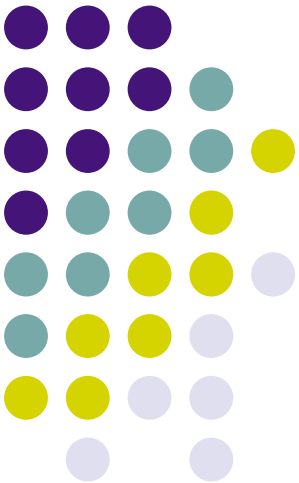
Curso de Computación Científica en Clusters
Administración de Sistemas Operativos III

Luis P. García González
Servicio de Apoyo a la Investigación Tecnológica

Universidad Politécnica de Cartagena



MPI: Introducción

- MPI (Message Passing Interface)
- Implementaciones:
  - **MPICH:** Versiones MPICH1 y MPICH2. Open-source y licencia libre. Preferiblemente se instalará **MPICH2**, a no ser que estemos en un cluster heterogéneo con equipos en los que hay diferentes representación de datos. Desarrollado en el Argone National Laboratory -> MPI Intel, MPI IBM, Cray, Microsoft, Ohio State University (MVAPICH), Myricom, Univ. British Columbia (MPI-STCP).
  - **LAM-MPI** y **Open-MPI:** Ohio Supercomputing Center y posteriormente por el Laboratory for Scientific Computing (University of Notre Dame) actualmente en la Indiana University.

MPI: Introducción

¿Qué hay que hacer desde el punto de vista del administrador del cluster para computación científica?

- **CONFIGURAR E INSTALAR:** LO MISMO DE SIEMPRE O CASI. Ya que habrá que indicar los dispositivos, channels o fabrics (Redes de Interconexión) disponibles en el sistema (Memoria, Ethernet, Myrinet, Infiniband, etc.).
- **COMPROBAR:** Que la instalación funciona en el sistema.

- **INTEGRACIÓN:** Con los compiladores y el gestor de recursos disponible en el cluster.

MPICH2
Configuración/Instalación/Integración

1. Prerequisitos: Lenguaje Python (versión 2.2) para el systema de gestión de procesos MPI.
2. Descargar MPICH2 desde: http://www.mcs.anl.gov/research/projects/mpich2/downloads/tarballs/1.2.1p1/mpich2-1.2.1p1.tar.gz
3. Extraer el paquete descargado, configurarlo y compilarlo:

```
$ tar zxvf mpich2-1.2.1p1.tar.gz
$ cd mpich2-1.2.1p1
$ ./configure --with-device=ch3:nemesis --disable-cxx --prefix=/
apps/mpich2-1.2.1p1/gnu/4.4.1 2>&1 | tee c.txt    # Unos 5 minutos
$ make 2>&1 | tee m.txt                            # Unos 6 minutos
```

4. Instalar:

```
$ su -          # Requiere instalación como root
# make install
```

MPICH2
Configuración/Instalación/Integración

5. Comprobación inicial en máquina local:

```
$ vi $HOME/.mpd.conf
secretword=lp67-g10g
$ export PATH=/apps/mpich2-1.2.1p1/gnu/4.4.1/bin:$PATH
$ mpd &
$ mpdtrace
$ mpdallexit
```

6. Comprobación lanzando un programa no MPI:

```
$ mpd &
$ mpiexec –n 1 /bin/hostname
$ mpdallexit
```

7. Comprobación usando varios nodos:

```
$ cat > mpd.hosts              # ccc-server ccc-nodo1
$ mpdboot –n 2 –f mpd.hosts
$ mpdtrace
$ mpdringtest 100
$ mpiexec –l –n 2 hostname
$ mpdallexit
```

MPICH2
Configuración/Instalación/Integración

8. Integración con PBS/Torque:

```
#!/bin/bash
#PBS –lnodes=2:ppn=2
#PBS –lwalltime=00:05:00

export PATH=/apps/mpich2-1.2.1p1/gnu/4.4.1/bin:$PATH
MPICH2_CONF=$HOME/.mpd.conf
export PATH=/apps/mpich2-1.2.1p1/gnu/4.4.1/bin:$PATH
if [ ! –r $MPICH2_CONF ] ; then
   SECRET=`mkpasswd –l 7 –s 0`
   echo “secretword=$SECRET” > $MPICH2_CONF
   chomod 600 $MPICH2_CONF
fi
NP=`wc –l $PBS_NODEFILE | awk ‘{print $1}’`
NN=`uniq $PBS_NODEFILE | wc –l | awk ‘{print $1}’`
uniq –c  $PBS_NODEFILE | awk ‘{printf(“%s:%s\n”, $2, $1);}’ > mpd.hosts
cd $PBS_O_WORKDIR
mpdboot –f mpd.hosts –n $NN
mpiexec –n $NP ./cpi
mpdallexit
```

