Programación de Alto Rendimiento: computación en paralelo

Giovanni Ramírez García, PhD

Escuela de Ciencias Físicas y Matemáticas
Universidad de San Carlos de Guatemala

Guatemala, 2 de febrero de 2021




Computación en paralelo

MPI

Referencias

Computación de alto rendimiento (HPC)

- Una computadora de alto rendimiento puede resolver problemas grandes en menos tiempo que una computadora de escritorio.
- Son capaces de hacer los cálculos secuenciales más rápido y también pueden hacer cálculos en paralelo.
- Características:
  - procesadores rápidos,
  - mucha memoria RAM,
  - alta velocidad de entrada/salida,
  - conexiones de red rápidas.
- [Fountain, cap 6 *System Performance*]
- Una forma de medir su desempeño es en *floating point operatios per second* (Flops o Flop/s)
  - Intel Core 4DP Flops/cycle [Dolbeau]
  - Intel Xeon Phi (KL) 32DP Flops/cycle [Dolbeau]
  - AMD Ryzen 8DP Flops/cycle [Dolbeau]
- La #1 del Top500 [top500.org, feb 2021]: Fugaku en RIKEN Center for Comp. Sci.: 7 630 848 cores, 5 087 232 GB de RAM, Rmax de 442 010 TFlop/s, Rpeak de 537 212 TFlop/s. Cons: 30MW.

Aplicaciones de la HPC

- Mecánica cuántica (DMRG [Delcompare, P], DFT, QMC)
- Meteorología y Clima (REGCM [García, L])
- Biología computacional: secuencias de ADN
- Farmacéuticas: diseño de nuevos medicamentos
- Geología: exploración sísmica
- Economía: análisis de mercados
- Física médica [Florian, E]
- Procesamiento de imágenes [Ballina, M; Barrientos, D]
- Física de partículas (*background substraction* en el CERN, modelos de nueva física en el LHC [preguntar a M.E. Cabrera], LAGO [Tun, L])
- Sistemas complejos [Estrada, E]
- Astrofísica [Lemus, B]
- Dinámica molecular
- Física de plasmas [Franco, J]
- Mecánica de Fluidos (ecuaciones de Navier-Stokes)
- Modelación matemática
- Inteligencia artificial

euclides

- clúster tipo Beowulf: grupo de computadoras de rendimiento semejante interconectadas en una red local
- la red local está en topología de estrella,
- un nodo maestro: procesador Intel Pentium 4 HT, 2GB RAM, 500GB HD
- cuatro nodos esclavos: procesador Core 2 Duo, 2GB RAM, 100GB HD
- SO: Debian Linux 7.9
- Compiladores: GNU gcc
- MPICH 3.1.4
- Rendimiento (LINPACK): 45GFlops

Información tomada de Alvarez, W.G.

Secuencial vs Paralelo

Secuencial

- el tiempo
- escribir un algoritmo
- escuchar una canción
- jugar ajedrez

Paralelo

- La primavera
- Una orquesta

Paralelismo en algoritmos

- Aprovechar que una computadora es por naturaleza un sistema paralelo.
- *Pero un algoritmo es una secuencia de pasos...*
- *Can parallel computers be used effectively for large scale scientific computation?* Parallel Computing Works! [Fox, Williams y Messina]

¿Qué es la computación en paralelo?

- Es hacer que una computadora haga más de un cálculo al mismo tiempo usando más de un procesador.
- Existe un límite en el rendimiento al usar un solo procesador, pero se puede incrementar el rendimiento usando más procesadores.
- Consideremos que un procesador hace una tarea en un tiempo t entonces, si usamos p procesadores la misma tarea se haría, idealmente, en un tiempo t/p.
- "Casi todos los algoritmos tienen cierta forma de paralelismo". – NCSA.
- Entonces, para *programar en paralelo* debemos estudiar si los datos o el algoritmo puede partirse en partes independientes que puedan realizarse simultáneamente. Este proceso se llama Decomposición.
- La decomposición establece dos esquemas: *data parallelism* y *task parallelism*.

Taxonomía de Flynn

SIMD

- *single instruction stream, multiple data stream*
- representa un arreglo de procesadores que ejecutan la misma instrucción al mismo instante
- requiere una conexión
- permite el paralelismo de los datos

MIMD

- *multiple instruction stream, multiple data stream*
- representa un arreglo de procesadores, cada uno ejecuta una tarea de forma independiente a los demás
- requiere una conexión, aunque no hace falta especificarla
- permite, el paralelismo de funciones

La taxonomía de Flynn incluye también: SISD, MISD [Fountain].

Data parallelism I

- El mismo segmento de código se ejecuta concurrentemente en cada procesador.
- Sin embargo, cada procesador trabaja en un subdominio de los datos.
- También se le llama *fine grain parallelism* porque se divide el trabajo computacional en subtareas [NCSA].
- Consideremos la multiplicación de matrices $C = AB$ donde A, B y C son matrices de dimensión $n \times m$, $m \times p$ y $n \times p$.
- el ij-ésimo elemento de C es

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

  donde $i \in [1, n], j \in [1, p]$
- Podemos usar OpenMP (Open Multi-Processing), un estándar para instrucciones multiproceso de memoria compartida. En este esquema se aplica al ciclo dominante del algoritmo.

