How to use Sophia's cluster

Nicolas Niclausse nicolas.niclausse@sophia.inria.fr

INRIA Sophia Antipolis - Service DREAM

June 20, 2005

Outline

Outline

Outline

Outline

Outline

Introduction

- Cluster = parallel computer built from multiples computers and an interconnection network
- Price/performance ratio much more affordable than big SMP supercomputers
- Resource sharing across users done by a *batch scheduler*
- Standard API to build parallel programs = MPI (but also OpenMP, PVM, ProActive, etc.)

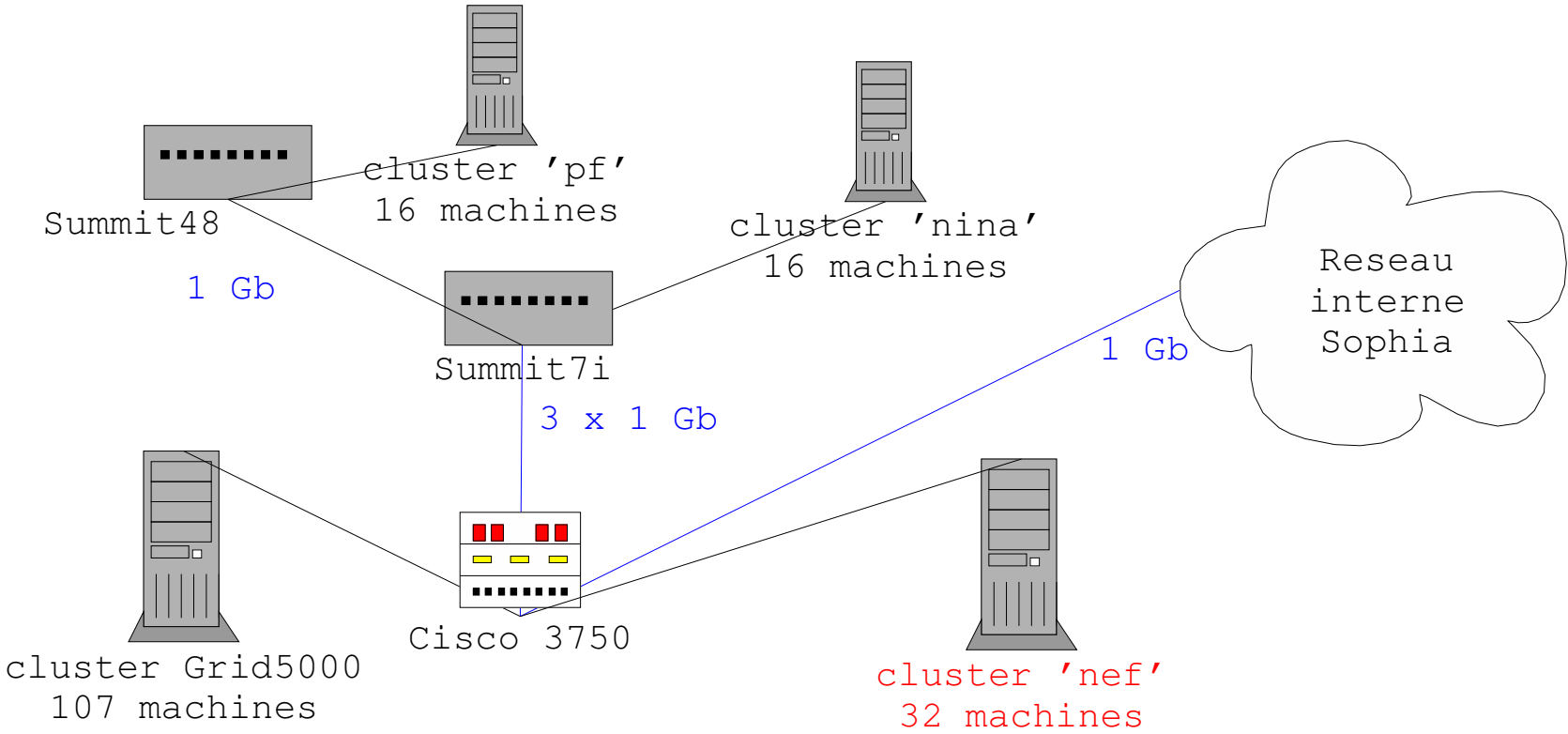
Outline

Current platform

- 2 clusters of PC :
  - Grid'5000 (experimental GRID platform)
  - PACA, divided into two groups of machines:
    1. built upon 16 PC Bi-Pentium 3 (`pf`)+ 16 PC Bi-Xeon (`nina`)
    2. the new one, built upon 32 Bi-Opteron nodes (`nef`)

