Curso clúster

06-07 Agosto 2014

Romina Ruscica

Modelo climático regional RCA

- Modelo Sueco desarollado en Rossby Centre (SMHI)
- Instalado en el CIMA en el 2005
- Versión RCA3 usada en CLARIS y CLARIS-LPB.

<u>Nueva versión RCA4-CORDEX</u>

- Reformulado en Fortran 90/95 y C/C++
- La mayor parte del programa es ahora modular y utiliza librerías MPI para la comunicación entre procesadores
- no es necesario el pre-procesamiento, ya que los datos usados en la simulación son “leídos” de datos globales, así como tampoco necesita ser re-compilado cuando hay cambios en el dominio ó en el número de niveles verticales.
- RCA4 tiene mayor flexibilidad que RCA3
  - Puede ser usado para casi cualquier región del planeta
  - Fácil de correr con diferentes escenarios (RCPs) y diferentes GCMs .

Algunas líneas de MPI en código RCA4

```
subroutine hlprog
      !    HLPROG - MAIN PROGRAM
##################################
……
ifdef MPI_SRC (use the MPI-library)
….
call mpi_comm_rank(localcomm,mype,ierr)
    call mpi_comm_size(localcomm,nproc,ierr)
    call mpi_get_version( ver, subver,ierr )
    if( mype==0)then
      write(*,*)"Initializing MPI Version ", ver,subver
…
```

Instalación/Compilación

Descargado con svn

svn co svn+ssh://
sm_romru@gimle.nsc.liu.se/home/rossby/svn/rca_repository/rca/branches/RCA4_CORDEX ***RCA4_CORDEX*** *(en* ***home_HYDRA)***

Compilación (hecha por gente de Rossby Centre)

OPT := -O3 .....lenta, porque hay optimización, pero corre mas rápido)

(OPT := -O0 .....rápida, porque no hay optimización, pero corre más lento)

CC := mpicc -cc=icc

FC := mpif90 -fc=ifort

NETCDFPATH :=/usr/local

LD_MPP := -L /usr/lib/gcc/x86_64-redhat-linux/4.1.2 -lstdc++ $(NCDF)

AR := xiar

Como correr en el clúster: Uso de PBS

Archivo **pbsRCA**

Solo hay colas largas ahora

```
#!/bin/bash

#PBS -l nodes=1:ppn=24
#PBS -q corta
#PBS -o output.out
#PBS -e errors.out
#####################################################
NPROCS=`wc -l < $PBS_NODEFILE`
NNODES=`uniq $PBS_NODEFILE | wc -l`

echo "This job is using $NPROCS CPU(s) on the following
      $NNODES node(s):"
echo "----------------------"
uniq $PBS_NODEFILE | sort
echo "----------------------"

EXEC_NAME="/home/ruscica/sep_2012/RCA4_CORDEX/hydra/bin/rca.x"
cd $PBS_O_WORKDIR
mpirun -np $NPROCS $EXEC_NAME >& logfile
```

