Modelación Aplicada del Océano

Curso Básico - CROCO

Andrés Sepúlveda

Departamento de Geofísica
Universidad de Concepción

11 Enero 2022

Anuncios

- Hoy: **Paralelización**

Paralelización

Aspectos Generales

- Definimos paralelización como el uso de códigos computacionales para que un programa, en este caso una simulación de CROCO pueda usar varios núcleos/cores (CPU) para calcular los resultados.
- Dada la necesidad y la tendencia a simular dominios cada vez mas grandes, y/o dominios a mayor resolución, es importante conocer los aspectos básicos de paralelización.
- El código CROCO tiene dos formas principales de paralelizar una simulación:
  - OpenMP
  - MPI
- Estos dos enfoques dependen principalmente de saber si los cores están en una sola máquina, o en varias.
- El propósito último de estas es **sincronizar** los procesos entre los núcleos, pues para acelerar el cálculo, vamos a subdividir nuestro dominio.
- Existe la tecnología llamada *hyperthreading* la cual duplica, de forma virtual, los cores. Esto no ha resultado en un aumento en la velocidad de cálculo con CROCO.

Paralelización

OpenMP

- OpenMP es una API (interfaz de programación) enfocada en multiprocesos de memoria compartida, es decir para máquinas con muchos núcleos que están conectados a un sólo banco de memoria RAM (memoria compartida).
- En CROCO se activa a través del **cppdef.h**

```
                              /* Parallelization */
# define  OPENMP
# undef   MPI
```

- Es relativamente fácil de implementar y no requiere un compilador especializado, se puede usar *ifort* o *gfortran*
- El *jobcomp* reconoce esta opción y agrega una opción (*flag*) de compilación al crear el ejecutable

```
gfortran -c -O3 -fdefault-real-8 -mcmodel=medium -fopenmp
```

Paralelización

OpenMP

- Es necesario especificar el número de núcleos a usar a través del **param.h**

```
#elif defined OPENMP
      parameter (NPP=28)
# ifdef AUTOTILING
      common/distrib/NSUB_X, NSUB_E
# else
      parameter (NSUB_X=2, NSUB_E=14)
# endif
```

- y también al lanzar la simulación, como en el **run_croco.bash**

```
#  Define environment variables for OPENMP
OMP_SCHEDULE=static
OMP_NUM_THREADS=28
OMP_DYNAMIC=false
OMP_NESTED=false
```

Paralelización

MPI

- MPI (message passing interface) es una interfaz de paralelización que no necesita que la memoria esté compartida, por lo que se usa en los sistemas donde la memora está distribuida en varias máquinas.
- En CROCO se activa a través del **cppdef.h**

```
                              /* Parallelization */
# undef  OPENMP
# define  MPI
```

- Es mas complicado de implementar pues requiere librerias especializadas y compiladores especializados *mpiifort* o *mpif90*
- El *jobcomp* reconoce esta opción cambia el compilador que se usa al crear el ejecutable

```
/usr/local/bin/mpif90 -c -O3 -mcmodel=medium  module_nh_.f90
```

Paralelización

MPI

- Es necesario especificar el número de núcleos a usar a través del **param.h**

```
#ifdef MPI
      integer NP_XI, NP_ETA, NNODES
      parameter (NP_XI=7,  NP_ETA=4,  NNODES=NP_XI*NP_ETA)
      parameter (NPP=1)
      parameter (NSUB_X=1, NSUB_E=1)
```

- y también al lanzar la simulación, como en el **run_croco.bash**

```
# number of processors for MPI run
NBPROCS=28

# command for running the mode :
RUNCMD='./'
#RUNCMD="mpirun -np $NBPROCS "
```

Paralelización

División del Dominio

- La velocidad de cálculo aumenta al paralelizar pues cada núcleo se dedica a calcular una porción de nuestro dominio.
- Según el número de núcleos que usemos, el dominio se puede dividir de muchas formas:

```
28 = 28 x 1 = 1 x 28 = 2 x 14 =14 x 2 = 4 x 7 = 7 x 4 = ....
```

- Y la velocidad de cálculo no es la misma para cada caso.
- Existen técnicas de *Descomposición de Dominios* que subdividen un dominio de forma irregular según ciertos criterios. No se ha aplicado a CROCO y es más común en modelos de grilla no estructurada (e.g. FVCOM).