OpenMPI
Configuración/Instalación/Integración

1. Prerequisitos: Lenguaje C++.
2. Descargar Open-MPI desde:

```
$ bzip2 –dc openmpi-1.4.1.tar.bz2 | tar xvf -
$ cd openmpi-1.4.1
$ ./configure --with-tm=/usr/local --prefix=/apps/openmpi-1.4.1/
gnu/4.4.1 2>&1 | tee c.txt    # Unos 10 minutos
$ make 2>&1 | tee m.txt       # Unos 20 minutos
```

3. Instalar:

```
$ su -
# make install
```

4. Comprobar que se incluye soporte para el *Task Manager* de Torque:

```
$ PATH=/apps/openmpi-1.4.1/bin:$PATH
$ LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/apps/openmpi-1.4.1/lib
$ ompi_info | grep tm
```

OPENMPI
Configuración/Instalación/Integración

5. Comprobación inicial en máquina local:

```
$ export PATH=/apps/openmpi-1.4.1/gnu/4.4.1/bin:$PATH
$ export LD_LIBRARY_PATH=/apps/openmpi-1.4.1/gnu/4.4.1/lib:$
{LD_LIBRAY_PATH}
$ mpicc –o hello_c hello_c
$ mpiexec –n 2 ./hello_c
```

6. Comprobación con el gestor de colas Torque:

```
#!/bin/bash
#PBS –lnodes=2:ppn=2,walltime=00:02:00

export PATH=/apps/openmpi-1.4.1/gnu/4.4.1/bin:$PATH
export LD_LIBRARY_PATH=/apps/openmpi-1.4.1/gnu/4.4.1/lib

cd $PBS_O_WORKDIR
mpiexec ./hello_c
```

7. Creación de OpenMPI para usar con los compiladores de Intel:

```
$ ./configure CC=icc CXX=icpc F77=ifort FC=ifort ...
```

BLAS/LAPACK: Introducción
API Estándar

- BLAS (Basic Linear Algebra Subprogram)
- LAPACK (Linear Algebra PACKage)
- Implementaciones:
  - **REFERENCIA:** http://www.netlib.org/{blas,lapack}
  - **OPTIMIZADAS PARA UNA PLATAFORMA**: Por un fabricante para sus máquinas
    - AMD: ACML (Opteron en Windows y Linux)
    - Apple: Velocity Engine (G4, G5, PowerPC en Mac OSX)
    - HP-Compaq: CXML (Alpha en Tru64 Unix y Linux-Alpha)
    - Cray: xt-libsci (Opteron en Cray Linux Environment)
    - HP: MLIB (HP PA-RISC 2.0 e Itanium en HP-UX)
    - Intel: MKL (Intel en Windows, Linux y Mac OSX)
    - SUN: SPL (Intel, SPARC en Solaris y Linux)
    - IBM: ESSL (POWER y PowerPC en AIX)

BLAS/LAPACK

- Optimizadas Multiplataforma:
  - **ATLAS:** Automatically Tuned Linear Algebra Software): Un subconjunto de BLAS/LAPACK. Aplican técnicas empíricas para obtener rendimineto portable.

    http://math-atlas.sourceforge.net

  - **GotoBLAS:** Versión optimizada de BLAS desarrollada por Kazushige Goto en el Texas Advanced Computing Center en la Universidad de Texas en Austin.

    http://wwww.tacc.utexas.edu

BLAS/LAPACK
Instalación ATLAS

1. Prerequisito para el “auto-tunning”: Desactivar Gestión de Energía y disponer de un compilador de Fortran:

```
$ su -
# /usr/bin/cpufreq-selector –g performance
# yum install gcc-gfortran
```

2. Última versión 3.9.23. Descargar de: http://sourceforge.net/projects/math-atlas/files

```
$ bzip2 –dc atlas3.9.23.tar.bz2 | tar xvf -
$ cd ATLAS
$ mkdir ATLAS_Linux_Core2Duo
$ ../configure
$ make                  # De 10 minutos a VARIAS HORAS!!!
$ make check            # Comprueba la corrección de las rutinas
$ make ptcheck          # Comprueba versión multi-hilo
$ make time             # Resumen del rendimiento en % frec. reloj
```

BLAS/LAPACK
Instalación ATLAS

BIG_MM N=1600, mf=9653.30,9661.80!

