Sistemas Operativos Distribuidos

Gestión de Procesos

Gestión de Procesos

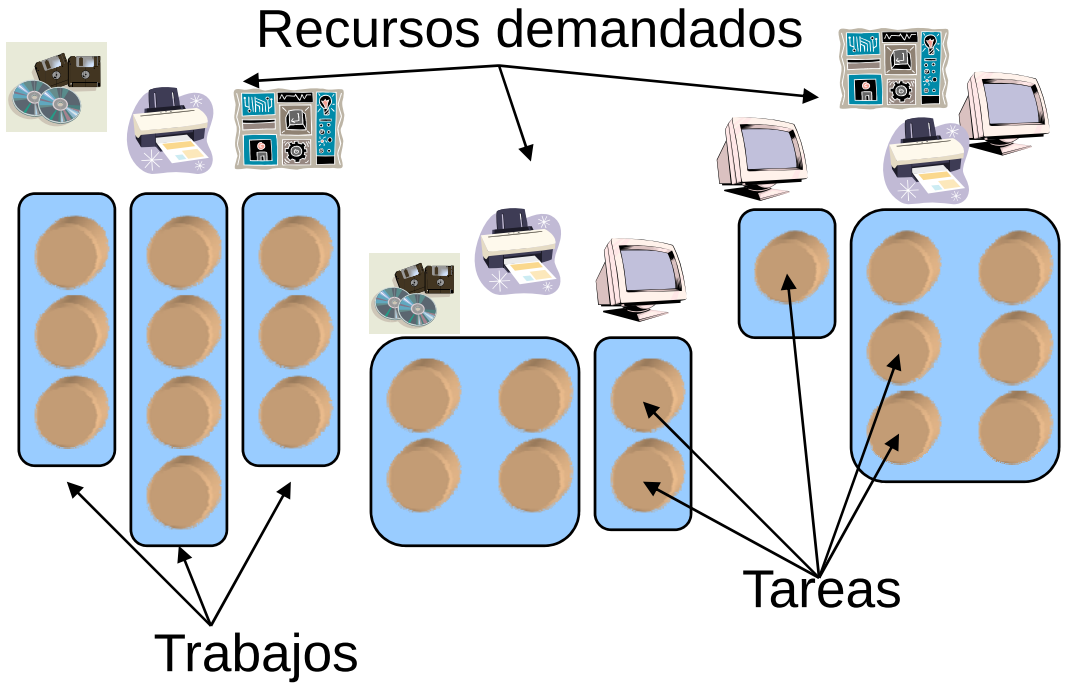
1. Conceptos y taxonomías: Trabajos y sistemas paralelos
2. Planificación estática:
   - Planificación de tareas dependientes
   - Planificación de tareas paralelas
   - Planificación de múltiples tareas
3. Planificación dinámica:
   - Equilibrado de carga
   - Migración de procesos
   - Migración de datos
   - Equilibrado de conexiones

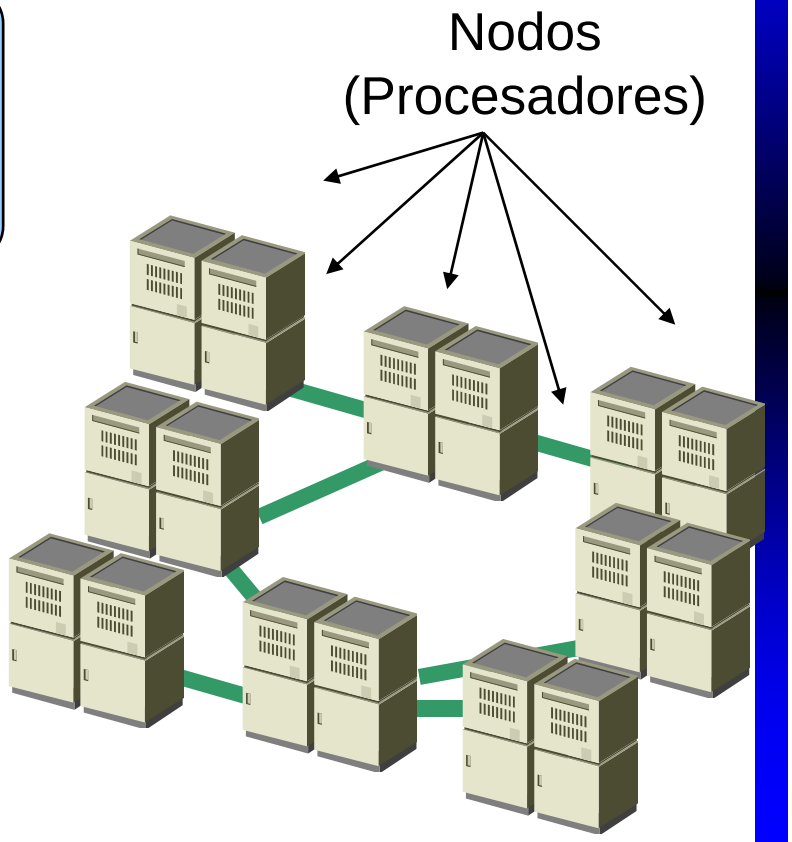
Fernando Pérez Costoya
José María Peña Sánchez

Escenario de Partida: Términos

- Trabajos: Conjuntos de tareas (procesos o hilos) que demandan: (recursos x tiempo)
  - Recursos: Datos, dispositivos, CPU u otros requisitos (finitos) necesarios para la realización de trabajos.
  - Tiempo: Periodo durante el cual los recursos están asignados (de forma exclusiva o no) a un determinado trabajo.
  - Relación entre las tareas: Las tareas se deben ejecutar siguiendo unas restricciones en relación a los datos que generan o necesitan (dependientes y concurrentes)
- Planificación: Asignación de trabajos a los nodos de proceso correspondientes. Puede implicar revisar, auditar y corregir esa asignación.

Escenario de Partida





OBJETIVO

Asignación de los trabajos de los usuarios a los distintos procesadores, con el objetivo de mejorar prestaciones frente a la solución tradicional

Fernando Pérez Costoya
José María Peña Sánchez

Características de un Sistema Distribuido

- Sistemas con memoria compartida
  - Recursos de un proceso accesibles desde todos los procesadores
    - Mapa de memoria
    - Recursos internos del SO (ficheros/dispositivos abiertos, puertos, etc.)
  - Reparto/equilibrio de carga (*load sharing/balancing*) automático
    - Si el procesador queda libre puede ejecutar cualquier proceso listo
  - Beneficios del reparto de carga:
    - Mejora uso de recursos y rendimiento en el sistema
    - Aplicación paralela usa automáticamente procesadores disponibles
- Sistemas distribuidos
  - Proceso ligado a procesador durante toda su vida
  - Recursos de un proceso accesibles sólo desde procesador local
    - No sólo mapa de memoria; También recursos internos del SO
  - Reparto de carga requiere migración de procesos

Escenario de Partida: Objetivos

¿Qué “mejoras de prestaciones” se espera conseguir?

Tipología de sistemas:

- Sistemas de alta disponibilidad
  - HAS: *High Availability Systems*
  - Que el servicio siempre esté operativo
  - Tolerancia a fallos
- Sistemas de alto rendimiento
  - HPC: *High Performance Computing*
  - Que se alcance una potencia de cómputo mayor
  - Ejecución de trabajos pesados en menor tiempo
- Sistemas de alto aprovechamiento
  - HTS: *High Troughput Systems*
  - Que el número de tareas servidas sea el máximo posible
  - Maximizar el uso de los recursos o servir a más clientes (puede no ser lo mismo).

Sistemas Operativos Distribuidos

Descripción de Tareas

- Coordinación
- Orquestación

Fernando Pérez Costoya
José María Peña Sánchez

Escenario de Partida: Trabajos

¿Qué se tiene que ejecutar?

