## 4.3. ASIGNACIÓN DE PROCESADORES

Por definición, un sistema distribuido consta de varios procesadores. Estos se pueden organizar como colección de estaciones de trabajo personales, una pila pública de procesadores o alguna forma híbrida. En todos los casos, se necesita cierto algoritmo para decidir cuál proceso hay que ejecutar y en qué máquina. Para el modelo de estaciones de trabajo, la pregunta es cuándo ejecutar el proceso de manera local y cuándo buscar una estación inactiva. Para el modelo de la pila de procesadores, hay que tomar una decisión por cada nuevo proceso.

En cuarto lugar, cada maquina puede tener un sistema de archivos auto contenido, con la posibilidad de montarlo o tener su sistema de archivos de otras maquinas. La idea aquí es que cada maquina esta auto contenida en lo fundamental y que el contacto con el mundo exterior sea limitado. Este sistema proporciona un tiempo de respuesta uniforme y garantizado para el usuario y pone poca carga en la red.

### Uso de estaciones de trabajo inactivas

Plantea el problema de encontrar estaciones de trabajo inactivas en la red que puedan ejecutar procesos. Por lo cual las estaciones de trabajo deben de anunciar cuando no cuentan con una carga de trabajo asignada, así todas las demás estaciones toman nota de esto y lo registran.
Ya sea que existan muchos o pocos registros, existe un peligro potencial de que aparezcan condiciones de competencia si dos usuarios llaman al mismo tiempo al comando remote y ambos descubren que la misma maquina esta inactiva, ambos intentaran iniciar procesos al mismo tiempo. Para detectar y evitar esta situación, el programa remote verifica la estación de trabajo inactiva, la cual si continua libre se elimina así misma del registro y da la señal de continuar, de esta manera quien hizo la llamada puede enviar su ambiente e iniciar el proceso remoto.

### Modelo de pila de procesadores

Este método consiste en construir una pila de procesadores, repleta de CPU, en un cuarto de maquinas, los cuales se pueden asignar de manera dinámica a los usuarios según la demanda.

Desde el punto de vista conceptual este método es mas parecido al tiempo compartido tradicional que al modelo de la computadora personal aunque se construye con la tecnología moderna. La motivación para la idea de la pila de procesadores proviene de dar un paso mas adelante en la idea de las estaciones de trabajo sin disco. Si el sistema de archivos se debe concentrar en un pequeño numero de servidores de archivos para mayor economía, debe ser posible hacer lo mismo con los servidores de computo, es decir si colocamos todos los CPU en un gabinete de gran tamaño en el cuarto de maquinas se pueden reducir los costos de suministro de energía y de empaquetamiento, lo cual produce un mayor poder de computo con una cantidad fija de dinero.
De hecho convertimos todo el poder de computo en “estaciones de trabajo inactivas” a las que se puede tener acceso de manera dinámica. Los usuarios obtienen tantos CPU como sea necesario, durante periodos cortos, después de lo cual regresan a la pila, de modo que otros usuarios puedan disponer de ellos.
El principal argumento para la centralización del poder de computo como pila de procesadores proviene de la teoría de colas. Un sistema de colas es una situación donde

los usuarios generan en forma aleatoria solicitudes de trabajo a un servidor. Cuando el servidor esta ocupado, los usuarios se forman por el servicio y se procesan según su turno.

**Un modelo híbrido**

Se puede establecer una mediación al proporcionar a cada usuario una estación de trabajo personal y además tener una pila de procesadores. Aunque esta solución es más cara que cualquiera de los dos modelos puros, combina las ventajas de arribos. El trabajo interactivo se puede llevar a cabo en las estaciones de trabajo, con una respuesta garantizada. Sin embargo, las estaciones inactivas no se utilizan, lo cual hace más sencillo el diseño del sistema. Sólo se dejan sin utilizar. En vez de esto, todos los no interactivos se ejecutan en la pila de procesadores, así como todo el cómputo pesado en general. Este modelo proporciona una respuesta interactiva más rápida, un uso eficiente de los recursos y un diseño sencillo.

**Administración de Procesadores**

Se genera un nuevo trabajo cuando un proceso en ejecución decide realizar una bifurcación o crear un subproceso. En ciertos casos, el proceso que se bifurca es el intérprete de comandos (shell) que inicia un nuevo trabajo en respuesta a un comando del usuario. En otros, el propio proceso usuario crea uno o más hijos; por ejemplo, para tener mejor desempeño con la ejecución en paralelo de todos los hijos.

Las estrategias de asignación de procesadores se pueden dividir en dos categorías amplias. En la primera, que llamaremos no migratoria, al crearse un proceso, se toma una decisión acerca de dónde colocarlo. Una vez colocado en una máquina, el proceso permanece ahí hasta que termina. No se puede mover, no importa lo sobrecargada que esté la máquina ni que existan muchas otras máquinas inactivas. Por el contrario, con los algoritmos migratorios, un proceso se puede trasladar aunque haya iniciado su ejecución. Mientras que las estrategias migratorias permiten un mejor balance de la carga, son más complejas y tienen un efecto fundamental en el diseño del sistema.

