COMPUTACIÓN DE ALTA PERFORMANCE

Curso 2024

Sergio Nesmachnow (sergion@fing.edu.uy)

Centro de Cálculo

cecal

FACULTAD DE INGENIERIA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

TEMA 4
EVALUACIÓN DE DESEMPEÑO

Contenido

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

TEMA 4: EVALUACIÓN DE DESEMPEÑO

1. Modelo de performance
2. Medidas: speedup, eficiencia, paralelicibilidad
3. Ley de Amdahl
4. Scheduling y balance de cargas

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

MEDIDAS DE PERFORMANCE
4.1: MODELO DE PERFORMANCE

Evaluación de desempeño

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Objetivos:
  - Estimación de desempeño de algoritmos paralelos
  - Comparación con algoritmos seriales
- Factores intuitivos para evaluar la performance:
  - Tiempo de ejecución
  - Utilización de los recursos disponibles

Evaluación de desempeño

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- El desempeño es un concepto complejo y polifacético.
- El tiempo de ejecución es la medida tradicionalmente utilizada para evaluar la eficiencia computacional de un programa.
- El almacenamiento de datos en dispositivos y la transmisión de datos entre procesos influyen en el tiempo de ejecución de una aplicación paralela.
- La utilización de recursos disponibles y la capacidad de utilizar mayor poder de cómputo para resolver problemas más complejos o de mayor dimensión, son las características más deseables para aplicaciones paralelas.

EVALUACIÓN DE DESEMPEÑO
TIEMPO DE EJECUCIÓN

- El tiempo total de ejecución se utiliza habitualmente como medida del desempeño:
  - Es simple de medir.
  - Es útil para evaluar el esfuerzo computacional requerido para resolver un problema.
- El modelo de performance que se utilizará en el curso considera como medida fundamental el tiempo de ejecución.

Modelo de performace

- Métricas como función del tamaño del problema (*n*), procesadores disponibles (*p*), número de procesos (*U*), y otras variables dependientes del algoritmo y/o de características del hardware sobre el que se ejecuta:

$$T = f(n, p, U, \ldots)$$

- Tiempo de ejecución de un programa paralelo
  - Tiempo que transcurre entre el inicio de la ejecución del primer proceso hasta el fin de ejecución del último proceso.
- Diferentes estados: procesamiento efectivo, comunicación y ocioso.

$$T = T_{PROC} + T_{COM} + T_{IDLE}$$

  - *p* tareas ejecutando en *p* procesadores, tiempos del procesador *i* en la etapa correspondiente (estadísticas sobre utilización de recursos).

$$T = \frac{1}{P}\left(\sum_{i=1}^{P} T_{PROC}^{i} + \sum_{i=1}^{P} T_{COM}^{i} + \sum_{i=1}^{P} T_{IDLE}^{i}\right)$$

Tiempo de ejecución

- El tiempo de cómputo depende de complejidad y dimensión del problema, del número de tareas utilizadas y de las características de los elementos de procesamiento (hardware, heterogeneidad, no dedicación).
- El tiempo de comunicación depende de localidad de procesos y datos (comunicación inter e intra-procesador, canal de comunicación).
- Costo de comunicación interprocesador: tiempo necesario para el establecimiento de la conexión (latencia) y tiempo de transferencia de información (dado por ancho de banda del canal físico).
- Para enviar un mensaje de L bits, se requiere un tiempo

$$T = latencia + L \,.\, T_{TR}$$

(siendo T_{TR} el tiempo necesario para transferir un bit).

Tiempo de ejecución

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- El tiempo en estado ocioso es consecuencia del no determinismo en la ejecución, minimizarlo debe ser un objetivo del diseño.
- Motivos: ausencia de recursos de cómputo disponibles o ausencia de datos sobre los cuales operar.
- Soluciones: aplicar técnicas de balance de carga para distribuir los requerimientos de cómputo o rediseñar el programa para distribuir los datos adecuadamente.

Mejora de desempeño

- SPEEDUP
  - Es una medida de la mejora de rendimiento (performance) de una aplicación al aumentar la cantidad de procesadores (comparado con el rendimiento al utilizar un solo procesador).

- SPEEDUP ABSOLUTO $S_N = T_0 / T_N$

  siendo:
  - T_0 el tiempo del mejor algoritmo secuencial que resuelve el problema (considerando el tiempo de ejecución, o sea el algoritmo más rápido que lo resuelve).
  - T_N el tiempo del algoritmo paralelo ejecutado sobre N recursos de cómputo.

Speedup

- Siendo T_K el tiempo total de ejecución de una aplicación utilizando K procesadores …
- Se define el SPEEDUP ALGORÍTMICO como

$$S_N = T_1 / T_N$$

siendo T_1 el tiempo en un procesador (secuencial) y T_N el tiempo del algoritmo paralelo utilizando N recursos de cómputo.