Paralelización

NOLAND

- Recientemente se ha incorporado a CROCO una técnica proveniente del modelo europeo NEMO
- Esta técnica apunta a evitar que se calculen las celdas que contienen tierra.
- Para esto modificamos el **cppdefs.h**

```
#  undef  MPI_NOLAND
```

- Como las grillas estructuradas, como la curvilínea ortogonal de CROCO, suelen tener una buena proporción de celdas en tierra, esto disminuye el tiempo de cálculo.
- Alternativamente podemos usar técnicas avanzadas de diseño de grilla para evitar al máximo las celdas sobre tierra (Daniel Brieva).

Paralelización

XIOS

- Otra herramienta que se ha incorporado a CROCO para disminuir el tiempo de cálculo es el uso del programa **XIOS**
- El objetivo de esta librería es dedicar un solo núcleo, donde corre el servidor el servidor **XIOS**, a escribir y leer los archivos. Sin este enfoque, cada procesador escribe y lee la información correspondiente a su zona.
- Esta librería se tiene que compilar previamente, y después es incorporada al compilar CROCO. Para esto modificamos el **cppdefs.h**

```
            /* I/O server */
 # define  XIOS
```

- Para dominios grandes, **XIOS** es muy eficiente al escribir archivos NetCDF en paralelo.

Paralelización

Escalamiento

- Para sacar el mejor provecho de un computador o un cluster, es indispensable realizar un estudio de escalamiento.

  `https://wiki.nlhpc.cl/Escalamiento_de_Aplicaciones`

- Esto significa identificar cuál es el número óptimo de núcleos de cálculo que podemos usar.
- Cómo los núcleos de cálculo tienen que comunicarse entre ellos para intercambiar información (si tienen fronteras comunes), usar muchos núcleos puede volver más lento el cálculo, pues los núcleos pasan más tiempo comunicándose, que calculando.
- Cada programa, y cada configuración, puede tener un escalamiento distinto.

