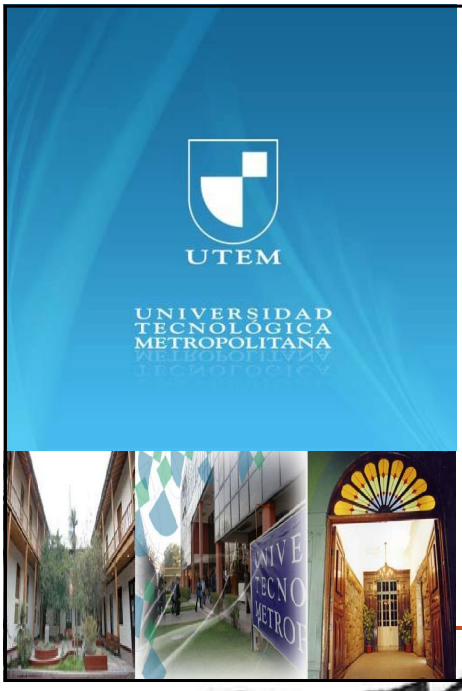


ASIGNATURA
COMPUTACIÓN PARALELA

Departamento de Computación e Informática

Facultad de Ingeniería

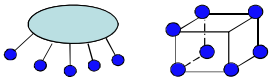
2012-II

Oscar Magna V.
Civil Engineering on Computer Science & MBA
Dr (c) in Business Management and Administration
Technological Metropolitan University
Santiago of Chile
omagna@utem.cl, osemav@gmail.com
http://omagna.tripod.com
(56-2) 787.7211
C H I L E.

UTEM
UNIVERSIDAD TECNOLÓGICA METROPOLITANA
del Estado de Chile

COMPUTACIÓN PARALELA

INF - 5141



INGENIERÍA CIVIL EN COMPUTACIÓN
Mención Informática

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Oscar E. Magna V.
Ingeniero Civil en Informática & MBA
Dr (c) en Administración y Dirección de Empresas
omagna@utem.cl
http:// omagna.tripod.com

2012

Slide-3
Análisis de Rendimiento
Parte 1

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Análisis de Rendimiento

Indices de Performance Básicos

- Speed Up
- Throughput
- Eficiencia
- Costo

Abril 2006

Análisis de Rendimiento

Speed Up

El incremento de velocidad de un algoritmo paralelo cuando se ejecuta Sobre *k* procesadores es:

$$S(n;k) = \frac{\text{Tiempo de ejecución sobre un procesador}}{\text{Tiempo de ejecución en } k \text{ procesadores}}$$

El incremento de velocidad (o *Speedup*) de un algoritmo paralelo cuando se ejecuta sobre *k* procesadores respecto al mejor algoritmo secuencial es:

$$S'(n;k) = \frac{\text{Tiempo de ejecución del secuencial más rapido}}{\text{Tiempo de ejecución del paralelo en } k \text{ procesadores}}$$

Marzo 2007

Análisis de Rendimiento

Throughput

Numero de trabajos por unidad de tiempo que realiza un procesador.

Throughput = f / (IC * CPI)

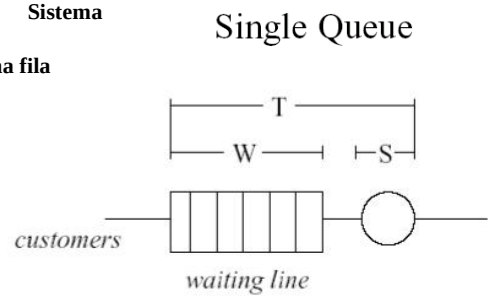
Donde:

f= frecuencia de reloj en Mhz.
IC= cantidad de instrucciones del programa que se esta ejecutando.
CPI= ciclos por instrucion.

Abril 2006

PERFORMANCE METRICS

MODELO elemental de Sistema Computacional.
Un Servidor y una fila



$$T = W + S$$

PERFORMANCE METRICS

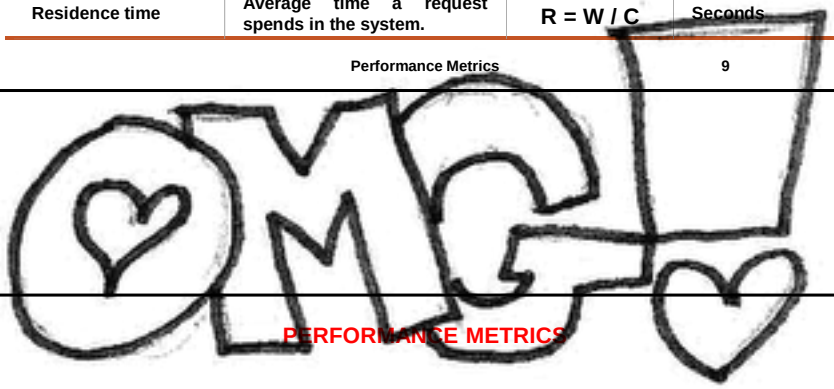
Arrivals ------> RESOURCE ------> Departures

B is the length of time that the resource was observed to be BUSY.

Note that the pictures may be equivalent; a system may have only one resource. A request is presented to a system (in which case it might be called a "job") or is presented to a particular resource.

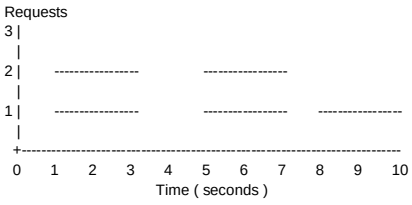
PERFORMANCE METRICS

Metric	Description	Formula	Units
Arrival Rate	How many requests arrive per time.	Y = A / T	Requests/second
Throughput *(Departure Rate)*	How many requests complete per time.	X = C / T	Requests/second
Utilization	Fraction of time the resource is in use.	U = B / T	Number in range 0 - 1.
Service Requirement	Time needed by a resource to handle a request.	S = B / C	Seconds
Requests in system	Average number of requests in the system.	N = W / T	Dimensionless
Residence time	Average time a request spends in the system.	R = W / C	Seconds



PERFORMANCE METRICS

Example:



Note that requests are serviced simultaneously - it's a multiprocessor?

What are:

T	Time interval
A	Arrivals
C	Completions
B	Time spent servicing requests

Y = A / T	Arrival rate
X = C / T	Throughput
U = B / T	Utilization
S = B / C	Service time/request

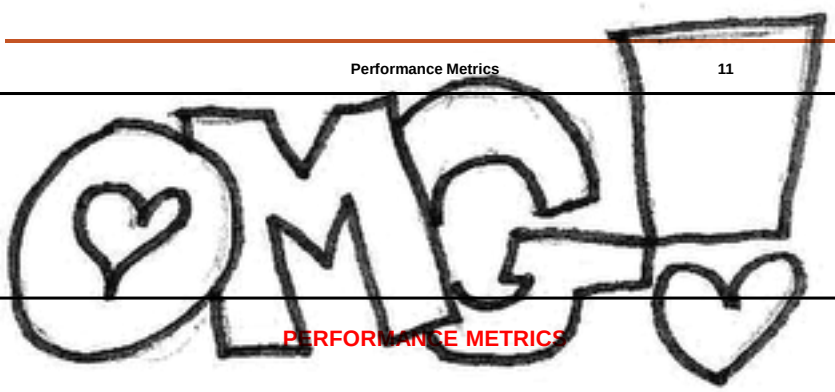
PERFORMANCE METRICS

UTILIZATION LAW U = X * S

Proof: **B / T = C / T * B / C**

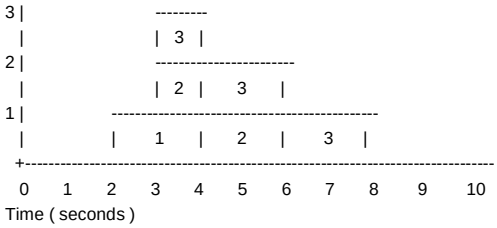
Example:

The server of dinners in the Cafeteria keeps busy 75% of the time between 12:00 and 1:00. During this time 90 people are served a dinner. How long does it take to serve each customer?