- El speedup algorítmico es el más utilizado frecuentemente en la práctica para evaluar la mejora de desempeño (considerando el tiempo de ejecución) de programas paralelos.
- El speedup absoluto es difícil de calcular, porque no es sencillo conocer el mejor algoritmo serial que resuelve un problema determinado.

Speedup

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

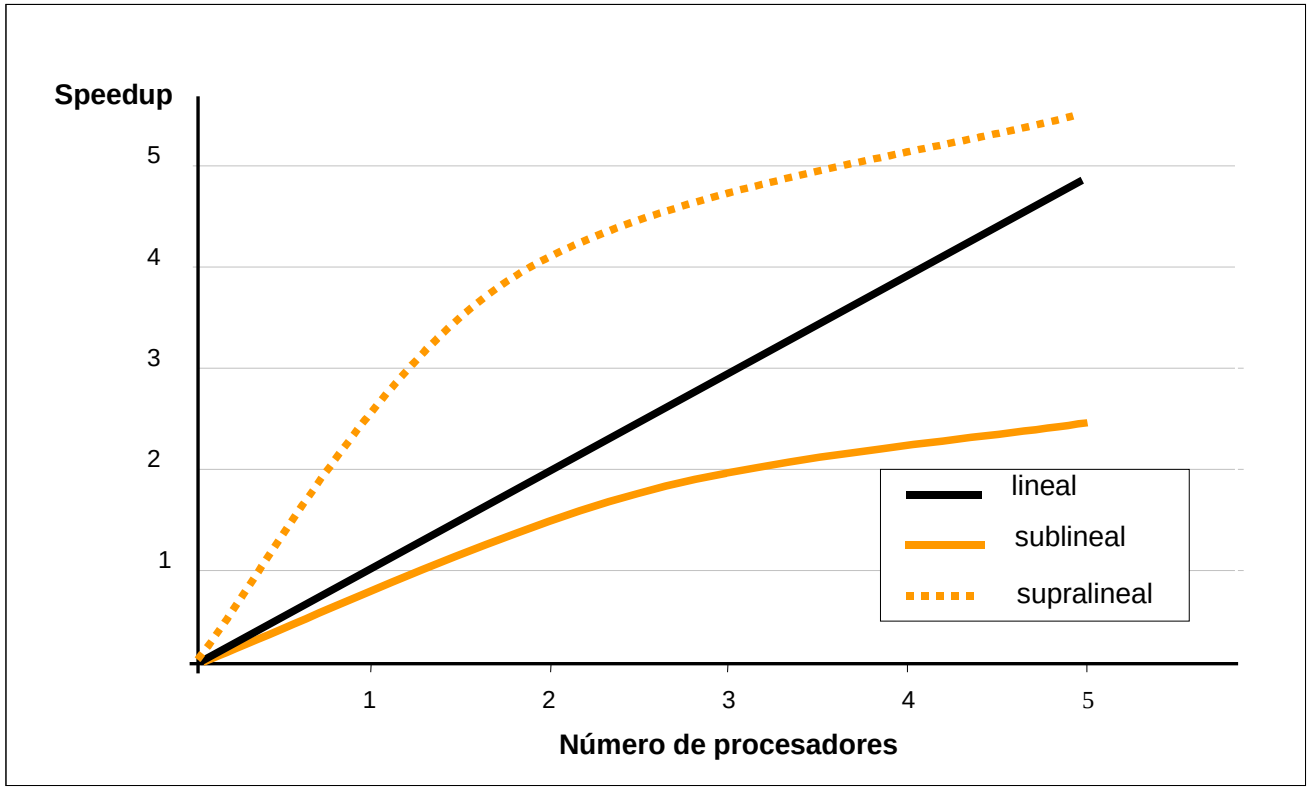
- Al medir los tiempos de ejecución, debe considerarse la configuración de la máquina paralela utilizada.

- Comparación justa:
  - Para calcular speedup absoluto utilizar el mejor algoritmo serial disponible (o el mejor conocido, en caso de no existir una cota inferior para su complejidad).
  - Debe analizarse el hardware disponible.
  - Coeficientes (“pesos”) asignados a los equipos en recursos de cómputo heterogéneos.
  - Tomar en cuenta el asincronismo: medidas estadísticas.
  - Existen algoritmos diseñados para la comprobación (benchmarks).

Speedup

- La situación ideal es lograr el speedup lineal.
  - Al utilizar *p* procesadores obtener una mejora de factor *p*.
- La realidad indica que es habitual obtener speedup sublineal.
  - Utilizar *p* procesadores no garantiza una mejora de factor *p*.
  - Causas: procesamiento no paralelizable, demoras en las comunicaciones y sincronizaciones, tiempos ociosos, etc.
- En ciertos casos es posible obtener valores de speedup superlineal.
  - Tomando partido de ciertas características especiales del problema o del hardware disponible.