The times labeled Reference are for ATLAS as installed by the authors.

NAMING ABBREVIATIONS:

kSelMM : selected matmul kernel (may be hand-tuned)

kGenMM : generated matmul kernel

kMM_NT : worst no-copy kernel

kMM_TN : best no-copy kernel

BIG_MM : large GEMM timing (usually N=1600); estimate of asymptotic peak

kMV_N : NoTranspose matvec kernel

kMV_T : Transpose matvec kernel

kGER : GER (rank-1 update) kernel

Kernel routines are not called by the user directly, and their performance is often somewhat different than the totalalgorithm (eg, dGER perf may differ from dkGER)

Reference clock rate=2493Mhz, new rate=2800Mhz

Refrenc : % of clock rate achieved by reference install

Present : % of clock rate achieved by present ATLAS install

Intel Xeon E5462 2,80 Ghz gcc 4.3.1 (10 minutos con 40 segundos en configurar)

	single precision				double precision			
	real		complex		real		complex	
Benchmark	Refrenc	Present	Refrenc	Present	Refrenc	Present	Refrenc	Present
kSelMM	643.0	625.3	562.2	643.1	370.1	372.0	357.3	357.6
kGenMM	189.4	187.0	189.7	188.8	175.8	175.4	176.2	177.9
kMM_NT	135.7	190.2	166.8	180.3	178.2	157.5	173.1	175.4
kMM_TN	141.8	183.0	158.2	176.0	168.7	171.4	159.8	169.0
BIG_MM	611.7	604.6	620.4	604.3	353.1	349.6	348.4	347.1
kMV_N	85.6	23.9	117.5	114.9	44.2	38.6	58.9	52.5
kMV_T	71.3	68.5	100.1	95.4	45.9	37.7	60.5	55.2
kGER	56.0	49.5	100.9	100.8	25.7	25.2	56.1	49.5

BLAS/LAPACK
Versiones Paralelizadas

- HAY QUE DISTINGUIR SI SON PARA MEMORIA COMPARTIDA O DISTRIBUIDA.
- MEMORIA COMPARTIDA:
  1. VERSIONES DEL FABRICANTE: Todas contemplan uso de memoria compartida. La llamada a la rutinas de BLAS o LAPACK no cambia.
  2. VERSIONES MULTIPLATAFORMA: ATLAS y GotoBLAS.
- MEMORIA DISTRIBUIDA:
  1. ScaLAPACK: http://www.netlib.org/scalapack/scalapack_home.html
  2. Desde hace algunos años ya comienza a incluirse en las versiones de los fabricantes:
     1. Intl MKL
     2. IBM Parallel ESSL.
     3. HP MLIB
     4. etc.
  3. Requiere alguna versión de MPI en el sistema.
  4. Algunos fabricantes contemplan el uso de diferentes versiones de MPI:
     1. Intel MKL: Intel MPI, OpenMPI, MPICH2.

Compiladores OpenMP

- Los compiladores GNU GCC soportan OpenMP desde la versión 4.2
- Todos los fabricantes de compiladores soportan OpenMP:
  - Intel C/C++ Fortran (x86, x86_64, IA64 en Linux y Windows)
  - Portland Group Fortran C/C++ (AMD, x86, x86_64 en Linux y Windows)
  - Absoft Fortran Pro Compiler (AMD, x86, x86_64, PowerPC en Linux y Windows)
  - PathScale C/C++ Fortran (AMD, x86, x86_64 en Linux)
  - Compiladores de HP, IBM y Sun para sus procesadores.

Distribuciones Linux específicas para CCC

Distribuciones de Linux de código abierto que posibilita la preparación de un cluster para cálculo:

- Rocks: http://www.rockscluster.org
  - Última Versión: 5.3 17 de diciembre de 2010
  - El cluster más grande es un entorno Grid de 8632 cores Intel EM64t gestionado por un Instituto tecnológico Alemán.

- Open Source Cluster Application Resources (OSCAR) http://svn.oscar.openclustergroup.org/trac/oscar
  - Última versión: 6.0.3 27 de mayo de 2009
  - El cluster más grande tiene 2048 cores Intel EM64t y es gestionado por el Instituto Turbooinstitut d.d. de Eslovenia.