Paralelización

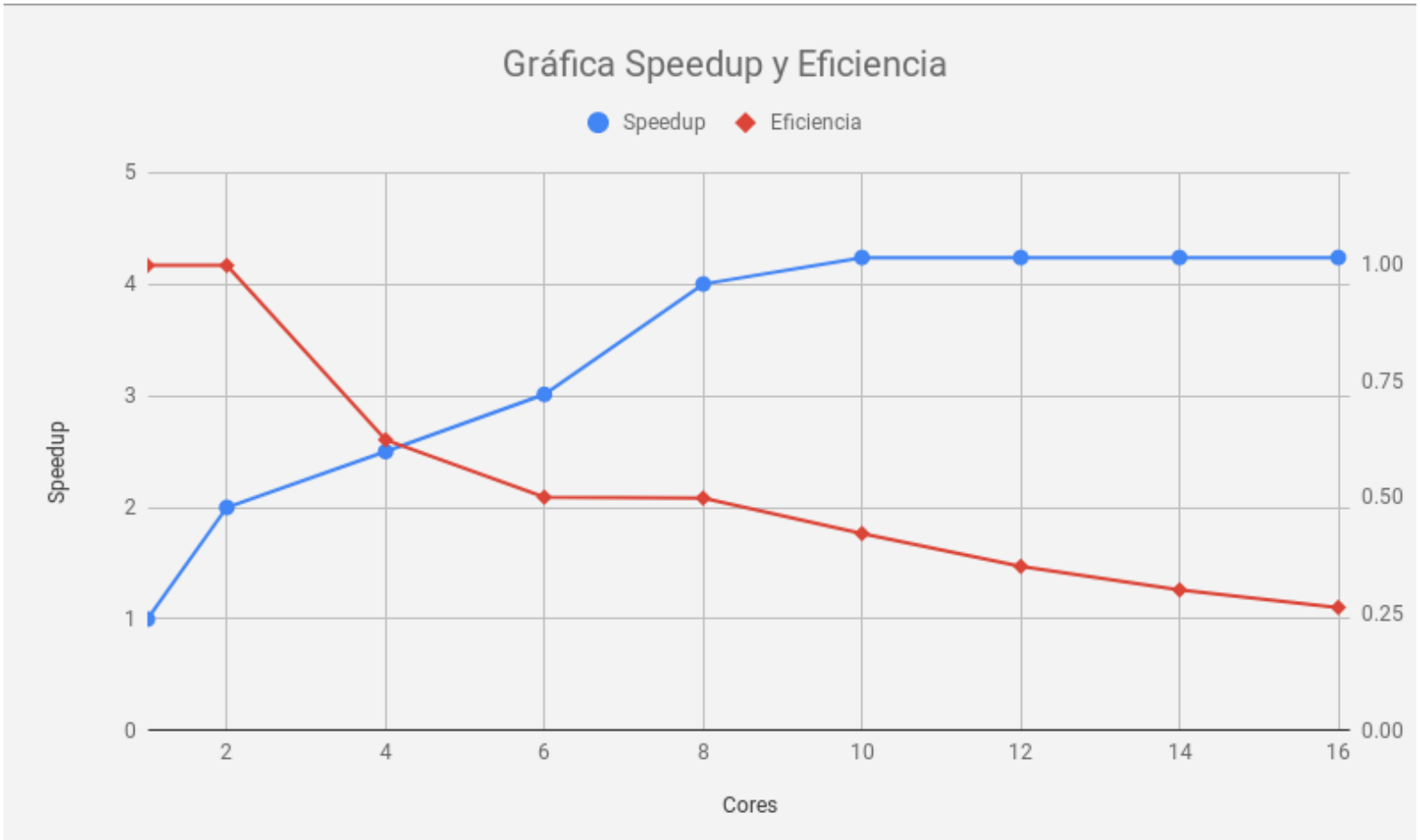

Figura: Gráfico de escalamiento. La eficiencia debe ser mayor a 0.5

Paralelización

Rutina cálculo eficiencia

```
%
% Jessica Crisostomo - Universidad de Concepcion 2019
%
cores=[1,2:2:16];
%tiempo_20m --> cuanto se demoraba en simular 20 minutos
tiempo_20m=[7.17 4.10 2.21 1.54 1.36 1.11 1.28 1.37 1.16 ]/1440  | ]
tiempo_hora=tiempo_20m*3;
tiempo_dia=tiempo_hora*24;
tiempo_semana=tiempo_dia*7;
tiempo_agno=tiempo_semana*52.1429
tiempo_agno_dias=tiempo_agno

Speedup=tiempo_agno_dias(1)./tiempo_agno_dias
Eficiencia=Speedup./cores
```