Speedup



Speedup lineal, sublineal y supralineal

Speedup

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Motivos que impiden el crecimiento lineal del SPEEDUP:
  - Demoras introducidas por las comunicaciones.
  - Overhead en intercambio de datos.
  - Overhead en trabajo de sincronización.
  - Existencia de tareas no paralelizables.
  - Cuellos de botella en el acceso a recursos de hardware necesarios.

- Los factores mencionados incluso pueden producir que el uso de más procesadores sea contraproducente para la performance de la aplicación.

Eficiencia computacional

- La eficiencia computacional se define mediante:

$$E_N = T_1/(N \times T_N)$$

  - Es decir, $E_N = S_N / N$.
- Corresponde a un valor normalizado del speedup (entre 0 y 1), respecto a la cantidad de recursos de cómputo utilizados.
- Es una medida relativa que permite la comparación de desempeño en diferentes entornos de computación paralela.

- Valores de eficiencia cercanos a uno identificarán situaciones casi ideales de mejora de performance.

Paralelicibilidad

- La paralelicibilidad de un algoritmo paralelo se define como:

$$P = TP_1 / TP_N$$

- Siendo:
  - TP1 el tiempo que toma a un computador paralelo ejecutar un algoritmo paralelo en un único recurso de cómputo.
  - TP_N el tiempo que toma al mismo computador paralelo ejecutar el mismo algoritmo paralelo en N recursos de cómputo.

Paralelicibilidad

- Diferencias con el speedup
  - El speedup considera el tiempo de un algoritmo secuencial (el mejor existente o conocido) para la comparación.
  - La paralelicibilidad toma en cuenta el tiempo de ejecución de un algoritmo paralelo en un único procesador.

  - El speedup evalúa la mejora de desempeño al utilizar técnicas de programación paralela.
  - La paralelicibilidad da una medida de cuán paralelizable o escalable resulta el algoritmo paralelo utilizado.

Escalabilidad

- Capacidad de mejorar el desempeño al utilizar recursos de cómputo adicionales para la ejecución de aplicaciones paralelas.
  - Eventualmente, para resolver instancias más complejas de un determinado problema.

- Constituye una de las principales características deseables de los algoritmos paralelos y distribuidos.

Ejemplo

- Si el mejor algoritmo secuencial para resolver un problema requiere 8 unidades de tiempo para ejecutar en uno de los nodos de un computador paralelo homogéneo y 2 unidades al utilizar 5 procesadores.
- El speedup obtenido al paralelizar la solución es:

$$S_5 = T_1 / T_5 = 8 / 2 = 4.$$

- La eficiencia toma en cuenta la cantidad de procesadores utilizados:

$$E_5 = S_5 / 5 = 4 / 5 = 0{,}8.$$

  - Corresponde a un speedup sublineal.

- La paralelicibilidad toma en cuenta al mismo algoritmo paralelo para la medición de tiempos. Suponiendo que el algoritmo paralelo toma ventajas de la estructura del problema y ejecuta en un único procesador en 6 unidades de tiempo, se tendrá :

$$P_5 = TP_1 / TP_5 = 6 / 2 = 3.$$

Otros modelos de desempeño

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

EVALUANDO la UTILIZACIÓN de RECURSOS DISPONIBLES

- UTILIZACIÓN
  - Mide el porcentaje de tiempo que un procesador es utilizado durante la ejecución de una aplicación paralela.

USO = tiempo ocupado / (tiempo ocioso + tiempo ocupado)

  - Lo ideal es mantener valores equitativos de utilización entre todos los procesadores de una máquina paralela.

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

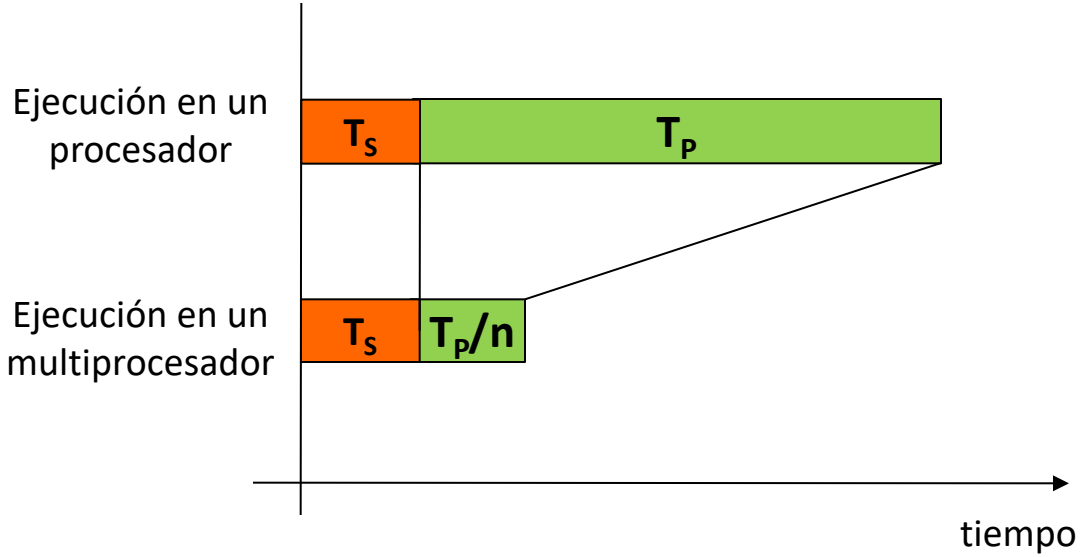
MEDIDAS DE PERFORMANCE
4.2: LEY DE AMDAHL

