CLUSTER FING:
PARALELISMO de MEMORIA DISTRIBUIDA

SERGIO NESMACHNOW

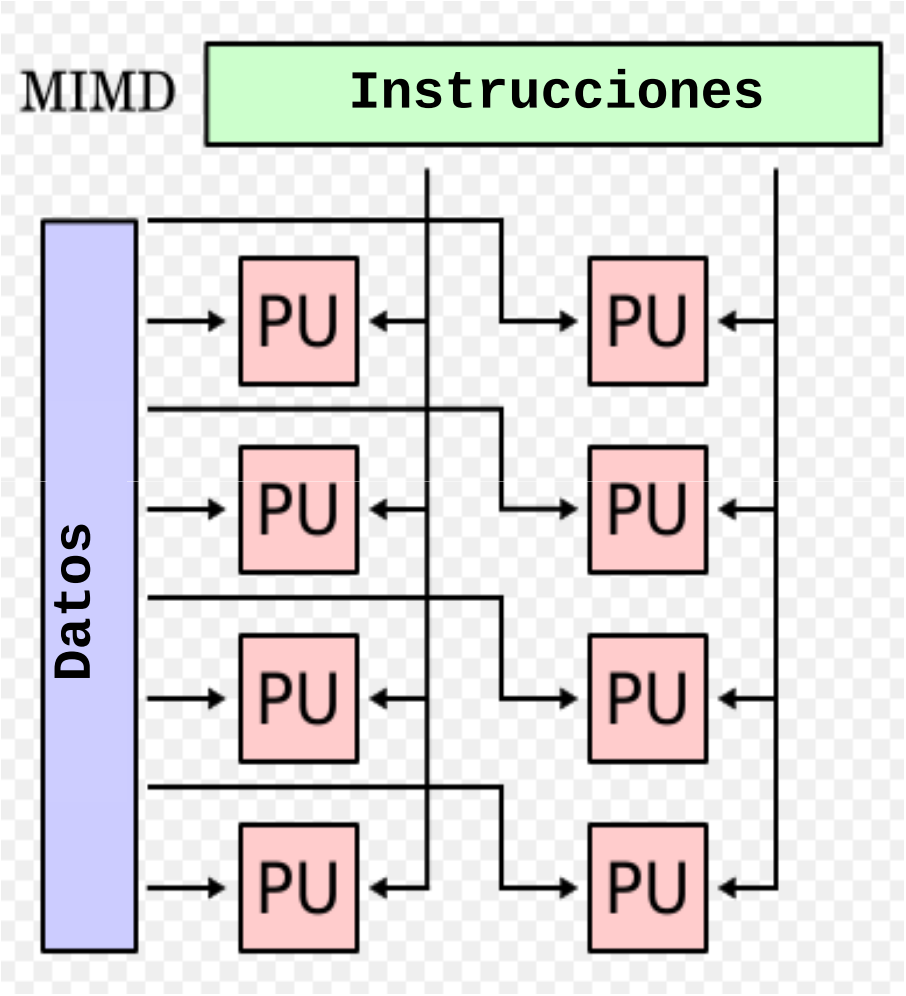
Centro de Cálculo, Instituto de Computación