PERFORMANCE METRICS

Requests



T	Time interval
A	Arrivals
C	Completions
B	Time spent servicing requests
W	Accumulated time within system

Y = A / T	Arrival rate
X = C / T	Throughput
R = W / C	Residence Time
U = B / T	Utilization
S = B / C	Service time/request
N = W / T	Average number of requests in the system.

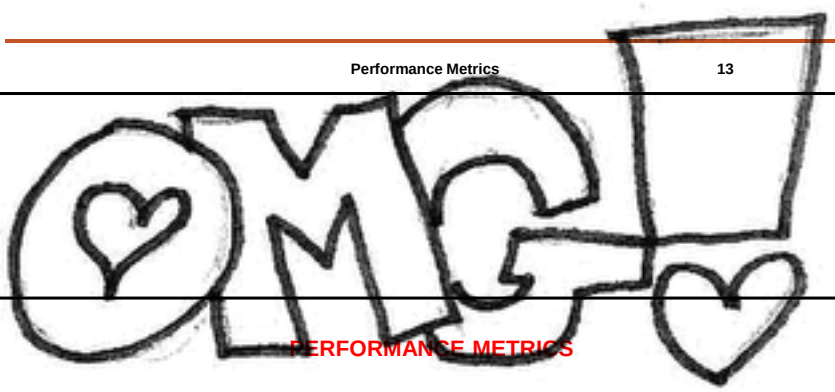
PERFORMANCE METRICS

LITTLE'S LAW N = X * R

Proof: **W / T = C / T * W / C**

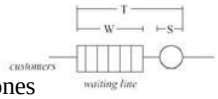
Example:

On an average day in the Cafeteria, four people are waiting in the sandwich line. Each of the two sandwich makers can produce a sandwich in one minute. How long must the diners wait in line until they are handed a sandwich? How many sandwiches can be produced in one hour?



PERFORMANCE METRICS

EJEMPLO : Ley de LITTLE



T = W + S

- Un Servidor NFS (Network File System) fue monitoreado durante 30 min y el número de operaciones I/O desarrolladas durante este período se determinó en 32.400. El número promedio de requerimientos activos (Nrew) fue de 9.
- ¿ Cuál fue el tiempo de respuesta promedio por requerimiento de NFS en el servidor?

Caja negra = Servidor NFS

Xserver = 32.40 / 1.800 = 18 req/seg

Rreq = Nreq / Xserver = 9 / 18 = 0,5 seg.

PERFORMANCE METRICS

Example:

One day I was at the Weston Toll Booth on the Mass Pike. There were 10 collectors working; when I joined the line, there were 8 cars ahead of me, and when my turn came it took me 12 seconds to pay my toll.

What information can be derived from the above paragraph? What information can NOT be deduced?

Example:

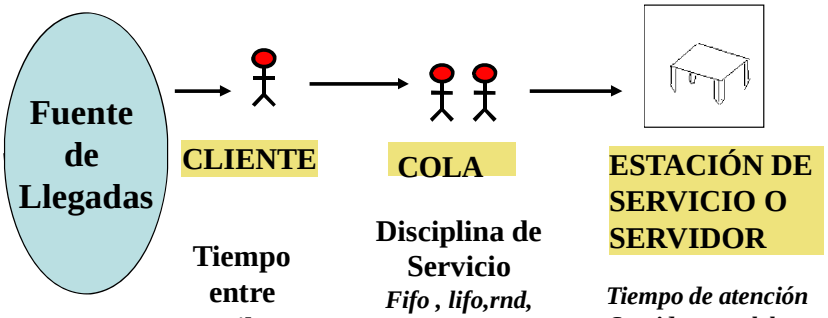
Recently in "Newsweek" it was reported that the current U.S. divorce rate is 4.6 per 1000 adults. It was also reported that half of all adults are currently married.

➔ **Develop a simple model/picture in order to make some sense of these numbers.**

TEORÍA DE FILAS

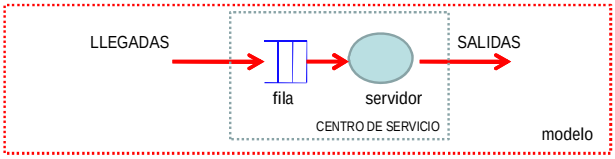
RECORDAR

FACTORES DE UNA FILA



PERFORMANCE METRICS

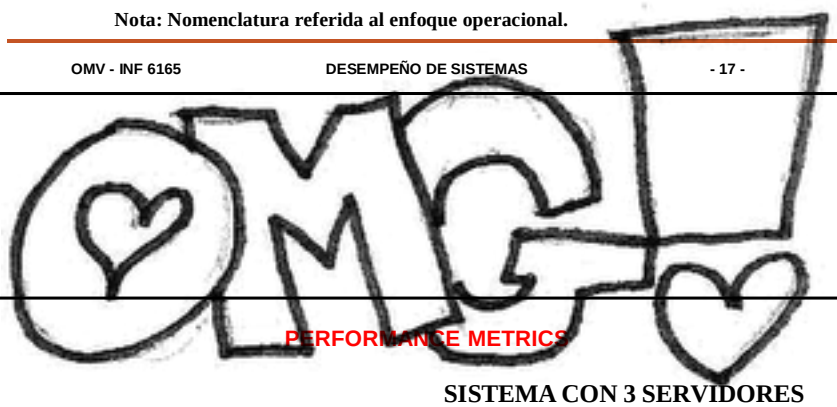
METRICAS DE FILAS – Modelo Operacional. Representación de un modelo con un servidor



- ✓ **X** = throughput (Tasa de salida)
- ✓ **U** = utilización del servidor (fracción del tiempo en que el servidor está ocupado)
- ✓ **R** = Tiempo promedio que un cliente se encuentra esperando en una fila) (tiempo promedio de espera + tiempo promedio de servicio) average wait time + average servicio time)
- ✓ **Q**= Número promedio de clientes en una fila.
- ✓ **S** = Tiempo promedio de servicio por cliente.

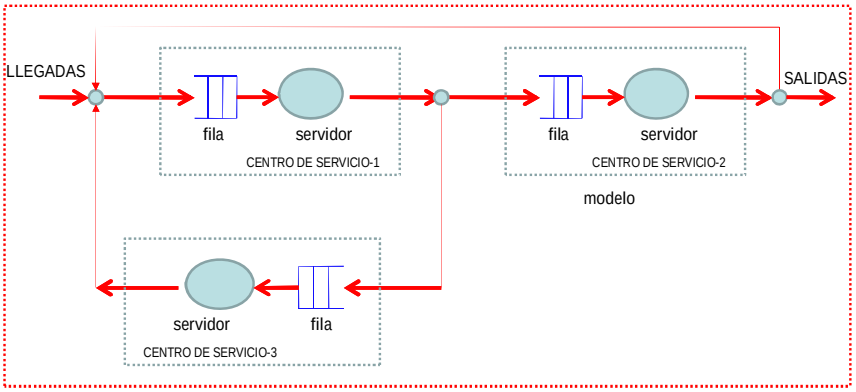
Nota: Nomenclatura referida al enfoque operacional.