Ley de Amdahl

LEY DE AMDAHL (1967):

"La parte serial de un programa determina una cota inferior para el tiempo de ejecución, aún cuando se utilicen al máximo técnicas de paralelismo."

Gene Amdahl



Ley de Amdahl

- Sea P la fracción de un programa que es paralelizable, y sea S = 1 – P la parte secuencial restante, entonces el speedup al trabajar con N procesadores estará acotado por:

$$S_N \leq 1 / (S + P / N)$$

- El valor asintótico para S_N es 1/S.
- Ejemplo:
  - Si solo el 50% de un programa es paralelizable, S = P = ½, se tiene que $S_N = 2 / (1 + 1/N)$ por lo que será $S_N \leq 2$ (valor asintótico), independientemente de N (cantidad de procesadores utilizados).

- Visión sombría del procesamiento paralelo (conjetura de Minsky)
  - La cota inferior del tiempo de ejecución de un programa paralelo es $O(\log_2 N)$, y funciona como valor asintótico para la mejora de tiempos de ejecución.

Ley de Amdahl
una visión más optimista

- Asumiendo que:
  - el tamaño (o la complejidad) del problema (n) crece con el numero de procesadores a usar.
  - la parte secuencial del programa (Ts) se mantiene constante.
  - la parte paralela del programa (Tp) crece según el tamaño del problema (n $\times$ Tp, $n^2 \times$ Tp, etc).

- Entonces las fracciones de tiempo de ejecución de las partes secuencial y paralela del programa resultan:

$$S = Ts / (Ts + n \times Tp)$$
$$P = n \times Tp / (Ts + n \times Tp)$$

Ley de Amdahl una interpretación más optimista

- Del razonamiento anterior es posible deducir que:

$$S_N \leq N \times (1 + n \times u) / (N + n \times u)$$

siendo u = Tp/Ts.

- Se concluye que dado un número de recursos de cómputo N, existirán problemas de complejidad n cuyas partes serial y paralela cumplan que $n \times u \sim N$ y entonces la cota teórica corresponde al speedup lineal: $S_N \leq N$.

LEY DE AMDAHL
ARGUMENTO DE GUSTAFFSON-BARSIS

- La idea detrás del argumento optimista tiene el valor de tomar en cuenta la COMPLEJIDAD de las tareas realizadas.
- Habitualmente las tareas no paralelizables son de complejidad lineal (O(n)), como las lecturas de datos de entrada, mientras que los algoritmos paralelizables tienen una complejidad mayor.
- Si se logra reducir el orden de complejidad del algoritmo mediante el uso de múltiples procesadores, el speedup ideal sería O(n).

$$speedup = O(n)+O(n^3) / (O(n)+O(n^2))$$

cuyo valor asintótico es O(n) (~ O(N))

- Gustaffson (1988): Es posible alcanzar valores de speedup de O(N).

Ley de Amdahl: conclusión

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

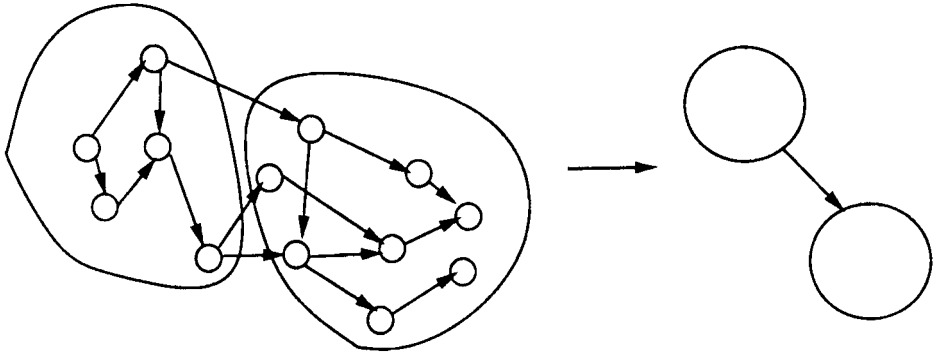
CONCLUSIÓN de la LEY de AMDAHL:

La razón para utilizar un número mayor de procesadores debe ser resolver problemas más grandes o más complejos, y no para resolver más rápido un problema de tamaño fijo.