FACULTAD DE INGENIERÍA, UNIVERSIDAD DE LA REPÚBLICA, URUGUAY

CONTENIDO

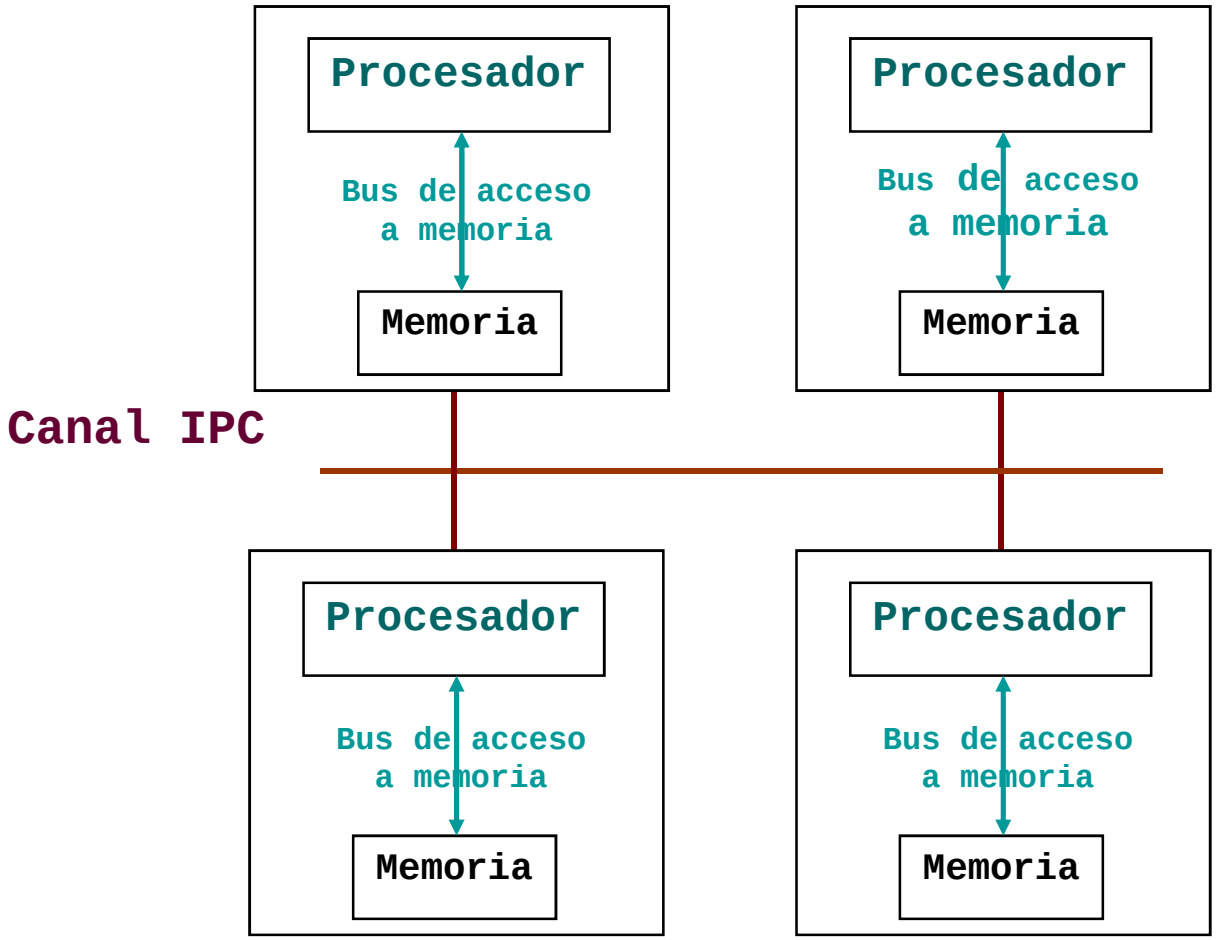
- Introducción: arquitecturas de memoria distribuida.
- Técnicas de programación paralela
  - Descomposición de dominio
  - Descomposición funcional
  - Ejemplos
- Modelos de comunicación entre procesos
  - Jerárquicos: maestro-esclavo
  - No jerárquicos: modelos totalmente distribuidos.
- Conclusiones

INTRODUCCIÓN: MIMD



MIMD: Multiple Instruction Multiple Data

MIMD con MEMORIA DISTRIBUIDA



Arquitectura MIMD con memoria distribuida

MIMD con MEMORIA DISTRIBUIDA

- No existe el concepto de memoria global.
- Comunicación y sincronización:
  - Mecanismos explícitos de IPC (mensajes) sobre una red (en escenario óptimo, red de alta velocidad).
  - Tiene mayor costo que en MIMD de memoria compartida.
- Las comunicaciones pueden ser cuellos de botella.
  - Influye la topología de conexión.
- Arquitectura escalable para aplicaciones apropiadas:
  - Decenas de miles de procesadores.
- Ventajas respecto a MIMD de memoria compartida
  - Fácilmente escalable.
  - Alta disponibilidad (el fallo de una CPU individual no afecta a todo el sistema)

PASAJE de MENSAJES

- Procesos se comunican mediante mensajes explícitos.
- Envío de mensajes (datos, sincronización)

  `send(origen,destino,tipo,msg,flags)`

  - msg contiene los datos o información de sincronización
- Recepción de mensajes

  `receive(origen,destino,tipo,msg,flags)`

  - El contenido de msg se almacena en memoria.

- Habitualmente se utilizan **buffers** (espacios reservados de memoria) para almacenar información y **colas** para almacenar mensajes.

PASAJE de MENSAJES: BIBLIOTECAS

- Bibliotecas simplifican la operativa de comunicación mediante pasaje de mensajes (sincronismo, garantía de recepción, etc.)
- PVM: Parallel Virtual Machine
  - Estándar de facto para bibliotecas de pasajes de mensajes.
  - Orientado al procesamiento paralelo en redes de computadores.
- MPI: Message Passing Interface
  - Estándar definido por investigadores, empresas e industrias.
  - Especificación de protocolo, tiene varias implementaciones
  - MPICH, LAM/MPI, OpenMPI, MS MPI
- Ambas bibliotecas siguen el paradigma presentado anteriormente.
- Son **bibliotecas**, no lenguajes y son de código libre.
- Tienen interfaces para C/C++ y FORTRAN.
- Versiones estables.

