Sistemas Operativos Distribuidos

Gestión de Procesos

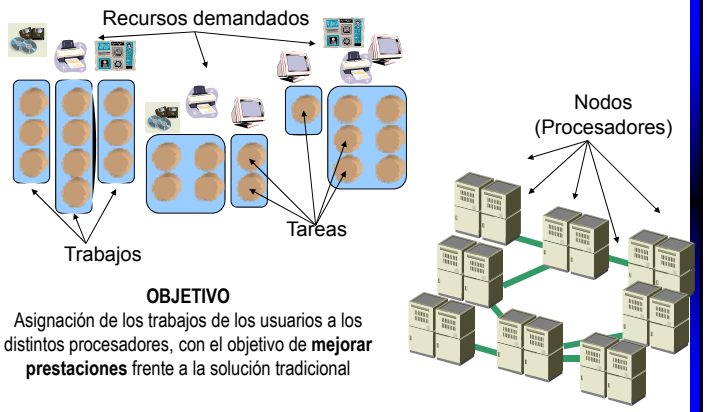
Gestión de Procesos

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

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

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: Trabajos

¿Qué se tiene que ejecutar?

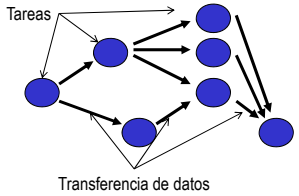
Tareas en las que se dividen los trabajos:

- 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

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 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

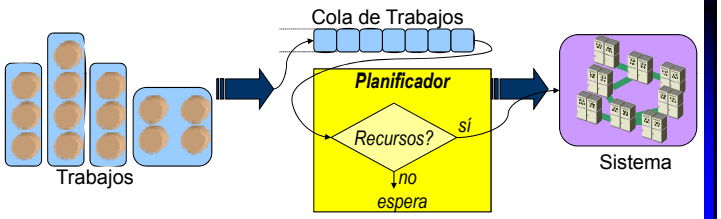
Gestión de Procesos

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.



Fernando Pérez Costoya
José María Peña Sánchez

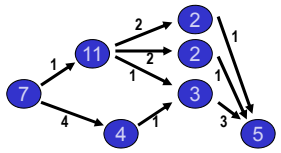
Descripción de los Trabajos

- 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.

Fernando Pérez Costoya
José María Peña Sánchez

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)

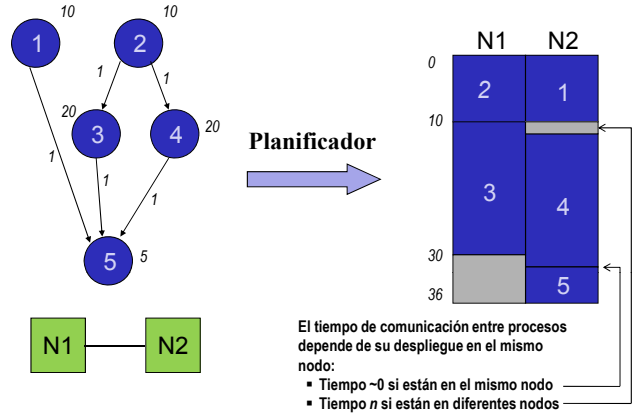
La Heterogeneidad complica estas estimación:
- Ejecución dependiente de procesador
- Comunicación dependiente de conexión

Fernando Pérez Costoya
José María Peña Sánchez

Planificación de Tareas Dependientes

- Planificación consiste en la asignación de tareas a procesadores en un instante determinado de tiempo:
  - Para un solo trabajo existen heurísticas eficientes: buscar camino crítico (camino más largo en grafo) y asignar tareas implicadas al mismo procesador.
  - Algoritmos con complejidad polinomial cuando sólo hay dos procesadores.
  - Es un problema NP-completo con N trabajos.
  - El modelo teórico se denomina *multiprocessor scheduling*.

Ejemplo de Tareas Dependientes



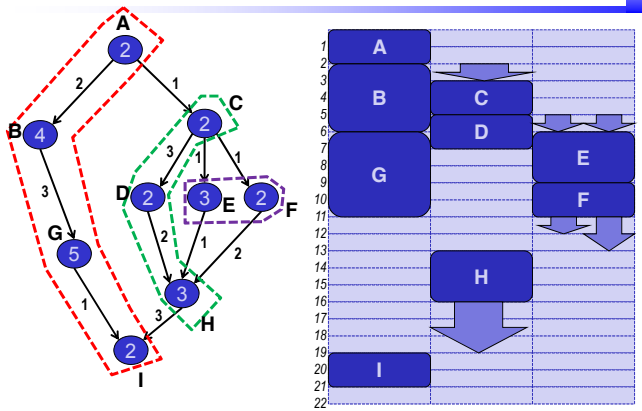
El tiempo de comunicación entre procesos depende de su despliegue en el mismo nodo:

- Tiempo ~0 si están en el mismo nodo
- Tiempo *n* si están en diferentes nodos

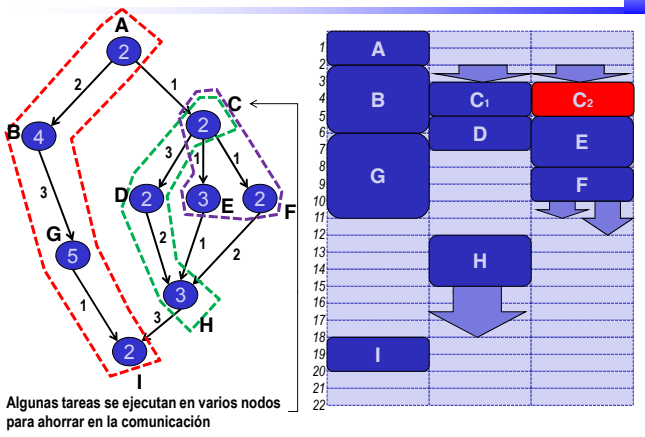
Algoritmos de *Clustering*

- Para los casos generales se aplican algoritmos de *clustering* que consisten en:
  - Agrupar las tareas en grupos (*clusters*).
  - Asignar cada *cluster* a un procesador.
  - La asignación óptima es NP-completa
  - Este modelo es aplicable a un solo trabajo o a varios en paralelo.
  - Los *clusters* se pueden construir con:
    - Métodos lineales
    - Métodos no-lineales
    - Búsqueda heurística/estocástica

Planificación con *Clustering*

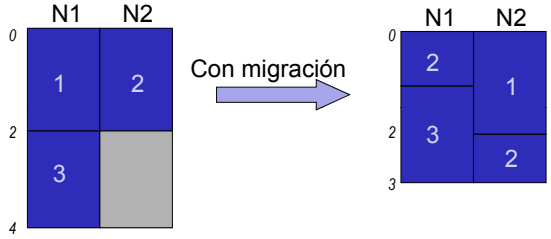


Uso de la Replicación



Migraciones con Procesos Dependientes

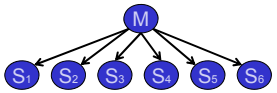
- El uso de estrategias migratorias permite mejorar el aprovechamiento.
  - Se pueden definir en la planificación estática.
  - Lo más habitual es que formen parte de las estrategias dinámicas.

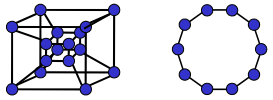


Planificación de Tareas Paralelas

- 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)





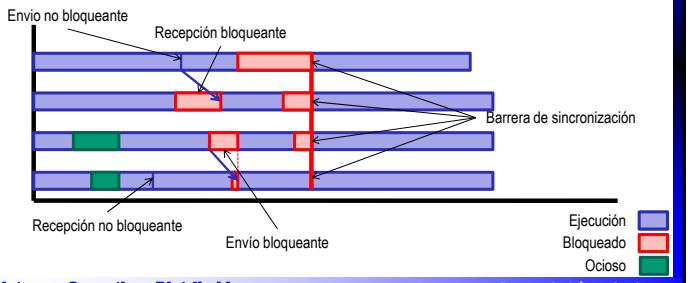
Modelo distribuido

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.

Rendimiento de la Planificación

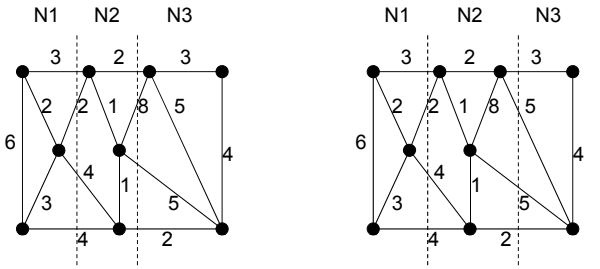
- El rendimiento del esquema de planificación depende:
  - Condiciones de bloqueo (equilibrado de carga)
  - Estado del sistema
  - Eficiencia de las comunicaciones: latencia y ancho de banda