Output.out

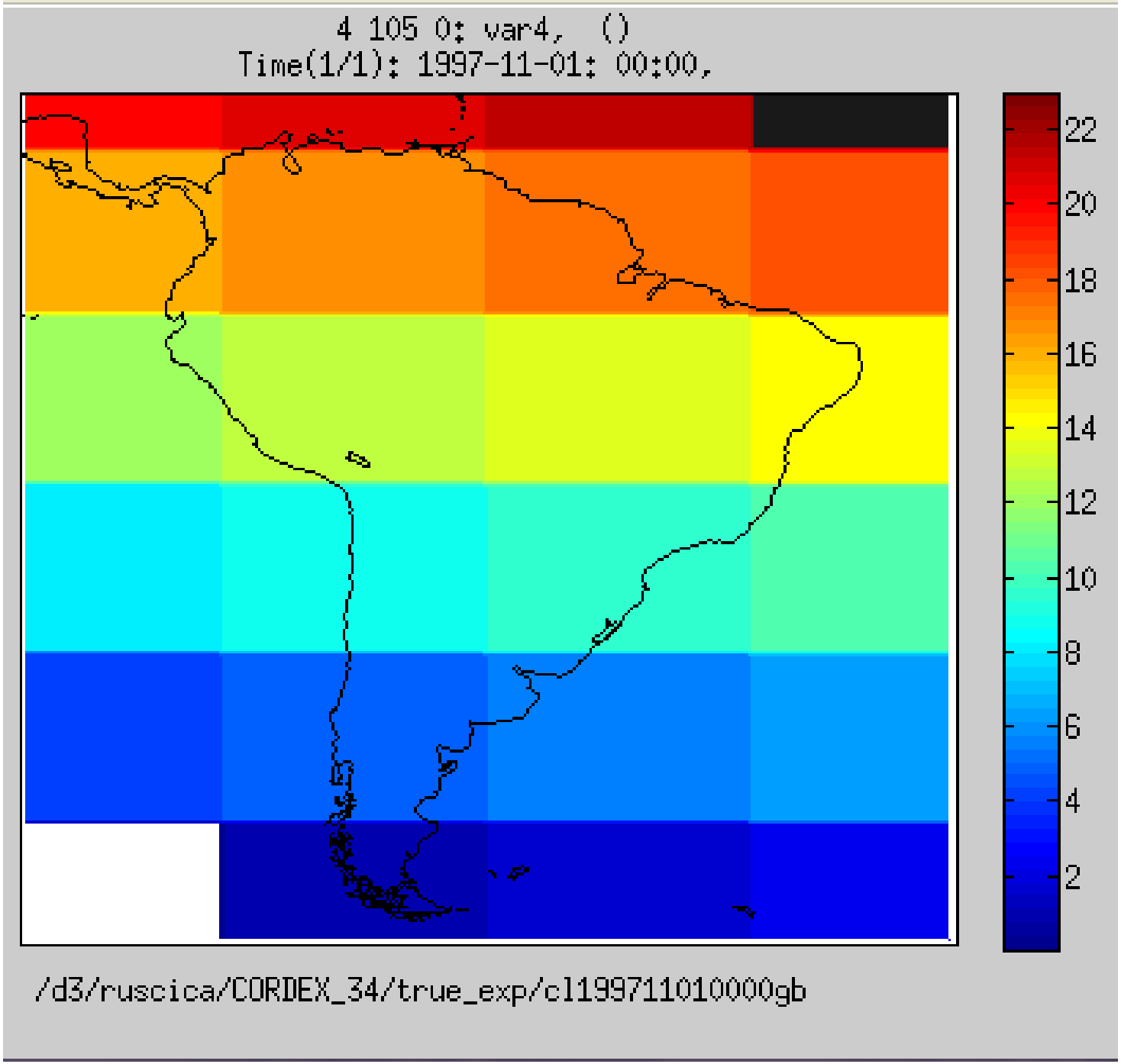
```
This job is using 24 CPU(s) on
the following 1 node(s):
----------------------
node40
----------------------
```

Dirección del archivo ejecutable

Imprime líneas de ejecución del código

Corre el modelo

Domino fraccionado en 24 (procesadores)



Problemas

- El código no compilaba sino no se usaba FLEX (fast lexical analyzer generator, a tool for generating scanners: Programs which recognize lexical patterns in text. More information at: http://flex.sourceforge.net/)
- El modelo no pudo correr en los nodos viejos (4 proc.), ya que no tenian SSE4.2 (necesario para la compilacion del modelo)
- C.i y C.C tenian que estar en el home, al igual que el codigo, ya que si estaban en otro disco, el modelo se caía
- Por ende, tuve que hacer restart varias veces, porque tenia que subir/bajar datos debido a la capacidad de disco en ese momento en el home/hydra

Tiempos

Corriendo sobre todo Sudamérica, con resolución espacial de 0.44° y 40 niveles verticales (166x198x40=1.314.720 puntos donde el modelo resuelve las ecuaciones en cada paso de iteración):

Con 24 procesadores (1 nodo) y usando 03:

1 mes los 20 años de simulación

Antes del clúster

- Instale y compile el modelo (yo solita) en Skogul, pero… se caía dando un error de (supuestamente) falta de memoria RAM.

