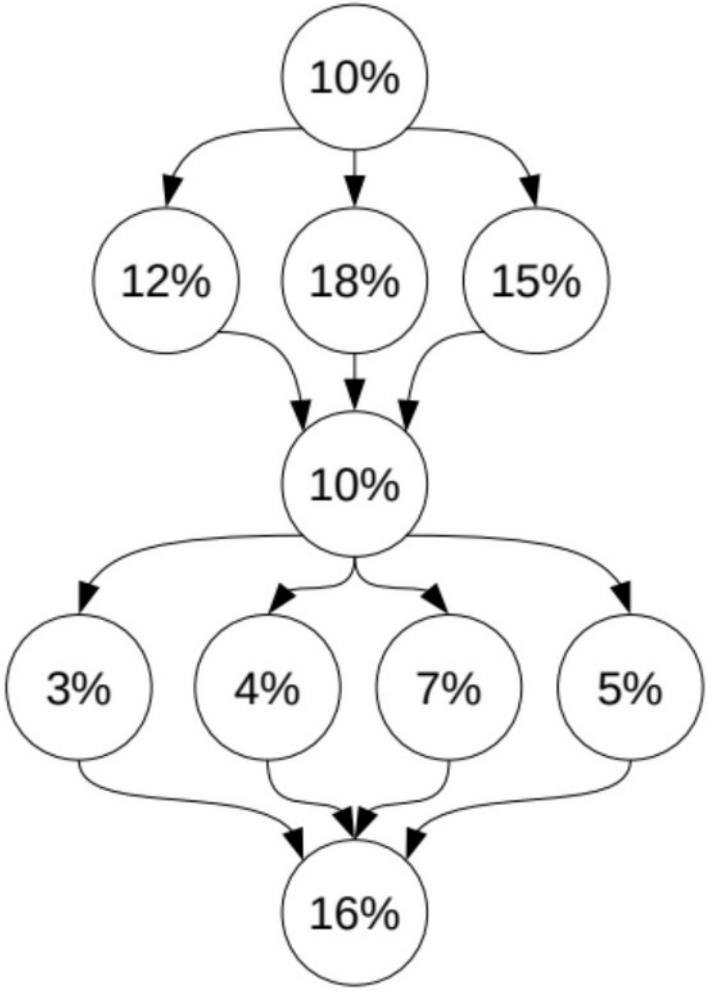
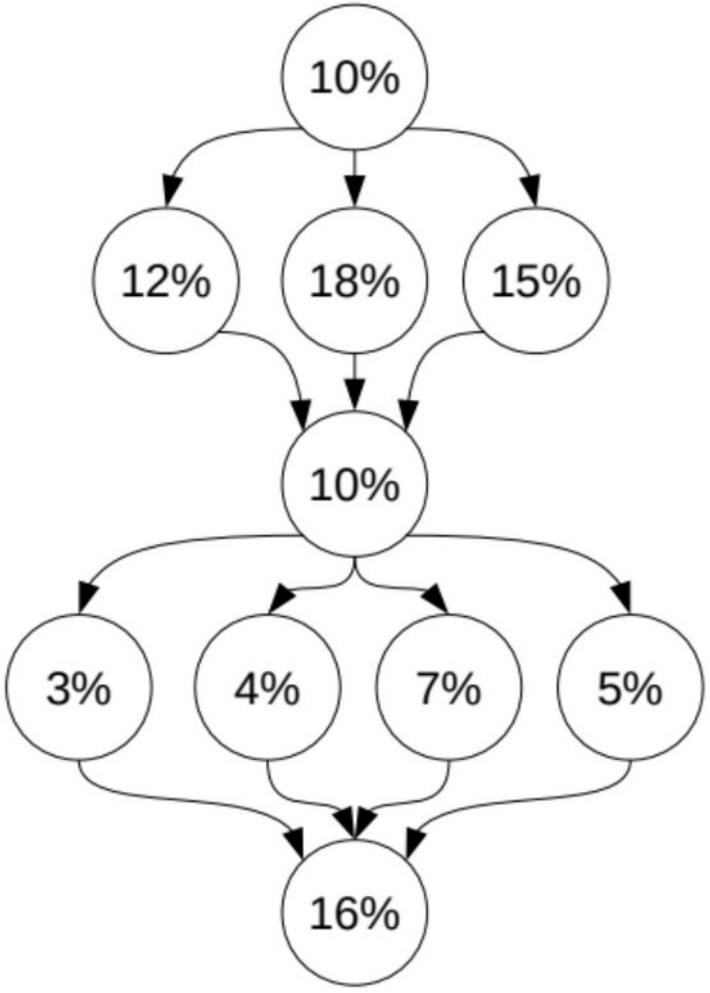