Data parallelism II

Código secuencial

```
DO K=1,N
DO J=1,N
DO I=1,N
C(I,J)=C(I,J)+A(I,K)*B(K,J)
END DO
END DO
END DO
```

Código paralelo

```
!$OMP PARALLEL DO
DO K=1,N
DO J=1,N
DO I=1,N
C(I,J)=C(I,J)+A(I,K)*B(K,J)
END DO
END DO
END DO
!$END PARALLEL DO
```

Data parallelism III

- Supongamos que todas las matrices son 20×20 y que usamos cuatro procesadores.
- Todos los procesadores ejecutan el código

```
DO J=1,N
DO I=1,N
C(I,J)=C(I,J)+A(I,K)*B(K,J)
END DO
END DO
```

- Pero usan distintos datos:
  1. proc0: $k \in [1, 5]$
  2. proc1: $k \in [6, 10]$
  3. proc2: $k \in [11, 15]$
  4. proc3: $k \in [16, 20]$
- OpenMP: estrategia incremental
  1. Paralelizar el ciclo dominante
  2. Calcular el rendimiento del código
  3. Si no le satisface, paralelice otro ciclo
  4. Repetir 2 y 3 cuantas veces necesite

Task parallelism I

- Ahora, en lugar de que cada procesador realice la misma tarea con distintos subconjuntos de los datos, cada proceso realiza diferentes operaciones.
- Se puede usar cuando el algoritmo se puede dividir en diferentes tareas o subrutinas que se pueden realizar de modo independiente.
- También se le llama *coarse grain parallelism* [NCSA].
- En este esquema hay más código ejecutándose en paralelo.
- Consideremos que un algoritmo se puede dividir en cuatro tareas o subrutinas, A, B, C y D; y que el podemos usar cuatro procesadores de modo que
  1. proc0: tarea A.
  2. proc1: tarea B.
  3. proc2: tarea C.
  4. proc3: tarea D.

Task parallelism II

Código secuencial

```
PROGRAM MAIN
CALL A
CALL B
CALL C
CALL D
END PROGRAM
```

Código paralelo

```
PROGRAM MAIN
!$OMP PARALLEL
!$OMP SECTIONS
CALL A
!$OMP SECTION
CALL B
!$OMP SECTION
CALL C
!$OMP SECTION
CALL D
!$OMP END SECTIONS
!$OMP END PARALLEL
END PROGRAM
```


¿Qué es MPI?

- MPI significa *Message Passing Interface*. Es un estándar para intercambio de mensajes.
- Existen varias versiones: openMPI, MPICH, Intel MPI, etc. Pero todas deben tener las mismas funciones y subrutinas, con la misma funcionalidad y con los mismos argumentos.
- La diferencia es entonces en la implementación, es por ello que algunas versiones son más eficientes que otras.
- Nosotros vamos a usar openMPI, que es una implementación de código abierto.
- Vamos a instalarnos las versiones que estén disponibles en los repositorios de la distribución de Linux que usemos. Pero necesitamos la versión de desarrollo.

Motivaciones para usar MPI

- MPI es un estándar definido mientras que OpenMP depende del compilador.
- MPI está diseñado para sistemas de memoria distribuida y de memoria compartida. OpenMP sólo funciona en sistemas de memoria compartida.

más sobre MPI

Esta es una breve descripción, ahora hay que pasar a practicar más sobre programación en FORTRAN y luego volveremos con más detalles, ejemplos y ejercicios.

Referencias I

- Alvarez, Walter Giovanni. Diseño e implementación de un clúster para la simulación computacional de la ecuación de onda en dos dimensiones.
- Dolbeau, Journal of Supercomputing 74(3), 1341, 2017.
- Delcompare, Paola. Métodos computacionales basados en Entrelazamiento cuántico para el análisis de transiciones de fase en Cadenas cuánticas de espín. ECFM-USAC, 2017.
- Estrada, Emilio. Soluciones solitónicas en el Modelo de Baby-Skyrme a través de un algoritmo de recocido simulado. ECFM-USAC, 2018.
- Florián, Eduardo. Simulación de la radiación dispersa de Rayos X de fluoroscopía por PMMA utilizando Geant4. ECFM-USAC, 2018.
- Fountain, T.J. Parrallel Computing: Principles and Practice. Cambridge University Press. 1994.

Referencias II

- Fox, Geoffrey, R Williams y G Messina. Parallel Computing Works!. Elsevier. 2014
- García, Lilian. Caracterización de la canícula en la región Guatemalteca usando el modelo climático regional REGCM. ECFM-USAC, 2018.
- Lemus, Brayan. Solución numérica de la dinámica relativista de un sistema de n cuerpos utilizando las ecuaciones de Einstein-Infeld-Hoffman. ECFM-USAC, 2018.
- National Center for Supercomputing Applications (NCSA). Parallel Computing Explained.
- Tun, Luis. Simulación de cascadas aéreas extensas en CORSIKA para la Colaboración LAGO en Guatemala. ECFM-USAC, 2017.
- top500.org. Lista de noviembre de 2018.

¡Muchas gracias!

Contacto:
Giovanni Ramírez García, PhD
ramirez@ecfm.usac.edu.gt
http://ecfm.usac.edu.gt/ramirez