    Each group has its own batch scheduler (LSF and Torque)
- Interconnection:
  - 1Gbit/s link to Grid'5000
  - 3Gbit/s link between the nina/pf and nef
  - 1Gbit/s link to INRIA-Sophia's network

Network


cluster 'pf'
16 machines
Summit48
1 Gb
cluster 'nina'
16 machines
Reseau
interne
Sophia
Summit7i
3 x 1 Gb
1 Gb
cluster Grid5000
107 machines
Cisco 3750
cluster 'nef'
32 machines

Feb 2005, new hardware: 140 machines

- 107 nodes reserved for GRID'5000
- 32 nodes for local use (PACA cluster)

Nef cluster

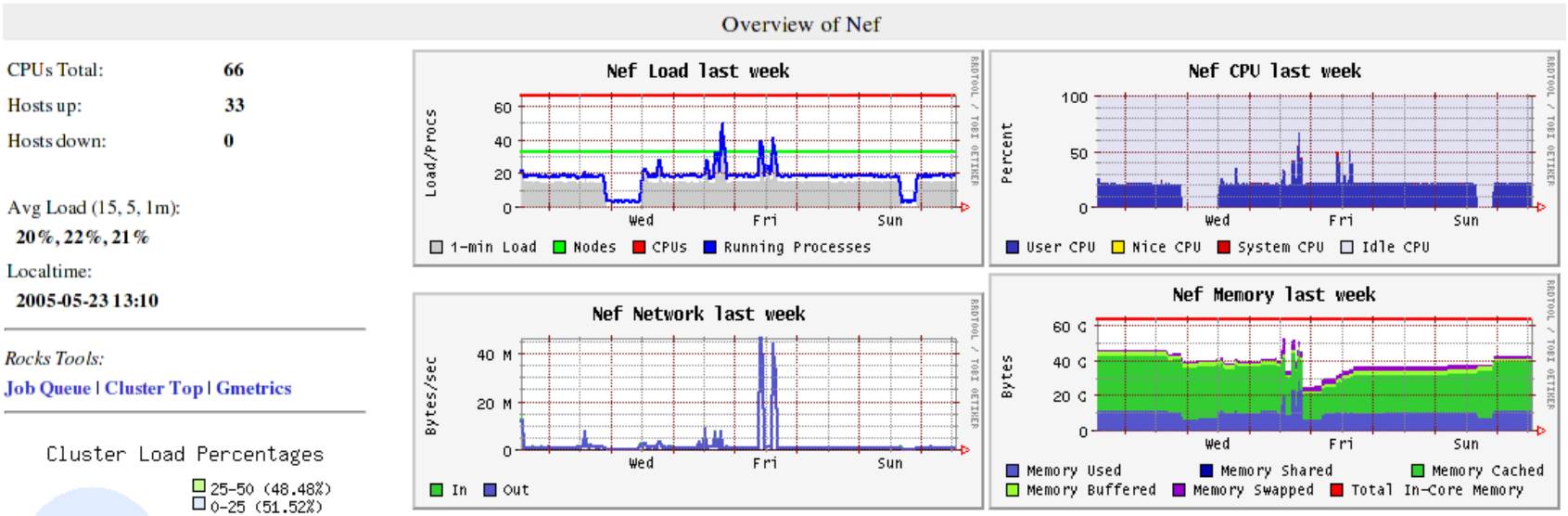
- One front-end (nef.inria.fr)
- 32 nodes
- 2 Opteron 246 CPU (2Ghz) per node
- 2 GB of RAM (1GB per processor)
- 80GB disk per node, 140GB on the front-end
- Gigabit Ethernet network
- Peak performance: $32 * 2 * 4 = 256$ Gflops