Tareas Paralelas: No-regulares

- En algunos casos las topologías de conexión no son regulares:
  - Modelado como grafo no dirigido donde:
  - Nodo representa un proceso con necesidades de CPU y memoria
  - Arcos incluye etiqueta que indica cantidad de datos que intercambian nodos implicados (tasa de comunicación)
- El problema en su forma general es NP-completo
- En el caso general se utilizan heurísticas:
  - P. ej. corte mínimo: Para P procesadores buscar P-1 cortes tal que se minimice el flujo entre cada partición
  - Resultado: Cada partición (procesador) engloba a un conjunto de procesos "fuertemente acoplados"
  - No tan sencillo en los modelos asíncronos

Tareas Paralelas: No-regulares



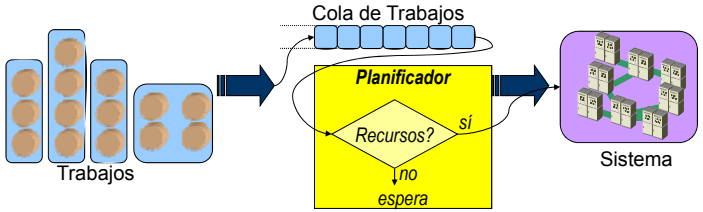
Tráfico entre nodos: 13+17=30

Tráfico entre nodos: 13+15=28

Tanenbaum. "Sistemas Operativos Distribuidos" © Prentice Hall 1996

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.

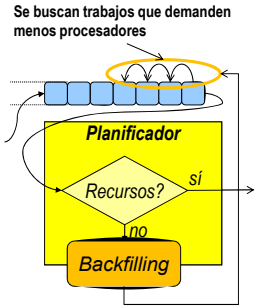


Planificación de Múltiples Trabajos

- ¿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*)

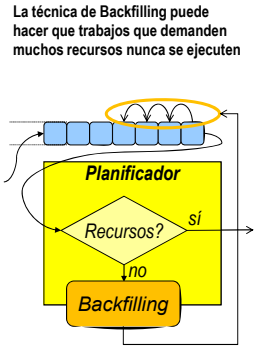
Backfilling

- *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



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.



Sistemas Operativos Distribuidos

Gestión de Procesos

Planificación Dinámica

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

Asignación

- *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 es un nodo inactivo?
  - Estación de trabajo: "lleva varios minutos sin recibir entrada del teclado o ratón y no está ejecutando procesos interactivos"
  - Nodo de proceso: "no ha ejecutado ningún proceso de usuario en un rango de tiempo".
  - Planificación estática: "no se le ha asignado ninguna tarea".
- ¿Qué ocurre cuando deja de estar inactivo?
  - No hacer nada → El nuevo proceso notará mal rendimiento.
  - Migrar el proceso a otro nodo inactivo (costoso)
  - Continuar ejecutando el proceso con prioridad baja.
- Si en lugar de considerar el estado (LS) se necesita conocer la carga (LB) hacen falta métricas específicas.

Políticas de Gestión de Trabajos

Toda la gestión de trabajos se basa en una serie de decisiones (políticas) que hay que definir para una serie de casos:

- **Política de información**: cuándo propagar la información necesaria para la toma de decisiones.
- **Política de transferencia**: decide cuándo transferir.
- **Política de selección**: decide qué proceso transferir.
- **Política de ubicación**: decide a qué nodo transferir.

Política de Información

Cuándo se transmite información sobre el nodo:

- **Bajo demanda**: únicamente cuando desea realizar una transferencia.
- **Periódicas**: se recoge información periódicamente, la información está disponible en el momento de realizar la transferencia (puede estar obsoleta).
- **Bajo cambio de estado**: cada vez que un nodo cambia de estado.

Ámbito de información:

- **Completa**: todos los conocen toda la información del sistema.
- **Parcial**: cada nodo sólo conoce el estado de parte de los nodos del sistema.

Política de Información

¿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).

Politica de Transferencia

- Generalmente, basada en **umbral**:
  - Si en nodo *S* carga > *T* unidades, *S* emisor de tareas
  - Si en nodo *S* carga < *T* unidades, *S* receptor de tareas

- Tipos de transferencias:
  - *Expulsivas*: se pueden transferir tareas ejecutadas parcialmente.
    - Supone: transferir el estado del proceso asociado a la tarea (migración) o reiniciar su ejecución.
  - *No expulsivas*: las tareas ya en ejecución no pueden ser transferidas.