Programación concurrente/paralela

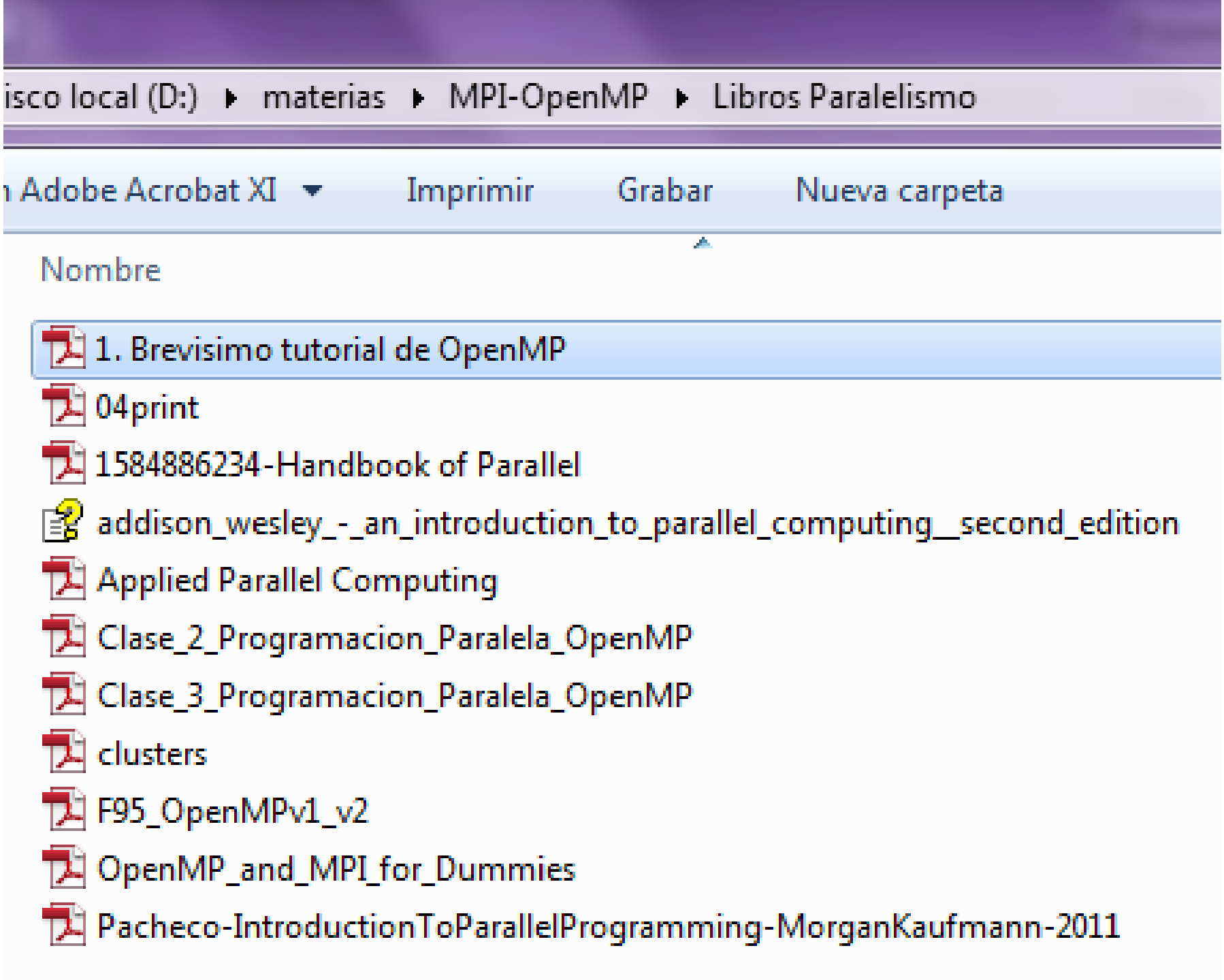
Motivación

Marcelo Arroyo

Dpto. de Computación - FCEFQyN - Universidad Nacional de Río Cuarto

16 de octubre de 2012

Clases disponibles para quien quiera


isco local (D:) ▸ materias ▸ MPI-OpenMP ▸ Libros Paralelismo
Adobe Acrobat XI
Imprimir
Grabar
Nueva carpeta
Nombre
1. Brevisimo tutorial de OpenMP
04print
1584886234-Handbook of Parallel
addison_wesley_-_an_introduction_to_parallel_computing__second_edition
Applied Parallel Computing
Clase_2_Programacion_Paralela_OpenMP
Clase_3_Programacion_Paralela_OpenMP
clusters
F95_OpenMPv1_v2
OpenMP_and_MPI_for_Dummies
Pacheco-IntroductionToParallelProgramming-MorganKaufmann-2011

FIN
y
GRACIAS

Notas

- Para compilar MPI en Fortran: mpif77
- Para compilar MPI en C: mpicc/mpich
- Para ejecutar MPI en C: mpirun –n(np) ‘n° procesos’ ‘ejecutable’
- Para compilar MP en C: gcc –fopenmp ‘fuente’
- Para compilar MP en Fortran: gfortran –fopenmp ‘fuente’
- Híbrido: mpicc –fopenmp ‘fuente’

Más notas

- La paralelizarían de los algoritmos/programas puede referirse a la descomposición de los datos o de las funciones.
- Puede paralelizarse un programa en tu PC de escritorio, usando Open MP (memoria compartida).
- No puede paralizarse todo un programa. Por otra parte conviene paralelizar solo la zona crítica (o cuello de botella) o sea donde la ejecución del programa es más “sensible”. Como por ejemplo los ciclos “for”.
- La comunicación entre nodos es mucho mas lenta que en un mismo nodo. Por eso es conveniente tener menos procesos que sean más largos que muchos cortos ya que la latencia trae más problemas que el ancho de banda
- La latencia es el tiempo que tarda el sistema en saber que se va a transferir y adonde. Es uno de los granes problemas en los clusters. Es conveniente mandar mensajes largos y pocos para minimizar la latencia.