Opteron specifications

- L2 cache: 1MB
- 64 bit architecture
- i386 binary compatibility (32 bit) (without loss of performance)
- Each CPU has its own memory and direct high speed access to other CPU's memory (NUMA system)

View on the system and network activity

https://nef.inria.fr/ganglia/



Outline

Operating System

- ROCKS distribution, based on RedHat Advanced Server
- Support for a mixed 32/64 bit system
- User accounts are served over NFS (server = front-end, 123GB available for users)
- All software is locally installed on each node
- 60GB available on each node (`/usertmp/username`) for temporary data
- Access to the front-end = ssh with RSA/DSA key (no password !)

Development tools

Languages:

- gcc, g++, g77 (version 3.2.3)
- PGI compiler (version 5.2)
  - 32 & 64 bit
  - C, C++, Fortran 77/90/95
  - MPI, OpenMP, LAPACK, SCALAPACK, BLAS
- JDK Java 1.5
- Perl, Python, Erlang/OTP, ocaml

Development tools

Libraries, tools:

- Paraview, gnuplot
- MPI and PVM Libraries, XMPI (for LAM/MPI)
- Scientific libraries:
  - ATLAS 3.6.0 (BLAS + LAPACK)
  - libgoto (highly optimized BLAS)
  - ACML 2.5.0 (AMD's BLAS, LAPACK & FFT)
  - GSL 1.5 (GNU Scientific Library)
  - PETSc 2.3.0

Missing a tool ? `clustermaster@sophia.inria.fr`

32/64 bit environment

- By default, 64 bit compilations
- GCC: add -m32 to build 32 bit binaries
- PGI:
  - 64 bit: `/usr/local/pgi/pgi_user.sh`
  - 32 bit: `/usr/local/pgi/pgi_user_i386.sh`
- 64 bit libraries are in /usr/lib64, /lib64, /usr/X11R6/lib64, etc.)

Outline

MPI: overview

- MPI = *Message Passing Interface*
- Standard for high performance message passing on parallel machines
- SPMD
- 2 versions of the standard: MPI-1 (1.2) and MPI-2
  - new in MPI-2: dynamic processes, parallel I/O, support for F95 and C++ , etc.
- Upward compatibility

MPI implementations

- LAM-MPI (MPI-1.2 and a big part of MPI-2)
  - 64 bit version in `/opt/lam/gnu`
  - 32 bit version in `/opt/lam/gnu32`
- MPICH (MPI-1)
  - 64 bit version in `/opt/mpich/gnu`
  - 32 bit version in `/opt/mpich/gnu32`
- MPICH 2 (MPI-1 & MPI-2)
  - 64 bit version in `/opt/mpich2/gnu`
- OpenMPI new standard implementation, no stable release yet
- `http://www-sop.inria.fr/parallel/benchmarks/`

MPI implementations

- LAM-MPI (MPI-1.2 and a big part of MPI-2)
  - 64 bit version in `/opt/lam/gnu`
  - 32 bit version in `/opt/lam/gnu32`
- MPICH (MPI-1)
  - 64 bit version in `/opt/mpich/gnu`
  - 32 bit version in `/opt/mpich/gnu32`
- MPICH 2 (MPI-1 & MPI-2)
  - 64 bit version in `/opt/mpich2/gnu`
- OpenMPI new standard implementation, no stable release yet
- `http://www-sop.inria.fr/parallel/benchmarks/`

