**Lección 4. Herramientas, estilos y estructuras en programación paralela**

**|Problemas que plantea la programación paralela |**

Nuevos problemas, respecto a programación secuencial:

* División en unidades de cómputo independientes (tareas).
* Agrupación/asignación de tareas ocarga de trabajo (código, datos) en procesos/threads.
* Asignación a procesadores/núcleos.
* Sincronización y comunicación.

Los debe abordar la herramienta deprogramación o el programador o ambos

Punto de partida

* Partir de una versión secuencial.
* Descripción o definición de la aplicación.

Apoyo:

- Programa paralelo que resuelva un problema parecido.

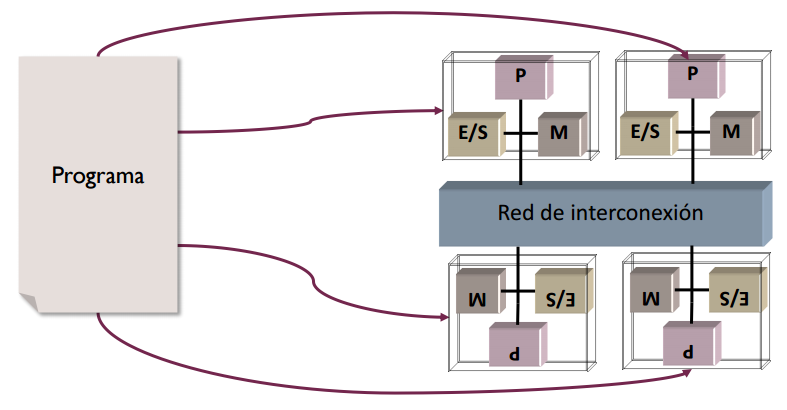
- Versiones paralelas u optimizadas de bibliotecas de funciones:

BLAS (Basic Linear Algebra Subroutine ),

LAPACK (Linear Algebra PACKage), …

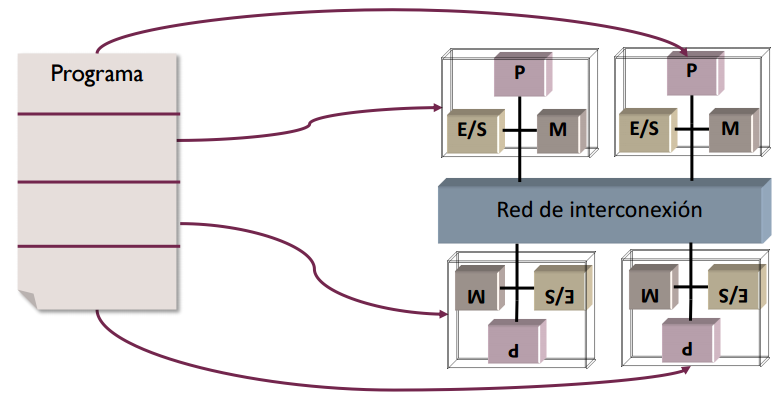
**|Modos de programación MIMD|**

SPMD (Single-Program Multiple Data)

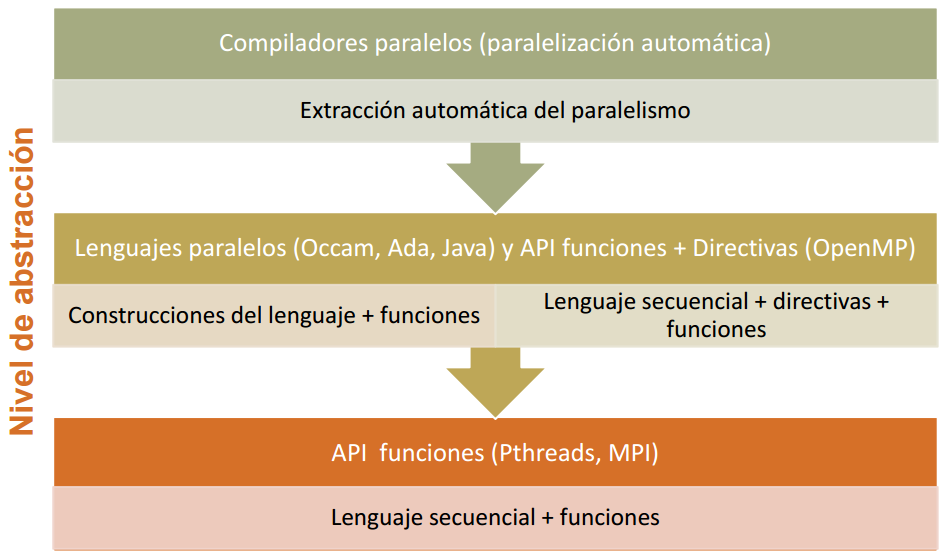
****

**|Modos de programación MIMD|**

MPMD (Multiple-Program Multiple Data)

