in group4b and long jobs in group4d
- CPU type:
  - Intel Xeon E5440 quad core, 2.83 GHz, 6 MB cache
  - Intel Xeon E5420 quad core, 2.50 GHz, 6 MB cache
  - Intel Xeon 5110 dual core, 1.6 GHz, 4 MB cache
- CPU Cores:
  - Memory per core (example: mem2GBpercore and ppn = 4)
  - Number of cores (4 or 8)

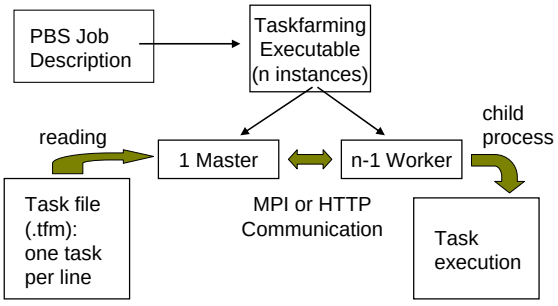
Selecting the Number of CPU cores (ppn)

- 4 or 8 CPU cores per node
- 1, 2 or 4 GB memory per core
- A single application may use more than 1 CPU
  - Java, C&J reranking parser, any sub-processes
- Limit memory usage
  - Command ulimit -v
- Processes compete for RAM
  - Swapping of one task effects the 3 other tasks
- If in doubt, reserve a full node
  - ppn=4:cores4 or ppn=8:cores8

CPU-Intensive Jobs

- Parallelisable, for example
  - Sentence by sentence processing
  - Cross-validation runs
  - Parameter search
- Split into parts
  - Run each part on a different CPU core
- Alternatives
  - Submit large number of jobs (ppn=1)
  - Taskfarming

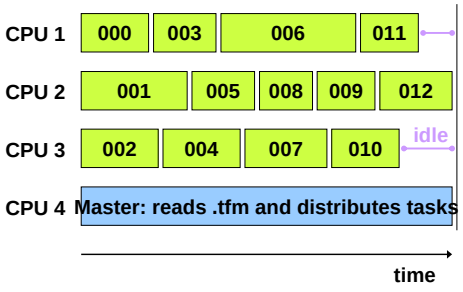
Taskfarming



Taskfarming Options

- Using individual PBS jobs
  - Can only allocate resources in multiples of 1/8 or ¼
    - Example: 3 GB task -> 4 GB job (ppn=2:cores8:mem16GB)
  - Floods job queue
- MPI-based taskfarming
  - All tasks inside one job
    - Example: 3 GB task -> 5 workers per 16 GB node
  - Master blocks one CPU core
- HTTP/XML-RPC-based taskfarming
  - Master runs on maia login node
  - Workers can run in multiple jobs
    - Example: 3 GB tasks -> one job with 5 workers for 16 GB nodes and one job with 8 workers for 32 GB nodes

Example: Taskfarming in Action



Estimating the PBS Walltime Parameter

- Collect durations from test run
- Usually high variance of execution time
  - Long sentences
  - Parameters
- Don't use #packages x avg. time per package
  - High risk (~50 %) that more time is needed
  - Prefix jobname with, for example, "24h-"
- Random sampling with observed package durations: /home/jwagner/tools/walltime.py

Questions

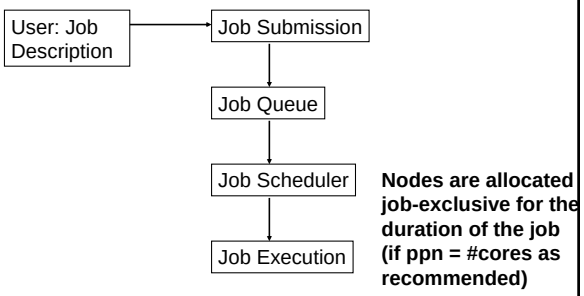
?

Contact:
Joachim Wagner
CNGL System Administrator
jwagner@computing.dcu.ie
(01) 700 6915

Installed Software

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

PBS Job Management



PBS Job Management Commands

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

PBS Job Description

```
#! /bin/sh

# Common PBS variables
#PBS -l nodes=5:ppn=8:mem16GB
#PBS -N test-mpi
#PBS -M jwagner@computing.dcu.ie
#PBS -m eba
#PBS -l walltime=00:05:00

source ${HOME}/.bashrc

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

# run multiple copies
mpiexec -n 40 /home/jwagner/intro/demo.py
```

Number of nodes #CPU cores/node

Notification: end, begin and abort

Maximum runtime

Number of processes to start

Example: Memory-Intensive Job

```
#! /bin/sh

# Common PBS variables
#PBS -l nodes=1:ppn=8:min32GB
#PBS -N my-test-job
#PBS -M jwagner@computing.dcu.ie
#PBS -m eba
#PBS -l walltime=01:30:00

source ${HOME}/.bashrc

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

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

Taskfarming Executable

- If Instance ID == 0
  - Run master code loop:
    - Read .tfm file (arg 1)
    - Send lines to worker
    - Exit if no more task and all worker finished
- Else
  - Run worker loop:
    - Ask master for a task
    - Execute task
    - Exit if master has no more tasks

Example: Taskfarming PBS File

```
#! /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
```

Example: Taskfarming TFM File

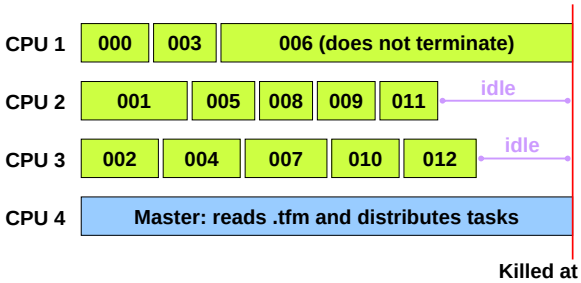
```
/home/jwagner/xle_parsing/run-package.sh 000
/home/jwagner/xle_parsing/run-package.sh 001
/home/jwagner/xle_parsing/run-package.sh 002
/home/jwagner/xle_parsing/run-package.sh 003
/home/jwagner/xle_parsing/run-package.sh 004
/home/jwagner/xle_parsing/run-package.sh 005
/home/jwagner/xle_parsing/run-package.sh 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
```

Example: Taskfarming Helper Script

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
```

Example: Non-Terminating Task



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