MPI implementations

- LAM-MPI (MPI-1.2 and a big part of MPI-2)
  - 64 bit version in `/opt/lam/gnu`
  - 32 bit version in `/opt/lam/gnu32`
- MPICH (MPI-1)
  - 64 bit version in `/opt/mpich/gnu`
  - 32 bit version in `/opt/mpich/gnu32`
- MPICH 2 (MPI-1 & MPI-2)
  - 64 bit version in `/opt/mpich2/gnu`
- OpenMPI new standard implementation, no stable release yet
- `http://www-sop.inria.fr/parallel/benchmarks/`

MPI implementations

- LAM-MPI (MPI-1.2 and a big part of MPI-2)
  - 64 bit version in `/opt/lam/gnu`
  - 32 bit version in `/opt/lam/gnu32`
- MPICH (MPI-1)
  - 64 bit version in `/opt/mpich/gnu`
  - 32 bit version in `/opt/mpich/gnu32`
- MPICH 2 (MPI-1 & MPI-2)
  - 64 bit version in `/opt/mpich2/gnu`
- OpenMPI new standard implementation, no stable release yet
- `http://www-sop.inria.fr/parallel/benchmarks/`

MPI implementations

- LAM-MPI (MPI-1.2 and a big part of MPI-2)
  - 64 bit version in `/opt/lam/gnu`
  - 32 bit version in `/opt/lam/gnu32`
- MPICH (MPI-1)
  - 64 bit version in `/opt/mpich/gnu`
  - 32 bit version in `/opt/mpich/gnu32`
- MPICH 2 (MPI-1 & MPI-2)
  - 64 bit version in `/opt/mpich2/gnu`
- OpenMPI new standard implementation, no stable release yet
- http://www-sop.inria.fr/parallel/benchmarks/

MPI examples

```
#include <stdio.h>
#include <mpi.h>
int main (int argc, char *argv []) {
    int size, rank, i , pid;
    int bcast_value = 1;

    MPI_Init (&argc, &argv);
    MPI_Comm_rank (MPI_COMM_WORLD, &rank);
    MPI_Comm_size (MPI_COMM_WORLD, &size);

    if (!rank) { bcast_value = 42; }
    MPI_Bcast (&bcast_value, 1 ,MPI_INT, 0, MPI_COMM_WORLD );

    printf("%d - %d - %d\n", rank, size, bcast_value);
    MPI_Finalize ();
}
```

MPI compilation

- LAM/MPI
  - `/opt/lam/gnu/mpicc -o prog prog.c`
- MPICH
  - `/opt/mpich/gnu/mpicc -o prog prog.c`

  or
  - `gcc -o prog prog.c -L/opt/mpich/gnu/lib -I/opt/mpich/gnu/include -lmpich`

  with PGI, there is a built-in MPICH library:
  - `pgcc -o prog prog.c -lmpich`
  - shared memory support not enabled (use LAM for better performance)

Outline

OpenMP: overview

Multi-platform shared-memory parallel programming

- A set of compiler directives and library routines for parallel applications
- Parallelization directives = comments in the source code
- *Fork and join* model: master thread spawns a team of threads when needed
- Only useful on SMP computers
- C/C++ and Fortran

OpenMP vs MPI

OpenMP vs MPI

- \+ Incremental parallelization
- – Only for SMP machines
  - ⇒ hybrid programming with OpenMP + MPI
    `http://www.lri.fr/~gk/QUID/papers/SPAA2003.pdf`
  - ⇒ OpenMP on top of SSI cluster (Kerrighed)
    `http://www.inria.fr/rrrt/rr-4947.html`

OpenMP: example

```
#include <stdio.h>
#include <omp.h>
main(){
     int i,b[10];
#pragma omp parallel for
     for (i=0; i<10; i++) {
          b[i]=i;
          printf("thread=%d, iter=%d\n",
                    omp_get_thread_num(),i);
}}
```

OpenMP: compilation & execution

How to use OpenMP:

Only available with PGI (Omni OpenMP not yet installed)

compilation: `pgcc -mp -o prog prog.c`

execution: `OMP_NUM_THREADS=2 ./prog`

Outline

Batch scheduler: Overview

Torque+Maui

- LSF: license/processor, too expensive
- Torque + Maui: open source, same functionalities.