Paralelización

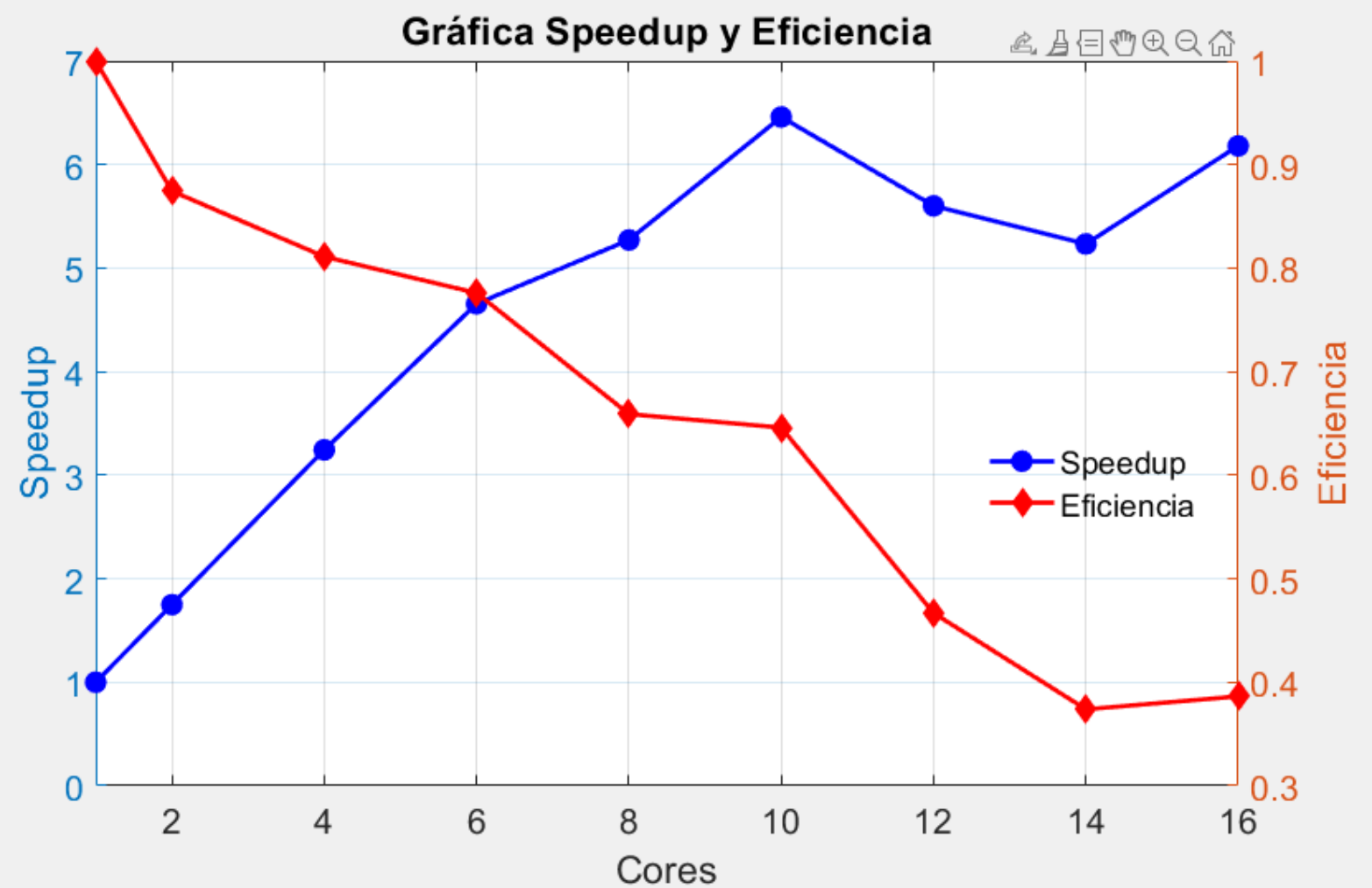

Figura: Gráfico de escalamiento WRF. La eficiencia debe ser mayor a 0.5

Paralelización

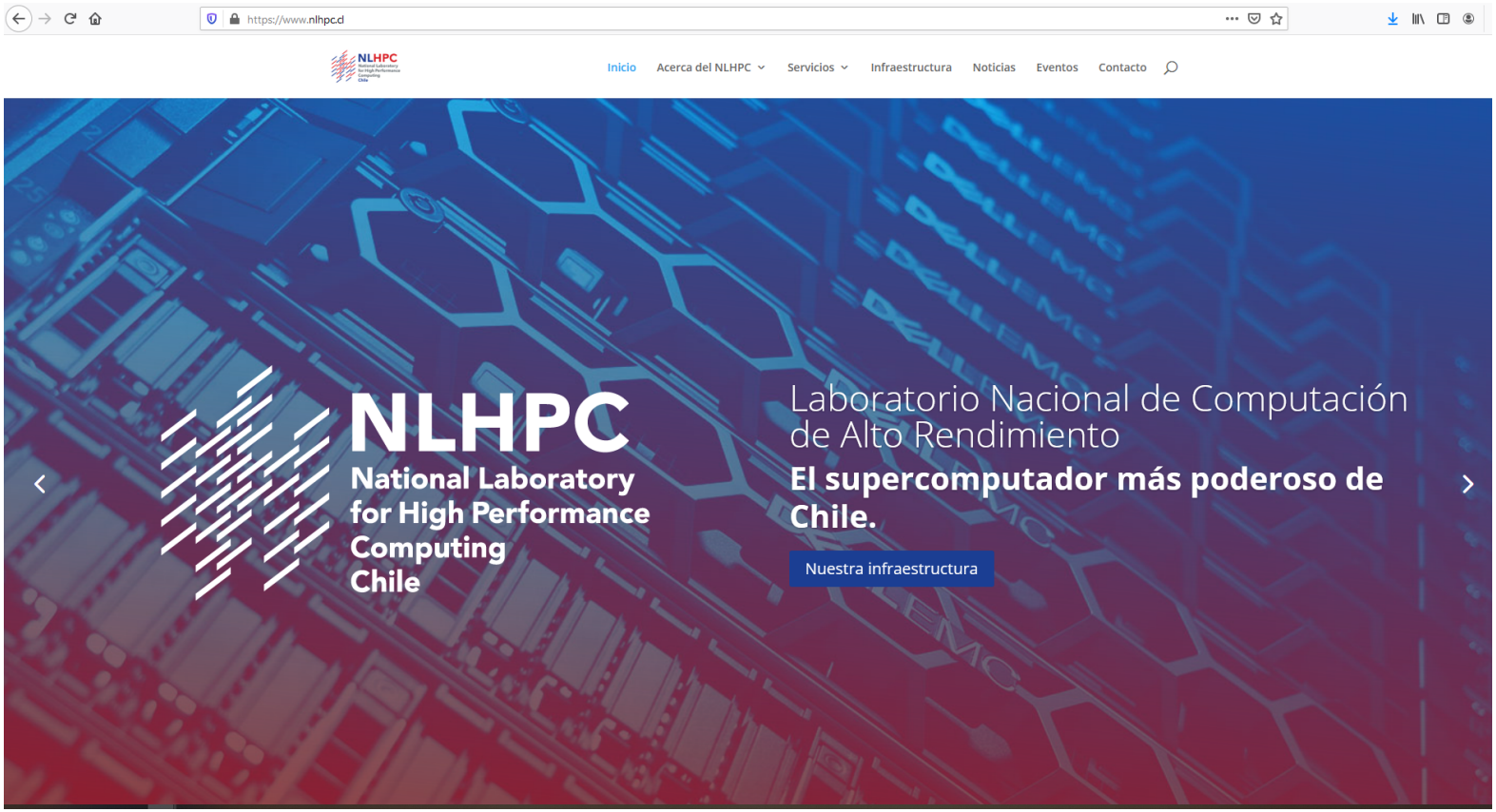

Figura: Página web del NLHPC

Paralelización

NLHPC

- El NLHPC (National Laboratory for High Performance Computing - Chile) es un centro de supercómputo que está albergado en la Universidad de Chile.
- Busca ofrecer el servicio de cálculo a las universidades y la industria (empresas, consultoras), para que no exista un cluster en cada oficina.
- Los programas mas comunes están ahí, en forma de código (CROCO) o precompilados (WRF), por lo que nos ahorra el tiempo de instalación (*module avail*).
- Es posible pedir horas de cálculo para el desarrollo de tésis de pregrado y también de postgrado.
- La Uiversidad de Concepción alberga un centro de supercómputo enfocado en la tecnología GPU.

Paralelización

NLHPC

El siguiente es un ejemplo básico de un script que puede ser usado para lanzar una simulación de CROCO en el NLHPC

```
#!/bin/bash
#SBATCH --job-name=test15
#SBATCH --partition=slims
#SBATCH -n 4
#SBATCH --ntasks-per-node=4

ml purge
ml intel/2019b
ml netCDF-Fortran/4.4.4
srun ./croco croco.in
```

El script generará archivo de registro (.log) y un archivo de errores (.err)

Paralelización

NLHPC

Algunas instrucciones básicas del NLHPC son

- **sbatch** Para enviar la simulación los nodos

```
sbatch run_croco_en_nlhpc.bash
```

- **squeue** Para ver el estado de una simulación

```
ifop@leftraru3:~$ squeue
    JOBID PARTITION     NAME     USER ST  TIME      NODES
21135972   general ROMSPHYS     ifop  R 1-03:11:26     1