Modelos de programación:

MP (multi-processor). Memoria compartida

- Se implementa con extensiones al lenguaje.
- POSIX Threads, openMP. Los threads se implementan como bibliotecas en algunos lenguajes y en otros no.

MPI (Message passing interface). Establece el pasaje de mensajes.

- Estos pueden ser asíncronos ó síncronos (punto a punto donde el sistema bloquea hasta que lleguen todos los procesos). Este último caso es más poderoso pero es más difícil de implementar también.
- No se cambia el lenguaje solo debe implementarse con librerías.
- MATLABatla

Programación de co-procesadores

- GPGPU (GPU es un co-procesador especializado para parte grafica (Grapic PU (similar a CPU)). GPU es procesamiento vectorial).
- OpenCL

La cantidad de procesos puede ser menor/igual/mayor que el número de procesadores. Se establece a priori o sea estáticamente.

Ley de Amdhal

- el speed up de la ejecución a partir de una determinada cantidad de procesadores ya no crece mas volviéndose asintótico porque el overhead y los tiempos de comunicación empiezan a pesar mas. En el overhead influyen: planificación; sincronización; llamadas al sistema.
- Escalabilidad: Si al aumentar la cantidad de procesadores el speed up aumenta proporcionalmente se dice que el problema es escalable. Esto implicaría que la eficiencia se mantiene constante. Según Amdhal esto se cumple hasta cierto valor de #procesadores.

Cluster al 19-9-12

- Operating system: Linux Centos release 6.2, 64 bits
- Compilers: Intel 12.03(Fortran, C, C++) and GNU GCC version 4.4.6
- (Fortran, C, C++)
- MPI: MPICH2 v.1.4.1p1 (compiled with intel)
- NetCDF: NetCDF v.4.1.3
- cmake: cmake v.2.6-patch 4