OMV - INF 6165 DESEMPEÑO DE SISTEMAS - 17 -



PERFORMANCE METRICS

SISTEMA CON 3 SERVIDORES



- ✓ Los clientes salen de un recurso compartido y pueden visitar un centro de espera u otro recurso compartido.
- ✓ La presión de las tareas no está representada aquí (durante el curso).
- ✓ La selección del camino es impredecible (visión abstracta para el Sw y contenido de los mensajes)

OMV - INF 6165 DESEMPEÑO DE SISTEMAS - 18 -

PERFORMANCE METRICS

SUMMARY OF PERFORMANCE METRICS

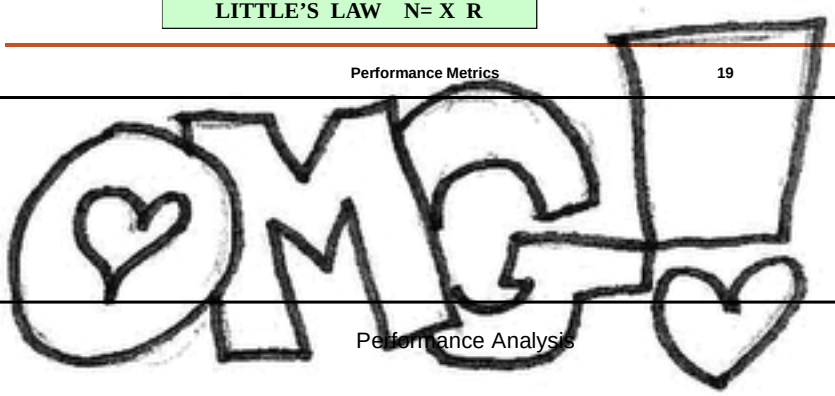
T	is the length of TIME we observed the system.
A	is the number of request ARRIVALS observed.
C	is the number of request DEPARTURES observed.
W	is the ACCUMULATED TIME for all requests within the system - time spent both waiting for and using resources.
B	is the length of time that the resource was observed to be BUSY.

Arrival Rate	Y = A / T
Throughput *(Departure Rate)*	X = C / T
Utilization	U = B / T
Service Requirement	S = B / C
Requests in system	N = W / T
Residence time	R = W / C

UTILIZATION LAW U = X S

LITTLE'S LAW N= X R

Performance Metrics 19



Performance Analysis

Eficiencia

La eficiencia de un algoritmo paralelo, respecto a sí mismo, es:

$$E(n;k)=\frac{S(n;k)}{k}$$

Siendo la eficiencia respecto al mejor algoritmo secuencial:

$$E'(n;k)=\frac{S'(n;k)}{k}$$

Abril 2006

Performance Analysis

Se desprende que:

$$E'(n;k) \leq E(n;k) \leq 1$$

Objetivo:

Algoritmos Óptimos

$$E'(n;k) \in \theta(1)$$

Algoritmos Eficientes

$$E'(n;k) \in \Omega\left(\frac{1}{\log n}\right)$$

Abril 2006

Performance Analysis

Escalabilidad

Para un sistema paralelo interesa que las prestaciones se sigan manteniendo en cierta medida al aumentar el tamaño de sistema.

Un sistema es escalable si es posible aumentar sus recursos si se requiere una mayor demanda de rendimiento o disminuir sus recursos para reducir los costos.

El rendimiento debe aumentar de forma proporcional, pero la eficiencia no tiene por qué mantenerse constante y por tanto el sistema podría no ser escalable.

$$C(n;k) = \frac{T(n;1)}{T(n;k)}k$$

Abril 2006

Performance Analysis

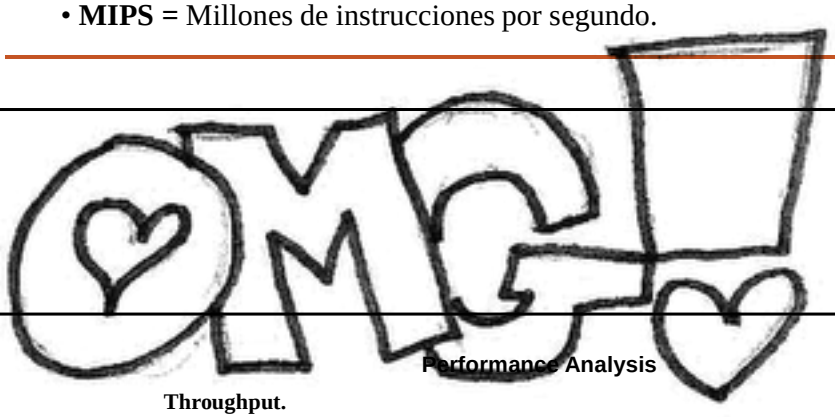
Throughput.

Corresponde a la cantidad de trabajos o programas por unidad de tiempo que un procesador puede ejecutar.

$$\textbf{Throughput} = (\textbf{MIPS} * 10^6) / \textbf{Ic} \quad (1)$$

$$\textbf{Throughput} = \textbf{f} / (\textbf{Ic} * \textbf{CPI}) \quad (2)$$

- **f** = Frecuencia de reloj (MHz).
- **Ic =** Cantidad de instrucciones del programa en ejecución.
- **CPI =** Ciclos por instrucción.
- **MIPS =** Millones de instrucciones por segundo.



Performance Analysis

Throughput.

Ejemplo: Se dispone de 2 CPU para ejecutar un programa de 1000 instrucciones. La CPU 1 tiene una frecuencia de reloj de 600 MHz con CPI = 1,5 . La CPU 2 tiene una frecuencia de reloj de 1200 MHz con CPI = 1. ¿ Que CPU tiene un mejor rendimiento?.

Respuesta: Para responder que CPU tiene un mejor rendimiento, se utilizará la ecuación (2) como sigue.

CPU 1. Troughtput = $\frac{600000000}{1000 * 1,5}$ = 400000

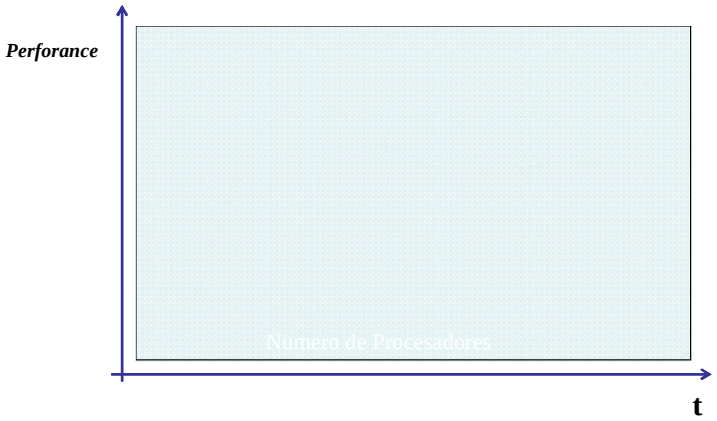
CPU 2. Troughtput = $\frac{1200000000}{1000 * 1}$ = 1200000

El Throughput es mucho mayor en la CPU 2.

Throughput.

Formas de aumentarlo.

- Aumentando la velocidad de reloj (Frecuencia).
- Disminuyendo la cantidad de ciclos por instrucción (CPI).
- Aumentar la concurrencia mediante el paralelismo de instrucciones (Pipelining) para lograr el punto anterior (disminuir los CPI).