Factores que afectan el desempeño

GRANULARIDAD

- Cantidad de trabajo que realiza cada nodo

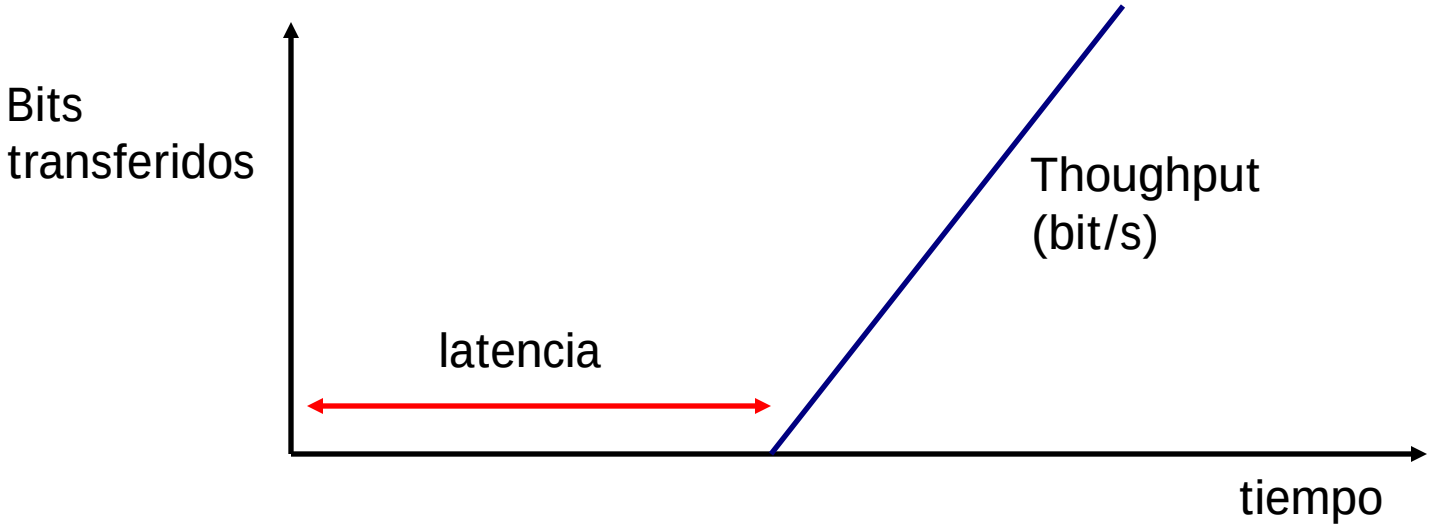


TAREA	GRANULARIDAD
operación sobre bits	super fino (extremely fine grain)
una instrucción	grano fino (fine grain)
un proceso	grano grueso (large grain)

- Aumentar la granularidad:
  - Disminuye el overhead de control y comunicaciones.
  - Disminuye el grado de paralelismo.

Factores que afectan el desempeño

LATENCIA



- Los tiempos de comunicación entre procesadores dependen del ancho de banda disponible y de la LATENCIA del canal.
- Latencias altas implicarán utilizar alta granularidad (comunicaciones menos frecuentes, mensajes mas largos, etc.).

Desempeño de algoritmos paralelos

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- El OBJETIVO del diseño de aplicaciones paralelas es lograr un compromiso entre:
  - El grado de paralelismo obtenido.
  - El overhead introducido por las tareas de sincronización y comunicación.
- Las técnicas de scheduling y de balance de carga son útiles para mejorar el desempeño de aplicaciones paralelas.

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

MEDIDAS DE PERFORMANCE

4.3: SCHEDULING Y BALANCE DE CARGA

Scheduling (mapeo)