TÉCNICAS de PROGRAMACIÓN PARALELA

- Técnicas de DESCOMPOSICIÓN o PARTICIONAMIENTO, que permiten dividir un problema en subproblemas a resolver en paralelo.
- Objetivo: dividir en forma equitativa los cálculos asociados con el problema y los datos sobre los cuales opera el algoritmo.
- ¿ Cómo lograr el objetivo de la descomposición ?
  - Definir al menos un orden de magnitud más de tareas que de procesadores disponibles (utilización de recursos).
  - Evitar cálculos y almacenamientos redundantes.
  - Generar tareas de tamaño comparable.
  - Generar tareas escalables con el tamaño del problema.
  - Considerar varias alternativas de descomposición, en caso de ser posible.
- Según se enfoque principalmente en la descomposición de **datos** o de **tareas**, resultará una técnica diferente de programación paralela.

DESCOMPOSICIÓN de DOMINIO

- Se concentra en el particionamiento de los datos del problema
  - “Data parallel”.
- Se trata de dividir los datos en piezas pequeñas, de (aproximadamente) el mismo tamaño.
- Luego se dividen los cálculos a realizar
  - Asociando a cada operación con los datos sobre los cuales opera.

- Los datos a dividir pueden ser:
  - La entrada del programa.
  - La salida calculada por el programa.
  - Datos intermedios calculados por el programa.