Análisis de Costo/Desempeño en Sistemas Paralelos

- ¿Siempre es conveniente una solución paralela?

➔ R: *No, pues aunque se consiga un tiempo de procesamiento menor, el tiempo de comunicación puede ser mayor que éste.*

Factor aceleración, y eficiencia

Factor aceleración (Speed-up):

- T(1): tiempo ejecución de un programa en un procesador
- T(N): tiempo ejecución de un programa en N procesadores

$$S(N) = \frac{t(1)}{t(N)}$$

Factor aceleración, y eficiencia

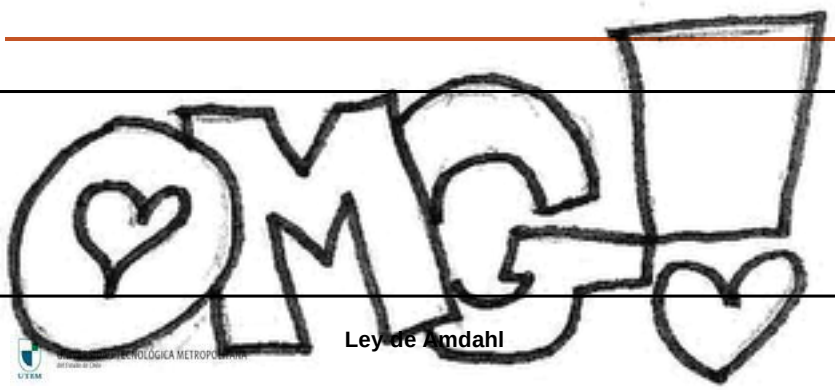
Speed-up ideal:

En condiciones ideales: T(N) = 1/N

$$S(N)_{ideal} = \frac{t(1)}{t(N)} = \frac{1}{\frac{1}{N}} = N$$

Eficiencia: Speed-up v/s Speed-up ideal

$$E(N) = \frac{S(N))}{S(N)_{ideal}} = \frac{S(N)}{N} = \frac{\frac{t(1)}{t(N)}}{N} = \frac{t(1)}{N * t(N)}$$



Ley de Amdahl

Dada una aplicación que se ejecuta en un computador paralelo, cómo afecta al rendimiento (Speed-up) el hecho de aumentar el número de procesadores

Supone un tamaño fijo del problema

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

Ganancia de velocidad S_p

$$S_p = \frac{1}{(1-f)+f/p}$$

Caso Mejora de Paralelización

- f: fracción de código paralelizable
- p: número de procesadores

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

S: parte no paralelizable

P: parte paralelizable

Tiempos de ejecución de un proceso:

$$t(1) = s + p \qquad t(N) = s + \frac{p}{N}$$

Ganancia de velocidad:

$$S(N)max = \frac{s+p}{s+\frac{p}{N}} = \frac{1}{\frac{s}{s+p}+\frac{p}{N(s+p)}}$$

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

f: fracción de tiempo no paralelizable

$$f = \frac{s}{s+p}$$

Ganancia de velocidad:

$$S(N)max = \frac{1}{f+\frac{1-f}{N}} = \frac{N}{Nf+1-f} = \frac{N}{1+(N-1)f}$$

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

Ganancia Máxima (S)

$$S_{lim} = lim_{N \to \infty} \frac{N}{1+(N-1)*f}$$

Debido a esta limitación de la ganancia máxima, la fracción no paralelizable recibe el nombre de **cuello de botella secuencial** de un proceso

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

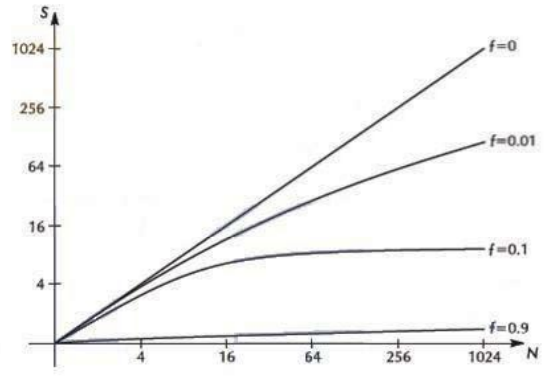
Eficiencia Máxima

$$Emax = \frac{S(N)max}{N} = \frac{1}{1+(N-1)f}$$

límite de eficiencia:

$$Elim = \lim_{N\to\infty} \frac{1}{1+(N-1)f} = 0$$

Ley de Amdahl



- Ganancia: velocidad en función número de procesadores

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

$$S_p = \frac{1}{(1-f)+f/p}$$

Ejemplo 1:

Mejorar el procesador de un servidor web. La nueva CPU es 10 veces mas rápida que la original. Suponiendo que la máquina original está un 40% del tiempo haciendo cálculos, y un 60% con operaciones de E/S, ¿Cuál es la mejora de velocidad que se obtiene?

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

$$S_p = \frac{1}{(1-f)+f/p}$$

Datos: f = 0.4, p = 10

$$S = \frac{1}{(1-0,4)+\frac{0,4}{10}} = 1,56$$

S ≡ Speed-Up ≡ mejora de velocidad ≡ Ganancia Máxima

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

$$S_p = \frac{1}{(1-f)+f/p}$$

Ejemplo 2:

- Suponer que la función FPSQR *(raíz cuadrada en coma flotante)* supone un 20% del tiempo de ejecución en un programa gráfico.
- Una propuesta para mejorar el hardware sobre el que se ejecuta es mejorar esta operación en un factor de 10.
- Otra posibilidad es mejorar todas las operaciones de coma flotante (FP suponen el 50% del total del tiempo de ejecución) haciéndolas un 1.6 más rápidas.

➔ **¿Qué opción es mejor?**

S ≡ Speed-Up ≡ mejora de velocidad ≡ Ganancia Máxima

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

$$S_p = \frac{1}{(1-f)+f/p}$$

Datos:

A) f = 0.2, p = 10

$$S = \frac{1}{(1-0{,}2)+\frac{0{,}2}{10}} = 1{,}22$$

B) f = 0.5, p = 1.6

$$S = \frac{1}{(1-0{,}5)+\frac{0{,}5}{1{,}6}} = 1{,}23$$

S ≡ Speed-Up ≡ mejora de velocidad ≡ Ganancia Máxima

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

Ejemplo 3:

$$S_{lim} = lim_{N \to \infty} \frac{N}{1+(N-1)*f}$$

Rendimiento de un sistema con 16 procesadores cuando sobre él se ejecuta un proceso con un 25% no paralelizable.

$$S(16)max = \frac{N}{1+(N-1)f} = \frac{16}{1+15*0,25} = 3,37$$

$$E(16)max = \frac{1}{1+(N-1)f} = \frac{1}{1+15*0,25} = 0,21$$

S ≡ Speed-Up ≡ mejora de velocidad ≡ Ganancia Máxima

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Amdahl

La parte no paralelizable provocó una disminución en el rendimiento del sistema.

Por lo tanto, el rendimiento no aumenta por aumentar indefinidamente el número de procesadores.

Ley de Gustafson

Referida al crecimiento del volumen de cálculo necesario para resolver un problema. Basándose en que el problema crece considerablemente en su parte paralela y no en la secuencial.

Por lo tanto, el cuello de botella secuencial tiende a cero cuando el volumen del problema aumenta.