Política de Selección

- Elegir los procesos de tareas nuevas (transferencia no expulsiva).
- Seleccionar los procesos con un tiempo de transferencia mínimo (poco estado, mínimo uso de los recursos locales).
- Seleccionar un proceso si su tiempo de finalización en un nodo remoto es menor que el tiempo de finalización local.
  - El tiempo de finalización remoto debe considerar el tiempo necesario para su migración.
  - El tiempo de finalización puede venir indicado en la descripción del trabajo (estimación del usuario).
  - De no ser así, el sistema debe estimarlo por sí mismo.

Política de Ubicación

- Muestreo: consulta de otros nodos para encontrar adecuado.
- Alternativas:
  - Sin muestreo (se manda a uno aleatoriamente).
  - Muestreo secuencial o paralelo.
  - Selección aleatoria.
  - Nodos más próximos.
  - Enviar un mensaje al resto de nodos (*broadcast*).
  - Basada en información recogida anteriormente.
- Tres tipos de políticas:
  - Iniciadas por el emisor (*Push*) → emisor busca receptores
  - Iniciadas por el receptor (*Pull*) → receptor solicita procesos
  - Simétrica → iniciada por el emisor y/o por el receptor.

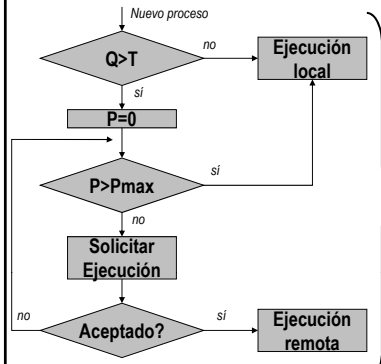
Algoritmos de Equilibrado de Carga

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

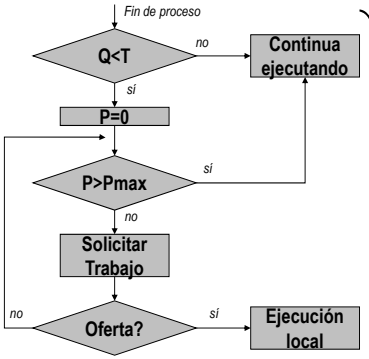


Q: Tamaño de la cola de procesos
T: Umbral máximo de la cola de procesos
Pmax: Número máximo de solicitudes

Política de transferencia: umbral basado en la longitud de la cola de procesos listos.
Política de selección: procesos nuevos.
Política de ubicación. 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



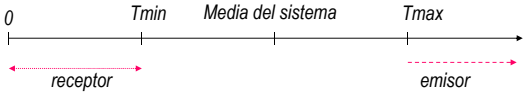
Q: Tamaño de la cola de procesos
T: Umbral máximo de la cola de procesos
Pmax: Número máximo de solicitudes

Política de transferencia: umbral basado en la longitud de la cola de procesos listos.
Política de selección: cualquier proceso.
Política de ubicación: 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

- **Política de transferencia**



- **Política de ubicación** dirigida 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.

Algoritmo Simétrico

- **Política de ubicación** iniciada 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.
- **Política de selección**: cualquier proceso.

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: Recursos

La migración de recursos también depende de la relación de éstos con las máquinas.

- *Unatachment resource*. Ejemplo: un fichero de datos del programa a migrar.
- *Fastened resource*. Son recursos movibles, pero con un alto coste. Ejemplo: una base de datos, un sitio web.
- *Fixed resource*. Están ligados estrechamente a una máquina y no se pueden mover. Ejemplo: dispositivos locales

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.

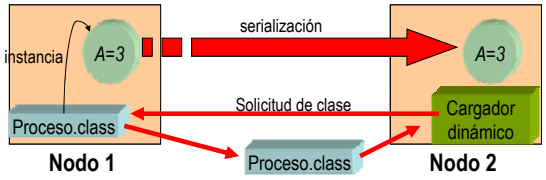
Migración Débil

- 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:

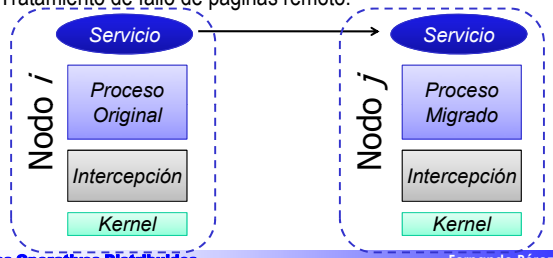
- A nivel de kernel:
  - Versiones modificadas del núcleo
  - Dispone de toda la información del proceso
- A nivel de usuario:
  - Librerias de checkpointing
  - Protocolos para desplazamiento de sockets
  - Intercepción de llamadas al sistema

Otros aspectos:

- PID único de sistema
- Credenciales y aspectos de seguridad

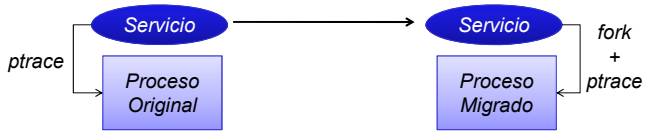
Migración Fuerte

- Desde espacio de usuario:
  - Servicio remoto de solicitud de migración.
  - Interceptores de llamadas al sistema.
    - Auditoría (reconstruir)
    - Redirección
  - Tratamiento de fallo de páginas remoto.