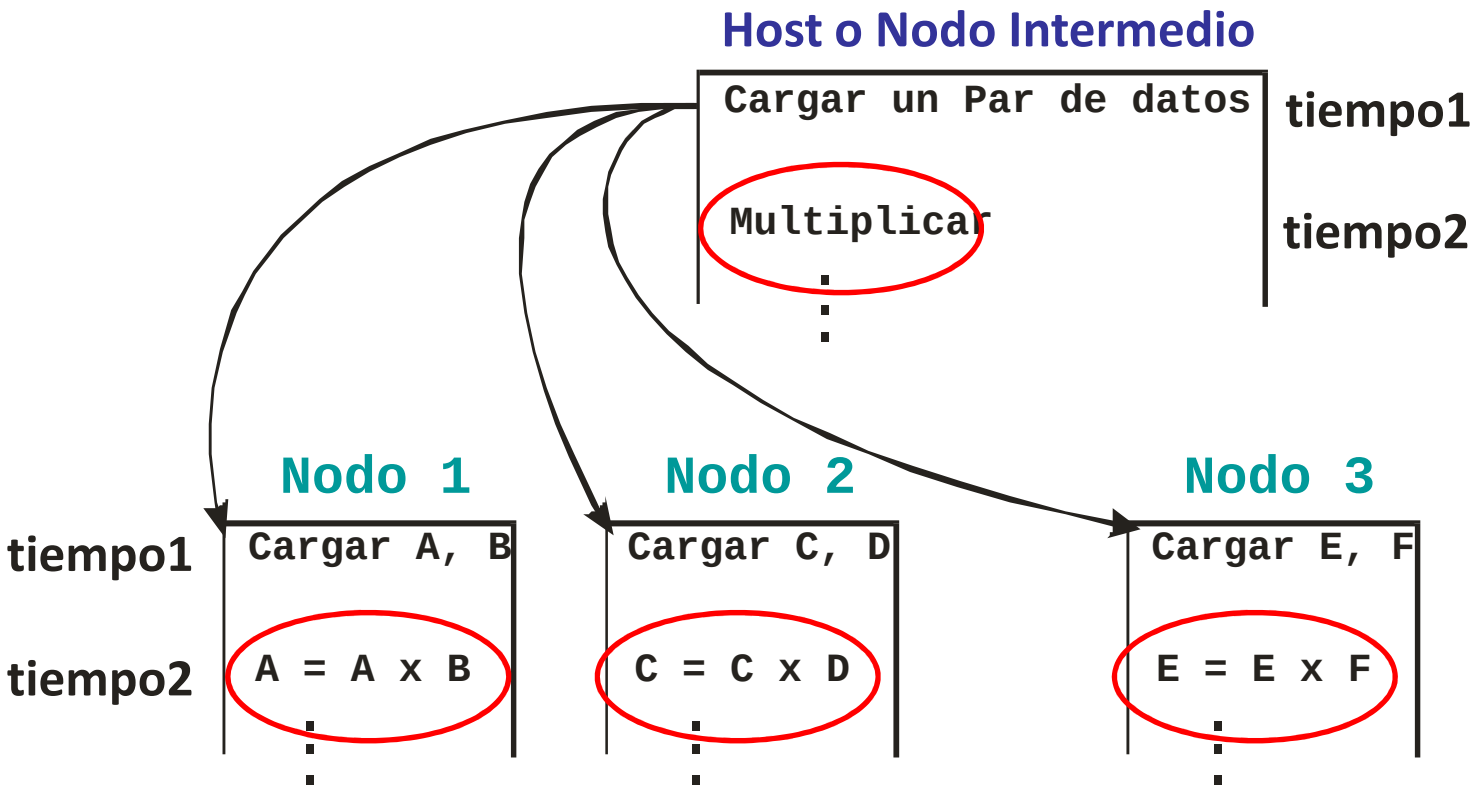
DESCOMPOSICIÓN de DOMINIO

- No existe regla general para la división de datos
- Existen algunas sugerencias obvias dadas por:
  - La estructura o "geometría" del problema.
  - La idea de concentrarse primero en las estructuras de datos más grandes o las accedidas con más frecuencia.
- La descomposición de dominio se asocia con la estrategia de "divide & conquer" y los modelos SIMD y SPMD de programas paralelos.
  - SIMD sobre arquitecturas de memoria compartida.
  - SPMD sobre arquitecturas de memoria distribuida.
- Ejemplo: Resolución de ecuaciones
  - Discretización de la solución.
  - División de dominios de cálculo.
  - Mismo programa en cada dominio.
  - Comunicación necesaria para cálculos en los bordes.

DESCOMPOSICIÓN de DOMINIO

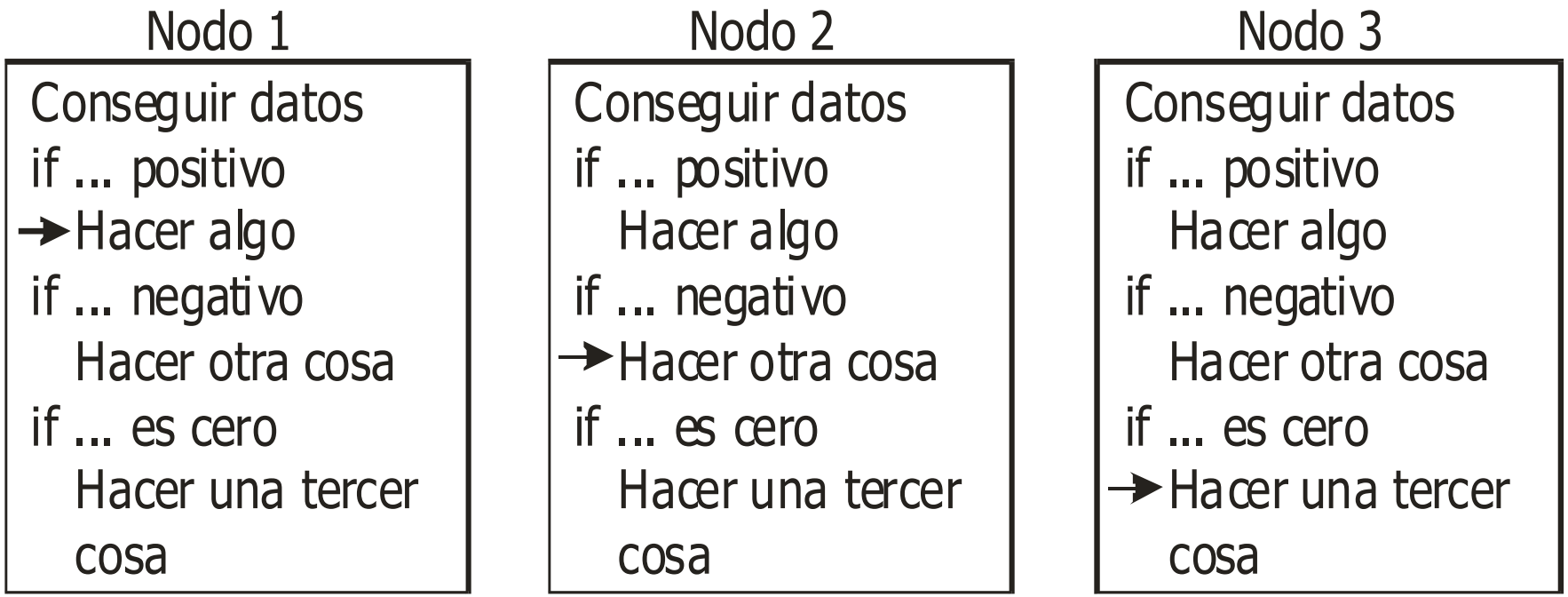
DESCOMPOSICIÓN de DOMINIO (SIMD)

- Cargar Programa
- Broadcast de Instrucción



Los nodos reciben y ejecutan

DESCOMPOSICIÓN de DOMINIO (SPMD)



- Todos los nodos ejecutan el mismo programa, pero no las mismas instrucciones

DESCOMPOSICIÓN de DOMINIO

Ejemplo: multiplicacion de matrices utilizando el metodo de Strassen

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{1,2} &= \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2} \\
\mathbf{C}_{2,1} &= \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{2,2} &= \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}
\end{aligned}$$

$$n^3 = n^{\log_2 8}$$

$$\begin{aligned}
\mathbf{M}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\
\mathbf{M}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\
\mathbf{M}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\
\mathbf{M}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\
\mathbf{M}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\
\mathbf{M}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\
\mathbf{M}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})
\end{aligned}$$

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{1,2} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{2,1} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{2,2} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}$$