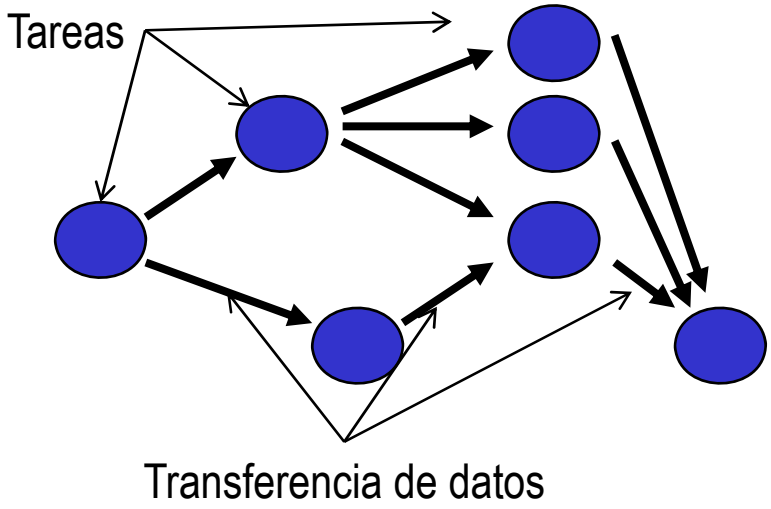
Tareas en las que se dividen los trabajos:

- Tareas disjuntas
  - Procesos independientes
  - Pertenecientes a distintos usuarios
- Tareas cooperantes
  - Interaccionan entre sí
  - Pertenecientes a una misma aplicación
  - Pueden presentar dependencias
  - O Pueden requerir ejecución en paralelo

Tareas Cooperantes

Dependencias entre tareas

- Modelizado por medio de un grafo dirigido acíclico (DAG).



- Ejemplo: Workflow

Ejecución paralela

- Implican un número de tareas concurrentes ejecutando simultáneamente:
  - De forma síncrona o asíncrona.
  - En base a una topología de conexión.
  - Siguiendo un modelo maestro/esclavo o distribuido.
  - Con unas tasas de comunicación y un intercambio de mensajes.

- Ejemplo: Código MPI

Orquestación vs. Coreografía

Existen dos términos relacionados en la gestión de servicios:

- Orquestación (*Orchestation*): Representa la ordenación y gestión de servicios desde la perspectiva de un participante (un proceso de negocio). Existe un solo coordinador.
- Coreografía (*Choreography*): Tiene un ámbito más amplio e implica la coordinación de todos los participantes de un sistema complejo entero. Existe una política en la que varios elementos se coordinan y se ajustan entre sí.

- Una diferencia muy sutil (en el plano teórico).
- En ambos casos representan definiciones declarativas de cómo se deben realizar uno o varios procesos, denominadas a veces como reglas de negocio (*business rules*)

Gestión de Negocio (I)

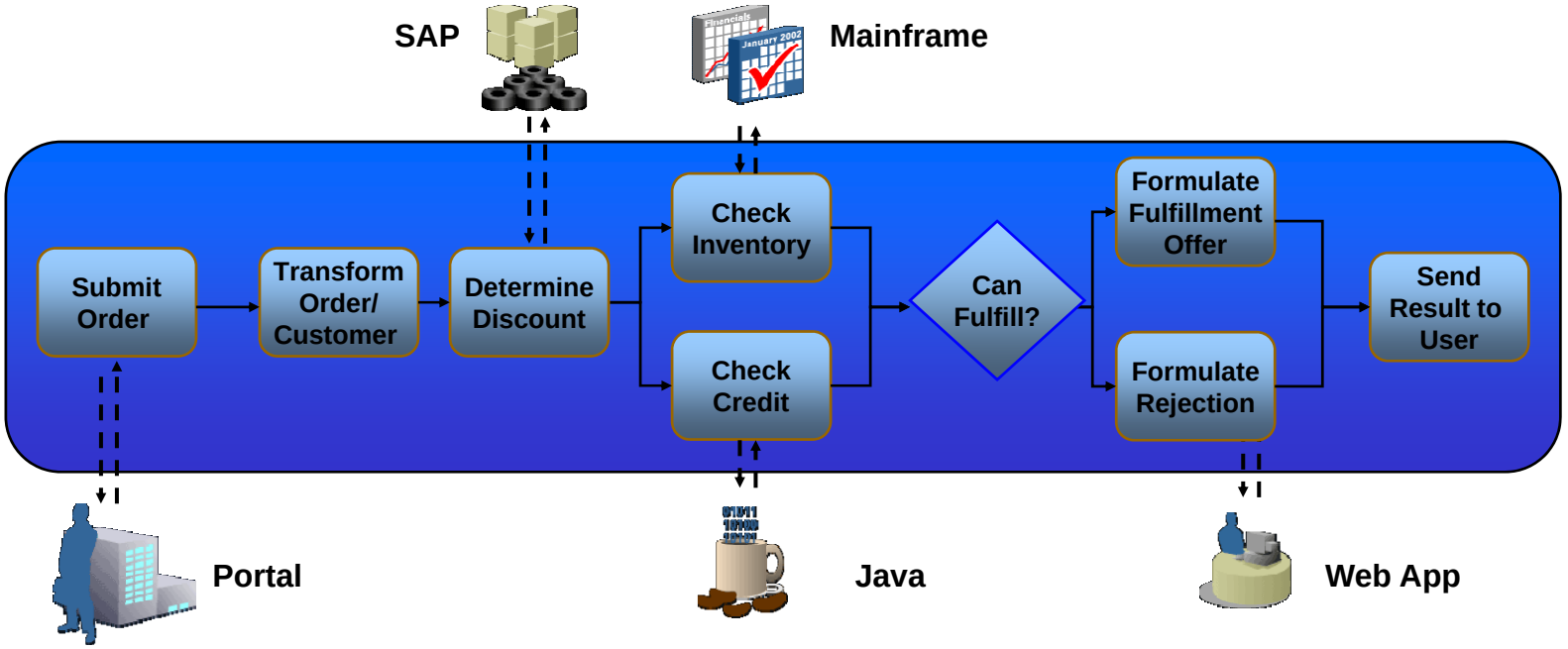
- Los sistemas que implementan BPM, denominados habitualmente Business Process Management System (BPMS) utilizan lenguajes de descripción de procesos:
  - BPEL (*Business Process Execution Language*) lenguaje XML de orquestación de servicios. Extensión de:
    - WSFL (de IBM)
    - XLANG (de BizTalk-Microsoft).

    Actualmente estandarizado por OASIS.
  - Otros lenguajes son (BPML – *Business Process Modeling Language* [anterior], y WS-CDL – *Web Services Choreography Description* [sin implementación]).

Gestión de Negocio (II)

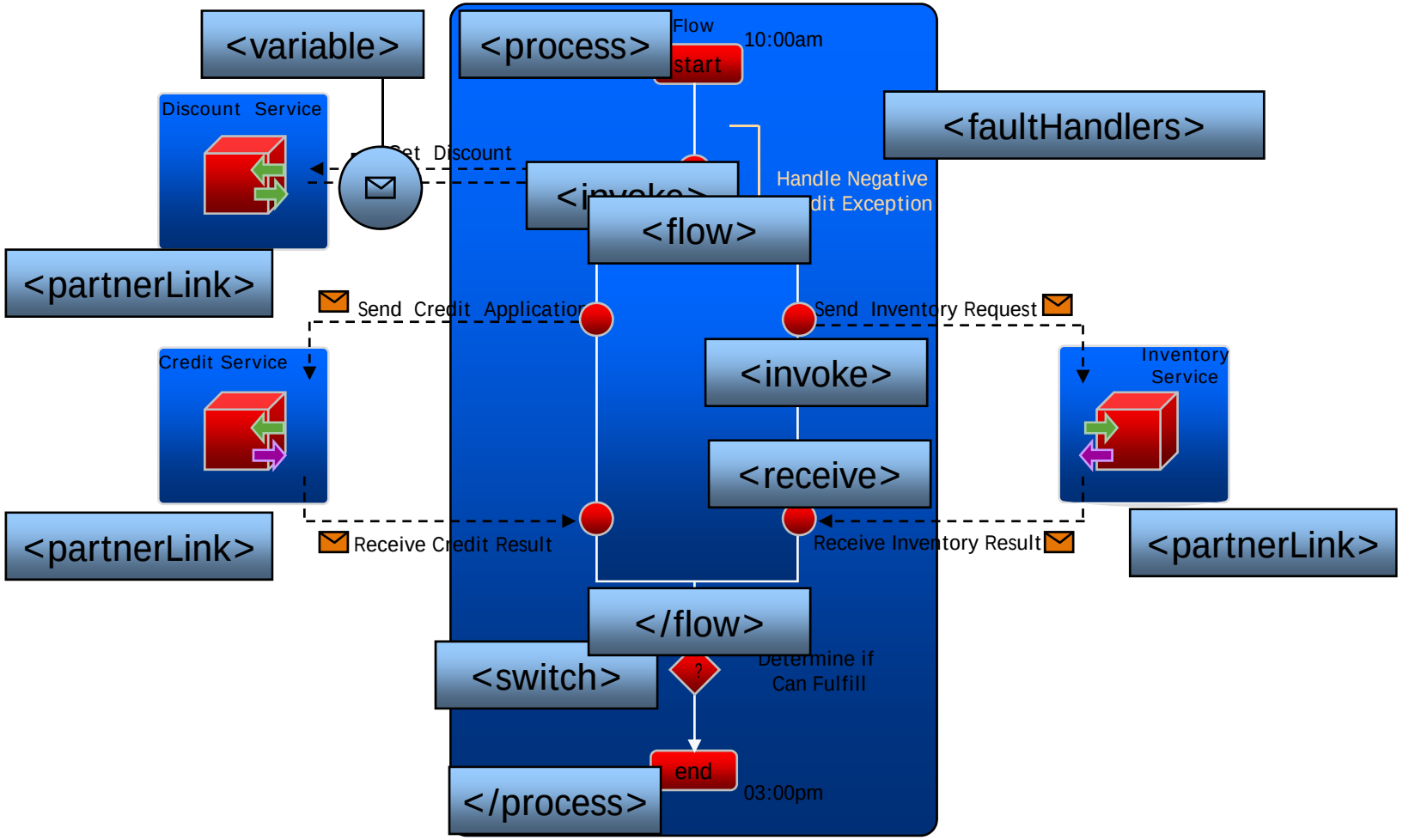
BELP:

- Define procesos de negocio interoperables y protocolos de negocio.
- Permite componer servicios nuevos a partir de otros.
- Define estructuras de control (if…then…else, while, sequence, flow)
- Gestiona variables del proceso y mensajes (entrantes y salientes).



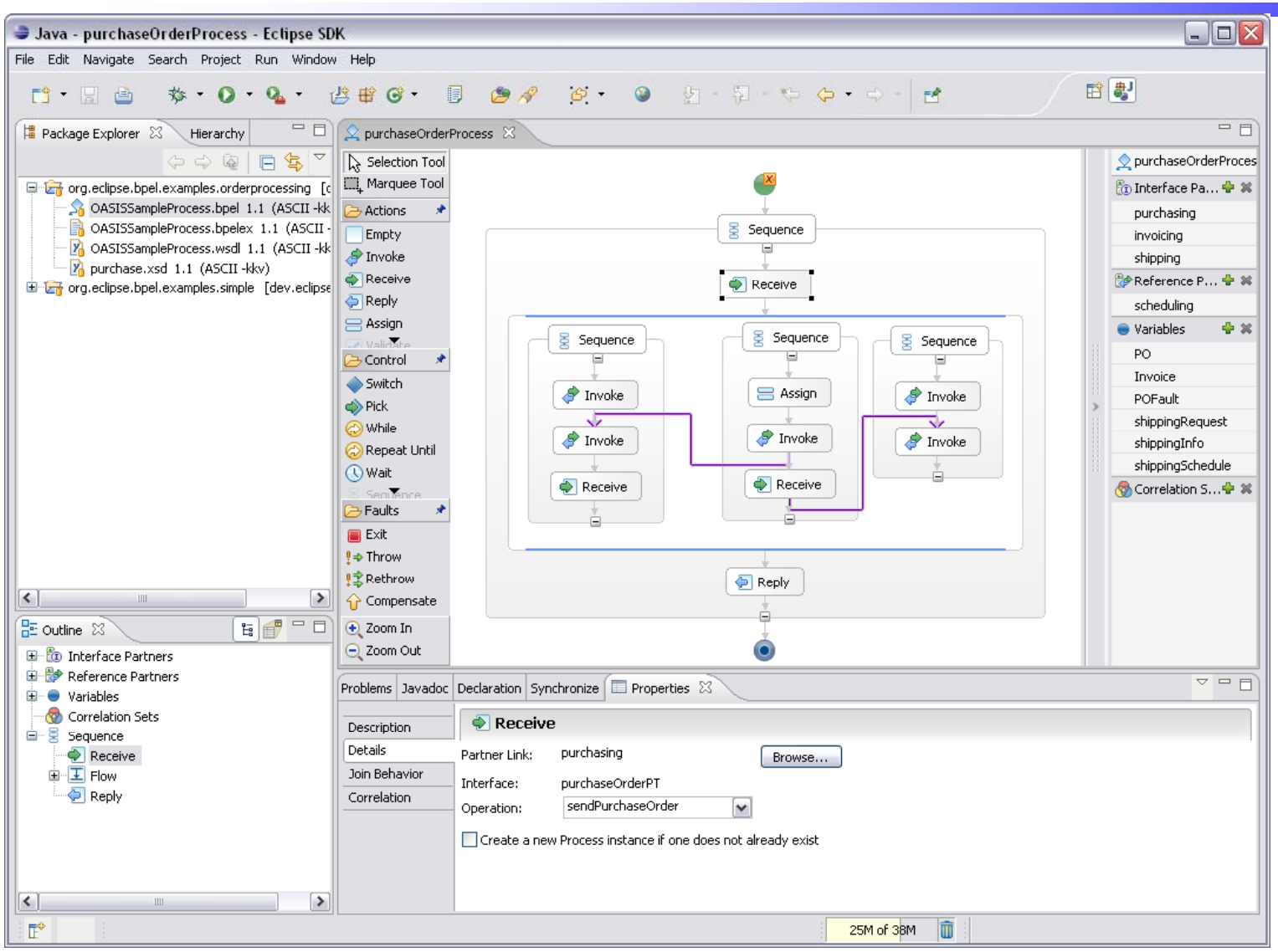
Fernando Pérez Costoya
José María Peña Sánchez

Gestión de Negocio (III)



© Oracle, SOA – Oracle Development Day

Gestión de Negocio (y IV)



Edición de procesos BPEL por medio de un entorno gráfico (Eclipse)
© BPEL project
http://www.eclipse.org/bpel/

Fernando Pérez Costoya
José María Peña Sánchez

Sistemas Operativos Distribuidos

Planificación

- Estática
- Dinámica

Fernando Pérez Costoya
José María Peña Sánchez

Sistemas de Cómputo

- Dependen de uso previsto del sistema:
  - Máquinas autónomas de usuarios independientes
    - Usuario cede uso de su máquina pero sólo cuando está desocupada
    - ¿Qué ocurre cuando deja de estarlo?
      - Migrar procesos externos a otros nodos inactivos
      - Continuar ejecutando procesos externos con prioridad baja
  - Sistema dedicado sólo a ejecutar trabajos paralelos
    - Se puede hacer una estrategia de asignación a priori
    - O ajustar el comportamiento del sistema dinámicamente
    - Se intenta optimizar tiempo de ejecución de la aplicación o el aprovechamiento de los recursos
  - Sistema distribuido general (múltiples usuarios y aplicaciones)
    - Se intenta lograr un reparto de carga adecuado

Tipología de Clusters

- High Performance Clusters
  - Beowulf; programas paralelos; MPI; dedicación a un problema
- High Availability Clusters
  - ServiceGuard, Lifekeeper, Failsafe, heartbeat
- High Throughput Clusters
  - Workload/resource managers; equilibrado de carga; instalaciones de supercomputación
- Según servicio de aplicación:
  - Web-Service Clusters
    - LVS/Piranha; equilibrado de conexiones TCP; datos replicados
  - Storage Clusters
    - GFS; sistemas de ficheros paralelos; identica visión de los datos desde cada nodo
  - Database Clusters
    - Oracle Parallel Server;

Planificación

- La planificación consiste en el despliegue de las tareas de un trabajo sobre unos nodos del sistema:
  - Atendiendo a las necesidades de recursos
  - Atendiendo a las dependencias entre las tareas
- El rendimiento final depende de diversos factores:
  - Concurrencia: Uso del mayor número de procesadores simultáneamente.
  - Grado de paralelismo: El grado más fino en el que se pueda descomponer la tarea.
  - Costes de comunicación: Diferentes entre procesadores dentro del mismo nodo y procesadores en diferentes nodos.
  - Recursos compartidos: Uso de recursos (como la memoria) comunes para varios procesadores dentro del mismo nodo.