21130217     slims  REA2009     ifop  R 6-04:45:39     3
```

- **scancel** Para terminar anticipadamente un proceso. Hay que conocer el ÍD del proceso.

```
scancel 21135972
```

Paralelización

Caveat Emptor

- El mundo de la paralelización es a veces extraño y contraintuitivo.
- Es bueno verificar que la simulación corre en modo serial, si hay problemas en la paralelización.
- Es recomendable verificar las diferencias entre dos simulaciones ïdénticas"pero paralelizables de distinta forma.
- Estas "variaciones"han llevado a la búsqueda de una reproducibilidad *bit-by-bit*, particularmente en el área de simulaciones de cambio climático.
- Potencialmente se pueden conectar varios computadores que tengan muchos núcleos cada uno y usar una combinación de OpenMP+MPI, pero esto no ha sido implementado en CROCO aún.
- Un cuello de botella importante en ambientes de memoria distribuida es el tipo de cables y el switch que conecta los nodos de cálculo. Para CROCO es necesario que esta tecnología sea tipo Infiniband o mejor. No Gigabit.

Paralelización

Cálculo en la nube

- Se aplican los mismos criterios mencionados anteriormente respecto a conectividad entre nodos
- Es importante agregar el costo ($) de almacenamiento y en subida y bajada de datos.
- Experiencias

  `https://sriha.net/articles/roms-on-aws`
  `https://github.com/poidl/awsroms`
  `https://www.mdpi.com/2077-1312/7/4/110` (¡Rick Signell!)
  `https://doi.org/10.3389/fmars.2019.00211` (¡Rick Signell!)

- ¿Más caro? Infraestructura, luz, mantención, aire acondicionado, salarios...
- Nada de alta gama, pero si algo propio (50 cores, 50 GB, 8 TB).