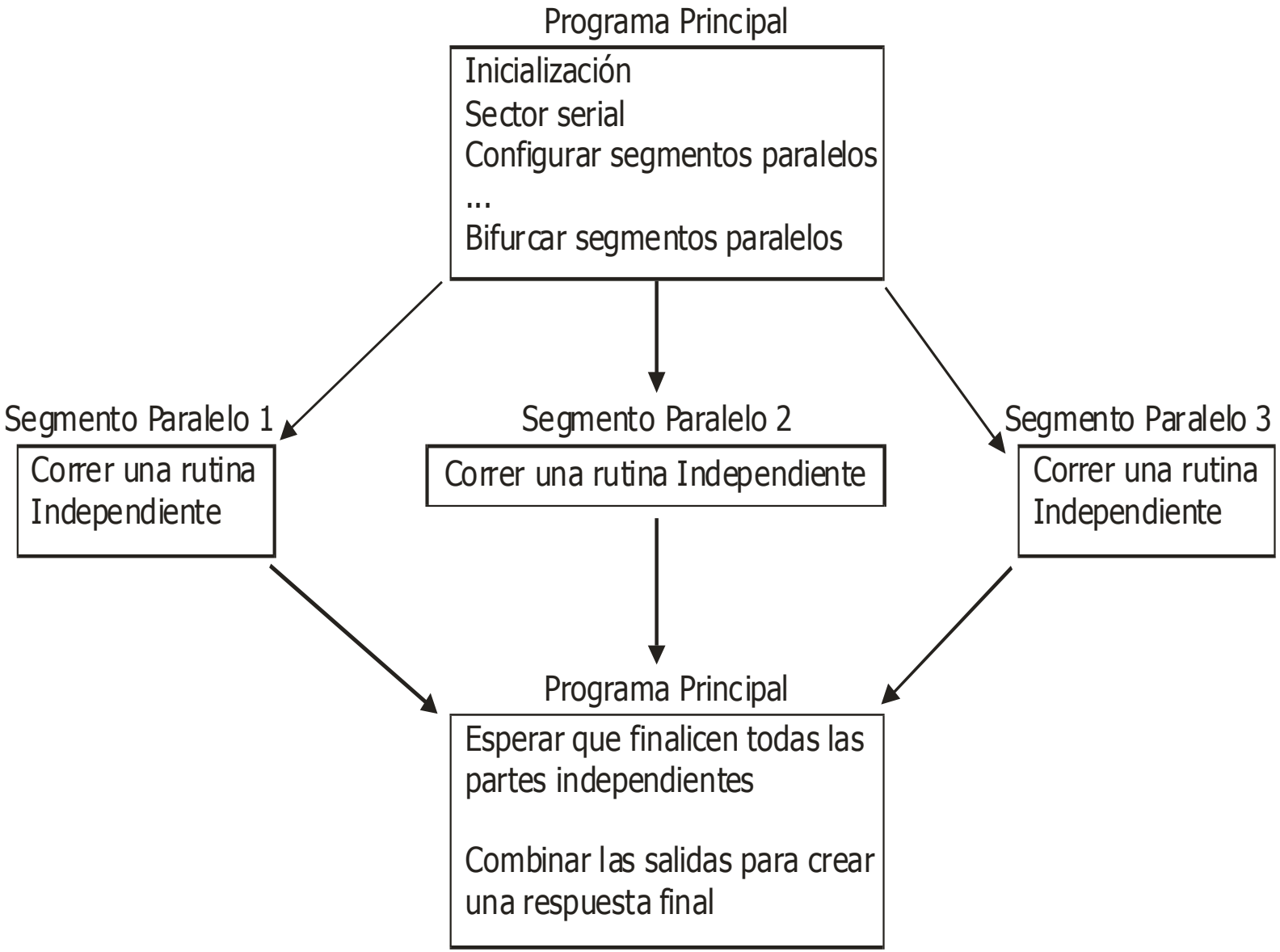
$$n^{\log_2 7} \approx n^{2.807}$$

Ref: http://mathworld.wolfram.com/StrassenFormulas.html
http://en.wikipedia.org/wiki/Strassen_algorithm

DESCOMPOSICIÓN FUNCIONAL

- Se concentra en el particionamiento de las **operaciones** del problema.
  - "Control parallel".
  - Se trata de dividir el procesamiento en tareas disjuntas.
- Luego se examinan los datos a utilizar por las tareas definidas.
  - Si los datos son disjuntos, resulta un PARTICIONAMIENTO COMPLETO.
  - Si los datos NO son disjuntos, resulta un PARTICIONAMIENTO INCOMPLETO. Se requiere replicar los datos o comunicarlos entre los procesos asociados a las diferentes tareas.
- Casos típicos:
  - Distribuir código para asociar requerimientos a recursos locales.
  - Cada tarea trabaja temporalmente con sus datos locales, pero debe existir comunicación.