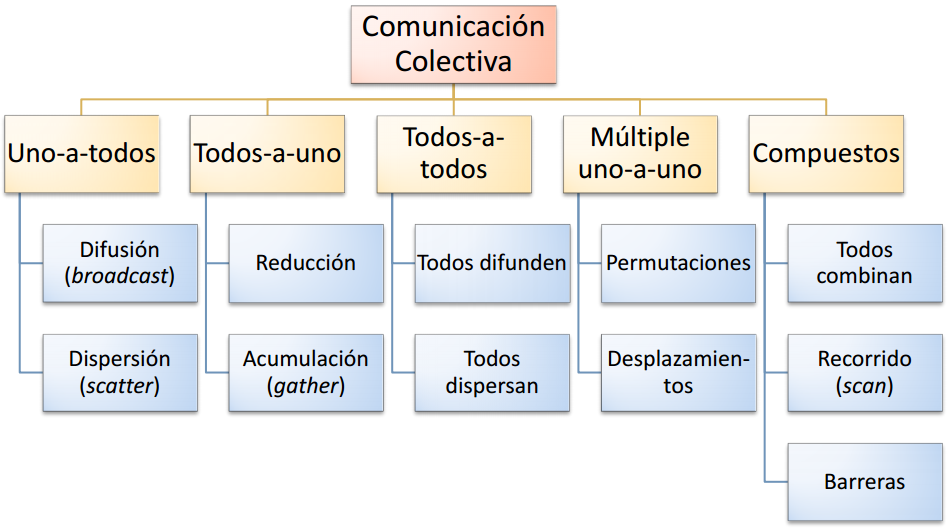
****

**|Herramientas de programación paralela|**

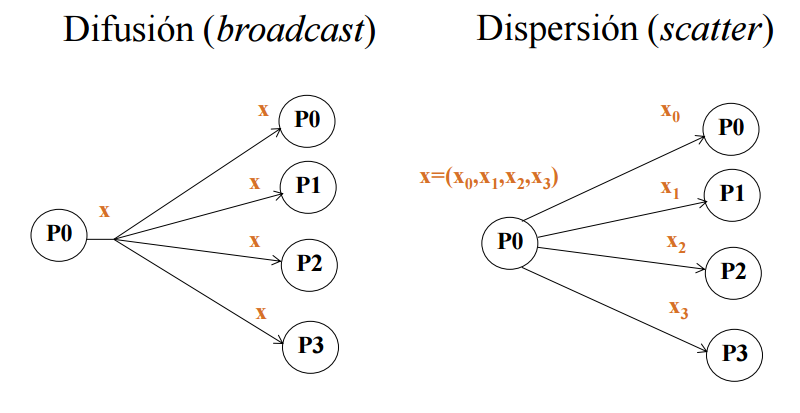


* Las herramientas permiten de forma implícita o explícita (lo hace el programador):
  + Localizar paralelismo o descomponer en tareasindependientes (descomposition)
  + Asignar las tareas, es decir, la carga de trabajo (código +datos), a procesos/threads (scheduling)
  + Crear y terminar procesos/threads (o enrolar y desenrolaren un grupo)
  + Comunicar y sincronizar procesos/threads
* El programador, la herramienta o el SO se encarga de
  + Asignar procesos/threads a unidades de procesamiento(mapping)