Ley de Gustafson

Suponiendo que el número de procesadores crece indefinidamente de la misma forma que las dimensiones del sistema:

$$\lim_{N\to\infty} f = \lim_{N\to\infty} \frac{s}{s+Np} = 0$$

Siendo

- s: parte secuencial
- p: parte paralela

$$S_{lim} = lim_{N \to \infty} \frac{N}{1+(N-1)*f}$$

S ≡ Speed-Up ≡ mejora de velocidad ≡ Ganancia Máxima

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Gustafson

Para calcular la ganancia de velocidad, supondremos que el tiempo que se tardaría en un solo procesador es:

$$t(1) = s + Np$$

Y en un sistema paralelo:

$$t(N) = s + p$$

Por lo tanto la ganancia de velocidad:

$$S = \frac{t(1)}{t(N)} = \frac{s+Np}{s+p}$$

•s: parte secuencial
•p: parte paralela

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Gustafson

Considerando la ecuación

$$f = \frac{s}{s+p}$$

La nueva ecuación formada es:

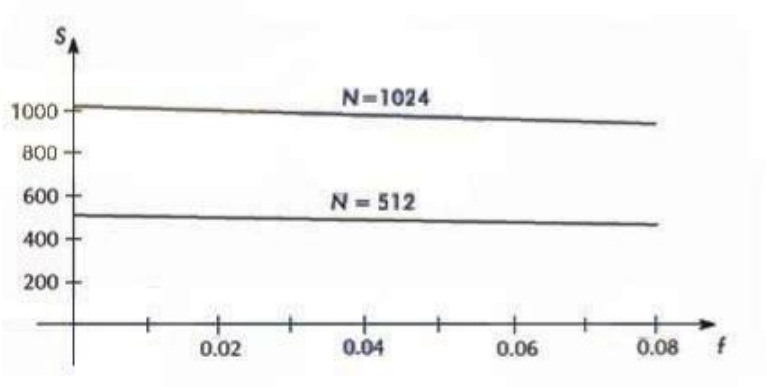
$$S = f + N * (1 - f) = N - (N-1) * f$$

•s: parte secuencial
•p: parte paralela

S ≡ Speed-Up ≡ mejora de velocidad ≡ Ganancia Máxima

UNIVERSIDAD TECNOLÓGICA METROPOLITANA

Ley de Gustafson



Aplicada para 512 y 1024 procesadores

Factores que afectan el rendimiento

GRANULARIDAD
Cantidad de trabajo que realiza cada Nodo

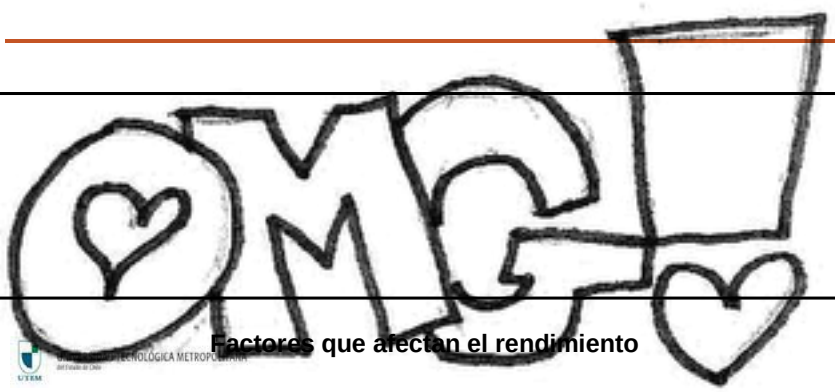
Aumentar la granularidad:

- Disminuye overhead de control y comunicaciones.
- Disminuye el grado de paralelismo.

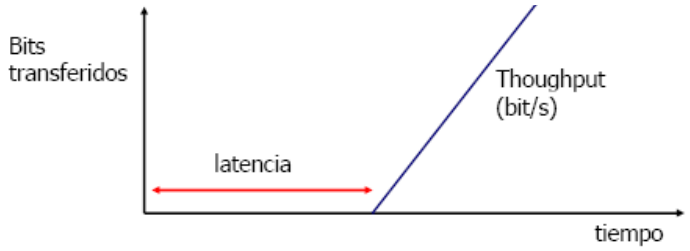
Factores que afectan el rendimiento

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.).



Factores que afectan el rendimiento



EJERCICIOS

DESEMPEÑO

Problema

Considere una aplicación de procesamiento de imágenes que consiste de:

- Doce procesos: un proceso de entrada.
- un proceso de salida.
- diez procesos de cómputo.

La tarea de entrada tiene que completarse antes que inicien las tareas de procesamiento. Las tareas de procesamiento tienen que completarse antes de que inicie la tarea de salida.
Cada una de las 12 tareas toma una unidad de tiempo.
Existe un solo dispositivo de entrada y un solo de salida.

1. ¿Cuál es la máxima aceleración que se puede lograr con este problema cuando se utilizan dos procesadores?
2. ¿Cuál es una cota superior a la aceleración que se logra con esta aplicación cuando se utiliza un computador paralelo?
3. ¿Cuál es el menor número de procesadores para lograr la aceleración dada en b)?
4. ¿Cuál es la máxima aceleración que se puede lograr resolviendo 5 instancias de este problema en dos procesadores?

DESEMPEÑO

Sol: Prob.

$$\text{Speedup} = \frac{t(n)}{t_s(n) + \frac{t_p(n)}{p}}$$

$$t(n) = 12$$

$$t_s(n) = 2$$

$$t_p(n) = \frac{10}{p}$$

DESEMPEÑO

Sol: Prob. Nº1

a. $\text{Speedup} = \frac{12}{2 + \frac{10}{2}} = 1.71$

b. $\text{Speedup} = \frac{12}{2 + \frac{10}{10}} = \frac{12}{3} = 4$

c. 10 procesadores

d. $t(n) = 12 \qquad t_s(n) = 2 \qquad t_p(n) = \frac{5}{2} = 3$

$$\text{Speedup} = \frac{12}{2+3} = \frac{12}{5} = 2{,}4$$

DESEMPEÑO

Problema

Se dispone de un equipo A que posee una CPU con una frecuencia de reloj de 600 Mhz con un CPI = 1,5 *(ciclos por instrucción)* y además se cuenta con un computador paralelo B capaz de operar a 1200 Mhz con un CPI = 1. Ambos computadores operan sobre un mismo programa, el cual cuenta con 1000 instrucciones.

a) Se desea evaluar cual de los 2 computadores convendría adquirir, basándose en su rendimiento (Throughput), ¿cuál sería su recomendación y porqué?
b) ¿Cuánto debería aumentar la frecuencia de reloj del computador con menos rendimiento calculado anteriormente, para igualar al otro. Explique además, que técnica utilizaría para llevar a cabo tal efecto.

$$\text{Throughput} = W_p = \frac{f}{(I_c \cdot CPI)}$$

DESEMPEÑO

Sol. Prob.

El Throughput se define de la siguiente manera: $\text{Throughput} = W_p = \frac{f}{(I_c \cdot CPI)}$

Donde:
f:Drecuencia del reloj.
I_c: Cantidad de Instrucciones del programa.
CPI: Ciclos por instrucción.

a.

Para el computador A:

$$W_{p(A)} = \frac{600}{(1000 \cdot 1{,}5)}[MHz]$$

$$W_{p(A)} = 0{,}4 \cdot 10^{-6}\left[\frac{programas}{segundos}\right]$$

Para el computador B:

$$W_{p(B)} = \frac{1200}{(1000 \cdot 1{,}0)}[MHz]$$

$$W_{p(B)} = 1{,}2 \cdot 10^{-6}\left[\frac{pro\ gra\ mas}{segundos}\right]$$

El computador B tiene un rendimiento 3 veces mayor que el A. Por lo tanto, la recomendación seria comprar el computador B ya que ejecuta más programas en el mismo tiempo.