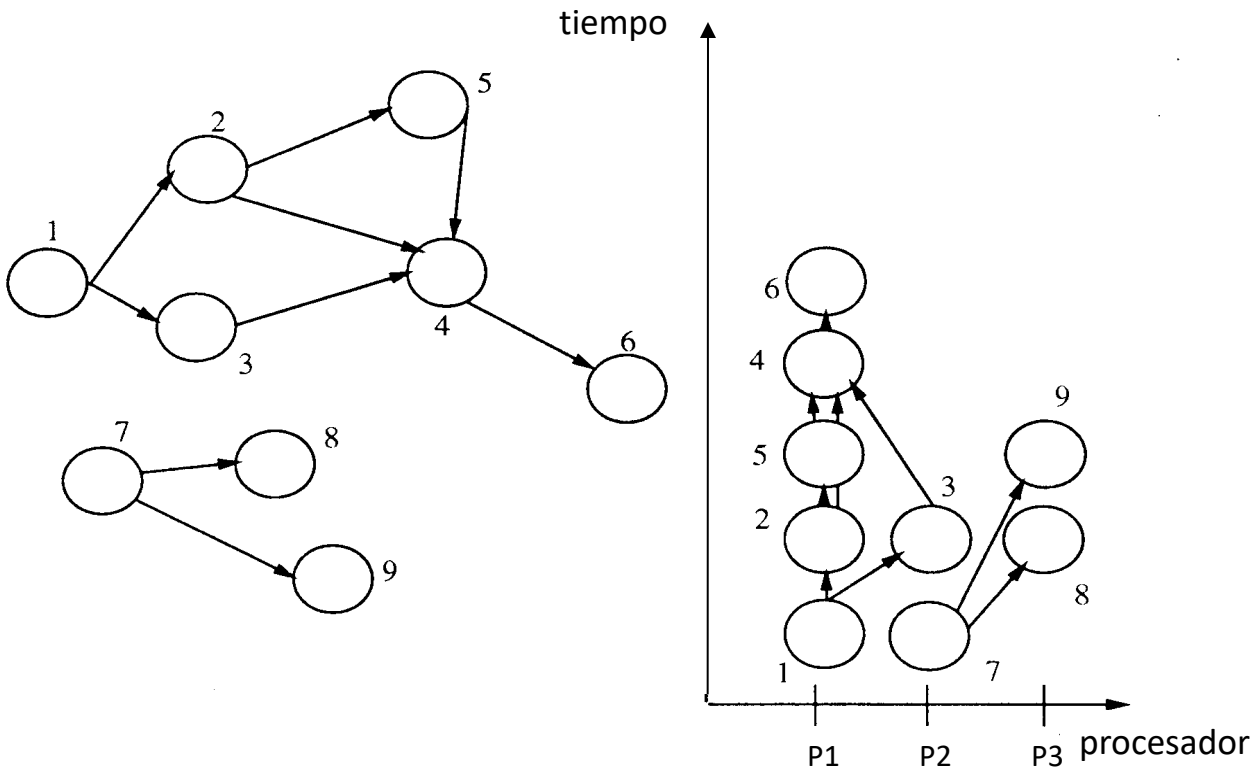
- Refiere a la etapa de asignación de recursos a los múltiples procesos que ejecutarán en paralelo.

- El mapeo determina dónde y cuándo se ejecutará una tarea.
- Las dependencias entre tareas definen pautas para la asignación.

- Usualmente se utilizan técnicas de investigación operativa para planificar la asignación de recursos de modo de optimizar un determinado criterio.
  - Tiempo total de ejecución.
  - Utilización de los recursos.
  - Balance de cargas entre recursos.

Scheduling

- El algoritmo de planificación relaciona el algoritmo (grafo de tareas) con el hardware disponible (procesador, tiempo)