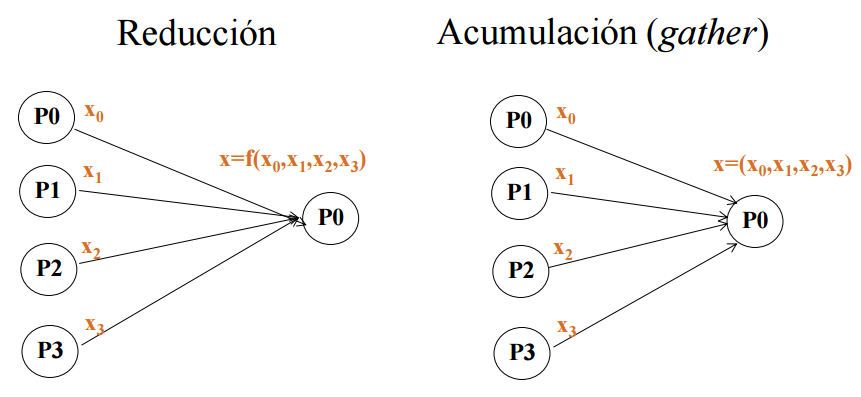
**|Comunicaciones colectivas|**



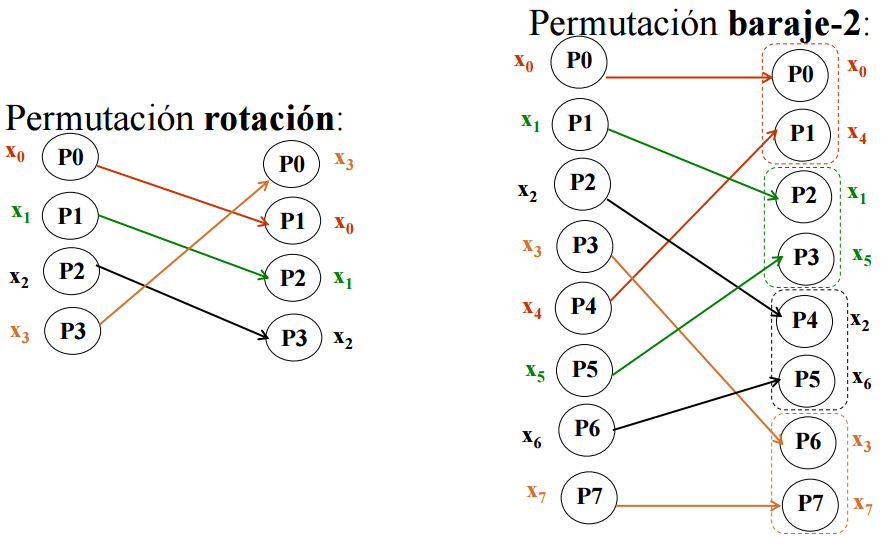
Comunicación uno-a-todos



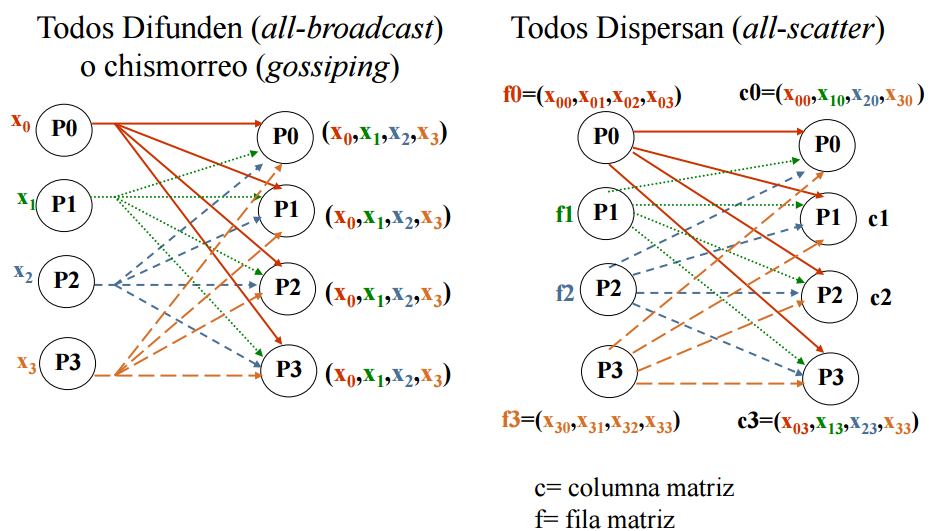
Comunicación todos-a-uno



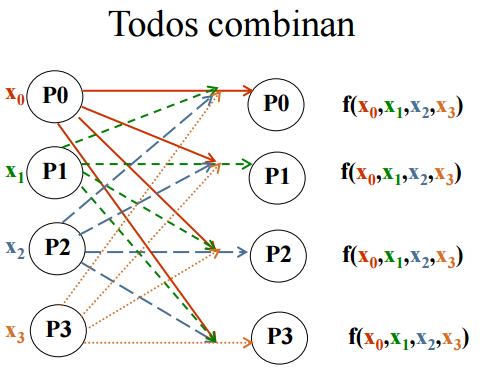
Comunicación múltiple uno-a-uno

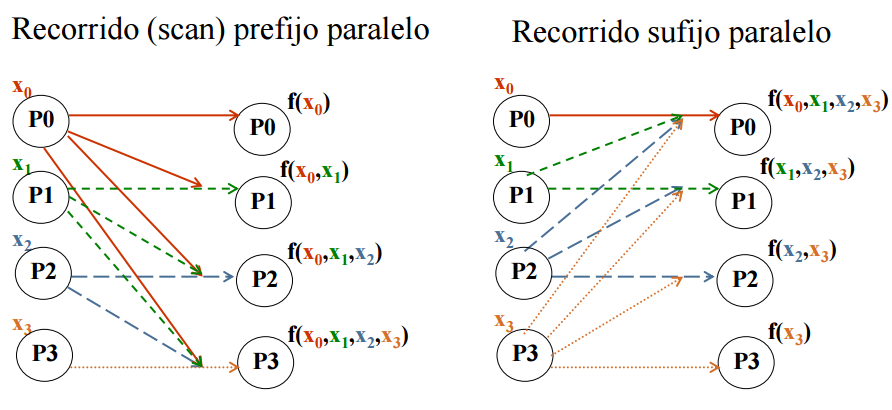


Comunicación todos-a-todos

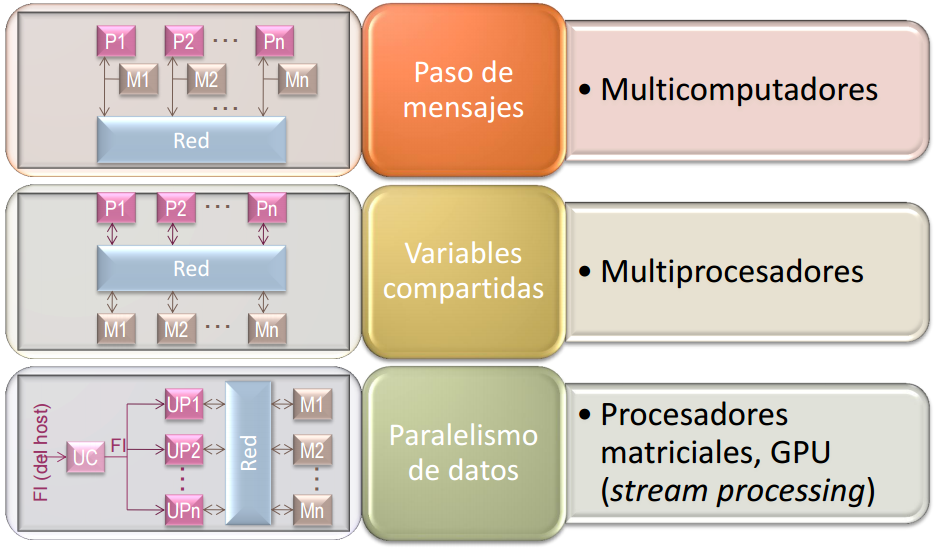


Servicios compuestos





**|Estilos de programación y arquitecturas paralelas|**

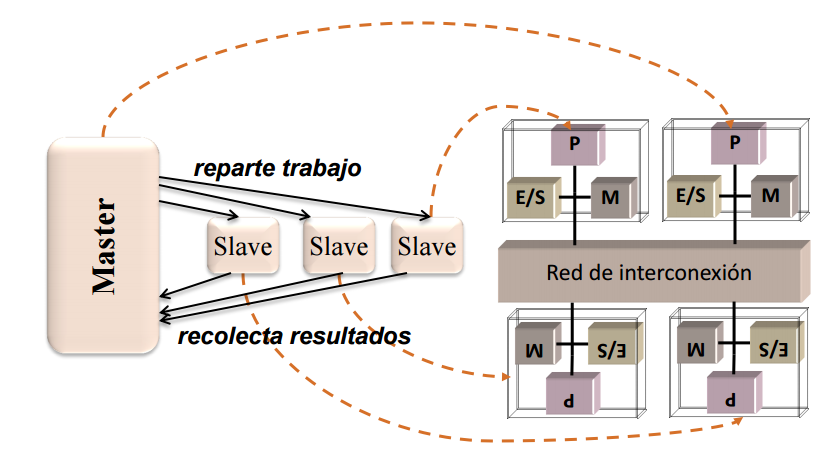


* Paso de mensajes (message passing)
  + Lenguajes de programación: Ada, Occam
  + API (Bibliotecas de funciones): MPI, PVM
* Variables compartidas (shared memory, shared variables)
  + Lenguajes de programación: Ada, Java
  + API (directivas del compilador + funciones): OpenMP
  + API (Bibliotecas de funciones): POSIX Threads, shmem, Intel TBB(Threading Building Blocks)
* Paralelismo de datos (data parallelism)
  + Lenguajes de programación: HPF (High Performance Fortran),Fortran 95 (forall, operaciones con matrices/vectores)
  + API (funciones - stream processing): Nvidia CUDA