Planificación

- Dedicación de los procesadores:
  - Exclusiva: Asignación de una tarea por procesador.
  - Tiempo compartido: En tareas de cómputo masivo con E/S reducida afecta dramáticamente en el rendimiento. Habitualmente no se hace.
- La planificación de un trabajo puede hacerse de dos formas:
  - **Planificación estática**: Inicialmente se determina dónde y cuándo se va a ejecutar las tareas asociadas a un determinado trabajo. Se determina antes de que el trabajo entre en máquina.
  - **Planificación dinámica**: Una vez desplegado un trabajo, y de acuerdo al comportamiento del sistema, se puede revisar este despliegue inicial. Considera que el trabajo ya está en ejecución en la máquina.

Sistemas Operativos Distribuidos

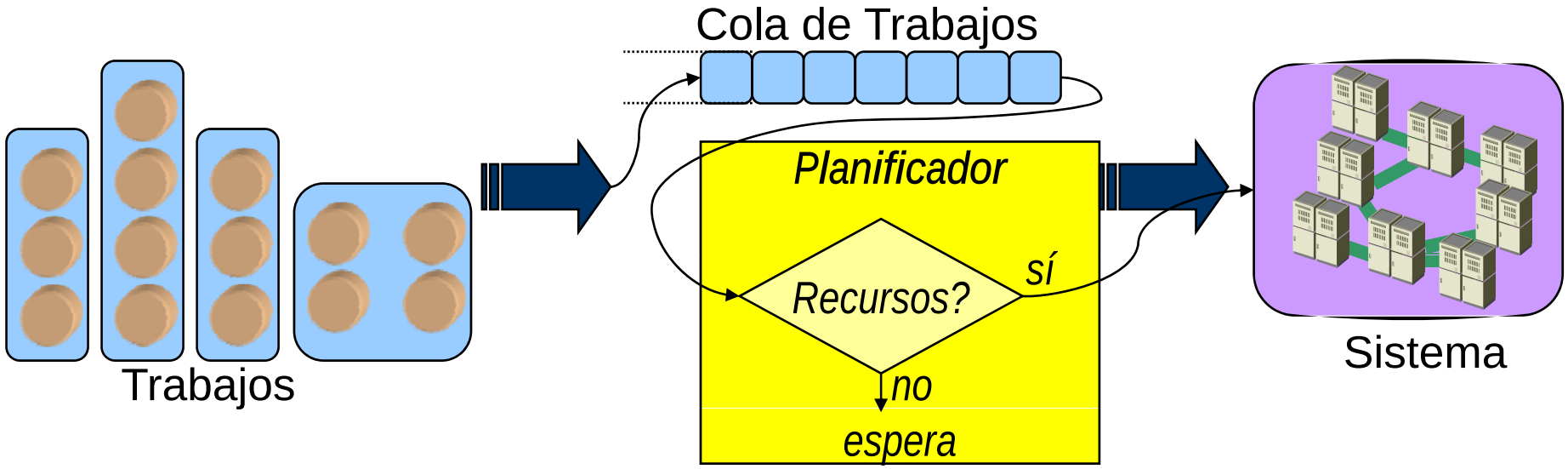
Gestión de Procesos

Planificación Estática

Fernando Pérez Costoya
José María Peña Sánchez

Planificación Estática

- Generalmente se aplica antes de permitir la ejecución del trabajo en el sistema.
- El planificador (a menudo llamado *resource manager*) selecciona un trabajo de la cola (según política) y si hay recursos disponibles lo pone en ejecución, si no espera.

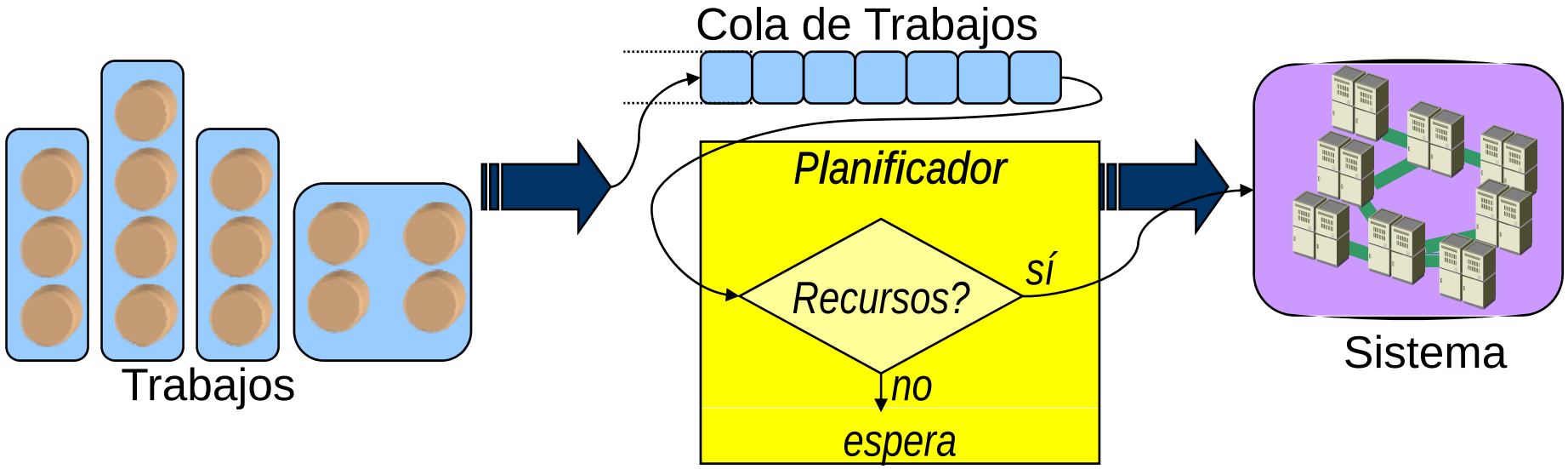


Descripción de los Trabajos

- Para poder tomar las decisiones correspondientes a la política del planificador, éste debe disponer de información sobre los trabajos:
  - Número de tareas (ejecutables correspondientes)
  - Prioridad
  - Relación entre ellas (DAG)
  - Estimación de consumo de recursos (procesadores, memoria, disco)
  - Estimación del tiempo de ejecución (por tarea)
  - Otros parámetros de ejecución
  - Restricciones aplicables
- Estas definiciones se incluyen en un fichero de descripción del trabajo, cuyo formato depende del planificador correspondiente.

Planificación de Múltiples Trabajos

- Cuando se deben planificar varios trabajos el planificador debe:
  - Seleccionar el siguiente trabajo a mandar a máquina.
  - Determinar si hay recursos (procesadores y de otro tipo) para poder lanzarlo.
  - De no ser así, esperar hasta que se liberen recursos.



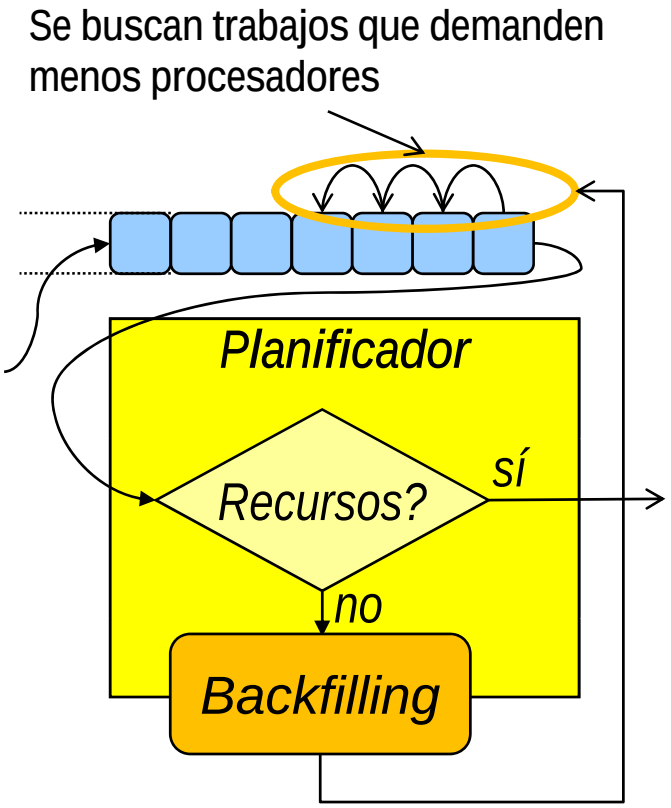
Fernando Pérez Costoya
José María Peña Sánchez