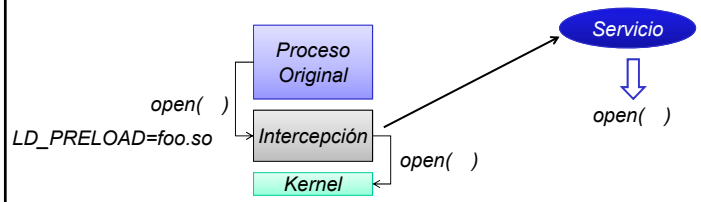
Migración Fuerte

- Servicio de solicitud de migración:
  1. Interactúa con los procesos (vía biblioteca *ptrace*, por ejemplo).
  2. Suspende la ejecución del proceso original
  3. Obtiene datos del proceso (valores de registros o mapa de memoria)
  4. Crea nuevo proceso destino
  5. Modifica valores de los registros
  6. Monitoriza, en términos generales, el estado de los procesos



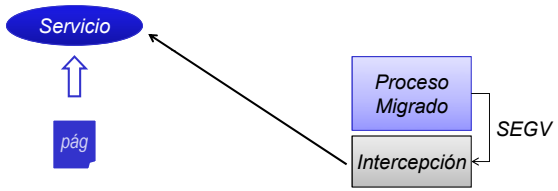
Migración Fuerte

- Interceptores de llamadas al sistema:
  1. Necesarios para conocer el estado externo
  2. Sobrescribir las funciones asociadas a las llamadas al sistema
  3. Despliegue por medio de bibliotecas dinámicas (e.g., LD_PRELOAD)
  4. Se conoce la secuencia de llamadas hechas (se puede repetir)
  5. Se puede redirigir una llamada a otro sitio (vía el servicio).



Migración Fuerte

- Tratamiento de fallo de página remoto:
  1. Acceso a páginas no cargadas da una señal SEGV.
  2. El interceptor captura la señal
  3. Solicita la petición remota a la página en el nodo original
  4. Copia la página



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

Migración Fuerte

- Migración de conexiones:
  - Afecta a las conexiones entrantes y salientes.
  - Solución por medio de proxies:
    - En realidad todas las direcciones son de un único servidor que redirige al nodo donde está el proceso en cada momento.
    - El problema es que no es escalable.
  - Solución a nivel IP:
    - Se asigna una dirección IP por proceso del sistema (vía DHCP)
    - Paquetes "Gratuitous ARP" para informar de cambio de MAC al migrar el proceso.
    - El problema es el rango de direcciones disponibles (red privada virtual).

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).

- 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).

Migración de Datos

- Mejoras de rendimiento:
  - Equilibrado de carga:
    - **Solución**: Definir muchos paquetes de trabajo e ir despachándolos bajo demanda (finalización del anterior).
  - Intentar lanzar rápido los trabajos iniciales:
    - **Solución**: Uso de broadcast en el envío de paquetes inicial.
  - Si todos los nodos son homogéneos y realizan el mismo trabajo finalizarán casi a la vez:
    - Plantea un posible problema de saturación del maestro que deriva en un retraso en despachar nuevos paquetes de trabajo.
    - **Solución**: Asignación inicial en distintos tamaños.

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

Planificación Dinámica vs. Estática

- Los sistemas pueden usar indistintamente una, otro o las dos.

	Estática	*No Estática*
Dinámica	***Planificación adaptativa**: Se mantiene un control central sobre los trabajos lanzados al sistema, pero se dispone de los mecanismos necesarios para reaccionar a errores en las estimaciones o reaccionar ante problemas.*	***Estrategias de equilibrado de carga**: Los trabajos se arrancan libremente en los nodos del sistema y por detrás un servicio de equilibrado de carga/estado ajusta la distribución de tareas a los nodos.*
No Dinámica	***Gestor de recursos** (con asignación batch): Los procesadores se asignan de forma exclusiva y el gestor de recursos mantiene información sobre los recursos comprometidos.*	***Cluster de máquinas sin planificación alguna**: Que sea lo que Dios quiera…*