DESEMPEÑO

Sol: Prob.

El Throughput se define de la siguiente manera: $\text{Throughput} = W_p = \frac{f}{(I_c \cdot CPI)}$

Donde:
f:Drecuencia del reloj.
I_c: Cantidad de Instrucciones del programa.
CPI: Ciclos por instrucción.

b.

$$W_{p(A)} = W_{p(B)}$$

$$\frac{X}{(1000 \cdot 1{,}5)} = \frac{1200}{(1000 \cdot 1{,}0)} [MHz]$$

$$X = 1800 [MHZ]$$

Por lo tanto, se debe triplicar la frecuencia de reloj del computador A.

Para lograr igualar los rendimientos de ambos computadores, se puede implementar un cluster con el computador A.

DESEMPEÑO

Problema Nº

Considere el algoritmo para obtener la matriz traspuesta de una matriz A de n x n en un multiprocesador UMA con "p" procesadores.

```
"Transpuesta"
Global n //Número de elementos
a[n][n] //matriz
p   //Número de procesadores
Local i,j, temp
Begin
    For all Pm Where 1 ≤ m ≤ P do
                    for i ←m to n step p do
                                for j ←i to n do
                                        temp ←a[i][j]
                                        a[i][j] ←a[j][i]
                                        a[j][i] ←temp
                                endfor
                    endfor
    endfor
End.
```

a) ¿Cuál es la granularidad?
b) ¿Cuál es el Costo?
c) Discuta el balance de carga

DESEMPEÑO

Sol: Prob. Nº

a. $\Theta\left(\frac{n^2}{2p}\right)$

b. Costo: $\Theta\left(\frac{n^2}{p}+p\right)$

El costo de sincronización es $\Theta(p)$ ya que se cuenta con p procesadores.

c. El balance de carga resulta ser desproporcionado. El procesador p trabaja la mitad que el procesador 1.

DESEMPEÑO

Problema Nº

La siguiente tabla registra los tiempos de ejecución del mejor algoritmo secuencial y algoritmo paralelo que resuelve el mismo problema. Estos algoritmos son ejecutados en un Multicomputador. Limitaciones de memoria restringen la ejecución de problemas grandes en 1 o 2 procesadores.

	Tiempo de Ejecución (mseg)				
Tamaño del		Algoritmo Paralelo			
problema	Mejor algoritmo secuencial	1 CPU	2 CPU	4 CPU	**8 CPU**
100	24	36	20	12	**10**
200	96	144	78	46	**30**
400	384	576	290	156	**80**
600	---	---	1000	512	**260**
1600	---		---	**1990**	**1024**

a) ¿Cuál es el Speedup del algoritmo paralelo que resuelve el problema de tamaño 100 en 4 procesadores?

b) ¿Cuál es la paralelizabilidad del algoritmo paralelo que resuelve el problema de tamaño 200 en 2 procesadores?

c) ¿Cuál es el Speedup escalado del algoritmo paralelo que resuelve el problema de tamaño 1600 con 8 procesadores?

DESEMPEÑO

Sol: Prob.

a.

b.

c.

DESEMPEÑO

Problema Nº

Supóngase que se tiene una instrucción con 5 etapas A, B, C, D y E (que pueden ser de lectura, escritura, carga, etc.), y que debemos ejecutar esta misma instrucción para 3 datos. Además las instrucciones se demoran 50, 50, 60, 50 y 50 ns cada una respectivamente.

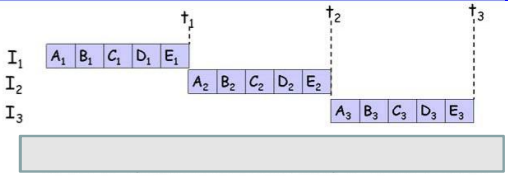
¿Qué posible solución se tendría para ejecutarlas y cuáles serían las conclusiones de cada solución?

a) **Secuencial:** Una instrucción tras otras en un único procesador.
b) **Paralelismo de datos.** "Uso de múltiples unidades funcionales para aplicar la misma operación simultáneamente a elementos de un conjunto de datos."
c) **Pipeling:** "Es una técnica de diseño que permite ejecutar, en una única CPU, más de una instrucción (no completa) al mismo tiempo, donde cada una de ellas está ejecutándose en una etapa distinta."

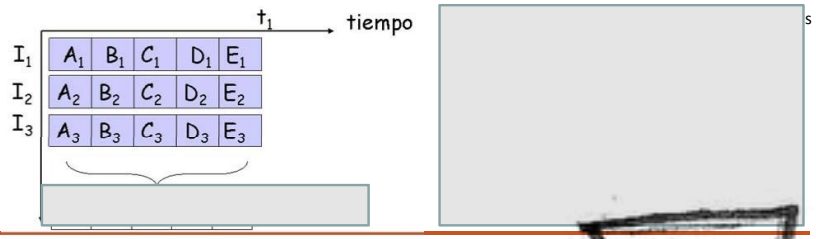
DESEMPEÑO

Sol: Prob.

a. Secuencial



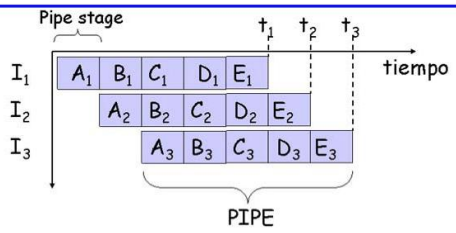
b. Paralelismo de Datos (implica una capacidad de tres procesadores)

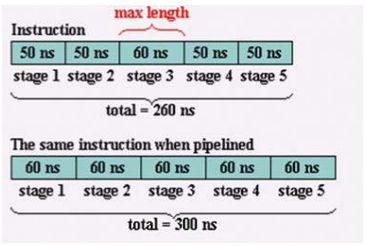


DESEMPEÑO

Sol: Prob.

b. Pipeling





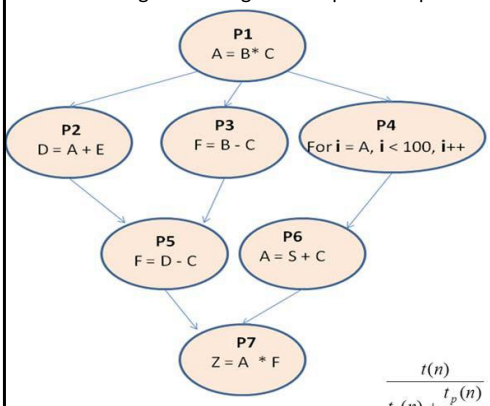
Conclusiones:

- El pipelining no disminuye el tiempo de ejecución de una instrucción.
- Se utiliza un solo procesador que ejecuta los distintos pasos de las instrucciones.
- El tiempo total de procesamiento de un conjunto de instrucciones es mayor que utilizando paralelismo de datos pero menor que en forma secuencial.
- Pueden existir problemas al tener dependencia de datos entre una instrucción y otra