Un algoritmo que asigne procesos a los procesadores lleva implícito el intento por optimizar algo. Si éste no fuera el caso, sólo haríamos la asignación en forma aleatoria o en orden numérico. Sin embargo, lo que hay que optimizar varía de un sistema a otro. Un posible objetivo podría ser maximizar el uso de los CPU; es decir, el número de ciclos de CPU que se ejecutan en beneficio de los trabajos del usuario, por cada hora de tiempo real.

La maximización del uso del CPU es otra forma de decir que hay que evitar a todo costo el tiempo inactivo del CPU. Cuando exista la duda, hay que garantizar que cada CPU tenga algo que hacer. Otro objetivo importante es la minimización del tiempo promedio de respuesta. Consideremos, por ejemplo, los dos procesos y procesadores de la figura 4-15. El procesador 1 ejecuta 10 MIPS; el procesador 2 ejecuta 100 MIPS, pero tiene una lista de espera de procesos atrasados, la cual tardará 5 segundos en terminar. El proceso A tiene 100 millones de instrucciones y el proceso B tiene 300 millones. En la figura se muestran los tiempos de respuesta para cada proceso en cada procesador (tiempo de espera incluido). Si asignamos el proceso A al procesador 1 y B al procesador 2, el tiempo promedio de respuesta será de (10 + 8)/2 = 9 segundos. Si los asignamos al revés, el tiempo promedio de respuesta será de (30 + 6)12 18 segundos. Es

claro que la primera asignación es mejor, en términos de minimizar el tiempo promedio de respuesta.

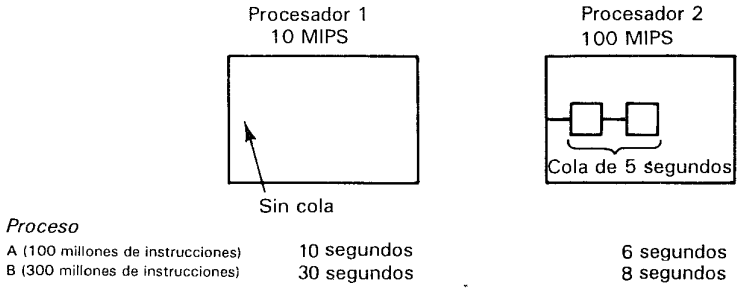


**Figura 4-15.** Tiempos de respuesta de dos procesos en dos procesadores.

Una variante de la minimización del tiempo de respuesta es la minimización de la tasa de respuesta, la cual se define como la cantidad de tiempo necesaria para ejecutar un proceso en cierta máquina, dividido entre el tiempo que tardaría en ejecutarse en cierto procesador de referencia, no cargado. Para muchos usuarios, la tasa de respuesta es una métrica más útil que el tiempo de respuesta, puesto que toma en cuenta el hecho de que los trabajos de gran tamaño tardan más que los pequeños. Para ver esto, ¿cuál es mejor, un trabajo de 1 segundo que tarda 5 segundos o un trabajo de 1 minuto que tarda 70 segundos? Mediante el tiempo de respuesta, el primero es mejor, pero con la tasa de respuesta, el segundo es mucho mejor, puesto que 5/1 » 70/60.

**Aspectos del diseño de algoritmos de asignación de procesadores**

Con el paso de los años, se han propuesto un gran número de algoritmos para la asignación de procesadores.
Las principales decisiones que deben tomar los diseñadores se pueden resumir en cinco aspectos:
1. Algoritmos deterministas Vs. Heurísticos.
2. Algoritmos centralizados Vs. Distribuidos.
3. Algoritmos óptimos vs. subóptimos.
4. Algoritmos locales vs. Globales.
5. Algoritmos iniciados por el emisor vs. Iniciados por el receptor.

También hay que tomar otras decisiones, pero éstas son las principales. Analizaremos cada una de ellas en su turno. Los algoritmos deterministas son adecuados cuando se sabe de antemano todo acerca del comportamiento de los procesos. En pocos, si no es que en ninguno de los sistemas, se tiene un conocimiento total de antemano, pero a veces se puede obtener una aproximación razonable. En el otro extremo están los sistemas donde la carga es por completo impredecible. Las solicitudes de trabajo dependen de quién esté haciendo qué, y puede variar de manera drástica cada hora, e incluso cada minuto. La asignación de procesadores en tales sistemas no se puede hacer de manera determinista o matemática, sino que por necesidad utiliza técnicas adhoc llamadas heurísticas.
El segundo aspecto del diseño es centralizado vs. distribuido. La recolección de toda la información en un lugar permite tomar una mejor decisión, pero menos robusta y coloca una carga pesada en la máquina central. Son preferibles los algoritmos descentralizados, pero se han propuesto algunos algoritmos centralizados por la carencia de alternativas descentralizadas adecuadas.