DESCOMPOSICIÓN FUNCIONAL



PARALELISMO OPTIMISTA

- Realizar operaciones adicionales previendo que deban ser ejecutadas en el futuro.
  - En la hipótesis que hay recursos disponibles para ejecutarlas.

```
if condicion then
        Funcion1()
else
        Funcion2()
end if
```

  - Evaluar `Función1` y `Funcion2` de antemano, independientemente del valor de la condición.

- Típico en aplicaciones de tiempo real.

MODELOS HÍBRIDOS

- **Dos o más tipos de programación paralela dentro del mismo programa.**

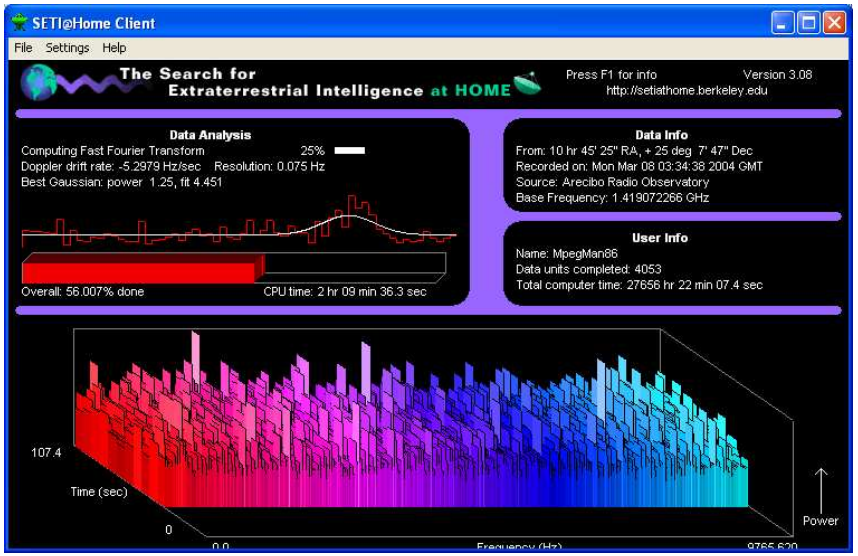
- **Comúnmente utilizados en programas paralelos distribuidos en Internet (donde casi siempre existe la posibilidad de conseguir recursos ociosos adicionales).**

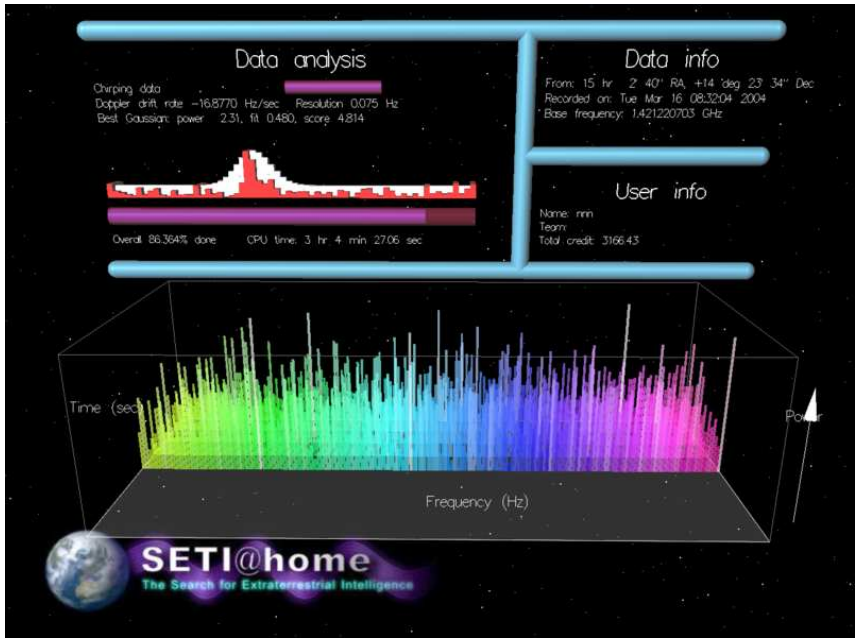
EJEMPLOS

- Cracking de passwords y claves publicas
  - www.distributed.net (R.I.P.)
- Calculo de mínimos globales
  - Función 'constr' de Matlab, algoritmos de optimización.
- Procesamiento de matrices y sistemas lineales
  - Aplicado a métodos paralelizables (Strassen, Jacobi, gradiente conjugado).
- Aplicación de filtros a imágenes
  - Típico de descomposición de datos.
- Modelo de corrientes del Río de la Plata
  - PTIDAL: descomposición de dominio de cálculo.
- Búsqueda de inteligencia extraterrestre
  - www.setiathome.ssl.berkeley.edu

EJEMPLO: SETI@HOME

- **Procesamiento distribuido en Internet.**
- **Objetivos:**
  - **Búsqueda de indicios de inteligencia extraterrestre.**
  - **Demostración práctica de la utilidad de la computación distribuida.**
  - **Lanzado en mayo de 1999.**
  - **Precursor de más de 100 proyectos científicos en el área.**





