Index

1. BSC & RES Introduction

2. MareNostrum Installation

3. Slurm at BSC

4. Future
1. BSC & RES Introduction

• **BSC**: Barcelona Supercomputing Center

• **RES**: Supercomputing Network of Spain
  - Barcelona*
  - Madrid
  - Zaragoza
  - Valencia
  - Canarias (La Palma, Gran Canaria)
  - Málaga
  - Santander
1. BSC & RES Introduction

**RES**: Supercomputing Network of Spain
1. BSC & RES Introduction

- **BSC** leading and selecting technology
- Other nodes with BSC structure and technology but in a minor scale
- There are other national scientific centers under BSC supervision: CNAG (Genomics)
1. BSC & RES Introduction

**RES Usage per Site**

- **80%**: Access committee assigned projects
- Each project is assigned with:
  - Number of cpu hours
  - Class: A or B
- **20%**: Site own projects
- From time to time: Special Priority projects (private companies)
1. BSC & RES Introduction

Access committee assigned projects

- **Class_A**: high priority, max wclimit = 72 hours
- **Class_B**: low priority, max wclimit = 36 hours

- **BSC creates**:
  - **Class_C**: do you have time for me? Max wclimit = 24 hours
1. BSC & RES Introduction

On Site Projects

- Life Sciences
- Computer Science
- Earth Science
- Case Engineering
- Deisa (European Project)
- PRACE
- Equity in hours and priorities
Index

1. BSC & RES Introduction
2. MareNostrum Installation
3. Slurm at BSC
4. Future
2. MareNostrum Installation

- MareNostrum is the biggest RES machine
- First installation in 2005
- Then fifth in the TOP 500 but …
- largest machine (number of nodes)

- Hard test for cluster tools …
- Loadleveler and ganglia did not work smoothly…
2. MareNostrum Installation

Current MareNostrum Machine
2. MareNostrum Installation

Current MareNostrum Machine

- 2554 JS21 blades PowerPC 970MP
  - 4 Cores, 8GBytes memory
- SLES10
- Gigabit Network (Maintenance and File System)
- Myrinet Network: 10 switches + 2 spines
- GPFS
2. MareNostrum Installation

• LoadLeveler did not work for us so...
• Slurm was choosen but scheduling was not what we needed, so ...
  ➔ Moab as scheduling
  ➔ Slurm as resource manager
• We are quite* happy but we were happier with access to Moab source code

• Current Moab scheduling is slow but we do not know why. Ticket opened with Moab support...
Index

1. BSC & RES Introduction

2. MareNostrum Installation

3. Slurm at BSC

4. Future
3. Slurm at BSC

- Slurm partitions
  - Main partition: parallel jobs
  - 4 node partition: hsm (backup)
  - 4 node partition: interactive (login)
- Moab QoS used for RES requisites
- Moab Fair Sharing for ensuring assigned hours
- QoS parameters used:
  - WCLIMIT, MAXNODE, MAXPROC, MAXIJOB
3. Slurm at BSC

- Current Slurm version 2.1.9
- Mnsubmit wrapper inherited from loadleveler usage

  - Cpus & nodes requested
  - Wclimit validity
  - QoS
  - node features
  - Special flags: X11, perfminer(debug)
  - sequential jobs
3. Slurm at BSC

- Node features created dynamically
  - Slurmd initialization
  - When job starts
  - Periodically: Each six hours

- Prolog & epilog checking node state
  - File systems: GPFS
  - Myrinet
  - Memory, cpu count, free local disk
3. Slurm at BSC

• Accounting:

  • BSC accounting based on Moab with some little help from Slurm jobcomp.log file

  • A complex process inherited from loadleveler

  • Interested in slurmdbd: CNAG site using it.
3. Slurm at BSC

• Homemade plugins

  • X11 forwarding: Users need to debug a parallel job using some interactive program like TotalView

  • PerfMiner: Per thread information. Configuring cpu perf counters, mpi statistics, collecting data from every job node and sending data to a external DB

  • Bandwith memory: Bachelor’s Degree Thesis by Carles Fenoy (not in production)
3. Slurm at BSC

• X11 forwarding (spank plugin)

  ▪ Users specify a flag: `# @ x11=1` when submitting jobs using mnsubmit wrapper
  ▪ Wrapper sets a new ENV variable SPANK_X11 with:
    ▪ hostname (machine where user logins)
    ▪ DISPLAY value set by the user

  ▪ Slurm_spank_init: gets SPANK_X11 value and creates a Xauth file. It allows limiting users/groups using a file as a whitelist.

  ▪ ssh redirection done between master node and login node
3. Slurm at BSC

- Perfminer (spank plugin): getting app execution info
  - Activated by default for all jobs
  - Uses PAPIEX/PAPI to access hardware cpu counters
  - A configuration file determines which metrics/counters
  - User can choose between several metrics sets
3. Slurm at BSC

- Perfminer (spank plugin): getting app execution info
  - slurm_spank_init: gets PATH to config file
  - slurm_spank_user_init:
    - parses config file
    - sets ENV for PAPI/PAPIEX
  - slurm_spank_exit:
    - PAPIEX library modified sending data from nodes to master node
    - Master node process data from all nodes
    - Master sends data to perfminer server
  - Fini(): clean perfminer metrics directory
3. Slurm at BSC

• Memory bandwidth plugin

  • Resource selection plugin

  • Allocation based on job memory bandwidth requirements

  • New parameter to configure nodes available memory bandwidth
Index

1. BSC & RES Introduction

2. MareNostrum Installation

3. Slurm at BSC

4. Future
4. Slurm Future

Can we avoid Moab?
- We do not like black boxes

What do we need from Slurm?
- Is Slurm Scheduling ready for us?
  - BFChunk(Duration, Size)
  - nodeset by default
  - free local disk per node
  - ...

4. Slurm Future

Memory usage per node

- We do no have memory control by job
- Current swapping problems
- How is cgroup-slurm implementation going?
4. Slurm Future

**BLCR**: We want to use it!

- Which is the BLCR state? Last news from June 2009
- We use Myrinet MPI implementation now but ...
- We expect new machine with Infiniband …
- … and MPI libraries with BLCR support
- By now BLCR could be used for single node jobs
4. Slurm Future

Slurm Simulator

- Needed for testing changes in scheduling configuration

- The higher the scheduling complexity the higher the necessity of a simulator

- Example: “which is the checkpointing + preemption impact for a real 3 month job load?”
4. Slurm Future

Slurm Simulator: *some ideas*

- It should avoid (main) Slurm source code modifications
- Scheduling implementation should be unaware of it
- A simulator controller can take over slurmctl
- LD_PRELOAD for slurmctl time related functions connecting with the simulator controller
4. Slurm Future

Slurm Simulator: *some ideas*

- The simulator controller:
  - Reads from a trace file about jobs, nodes and reservations
  - Creates a time domain for slurmctl
  - Does job submission and reservation requests
  - Controls jobs completing time based on:
    - Trace file: job duration
    - Job starting time by slurmctl (simulation time)
  - Do we need a super slurmd?
4. Slurm Future

Slurm Simulator: *some ideas*

Super slurmd:

- Just one slurmd for all the nodes (cheating slurmctl)
- No jobs execution, no slurmstepd, no jobs errors

- Responding:
  - SLURM_OK
  - REQUEST_COMPLETE_BATCH_SCRIPT
  - MESSAGE_EPILOG_COMPLETE

- It needs to interact with the simulator controller
Thank you