Planificación de Múltiples Trabajos

- ¿Cómo se selecciona el siguiente trabajo a intentar ejecutar?:
  - Política FCFS (*first-come-first-serve*): Se respeta el orden de remisión de trabajos.
  - Política SJF (*shortest-job-first*): El trabajo más pequeño en primer lugar, medido en:
    - Recursos, número de procesadores, o
    - Tiempo solicitado (estimación del usuario).
  - Política LJF (*longest-job-first*): Ídem pero en el caso inverso.
  - Basadas en prioridades: Administrativamente se define unos criterios de prioridad, que pueden contemplar:
    - Facturación del coste de recursos.
    - Número de trabajos enviados.
    - *Deadlines* de finalización de trabajos. (EDF – *Earliest-deadline-first*)

Backfilling

- *Backfilling* es una modificación aplicable a cualquiera de las políticas anteriores:
  - Si el trabajo seleccionado por la política no tiene recursos para entrar entonces,
  - Se busca otro proceso en la cola que demande menos recursos y pueda entrar.
  - Permite aprovechar mejor el sistema

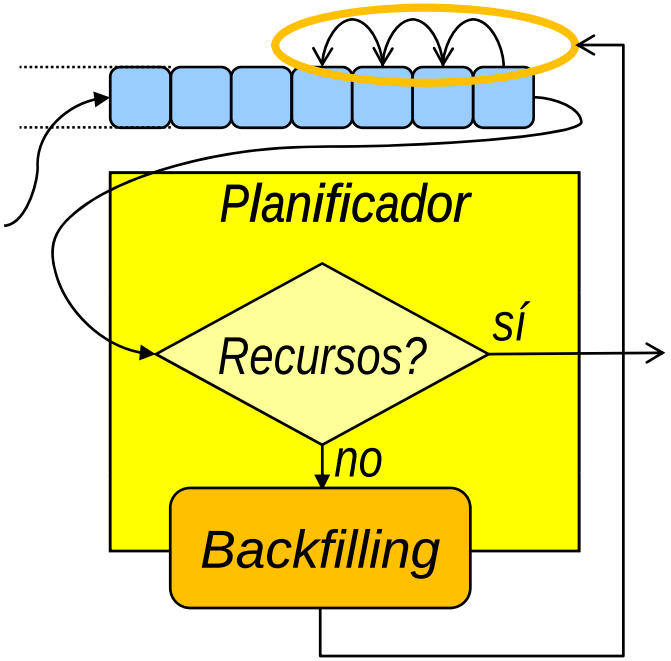


Fernando Pérez Costoya
José María Peña Sánchez

Backfilling con Reservas

- Las reservas consisten en:
  - Determinar cuándo se podría ejecutar la tarea inicialmente seleccionada, en base a las estimaciones de tiempos (*deadline*)
  - Se dejan entrar trabajos que demandan menos recursos (*backfilling*) siempre y cuando finalicen antes del *deadline*.
  - Aumenta el aprovechamiento del sistema, pero no retrasa indefinidamente a los trabajos grandes.

La técnica de Backfilling puede hacer que trabajos que demanden muchos recursos nunca se ejecuten

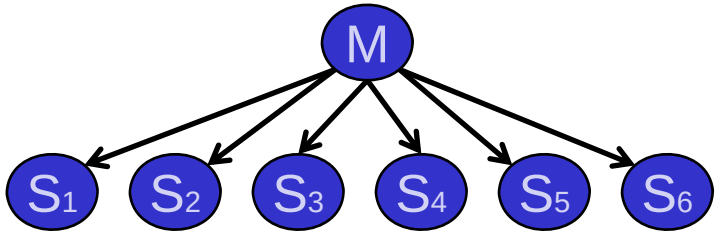


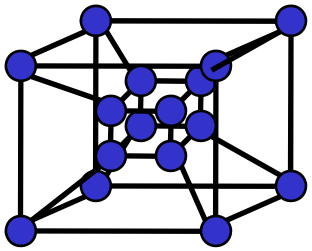
Fernando Pérez Costoya
José María Peña Sánchez

Planificación de Tareas Paralelas

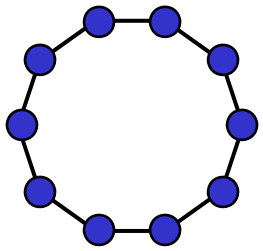
- Considera los siguientes aspectos:
  - Las tareas requieren ejecutarse en paralelo
  - Intercambian mensajes a lo largo de la ejecución.
  - Consumo de recursos locales (memoria o E/S) de cada tarea.

Modelo Centralizado (Maestro/Esclavo)





Hipercubo



Anillo

Modelo distribuido

Diferentes parámetros de comunicación:

- Tasas de comunicación: Frecuencia, volumen de datos.
- Topología de conexión: ¿Cómo intercambian los mensajes?
- Modelo de comunicación: Síncrono (las tareas se bloquea a la espera de datos) o Asíncrono.

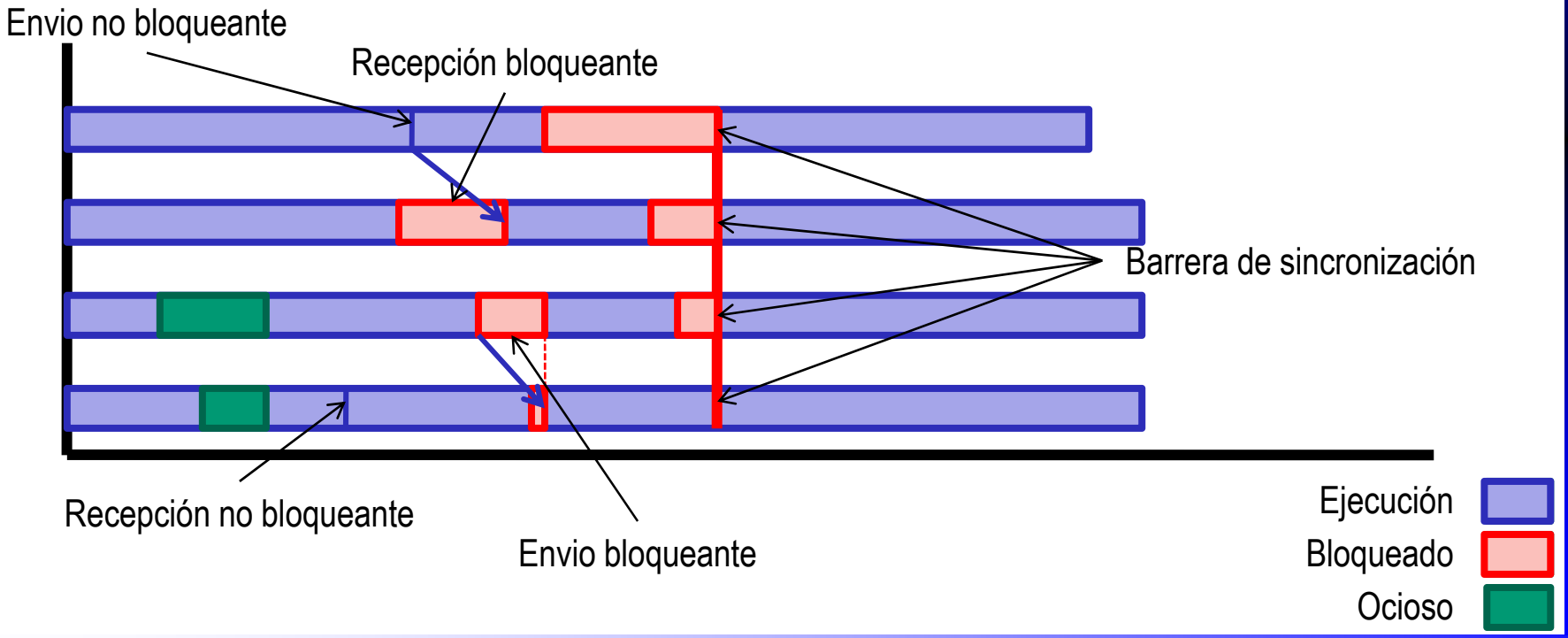
Restricciones:

- La propia topología física de la red de interconexión
- Prestaciones de la red.