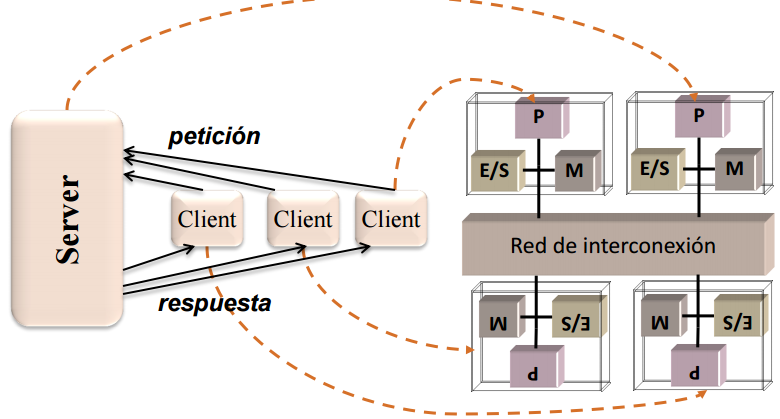
**|Estructuras típicas de procesos/threads/tareas|**

****

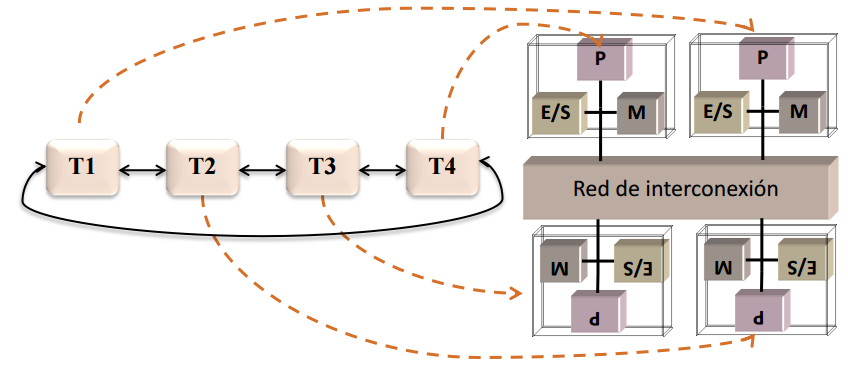
Master-Slave o granja de tareas



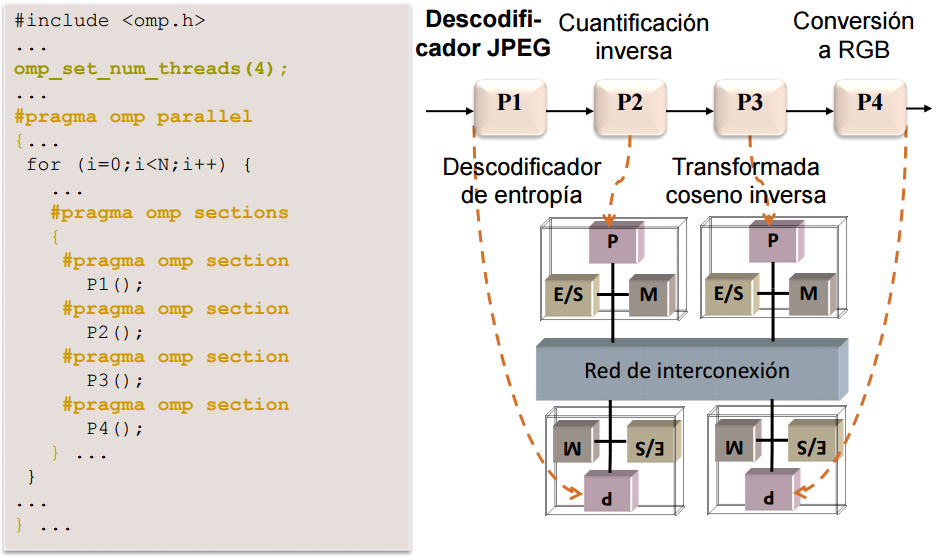
Cliente-Servidor



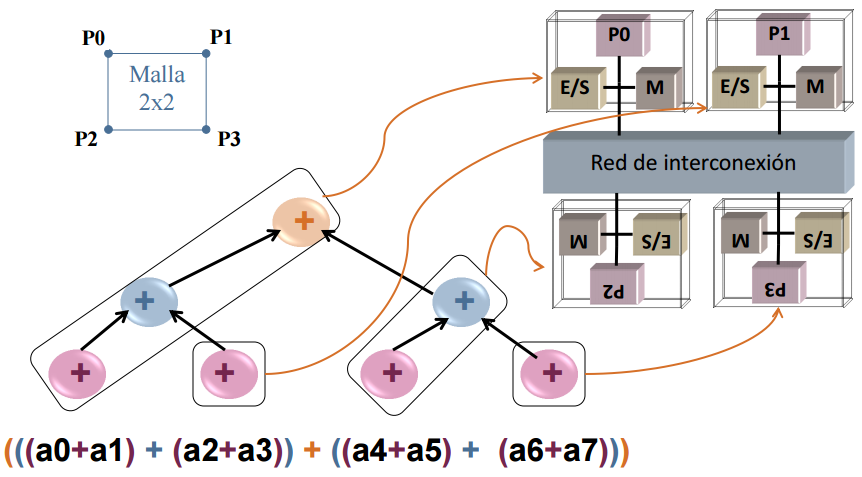
Descomposición de dominio o descomposición de datos