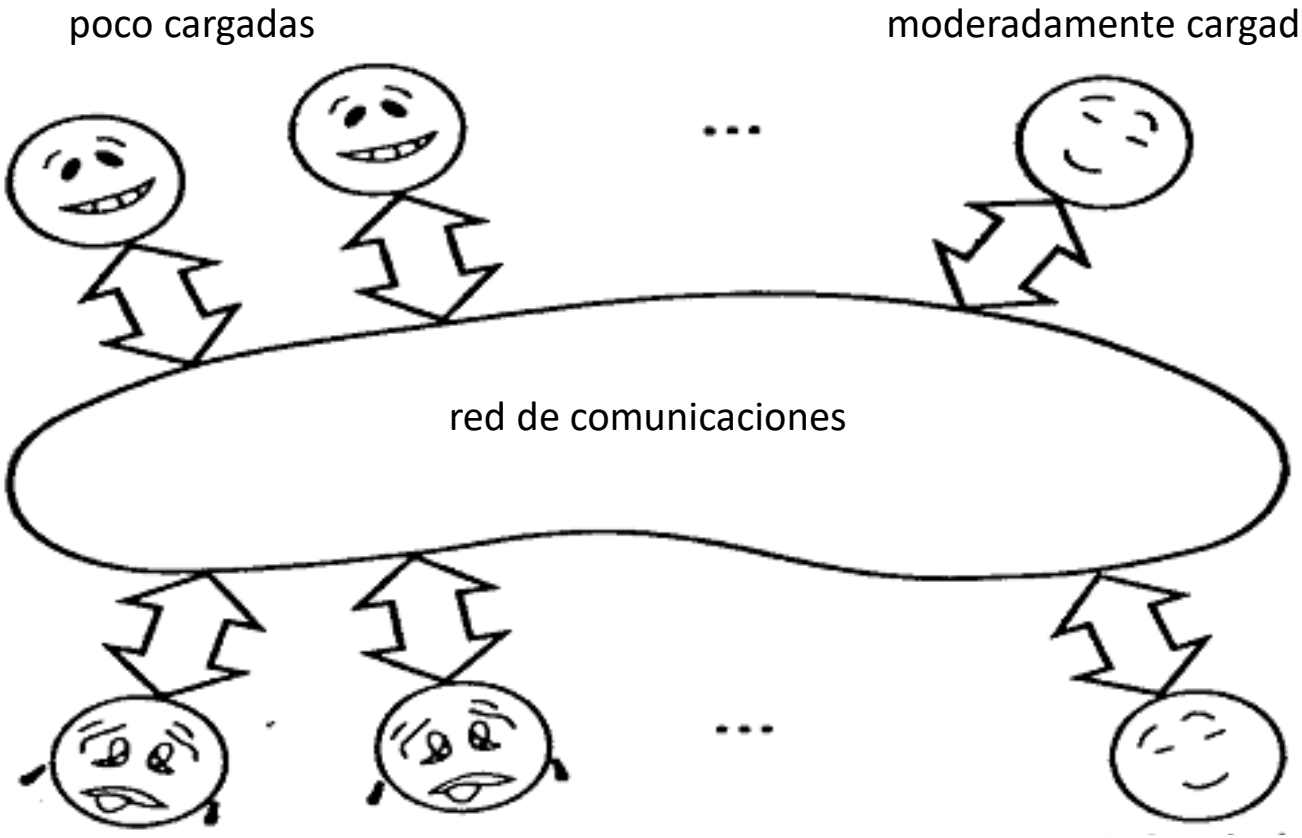
Scheduling

- Debe tomarse en cuenta la topología de la red, conjuntamente con las dependencias de datos y las comunicaciones, de modo de no afectar los costos de comunicaciones
  - Aprovechar la “localidad” de datos
  - Reducir las comunicaciones
- Estrategias
  - Procesos que pueden ejecutar concurrentemente se colocan en procesadores diferentes
  - Procesos que se comunican con alta frecuencia se colocan en el mismo procesador, o en procesadores “cercanos”
- Existen mecanismos teóricos de asignación de recursos para los diferentes modelos de descomposición en tareas y arquitecturas paralelas estudiadas
  - Ejemplo: los grafos de algoritmos (árboles, anillos, mallas) sobre mallas 2D, hipercubos, etc.

Balance de carga

- La distribución de la carga de trabajo es un factor relevante sobre el desempeño de aplicaciones paralelas y distribuidas ejecutando en un computador paralelo
- El objetivo consiste en evitar que la performance global del sistema se degrade a causa de la demora en tareas individuales
- El balance de cargas es muy importante en entornos de cómputo no dedicados

Balance de carga



Situación en un ambiente donde no se aplican técnicas de balance de carga

Técnicas de balance de carga

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- También conocidas como “técnicas de despacho”
- Clasificación:
  - Técnicas estáticas (planificación)
  - Técnicas dinámicas (al momento del despacho)
  - Técnicas adaptativas

- Principales criterios utilizados:
  - Mantener los procesadores ocupados la mayor parte del tiempo
  - Minimizar las comunicaciones entre procesos

Técnicas de despacho estáticas

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Se toman decisiones “tempranas”
- Se utilizan técnicas de planificación de investigación operativa
- Requieren una estimación (precisa) del tiempo de ejecución de cada tarea en cada recurso de cómputo
- La asignación inicial se mantiene, independientemente de lo que suceda
- Efectiva en ambientes de redes poco cargadas
- Falla en ambientes compartidos de carga variable

NO TIENEN EN CUENTA FLUCTUACIONES DE CARGA DE LA RED

Técnicas de despacho dinámicas

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Involucran estrategias para determinar el procesador que se asigna a una tarea durante la ejecución de la aplicación
- Usuales en modelo maestro-esclavo
- La asignación se realiza en el momento de creación de una nueva tarea
- Usualmente consideran la situación en el instante del despacho exclusivamente
- Efectivas en ambientes compartidos de carga variable

TRATAN DE APROVECHAR LAS FLUCTUACIONES DE CARGA DE LA RED

Técnicas de despacho adaptativas

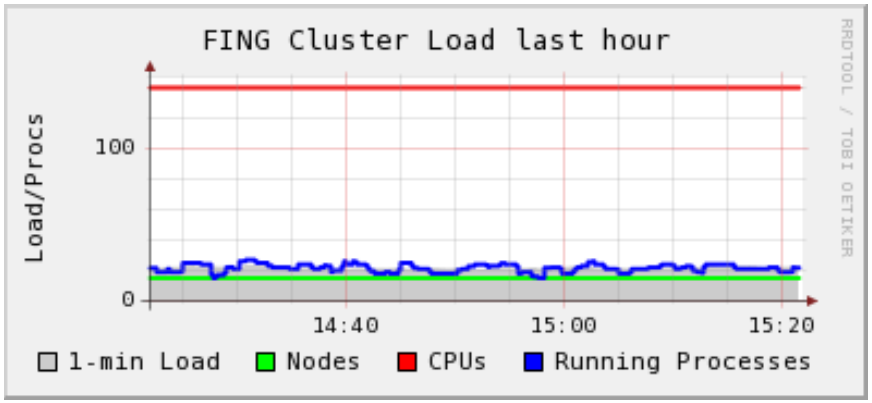
UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Realizan el despacho de acuerdo al estado actual de la red
- Pueden incorporar herramientas de predicción del futuro
- Utilizan técnicas de migración de procesos como mecanismo de eficiencia y para proveer tolerancia a fallos