DESEMPEÑO

Problema Nº

Sea el siguiente diagrama de procesos paralelos



Eficiencia = Speedup / p

$$\frac{t(n)}{t_s(n) + \frac{t_p(n)}{p}}$$

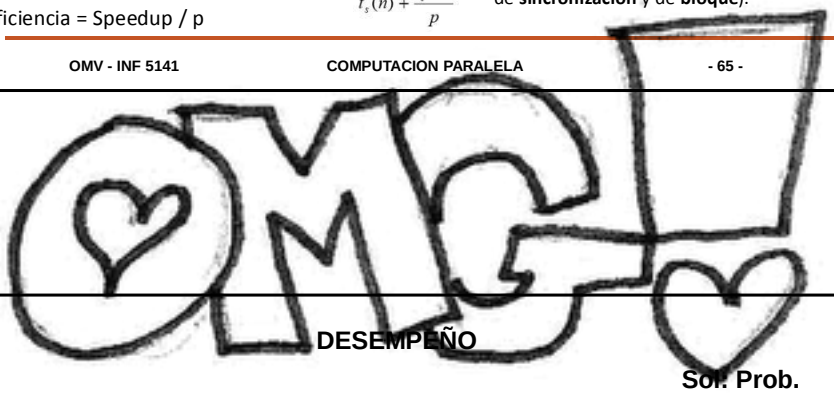
a) Escriba el programa maestro que "lanza" los procesos anteriores. Utilice las seudo instrucciones **Esperar_proceso (P_i)** para sincronizar el Proceso Pi, y el bloque:

En_paralelo
P_1
:
:
P_n
Fin_paralelo;

que lanza **n** procesos a ejecutarse en paralelo.

b) Escriba el programa secuencial.

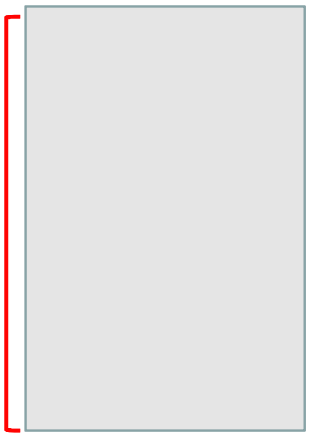
c) Suponiendo que cada instrucción demora 1 unidad de tiempo. Calcular Speedup y Eficiencia (considere que en la instrucción **for** la variable **i** toma el valor 30 y obvie el tiempo de comparación de la instrucción **for,** de **sincronización** y de **bloque**).



DESEMPEÑO

Sol. Prob.

a.



b.

DESEMPEÑO

Sol: Prob.

c.

DESEMPEÑO

Problema Nº

1. Cuales son los factores que limitan el Speedup.
2. ¿Qué significa el indicador "**Eficiencia**"?
3. ¿Qué significa que al calcular la eficiencia nos de le valor 1?

A)

- Entrada/Salida (el % es alto respecto de la computación)
- Algoritmo
- Tamaño del problema (puede ser pequeño, o fijo y no crecer con **p**)
- Desbalance de carga (produciendo esperas ociosas en algunos procesadores)
- Alto porcentaje de código secuencial

B) Mide la fracción de tiempo en que los procesadores son útiles para el cómputo.

C) Cuando es 1 corresponde al Speedup perfecto, es decir hay una máxima efectividad del uso de los procesadores.

DESEMPEÑO

Probl.

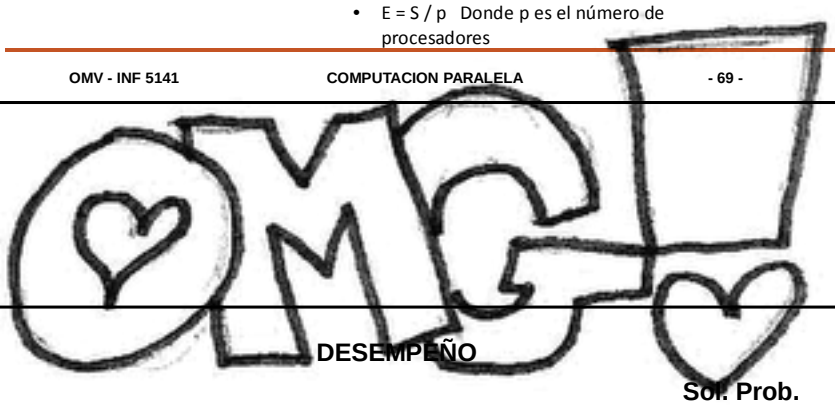
- Dado el algoritmo secuencial:

  para y <=1 hasta 100 hacer A[y]<=1

  Operaciones: Asigno a **y** el valor 1
  Comparo **y** con 100
  Asigno a **A[y]** el valor 1

- Si se implementa este algoritmo en paralelo con 100 procesadores, determine:.
  - Número de operaciones realizadas.
  - Factor de aceleración (S).
  - Costo (C).
  - Eficiencia (E).

a) Factor de aceleración (S): Speedup =
b) Costo (C): Costo = Tiempo que demora en ejecutar el algoritmo paralelo * p
d) Eficiencia (E):
- E = S / p Donde p es el número de procesadores



DESEMPEÑO

Sol. Prob.

Operaciones:

Si se implementa este algoritmo en paralelo con 100 procesadores,:

a)

a)
c)

d)

DESEMPEÑO

Problema Nº

Dada una lista de n números x0,…,xn-1
Se desea implementar la operación de prefijo que calcula todas las sumas parciales repartiendo los resultados en n procesos:

RESULTADO	PROCESADOR
x_0	P_0
x_1+x_0	P_1
.....	.....
$x_{n-1}+\ldots x_1+x_0$	P_{n-1}

a) Derivar una solución paralela a este problema siguiendo una estructura de hipercubo (suponiendo dimensión n).
b) Evaluar el tiempo de computación y comunicación del algoritmo

DESEMPEÑO

Problema Nº

1. Describir gráficamente cómo se podría implementar una operación colectiva **MPI_gather** siendo raíz el proceso 0 en un hipercubo de dimensión 2 (con 4 procesadores). Obtener una fórmula de tiempo de ejecución para esta operación en función del tamaño del bloque que aporta cada procesador N y los parámetros de comunicación $\mathbf{t_s}$ **(latencia)** y $\mathbf{t_w}$ **(ancho de banda)** de la arquitectura.
2. Intentar extender la fórmula para un hipercubo de cualquier dimensión.

DESEMPEÑO

Prob. Nº

Suponga la siguiente solución eficiente para realizar una operación de reducción global con 2n tareas (suma, máximo, producto, etc.) para los dos casos siguientes:

a) que la solución debe obtenerse sólo en una de las tareas,
b) que la solución debe obtenerse en todas las tareas.