Estructura segmentada o de flujo de datos



Divide y vencerás o descomposición recursiva

****

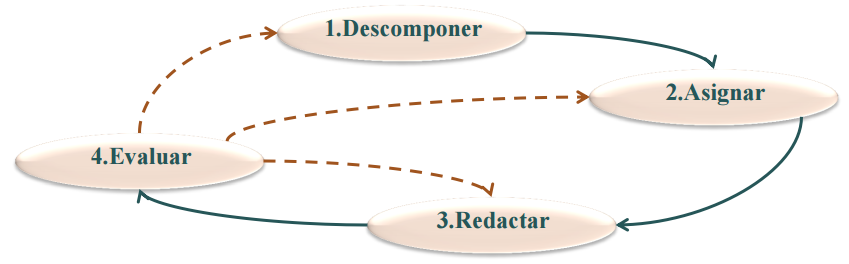
**Lección 5. Proceso de paralelización**

**|** **Proceso de paralelización |**

1.Descomponer (descomposition) en tareas independientes o Localizar paralelismo 2.Asignar (planificar+mapear) tareas a procesos y/o threads.

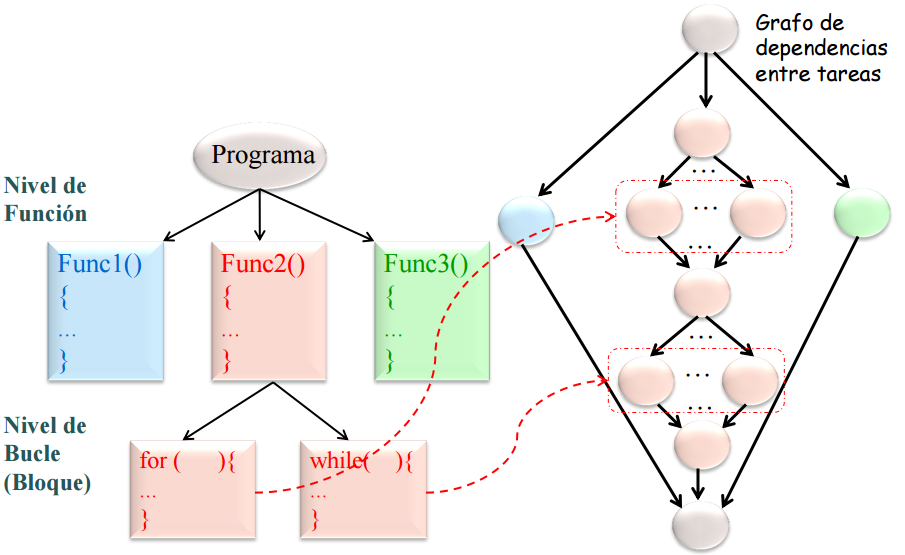
3.Redactar código paralelo.

4.Evaluar prestaciones.

****

**1.Descomposición en tareas independientes**

* Análisis de dependencia entre funciones.
* Análisis de dependencia entre iteraciones de bucles.

****

**2.Asignar (planificar+mapear) tareas a procesos y/o threads.**

Planificación: agrupar tareas en threads

Mapeo: asignar threads a cores/ procesadores

* Incluimos: agrupación de tareas en procesos/threads (scheduling) y mapeo a procesadores/cores (mapping)
* La granularidad de la carga asignada a losprocesos/threads depende de:
  + número de cores o procesadores o elementos deprocesamiento
  + tiempo de comunicación/sincronización frente a tiempo decálculo
* Equilibrado de la carga (tareas = código + datos) o loadbalancing:
  + Objetivo: unos procesos/threads no deben hacer esperar aotros
* ¿De qué depende el equilibrado?
  + La arquitectura:
    - homogénea frente a heterogénea,
    - uniforme frente a no uniforme
  + La aplicación/descomposición
* Tipos de asignación:
  + Estática
    - Está determinado qué tarea va a realizar cada procesador o core
    - Explícita: programador
    - Implícita: herramienta de programación al generar el código ejecutable
  + Dinámica (en tiempo de ejecución)
    - Distintas ejecuciones pueden asignar distintas tareas a un procesador o core
    - Explícita: el programador
    - Implícita: herramienta de programación al generar el código ejecutable

Mapeo de procesos/threads a unidades de procesamiento

Normalmente se deja al SO el mapeo de threads (light process)

Lo puede hacer el entorno o sistema en tiempo de ejecución (runtime system de la herramienta de programación)

El programador puede influir

**3.Redactar código paralelo.**

Haciendo uso de OpenMP/C o MPI/C

**4.Evaluar prestaciones.**

Siguiente Leccion.