El gráfico muestra el grafo de dependencias entre tareas de una aplicación paralelizada. Suponga $t_{overhead}(p)$ despreciable. Se pide:

a) $S_p(p)$ para p= 2 y 3

b) Tenemos 3 máquinas con n=2,3 y 4, ¿cuál es la más eficiente?

Ojo: Es un problema de asignación de tareas a hebras/procesos.



The graph shows the network of dependencies between tasks of a parallelized application. Suppose $t_{overhead}(p)$ is negligible. Calculate:

a) Sp(p) for p= 2 & 3

b) If we have 3 parallel machines with n=2, 3 and 4, which is the most efficient?

Note: This is a problem of task (threads/processes) assignment.

El **25%** de un programa no se puede paralelizar. **El resto sí**, es decir, podemos distribuir la carga entre un número cualquiera de elementos de procesamiento (*threads*, procesos, cores, nodos, ...).
[English] If only 75% of a program can be parallelized, so we can distribute the load between any number of processing elements (threads, processes, cores, nodes, …), calculate:

1) Calcule la ganancia paralela en velocidad para p unidades de procesamiento. Calculate the speed-up for p processing elements
2) ¿Qué pasa cuando p tiende a + infinito? What if p tends to +infinity?
3) Calcule la eficiencia paralela para el caso 1) y 2). Revisit 1) & 2) calculating the parallel eficiency.
4) ¿A partir de cuántas unidades de ejecución la ganancia paralela de velocidad es superiror a 2? Value of p, from which, the speed-up are >2

Un programa totalmente paralelizable tarda 20s en ejecutarse en un procesador **P1** y 30s en otro procesador **P2**. [English] A fully parallelizable program takes 20s to run on a given processor P1 and 30s on another processor P2.

Despreciando la sobrecarga calcule: Neglecting the overload, calculate:

a) Tiempo de ejecución paralela si distribuimos equitativamente la carga entre P1 y P2. Ganancia paralela en velocidad y eficiencia paralela. Parallel execution time if we distribute the load equally between P1 and P2. Parallel speed-up and efficiency.

b) Distribución de carga que optimiza la ejecución paralela. Calcule en este caso el tiempo de ejecución paralela, la ganancia paralela en velocidad y la eficiencia paralela. Load distribution that optimizes parallel execution. Calculate in this case the parallel time, parallel speed-up and efficiency

Se ha paralelizado una aplicación C++ para aprovechar el paralelismo de un multicomputador de **N** nodos. Como resultado del proceso de paralelización, se ha resuelto que:

- un **30%** de la aplicación sólo se puede ejecutar en un nodo.
- otro **20%** se puede ejecutar en cualquier número de nodos.
- otro **20%** se puede ejecutar en 1 nodo (trivial) o en, exactamente 16 nodos.
- otro **30%** se puede ejecutar de 1 a 4 nodos.

Nota: Suponga que se dispone de 16 nodos.

a) Proponga un posible árbol de precedencia entre tareas que case con los datos del problema. Proponga el número de tareas que necesite indicando claramente los porcentajes indicativos del peso relativo de cada tarea respecto de su ejecución secuencial
b) Calcule la ganancia y eficiencia paralela en velocidad en función del número de nodos usados.
c) Suponiendo una sobrecarga temporal de $t_{overhead} = K \cdot p$, con K una constante menor que 1 y p el número total de nodos, demuestre que existe necesariamente un mínimo para el tiempo paralelo en función de p.