Describir la estructura de comunicación así como las operaciones que realiza cada tarea, y evaluar el tiempo de comunicación y computación que requiere el algoritmo

DESEMPEÑO

Prob. Nº

Estructura de comunicación: Hipercubo de dimensión n: $P=2^n$
Suponemos que cada tarea mantiene un bloque de elementos B[1:N/P].
- $N_k(x)$=Vecino del procesador x en la dimensión k del hipercubo.
- ⊗ = operación a aplicar (suma, producto, mínimo, etc.).
- Digit(k,x)=valor del dígito k-ésimo en repres. binaria del nº x.

a) Proceso x, 0 <= x < P
```
k = n-1;
centinela=true;
While (k>0 and centinela) {
If (digit(k,x)==1)  {send(B, Nk(x));
                     centinela=false;}
Else{ receive (B2, Nk(x));
      for i=1,N/P
          {B[i] = B[i] ⊗ B2[i];}}
k=k-1;}
```

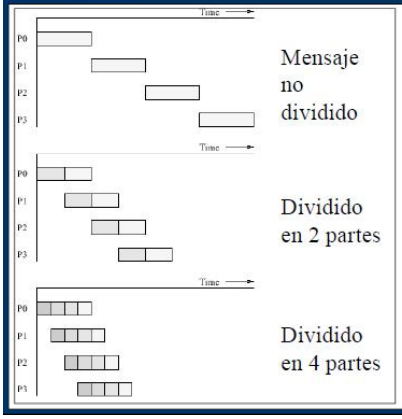
b) Proceso x, 0 <= x < P
```
For k = n-1 downto 0 {
   send(B, Nk(x))
   receive (B2, Nk(x));
   for i=1,N/P
         {B[i] = B[i] ⊗ B2[i];}
}
```

DESEMPEÑO

Costo de Comunicación en Sistemas Paralelos

- Paso de mensajes. El coste de comunicación de una operación de transferencia depende de:
  - Tiempo de inicio t_s: Añadir cabecera, corrección de errores, ejecución del algoritmo de enrutamiento, conexión entre fuente y destino.
  - Tiempo de salto t_h: Tiempo de desplazamiento entre dos nodos conectados directamente.
  - Tiempo de transferencia de palabra t_w: Inverso del ancho del canal de comunicación.

Store-and-forward y Cut-through



Mensaje no dividido

$t_{com} = t_s + (m \cdot t_w + t_h) \cdot l$

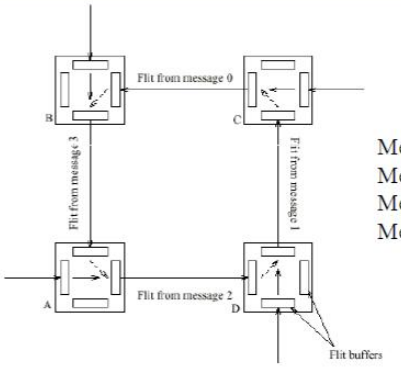
$t_{com} = t_s + m \cdot l \cdot t_w$

Dividido en 2 partes

Dividido en 4 partes

$t_{com} = t_s + l \cdot t_h + t_w \cdot m$

Enrutamiento *Cut-through*: Interbloqueos



Mensaje 0 → Nodo A
Mensaje 1 → Nodo B
Mensaje 2 → Nodo C
Mensaje 3 → Nodo D

Modelo de Costo de Comunicación

- Coste del envío de un mensaje de tamaño m:

$$t_{com} = t_s + t_w \cdot m$$

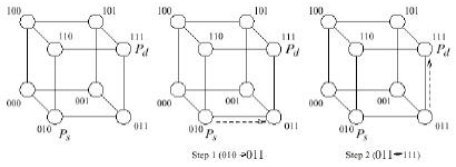
- t_s es mucho más grande que t_h, y en la mayoría de los casos, $t_w \cdot m$ es más grande que $t_h \cdot l$.

Mecanismos de enrutamiento

- Enrutamiento:
  - Algoritmo para determinar el camino que un mensaje tomará desde la fuente hasta el destino.
- Varias clasificaciones:
  - Mínimo vs. No-mínimo.
  - Determinista vs. Adaptativo.

Enrutamiento ordenado por dimensión

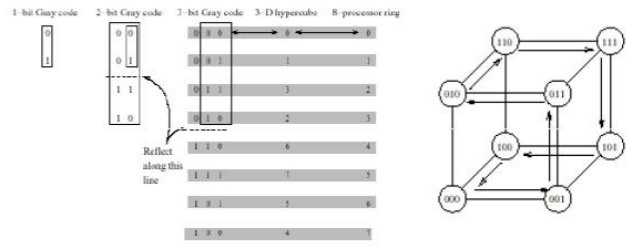
- Orden predefinido de las dimensiones.
- Los mensajes se encaminan por cada dimensión, en el orden establecido, hasta que no es posible continuar:
  - X-Y para mallas
  - E-cubo para hipercubos



Transformaciones en la Topologías

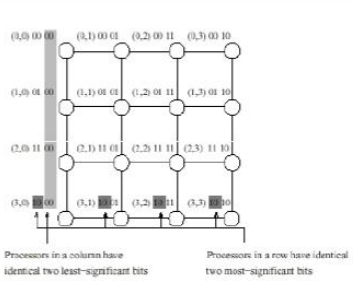
- Mapeo entre redes:
  - Util en los comienzos de la computación paralela, cuando los algoritmos dependían de las topologías.
- Métricas de calidad de las transformaciones:
  - Congestión: Máximo número de enlaces de la topología inicial mapeados en un único enlace de la topología final.
  - Dilatación: Máximo número de enlaces de la topología final, sobre los que se mapea un único enlace de la topología inicial.
  - Expansión: Relación entre el número de nodos de ambas topologías.

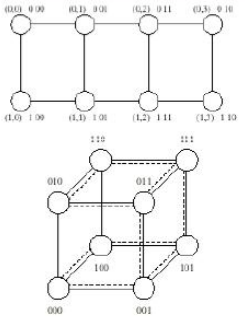
Anillo a Hipercubo



- Los nodos del anillo se mapean al hipercubo siguiendo el código Gray reflejado.
- La dilatación y congestión es 1.

Malla 2-D a Hipercubo

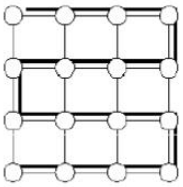




Malla 4x4 a Hipercubo 4-D

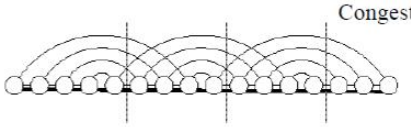
Malla 2x4 a Hipercubo 3-D

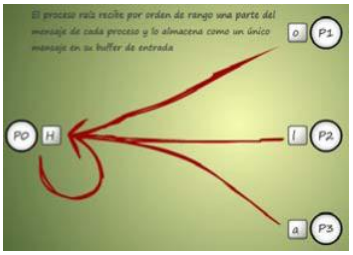
Array lineal a Malla 2-D



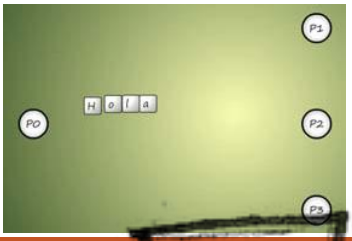
Array lineal a Malla 2-D
Congestión: 1

Malla 2-D a Array lineal
Congestión: 5 ($\sqrt{p}+1$)






```
#include <mpi.h> void MPI::Comm::Gather(const
        void* sendbuf, int sendcount, const
        MPI::Datatype& sendtype, void*
        recvbuf, int recvcount, const
        MPI::Datatype& recvtype, int root,
        const = 0
```



1. Recoge una serie de datos de varios procesos en un único proceso raíz (operación en la cual interviene también el propio proceso raíz).

Parámetros de Entrada	**sendbuf**	Dirección inicial del buffer de envío.
	sendcount	Número de elementos que va a enviar cada proceso individualmente, en general, el número de elementos del buffer de envío (int).
	sendtype	Tipo de dato de cada elemento del buffer de envío.
	recvcount	Número de elementos que se espera recibir de cada uno de los procesos (solo el proceso raíz tendrá este parámetro en cuenta) (int). Nótese que este valor es igual que sendcount siempre y cuando los tipos de envío y recepción sean los mismos.
	recvtype	Tipo de dato que se espera recibir en el buffer de entrada (útil únicamente para el proceso raíz).
	root	Rango del proceso raíz (el proceso receptor) (int).
	comm	Comunicador por el que se realiza la transferencia de datos