**Lección 6. Evaluación de prestaciones en procesamiento paralelo**

**|** **Evaluación de prestaciones|**

Medidas usuales

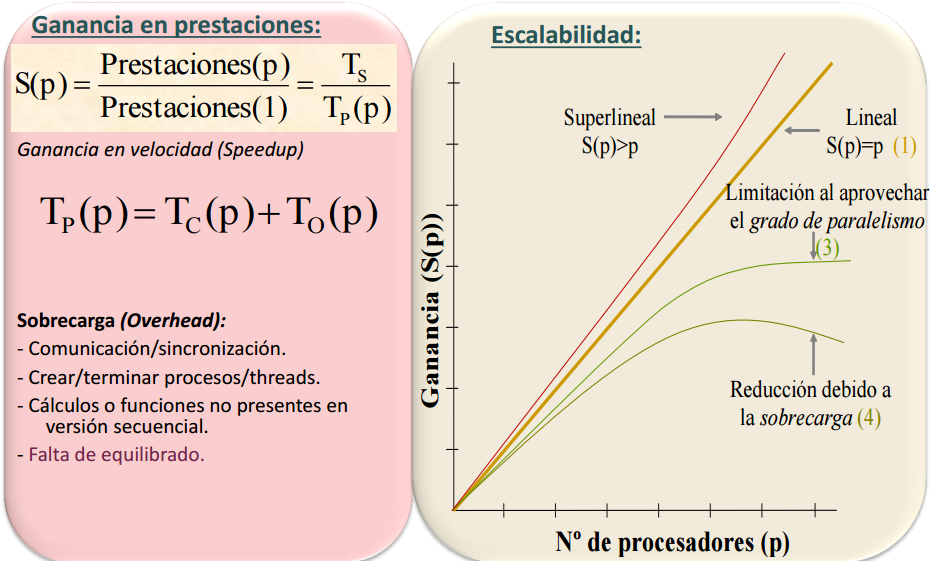
* Tiempo de respuesta
  + Real (wall-clock time, elapsed time) (/usr/bin/time)
  + Usuario, sistema, CPU time = user + sys
* Productividad

Escalabilidad

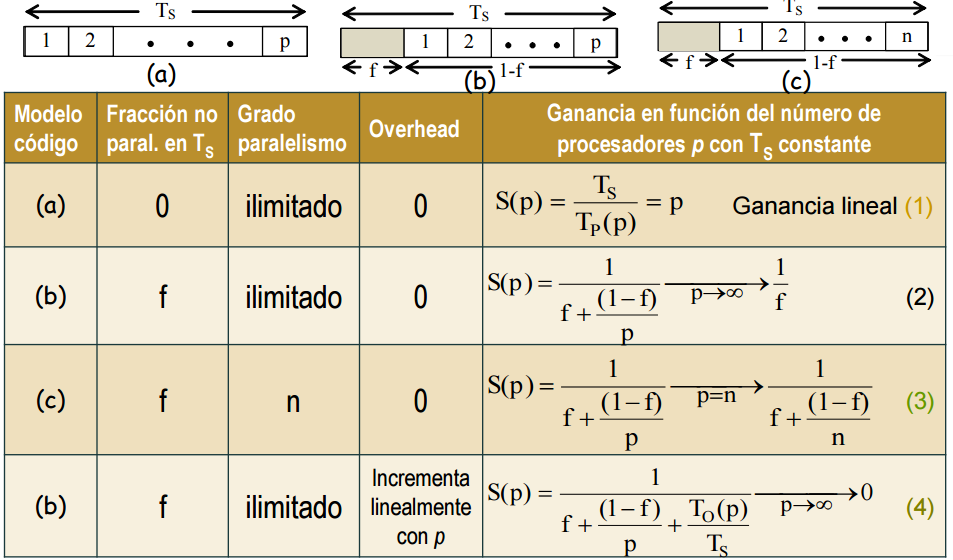
Eficiencia

* Relación prestaciones/prestaciones máximas
* Rendimiento = prestaciones/nº\_recursos
* Otras: Prestaciones/consumo\_potencia,prestaciones/área\_ocupada