Fernando Pérez Costoya
José María Peña Sánchez

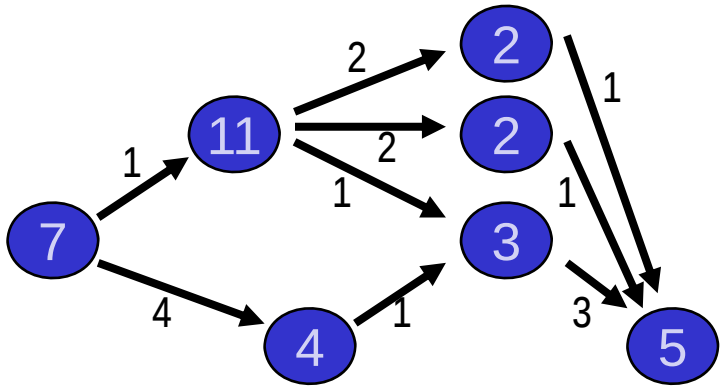
Rendimiento de la Planificación

- El rendimiento del esquema de planificación depende:
  - Condiciones de bloqueo (equilibrado de carga)
  - Estado del sistema
  - Eficiencia de las comunicaciones: latencia y ancho de banda



Planificación de Tareas Dependientes

- Considera los siguientes aspectos:
  - Duración (estimada) de cada tarea.
  - Volumen de datos transmitido al finalizar la tarea (e.g. fichero)
  - Precedencia entre tareas (una tarea requiere la finalización previa de otras).
  - Restricciones debidas a la necesidad de recursos especiales.



Representado por medio de un grafo acíclico dirigido (DAG)

Una opción consiste en transformar todos los datos a las mismas unidades (tiempo):

- Tiempo de ejecución (tareas)
- Tiempo de transmisión (datos)

La Heterogeneidad complica estas estimación:

- Ejecución dependiente de procesador
- Comunicación dependiente de conexión

Sistemas Operativos Distribuidos

Gestión de Procesos

Planificación Dinámica

Fernando Pérez Costoya
José María Peña Sánchez

Planificación Dinámica

- La planificación estática decide si un proceso se ejecuta en el sistema o no, pero una vez lanzado no se realiza seguimiento de él.
- La planificación dinámica:
  - Evalúa el estado del sistema y toma acciones correctivas.
  - Resuelve problemas debidos a la paralelización del problema (desequilibrio entre las tareas).
  - Reacciona ante fallos en nodos del sistema (caídas o falos parciales).
  - Permite un uso no dedicado o exclusivo del sistema.
  - Requiere una monitorización del sistema (políticas de gestión de trabajos):
    - En la planificación estática se contabilizan los recursos comprometidos.

Load Balancing vs. Load Sharing

- *Load Sharing*:
  - Que el estado de los procesadores no sea diferente
  - Un procesador ocioso
  - Una tarea esperando a ser servida en otro procesador

- *Load Balancing*:
  - Que la carga de los procesadores sea igual.
  - La carga varía durante la ejecución de un trabajo
  - ¿Cómo se mide la carga?

- Son conceptos muy similares, gran parte de las estrategias usadas para LS vale para LB (considerando objetivos relativamente diferentes). LB tiene unas matizaciones particulares.

Medición de la Carga

¿Qué información se transmite?:

- La carga del nodo→ ¿qué es la carga?

Diferentes medidas:

- %CPU en un instante de tiempo
- Número de procesos listos para ejecutar (esperando)
- Números de fallos de página / *swaping*
- Consideración de varios factores.

- Se pueden considerar casos de nodos heterogéneos (con diferentes capacidades).

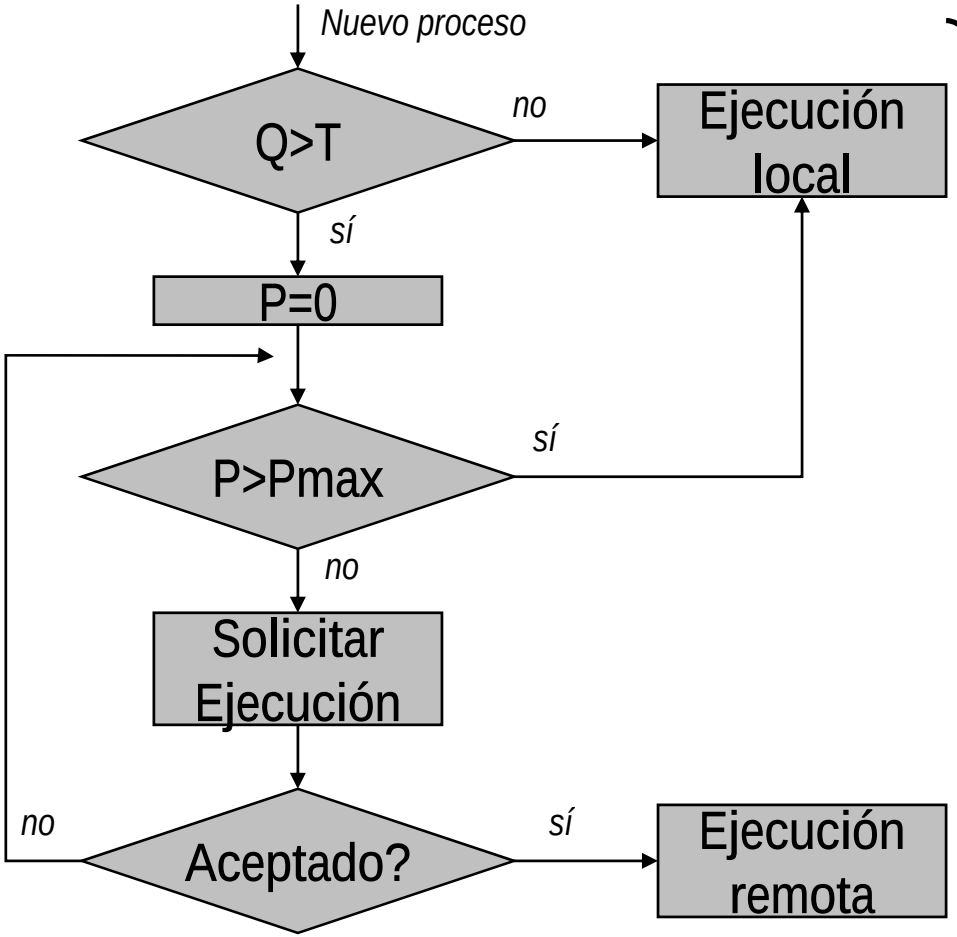
Algoritmos de Equilibrado de Carga

Situación:

- El estado del sistema es que ciertos nodos tienen una carga más alta que otros.

- Ejemplos de tipos de algoritmos:
  - Iniciados por el emisor
  - Iniciados por el receptor
  - Simétricos

Algoritmo Iniciado por el Emisor



Q: Tamaño de la cola de procesos
T: Umbral máximo de la cola de procesos
Pmax: Número máximo de solicitudes

Selección de destino:

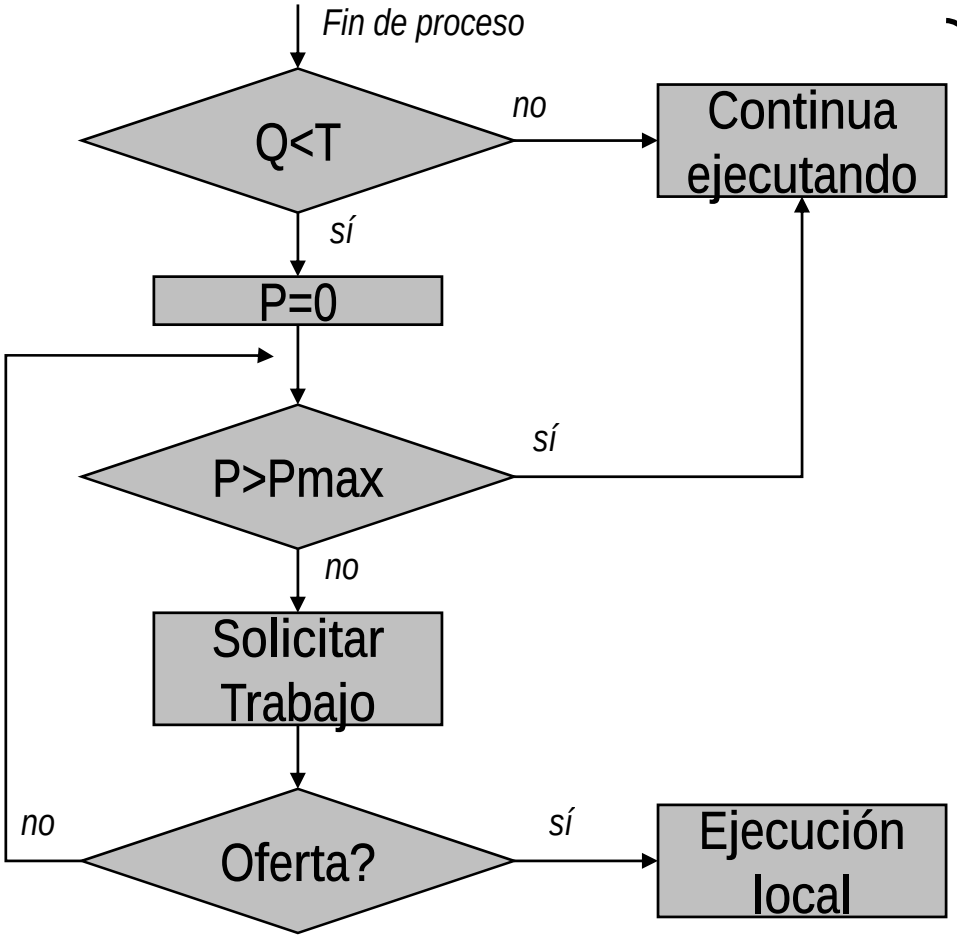
Distintas alternativas:

- Elegir un nodo al azar.
- Probar con un nº de nodos hasta encontrar un receptor.
- Probar con un nº de nodos y elegir aquél con menos carga.

Estabilidad: inestable con alta carga

- Difícil encontrar receptores y los muestreos consumen CPU

Algoritmo Iniciado por el Receptor



Q: Tamaño de la cola de procesos
T: Umbral máximo de la cola de procesos
Pmax: Número máximo de solicitudes

Selección de destino:

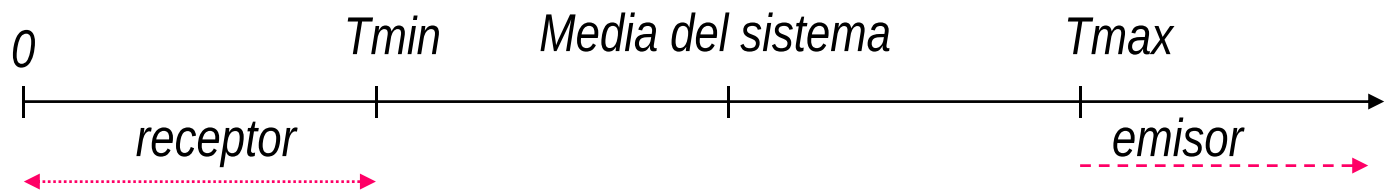
Ejemplo:

- Muestreo aleatorio de un nº limitado de nodos hasta encontrar uno con un nivel de carga > umbral.
- Si la búsqueda falla, esperar hasta que otro proceso termine o un periodo predeterminado antes de reintentar.

Estabilidad: estable

- Con altas cargas, probable que receptores encuentren emisores.

Algoritmo Simétrico



Iniciado por el emisor:

Emisor difunde mensaje *SOBRECARGADO* y espera *ACEPTAR*.

Un receptor envía *ACEPTAR*.

Si llega *ACEPTAR*: y el nodo todavía es emisor, transfiere el proceso más adecuado.

Si no, difundir un mensaje *CAMBIO-MEDIA* para incrementar la carga media estimada en el resto de nodos.

Iniciado por el receptor:

Un receptor difunde un mensaje DESCARGADO y espera por mensajes SOBRECARGADO.

Si llega un mensaje SOBRECARGADO, se envía un mensaje ACEPTAR.

Si no, difundir un mensaje CAMBIO-MEDIA para decrementar la carga media estimada en el resto de nodos.

Ejecución Remota de Procesos

- ¿Cómo ejecutar un proceso de forma remota?
  - Crear el mismo entorno de trabajo:
    - Variables de entorno, directorio actual, etc.
  - Redirigir ciertas llamadas al sistema a máquina origen:
    - P. ej. interacción con el terminal
- Migración (transferencia expulsiva) mucho más compleja:
  - "Congelar" el estado del proceso
  - Transferir a máquina destino
  - "Descongelar" el estado del proceso
- Numerosos aspectos complejos:
  - Redirigir mensajes y señales
  - ¿Copiar espacio de *swap* o servir fallos de pág. desde origen?

Migración de Procesos

Diferentes modelos de migración:

- Migración débil:
  - Restringida a determinadas aplicaciones (ejecutadas en máquinas virtuales) o en ciertos momentos.
- Migración fuerte:
  - Realizado a nivel de código nativo y una vez que la tarea ha iniciado su ejecución (en cualquier momento)
  - De propósito general: Más flexible y más compleja
- Migración de datos:
  - No se migran procesos sino sólo los datos sobre los que estaba trabajando.

Migración: Datos de las Tareas

- Los datos que usa una tarea también deben migrarse:
  - Datos en disco: Existencia de un sistema de ficheros común.
  - Datos en memoria: Requiere “congelar” todos los datos del proceso correspondiente (páginas de memoria y valores de registros).
    Técnicas de *checkpointing*:
    - Las páginas de datos del proceso se guardan a disco.
    - Se puede ser más selectivo si las regiones que definen el estado están declaradas de alguna forma específica (lenguajes/librerías especiales).
    - Es necesario guardar también los mensajes enviados que potencialmente no hayan sido entregados.
    - Útiles también para casos en los que no hay migración: Fallos en el sistema.

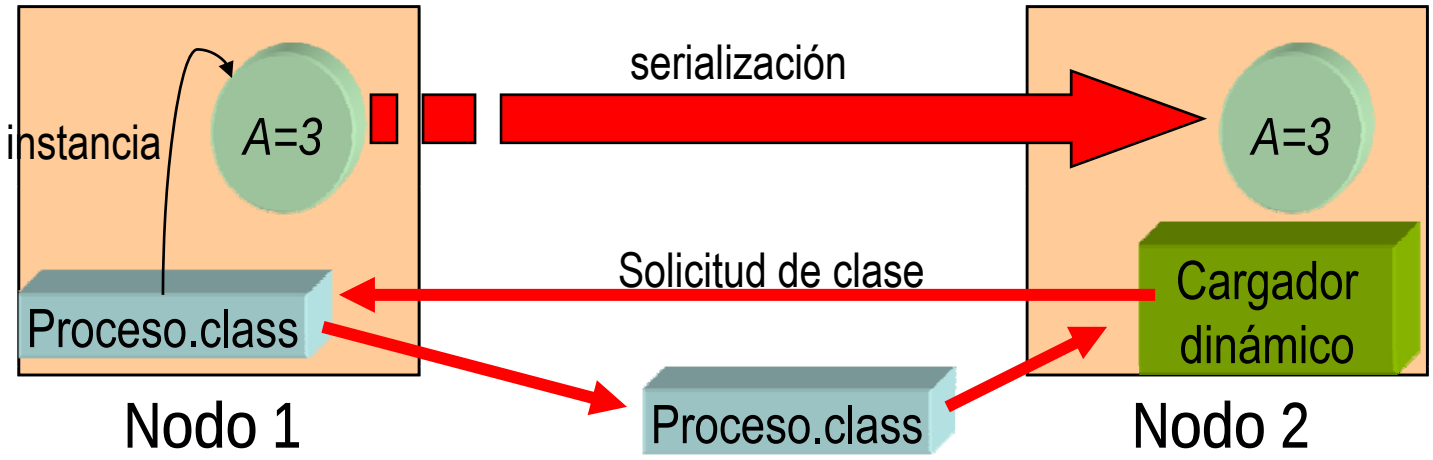
Migración Débil

- La migración débil se puede articular de la siguiente forma:
  - Ejecución remota de un nuevo proceso/programa
    - En UNIX podría ser en FORK o en EXEC
    - Es más eficiente que nuevos procesos se ejecuten en nodo donde se crearon pero eso no permite reparto de carga
  - Hay que transferir cierta información de estado aunque no esté iniciado
    - Argumentos, entorno, ficheros abiertos que recibe el proceso, etc.
  - Ciertas librerías pueden permitir al programador establecer puntos en los cuales el estado del sistema de almacena/recupera y que pueden ser usados para realizar la migración.
  - En cualquier caso el código del ejecutable debe ser accesible en el nodo destino:
    - Sistema de ficheros común.