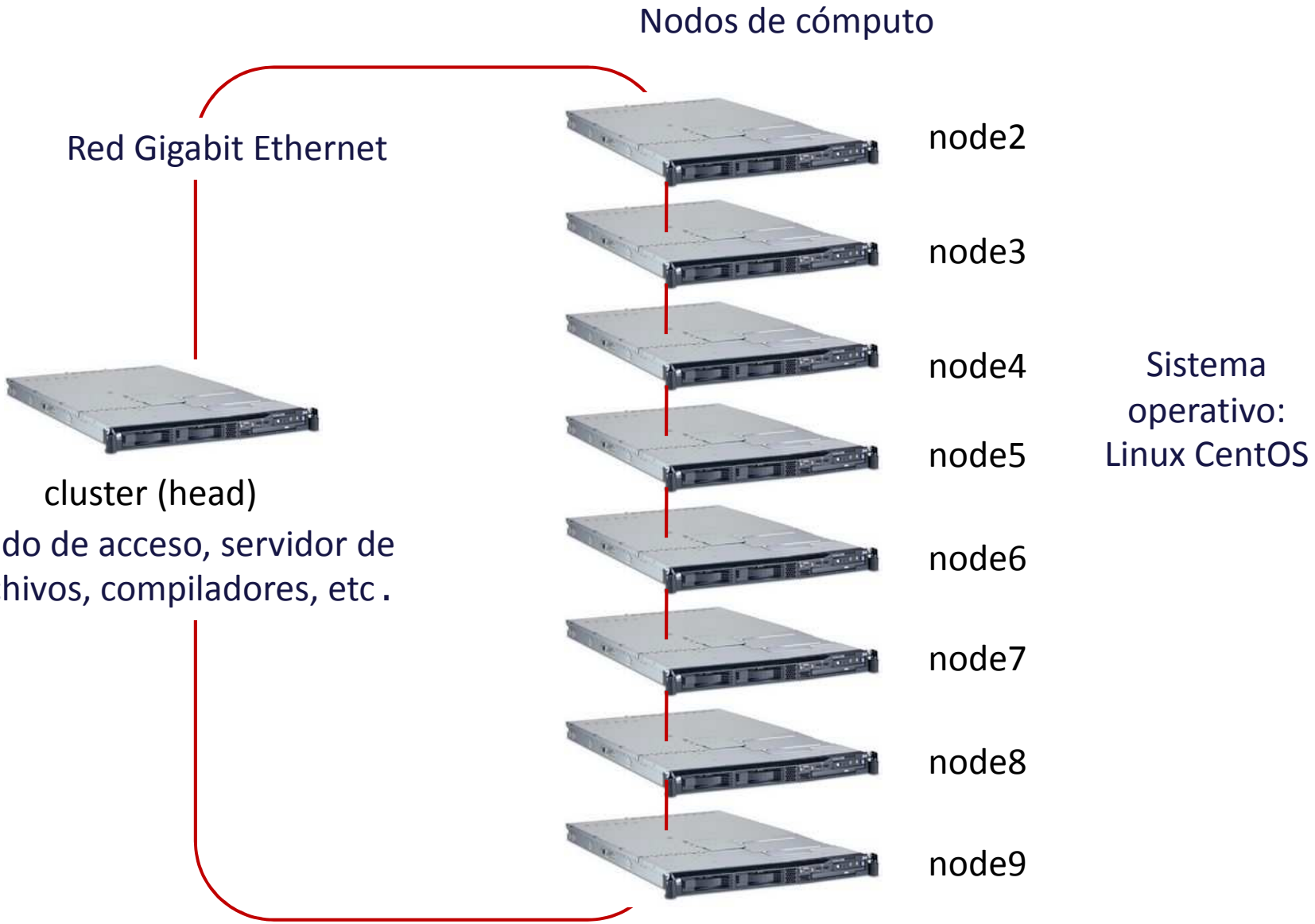
DISEÑO y "PARALELIZACIÓN"

- No siempre se dispone de recursos (en especial de tiempo) para diseñar una aplicación paralela-distribuida de acuerdo a los criterios y técnicas de programación especificadas.
- En múltiples ocasiones, se trata de obtener resultados de eficiencia computacional aceptable adaptando programas secuenciales a los modelos de programación paralela:
  - "Paralelizar" una aplicación existente.
- Problemas
  - Utilización de un código existente que no fue diseñado para ejecutar sobre múltiples recursos computacionales.
  - Modelos enmarañados, "contaminación" del código heredado.