APROVECHAN COMPLETAMENTE LAS FLUCTUACIONES DE CARGA DE LA RED

Parámetros relevantes en la determinación de la carga

- Consumo de CPU
  - Porcentaje de uso u operaciones/segundo
- Uso de disco
  - Bloques transferidos del controlador al dispositivo
- Tráfico de red
  - Paquetes transmitidos y recibidos



Proyectos desarrollados en FING

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Despacho mejorado de cálculo científico distribuido en redes de computadoras no dedicadas usando PVM
- Proyecto CSIC realizado en el CeCal (1997–1999)
- Utilización de información histórica para predecir la carga de la red en un futuro cercano
- Métodos de predicción:
  - Naive
  - Mínimos Cuadrados
  - Redes Neuronales
- Integración a la biblioteca PVM
- Evaluación de desempeño:
  - Mejora del 6%, correspondiente al 50% de la mejora ideal

Proyectos desarrollados en FING

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Checkpoint y migración de procesos en un ambiente de computación distribuido (CeCal, 1999)
  - Bajar a disco información de procesos (dumping)
  - Levantar proceso en equipo remoto
  - Integración a la biblioteca PVM
  - Buenos resultados de eficiencia y utilización de equipamiento disponible
- Replicación de carga en redes de computadoras (CeCal, 1998)
  - Replicar una serie temporal con información de carga de N estaciones de trabajo conectadas en red
  - Útil para experimentos de evaluación de performance
  - Utilizado en el proyecto “Despacho mejorado de cálculo científico distribuido en redes de computadoras no dedicadas usando PVM”

Proyectos desarrollados en FING

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Algoritmos genéticos incrementales
- Proyecto realizado en el CeCal (2003)
- Capacidad de utilización de entorno no dedicado
- Utilización de información del pasado para predecir la carga de la red en un futuro cercano (mediante mínimos cuadrados)
- Incorporó técnicas de migración de procesos
- Integración a la biblioteca MALLBA (algoritmos evolutivos)
- Evaluación de desempeño
  - Permitió utilizar “anónimamente” el entorno no dedicado del Instituto de Computación.
  - Buenos resultados de eficiencia computacional.

Proyectos desarrollados en FING

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Algoritmos de planificación para entornos de computación heterogénea (CeCal, 2008-2013)
  - Minimización de makespan (tiempo total de ejecución–*system oriented*)
  - Minimización de makespan y flowtime (tiempo de respuesta–*user oriented*)
  - Abordados como problemas de optimización single y multi-objetivo
  - Algoritmos metaheurísticos: computación evolutiva
  - URL: http://www.fing.edu.uy/inco/grupos/cecal/hpc/HCSP
- Planificación considerando eficiencia energética (2013-2015)
  - Multiobjetivo, considerando métricas system y user related
  - Heurísticas y metaheurísticas
  - URL: http://www.fing.edu.uy/inco/grupos/cecal/hpc/MScIturriaga
- Mejoras entre 10-25% en métricas de tiempo, hasta 50% en energía

Proyectos desarrollados en FING

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Algoritmos de planificación para el middleware Ourgrid (CeCal, 2012-2013)
  - Planificación para dar soporte a heterogeneidad
  - Estrategias de planificación greedy
  - Problema de optimización single y multiobjetivo: makespan, energía, confiabilidad
  - Proyecto conjunto con Universidade Federal de Campina Grande (Brasil), Universidad de Buenos Aires (Argentina) y Universidad Veracruzana (México)
- Algoritmos implementados y distribuidos en el código oficial de Ourgrid
- URL: www.fing.edu.uy/inco/grupos/cecal/hpc/GSOS
- El proyecto será presentado en las charlas de invitados al final del curso

Proyectos desarrollados en FING

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Affinity scheduling with hwloc (CeCal, 2013)
  - Algoritmos de planificación considerando afinidad de procesos con el hardware
  - Motivación: utilización eficiente de recursos de hardware: comunicaciones, uso de memoria cache, etc.
  - Desarrollo utilizando la biblioteca MPI
  - Heurísticas y metaheurísticas.
  - Proyecto conjunto con Universidad de San Luis (Argentina) e Universidad de Rennes, INRIA (Francia)
  - Objetivo: integrar los planificadores en middleware de gestión de clusters (TORQUE, Maui)
  - URL: http://runtime.bordeaux.inria.fr/sehloc