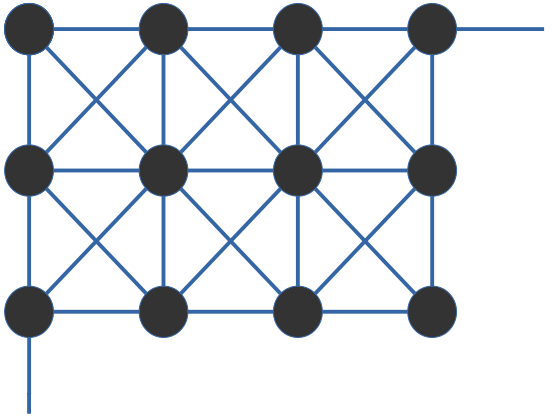
A C++ application has been parallelized to speed-up its execution in a multicomputer system.
In this way, it has been stated that the 25% of the application can be executed only in a single node, whereas the remaining percentage can be executed in several nodes as indicated in the next list.

- 25% can be executed in any number of nodes.
- 30% only can be executed in one (trivial) or 4 nodes.
- 20% only can be executed in one (trivial) up to 8 nodes.

Calculate:

a) Parallel speed-up and Efficiency as a function of the number of nodes.
b) Supposing that the time overhead is t overhead (p) = K·p (with K a constant less an 1 and p the number of nodes), demonstrate that there exists a minimum for the parallel time. What is this minimum?

La red de comunicación de una máquina paralela es tipo malla 2D **16x16**.
Sobre ella se ejecuta una aplicación que envía mensajes entre los nodos a distancia **d** con la siguiente probabilidad:



D (distancia)	**1**	**3**	**7**
Prob	**0.6**	**0.3**	**0.1**

Responda a las siguientes cuestiones:
a) Diámetro de la red.
b) Distancia media de la aplicación.
c) Distancia mínima para comunicar el nodo (3,5) y elnodo (12,12)
d) El ancho de banda de bisección es 184 Gbit/s. Calcule el ancho de banda del canal.

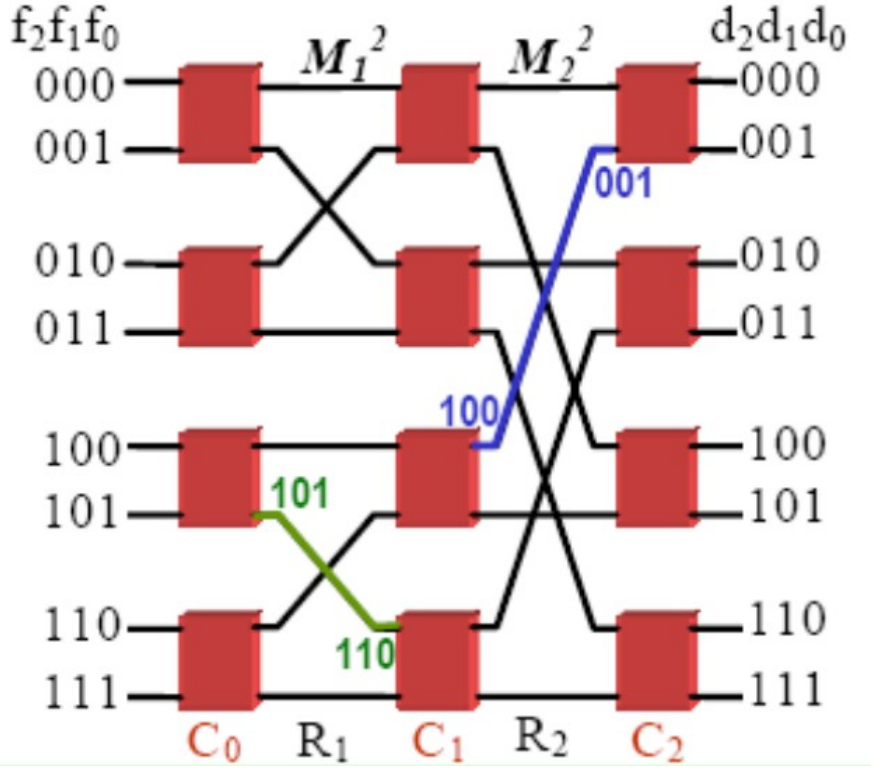
The communication network of a parallel machine is 2D 16x16 mesh type.
On it is mailed an application that sends messages between the nodes at distance d with the following probability:

Answer the following questions:
a) Network diameter.
b) Average distance of the application.
c) Minimum distance to communicate the node (3.5) and the node (12.12)
d) The bisection bandwidth is 184 Gbit/s. Calculate the channel bandwidth.