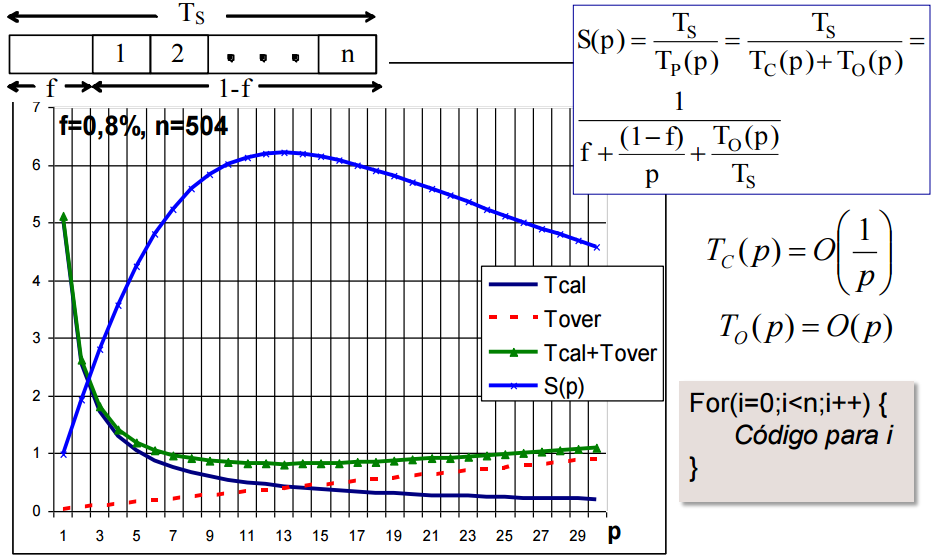
**|Ganancia en prestaciones. Escalabilidad|**

****

**|Ganancia en prestaciones. Ganancia máxima|**

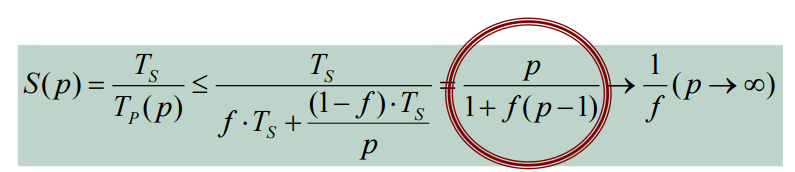
****

**|Número de procesadores óptimo|**

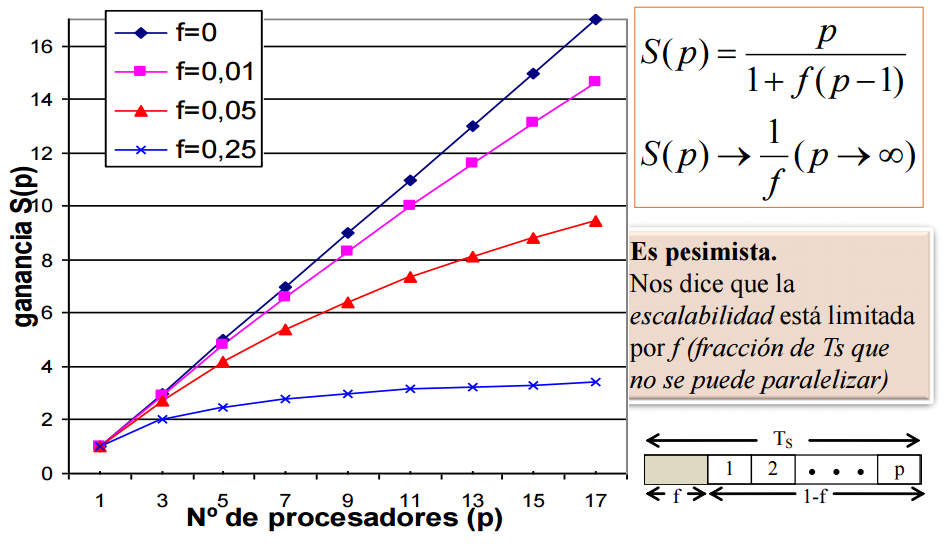
****

**|Ley de Amdahl|**

Ley de Amdahl: la ganancia en prestaciones utilizando pprocesadores está limitada por la fracción de código que no se puede paralelizar (2):



* S : Incremento en velocidad que se consigue al aplicar unamejora. (paralelismo)
* p : Incremento en velocidad máximo que se puedeconseguir si se aplica la mejora todo el tiempo. (número de procesadores)
* f : fracción de tiempo en el que no se puede aplicar lamejora. (fracción de t. no paralelizable)



**|Ganancia escalable o Ley de Gustafson |**