  Torque: resource manager (based on OpenPBS)

  Maui: scheduler
- You must use the batch scheduler (on the front-end) to start jobs on the cluster
- Each job has an expected duration (walltime)
- You start a job by using a script or interactively

Basic commands

Basic commands

qsub : submit a job

qstat : see current jobs in the queues

qdel : kill/cancel a job

qpeek : print the job output (stdout or stderr) during execution

showstart : expected starting time of a job

showq : number of active processors & jobs, jobs status

Queues

Available queues:

short	less than 15min
normal	more than 15min but less than 2 hours
long	more than 2hrs but less than 2 days
idle	sequential or very long jobs (may be suspended by the batch scheduler if new jobs are queued with higher priority)

Torque script example: sequential job

```
# 1 node with one processor per node
#PBS -l nodes=1:ppn=1
# job duration less than 10min
#PBS -l walltime=10:00

# go to the directory where the script is, By default, torque
# starts the job in the user's home directory !
cd $PBS_O_WORKDIR

# path to the binary
./helloWorld
```

Torque script example: MPICH job

```
# 8 nodes with one processor per node
#PBS -l nodes=8:ppn=1
# job duration less than 10min
#PBS -l walltime=10:00

cd $PBS_O_WORKDIR

# use mpiexec to starts MPICH jobs
mpiexec ./MPIprog
```

Torque script example: LAM/MPI job

```
# 8 nodes with two processors per node
#PBS -l nodes=8:ppn=2
# job duration less than 10min
#PBS -l walltime=10:00

cd $PBS_O_WORKDIR

# you must have /opt/lam/gnu/bin in your PATH
lamboot
mpirun C ./MPIprog
lamhalt
```

Job submission

Submit:

batch mode: `qsub myscript.pbs`

interactive mode: `qsub -I -l nodes=4:ppn=2 /bin/bash`

Available variables: (man qsub for more)

PBS_NODEFILE the nodes available. One entry (hostname) per processor (duplicated if SMP)

PBS_O_WORKDIR where the job has been submitted

Job submission

Submit:

batch mode: `qsub myscript.pbs`

interactive mode: `qsub -I -l nodes=4:ppn=2 /bin/bash`

Available variables: (man qsub for more)

PBS_NODEFILE the nodes available. One entry (hostname) per processor (duplicated if SMP)

PBS_O_WORKDIR where the job has been submitted

Job submission

Tips:

- To add argument(s), use environment variables and -v:
  `ARG1=foo; ARG2=bar`
  `qsub -v ARG1,ARG2 myscript.pbs`
  and use $ARG1 and $ARG2 in myscript.pbs
- Interactive mode: the script is not executed:
  `cd PBS_O_WORKDIR`
  `./myscript.pbs`

Job submission

Tips:

- To add argument(s), use environment variables and -v:
  `ARG1=foo; ARG2=bar`
  `qsub -v ARG1,ARG2 myscript.pbs`
  and use $ARG1 and $ARG2 in myscript.pbs
- Interactive mode: the script is not executed:
  `cd PBS_O_WORKDIR`
  `./myscript.pbs`

Outline

Links & References

- http://www-sop.inria.fr/parallel/
  Web page for Sophia's Cluster.
- http://www.idris.fr/docs/docu/support_cours/
  Cours de l'IDRIS.
- http://www.lifl.fr/west/courses/cshp/
  Cours de l'LIFL.

M.J. Quinn
Parallel Programming in C with MPI and OpenMP.
McGrawHill 2004.

Demonstration

- OpenMP example
- MPI example
- MPI job submission with torque