D (distance)	**1**	**3**	**7**
Prob	**0.6**	**0.3**	**0.1**

En una red mariposa **16x16** (N=16), el coste de los conmutadores 4x4 es 4 veces superior a los conmutadores de 2x2. Calcule el precio de cada implementación y quédese con el más barato.

In a 16x16 butterfly network (N=16), the cost of 4x4 switches is 4 times higher than 2x2 switches. Calculate the price of each deployment and select the cheapest one.



Un multicomputador usa una red de comunicación con enlaces de **1Gbps** y la comunicación es tipo Almacenamiento y Reenvío (Store & Forward). Mandar un paquete de 32 bytes a un nodo a distancia D=6 tarda 1.56μs. ¿Cuántas veces más rápido sería dicha transmsión si la técnica de conmitación fuera VCT? Suponga tráfico 0, flits de 8 bits y cabecera 1 flit.
Ayuda: Los paquetes tienen, por tanto, 31 flits de datos y 1 flit de cabecera, es decir 256 bits/paquete

A multicomputer uses a communication network with 1Gbps links. The switching technique is Store&Forward. Sending a 32-byte packet to a remote node at a distance of 6 takes 1.56 microseconds. How many times would the transmission be faster if the switching technique were wormhole? Assume zero traffic, 8-bit flits and 1 flit headers (31 data flits and 1 header flit).

Se tiene un cierto multiprocesador de **4 núcleos** conectados a un bus común que tiene 64GiB de memoria principal e implementa el protocolo MESI para mantener la coherencia entre sus cachés. **El sistema de caché tiene 8 bytes por línea o bloque de caché.** El tamaño de cada caché es de tan sólo **2 líneas** y la política de reemplazo es LRU.

En un determinado momento se producen los siguientes 6 accesos a memoria (ver abajo). Indique mediante una tabla o una **lista de eventos** la siguiente información para cada evento:

- Estado de cada línea de caché
- Señales que se inyectan en el bus (Ej. BusRdX en P2)
- ¿Está la memoria principal actualizada?
- Tipo de transferencia (Ej. $C_i \rightarrow C_j$, $MP \rightarrow C_i$)

Estado inicial:

	C0	C1	C2	C3
L0	I	I	I	I
L1	I	I	I	I

Acceso 1: `P2_Read@0x02`
Acceso 2: `P0_Read@0x00`
Acceso 3: `P2_Write(@0x02,7)`
Acceso 4: `P0_Read@0x12`
Acceso 5: `P0_Write(@0x00`
Acceso 6: `P1_Write(@0xFF,1)`

You have a 4-core multiprocessor connected to a common bus that has 64GiB of main memory and implements the MESI protocol to maintain consistency between its caches. The cache system has 8 bytes per line (and block) of cache. The size of each cache is only **2 lines** and the replacement policy is LRU.

At a given time, the following 6 memory accesses occur (see below). Indicate by means of a table or an event list the following information for each event:

- Status of each cache line
- Signs that are injected into the bus (I.e. BusRdX on P2)
- Is the main memory updated?
- Transfer type (I.e. $C_i \rightarrow C_j$, $MP \rightarrow C_i$)

Access 1: `P2_Read@0x02`
Access 2: `P0_Read@0x00`
Access 3: `P2_Write(@0x02,7)`
Access 4: `P0_Read@0x12`
Access 5: `P0_Write(@0x00,8)`
Access 6: `P1_Write(@0xFF,1)`

Initial state:

	C0	C1	C2	C3
L0	I	I	I	I
L1	I	I	I	I