Migración Débil

En lenguajes (como Java):

- Existe un mecanismo de serialización que permite transferir el estado de un objeto en forma de “serie de bytes”.
- Se porporciona un mecanismo de carga bajo demanda de las clases de forma remota.



Migración Fuerte

- Solución naïve:
  - Copiar el mapa de memoria: código, datos, pila, ...
  - Crear un nuevo BCP (con toda la información salvaguardada en el cambio de contexto).

- Hay otros datos (almacenados por el núcleo) que son necesarios: Denominado estado externo del proceso
  - Ficheros abiertos
  - Señales pendientes
  - Sockets
  - Semáforos
  - Regiones de memoria compartida
  - .....

Migración Fuerte

Existen diferentes aproximaciones a posibles implementaciones:

- A nivel de kernel:
  - Versiones modificadas del núcleo
  - Dispone de toda la información del proceso
- A nivel de usuario:
  - Librerias de checkpointing
  - Protocolos para desplazamiento de sockets
  - Intercepción de llamadas al sistema

Otros aspectos:

- PID único de sistema
- Credenciales y aspectos de seguridad

Migración Fuerte

- Se debe intentar que proceso remoto se inicie cuanto antes
  - Copiar todo el espacio de direcciones al destino
  - Copiar sólo páginas modificadas al destino; resto se pedirán como fallos de página desde nodo remoto servidas de *swap* de origen
  - No copiar nada al destino; las páginas se pedirán como fallos de página desde el nodo remoto
    - servidas de memoria de nodo origen si estaban modificadas
    - servidas de *swap* de nodo origen si no estaban modificadas
  - Volcar a *swap* de nodo origen páginas modificadas y no copiar nada al destino: todas las páginas se sirven de *swap* de origen
  - Precopia: Copia de páginas mientras ejecuta proceso en origen
- Páginas de código (sólo lectura) no hay que pedirlas:
  - Se sirven en nodo remoto usando SFD

Beneficios de la Migración de Procesos

- Mejora rendimiento del sistema por reparto de carga
- Permite aprovechar proximidad de recursos
  - Proceso que usa mucho un recurso: migrarlo al nodo del mismo
- Puede mejorar algunas aplicaciones cliente/servidor
  - Para minimizar transferencias si hay un gran volumen de datos:
    - Servidor envía código en vez de datos (p. ej. *applets)*
    - O cliente envía código a servidor (p. ej. cjto. de accesos a b. de datos)
- Tolerancia a fallos ante un fallo parcial en un nodo
- Desarrollo de “aplicaciones de red”
  - Aplicaciones conscientes de su ejecución en una red
  - Solicitan migración de forma explícita
  - Ejemplo: Sistemas de agentes móviles

Migración de Datos

Usando en aplicaciones de tipo maestro/esclavo.

- Maestro: Distribuye el trabajo entre los trabajadores.
- Esclavo: Trabajador (el mismo código pero con diferentes datos).

¿Cómo se reparten las iteraciones de un bucle entre los procesadores?

Si hay tantos procesadores como iteraciones, tal vez una por procesador.

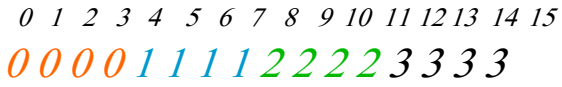
- Pero si hay menos (lo normal), hay que repartir. El reparto puede ser:
  - estático: en tiempo de compilación.
  - dinámico: en ejecución.

Migración de Datos: Objetivo

- Intentar que el tiempo de ejecución de los trozos que se reparten a cada procesador sea similar, para evitar tiempos muertos (*load balancing*).
- Representa un algoritmo de distribución de trabajo (en este caso datos) que:
  - Evite que un trabajador esté parado porque el maestro no transmite datos.
  - No asigne demasiado trabajo a un nodo (tiempo final del proceso es el del más lento)
  - Solución: Asignación de trabajos por bloques (más o menos pequeños).
  - Posibles dependencias (sincronización), el tamaño de grano, la localidad de los accesos y el coste del propio reparto.

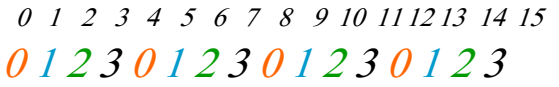
Migración de Datos: Estático

Consecutivo




```
for(pid=0;pid<P;pid++) {
  principio = pid * N/P
  fin = (pid + 1) * N/P - 1
  for (i=principio;i<fin,i++)
  {...}
}
```

Entrelazado




```
for(i=0;i<N;i++) {
   asignar(i,i%P);
}
```

No añade carga a la ejecución de los threads.

Pero no asegura el equilibrio de la carga entre los procesos.

Permite cierto control sobre la localidad de los accesos a cache.

Migración de Datos: Dinámico

- Para intentar mantener la carga equilibrada, las tareas se van escogiendo en tiempo de ejecución de un cola de tareas. Cuando un proceso acaba con una tarea (un trozo del bucle) se asigna un nuevo trozo.
- Dos opciones básicas:
  - los trozos que se van repartiendo son de tamaño constante
  - son cada vez más pequeños:
    - Asignación propia.
    - Centralizado: Guiado o Trapezoidal

Migración de Datos: Dinámico

Asignación propia

Las iteraciones se reparten una a una o por trozos

Añade carga a la ejecución de los threads.
Hay que comparar ejecución y reparto.

```
LOCK (C);
  mia = i;
  i = i + Z;    z = 1 → self
UNLOCK (C);

while (mia <= N-1) {
 limite=min(mia+Z,N);
 for(j=mia;j<limite;j++)
  {...}

 LOCK (C)
  mia = i;
  i = i + Z;
 UNLOCK (C)
}
```

Fernando Pérez Costoya
José María Peña Sánchez

Migración de Datos: Dinámico

Guiado / Trapezoidal

Los trozos de bucle que se reparten son cada vez más pequeños según nos acercamos al final.

- Guiado parte proporcional de lo que queda por ejecutar:

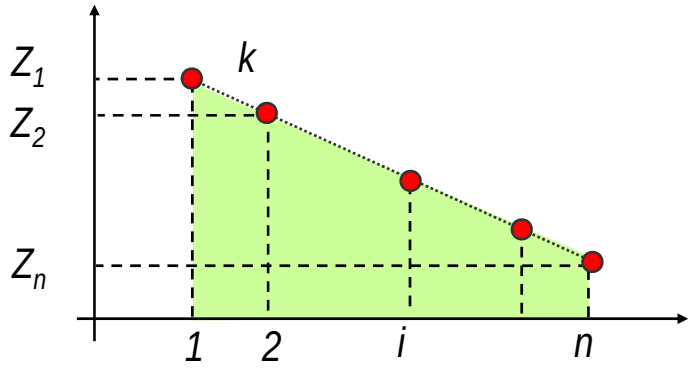
$$Z_s = (N - i) / P \quad \textit{(entero superior)}$$

que equivale a

$$Z_i = Z_{i-1} (1 - 1/P)$$

Migración de Datos: Dinámico

Trapezoidal reduciendo el trozo anterior en una constante: $Z_i = Z_{i-1} - k$



op. de planificación

$$\sum_{s=1}^{n} Z_s = \frac{Z_1 + Z_n}{2}\, n = \frac{Z_1 + Z_n}{2}\left(\frac{Z_1 - Z_n}{k} + 1\right) = N \quad \rightarrow \quad k = \frac{Z_1^2 - Z_n^2}{2N - (Z_1 + Z_n)}$$

Fernando Pérez Costoya
José María Peña Sánchez

Equilibrado de Conexiones

- Algunos sistemas (e.g. servidores web) consideran que un trabajo es una conexión remota que realiza una solicitud:
  - En estos casos se debe intentar repartir la carga de las peticiones entre varios servidores.
  - Problemática: La dirección del servicio es única.
  - Solución: Equilibrado de conexiones:
    - Redirección a nivel de DNS
    - Redirección a nivel IP (Reescritura NAT o encapsulado)
    - Redirección a nivel MAC