

1.2 Experimentación y Rendimiento Secuencial

Balance de Carga de Experimentos Climáticos: presentación *adjunta*

Ejemplo de multiplicación de matrices optimizada

1.3 Utilización de Clusters

Dos (¿Tres?) Grandes Areas:

- High Performance Computing/Applications
- High Throughput ¿Paralelo?
- ¿Desarrollo?
- Server Farms ¿Paralelo?

1.- High Performance:

- Motivación
 - Tienen todo: CPUs y comunicación entre ellas
 - Costo/rendimiento (costo por Mflop/s)
- Pasaje de mensajes disponible
 - Sockets
 - PVM, MPI (ambos portables)
- Muchas aplicaciones ya hechas
 - Estabilidad (sucess stories)
 - Reusabilidad
 - Diferentes clases-áreas

1.1 High Performance - Problemas:

- No todos los problemas resueltos
- No todos los algoritmos son “usables”
- No se conoce el grado de granularidad (imprecisa, muy gruesa)
- Aún no hay estabilidad en desarrollo
- Aún no hay estabilidad en debugging
- Se está estabilizando la terminología
- Aún hay más “success stories” que cosas estables

2.- High Throughput:

- Resource Management and Scheduling (RMS)
- Motivación: CPUs disponibles (M. Livny, University of Wisconsin-Madison)
- Requerimientos
 - Computadoras en red - Instituciones
 - Identificación de carga de trabajo
 - Mecanismo de ejecución remota
 - Mecanismo de monitorización
 - Mantenimiento de colas de trabajos (batch)
 - Mecanismo de cancelación o migración

2.1 High Throughput - Funcionamiento (CONDOR):

- Manejador de colas + utilización de ciclos libres
- Red local monitoreada (procesamiento)
- Identificación de carga (libre o con usuario)
- Disparo de procesos en computadoras libres
- Migración de procesos a computadoras libres
- Acceso a archivos remotos abiertos
- Aplicaciones “linkeadas” y “no linkeadas”
- Manejador de colas

2.2 High Throughput - ¿Paralelo?

- Para el que lo hace quizás lo sea
- Los trabajos que corren no necesariamente...
- ¿Gang-Coscheduling?
- Manejadores de colas \implies CPU intensivos
- Carga en la red... “sucess stories”

3.- Desarrollo:

- Aún en las universidades
- “Ejemplo Clementina”
- Clusters de Producción y de Desarrollo
- Implica
 - No escribir sobre los de producción
 - Prioridad para desarrollo... inversión
 - Debugging
 - Evaluación de rendimiento
 - Monitorización-Sintonización
- Trabajos

2 Linear Algebra

Reasons:

- Applications and users
- Background \implies starting point
- Parallel: processing requirements

General characteristics:

- Matrix processing. Matrix Computations, Golub G., C. Van Loan, 2nd Edition, The John Hopkins University Press, 1989.
- Background \implies starting point
- (Huge) processing requirements \implies parallel approaches

Classical Problems:

- Matrix Multiplication: simple, benchmark, “the best”
- LU Matrix Factorization: relatively complex, *the benchmark, the problem* (applications)

Matrix Multiplication. Given

$$A \in \mathbb{R}^{m \times k}; a_{ij}, 1 \leq i \leq m, 1 \leq j \leq k$$

and

$$B \in \mathbb{R}^{k \times n}; b_{ij}, 1 \leq i \leq k, 1 \leq j \leq n$$

matrix

$$C = A \times B; C \in \mathbb{R}^{m \times n}; c_{ij}, 1 \leq i \leq m, 1 \leq j \leq n$$

is obtained by

$$c_{ij} = \sum_{r=1}^k a_{ir} b_{rj}$$

LU Matrix Factorization. Given $A \in \mathbb{R}^{n \times n}$, find out two matrices, L and U such that

- $L \in \mathbb{R}^{n \times n}$ is a lower triangular, unit diagonal matrix.
- $U \in \mathbb{R}^{n \times n}$ is an upper triangular matrix.
- $A = L \times U$.

3 A Few Comments on Performance

Evaluación de rendimiento: una vez que *no hay errores...*

¿Cómo funciona?

¿Podría ser más rápido?

¿Por qué es importante?

¿Cómo evaluarían el rendimiento? (no hay nada aquí a propósito, ¿eh?)

Más específicamente:

¿Cómo evaluarían el rendimiento secuencial?

¿Cómo evaluarían el rendimiento paralelo?