PARALELIZANDO APLICACIONES EXISTENTES

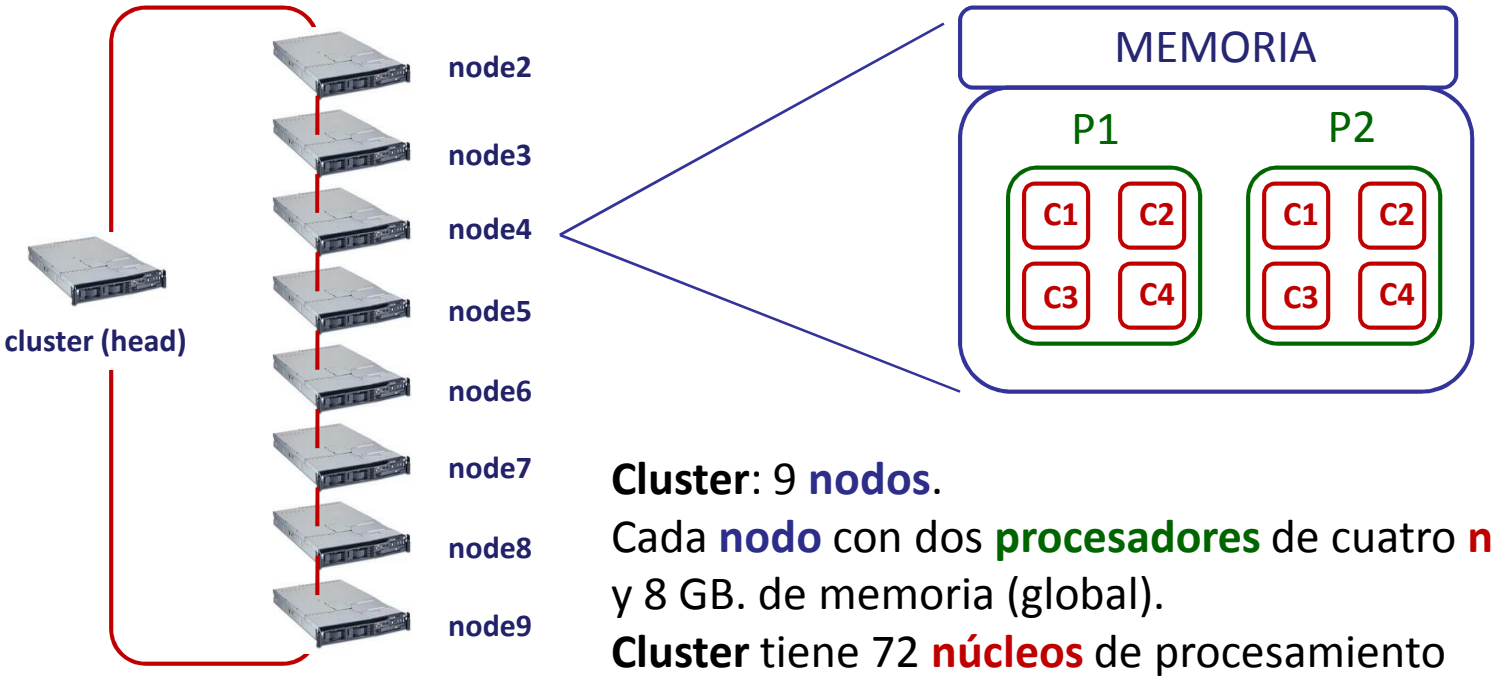
- Aspectos a analizar:
  - ¿Existe una partición funcional evidente?
    - El código modular es el más fácil de paralelizar.
  - ¿Existe forma de particionar los datos?
    - Si existe, ¿cuál es la relación entre procesamiento y datos?
  - ¿Existen muchas variables globales?
    - El manejo de recursos compartidos es un problema a resolver.
    - Considerar el uso de un servidor de variables globales.
  - ¿Es la seguridad un requerimiento importante?
    - Autenticación en ambientes distribuidos.
  - ¿Qué nivel de tolerancia a fallas se requiere?
    - En las computadoras (relanzado de tareas).
    - En la red (reenvio de mensajes).
  - ¿La aplicación utiliza otras formas de IPC?
    - No todos estos servicios existen en computación distribuida.

CLUSTER FING: ESTRUCTURA



CLUSTER FING: ESTRUCTURA

- **Combina arquitectura de cluster (memoria distribuida) y multi-core (memoria compartida).**
- **Permite aprovechar características de ambos modelos de paralelismo: paralelismo de dos niveles.**



Cluster: 9 **nodos**.
Cada **nodo** con dos **procesadores** de cuatro **núcleos** y 8 GB. de memoria (global).
Cluster tiene 72 **núcleos** de procesamiento

CLUSTER FING: APLICACIONES

- ¿Cómo aprovechar las características del cluster FING ?
- Paralelismo de memoria distribuida
  - Primitivas IPC (en C, C++).
  - Bibliotecas de programación paralela:
    - MPI, MPI-2, PVM (para C, C++, FORTRAN).

- Uso óptimo: paralelismo de dos niveles
  - Procesos en diferentes nodos (memoria distribuida).
  - Hilos en multicore (memoria compartida).