El tercer aspecto está relacionado con los dos anteriores: ¿Intentamos encontrar la mejor asignación, o sólo una que sea aceptable? Se pueden obtener las soluciones óptimas tanto en los sistemas centralizados como en los descentralizados, pero por regla son más caros que los subóptimos. Hay que recolectar más información y procesarla un poco más. En la práctica, la mayoría de los sistemas distribuidos reales buscan soluciones subóptimas, heurísticas y distribuidas, debido a la dificultad para obtener las óptimas.

El cuarto aspecto se relaciona con lo que se llama a menudo política de transferencia. Cuando se está a punto de crear un proceso, hay que tomar una decisión para ver si se ejecuta o no en la máquina que lo genera. Si esa máquina está muy ocupada, hay que transferir a otro lugar al nuevo proceso. La opción en este aspecto consiste en basar o no la decisión de transferencia por completo en la información local. Cada una de estas opciones tiene sus puntos a favor. Los algoritmos locales son sencillos, pero están muy lejos de ser los óptimos, mientras que los globales sólo dan un resultado poco mejor a un costo mayor. Más relacionados con los detalles de la implantación real de los algoritmos para la asignación de procesadores que con los grandes principios detrás de ellos.

Para comenzar, casi todos los algoritmos suponen que las máquinas conocen su carga, de modo que pueden decir si están subcargados o sobrecargados y pueden informar a las demás máquinas de su estado. La medición de la carga no es tan sencilla como parece. Un método consiste en contar el número de procesos en cada máquina y utilizar ese número como la carga. Sin embargo, como ya hemos señalado antes, incluso en un sistema inactivo pueden ejecutarse muchos procesos, como los demonios de correo o noticias, administra dores de ventanas y otros. Así, el contador del proceso casi no dice nada de la carga actual.

El siguiente paso consiste en contar sólo los procesos en ejecución o listos. Después de todo, cada proceso en ejecución o que se pueda ejecutar impone cierta carga a la máquina, incluso aunque sea un proceso secundario. Sin embargo, muchos de estos demonios despiertan de forma periódica, verifican si ocurre algo interesante y, en caso contrario, vuelven a dormir. La mayoría sólo pone una pequeña carga en el sistema.

Una medida más directa, aunque requiere de un mayor trabajo para su registro, es la fracción de tiempo que el CPU está ocupado. Es claro que una máquina con 20% de uso del CPU tiene una carga mayor que la de una máquina con 10% de uso del CPU, sin importar si ejecuta programas del usuario o demonios. Una forma de medir el uso del CPU es con figurar un cronómetro y dejarlo que interrumpa a la máquina en forma periódica. En cada interrupción, se observa el estado del CPU. De esta forma, se puede observar la fracción de tiempo que se gasta en el ciclo inactivo.

Un problema con las interrupciones por medio de cronómetros es que cuando el núcleo ejecuta un código crítico, éste desactiva por lo general todas las interrupciones, entre las cuales se encuentra la interrupción del cronómetro. Así, si el tiempo del cronómetro se termina mientras el núcleo está activo, la interrupción se retrasa hasta que el núcleo termina. Si el núcleo estaba en e! proceso de bloquear los últimos procesos activos, el tiempo del cronómetro no se agotará sino hasta que el núcleo termine (y entre al ciclo inactivo). Este efecto tiende a subestimar el verdadero uso del CPU.

Otro aspecto de la implantación es el enfrentamiento con el costo excesivo. Muchos de los algoritmos teóricos para la asignación de procesos ignoran el costo de recolectar medidas y desplazar los procesos de aquí para allá. Si un algoritmo descubre que el traslado de un proceso recién creado a una máquina distante puede mejorar el desempeño del sistema en un 10%, tal vez sería mejor no hacer nada, puesto que el costo del traslado del proceso puede engullirse todo el beneficio. Un algoritmo adecuado tomaría en cuenta el tiempo de CPU, uso de memoria y el ancho de banda de

la red utilizada por el propio algoritmo para asignación de procesadores. Pocos lo hacen, lo cual se debe principalmente a que no es fácil.
Nuestra siguiente consideración en torno a la implantación es la complejidad. Casi todos los investigadores miden la calidad de sus algoritmos mediante el análisis de datos analíticos, simulados o experimentales del uso del CPU y de la red, así como el tiempo de respuesta. Pocas veces se considera también la complejidad del software en cuestión, a pesar de las obvias implicaciones para el desempeño, precisión y consistencia del sistema.

Él ultimo aspecto de nuestra lista trata de la política de localización. Una vez que la política de transferencia ha decidido deshacerse de un proceso, la política de localización debe decidir dónde enviarlo. Es claro que esta política no puede ser local. Necesita información de la carga en todas partes para tornar una decisión inteligente. Sin embargo, esta información se puede dispersar de dos maneras. En uno de los métodos, los emisores inician el intercambio de información. En el otro, el receptor toma la iniciativa.
Corno ejemplo sencillo, observemos la figura 4-16(a). En ésta, una máquina sobrecargada envía una solicitud de ayuda a las demás máquinas, con la esperanza de que descarguen el nuevo proceso en alguna otra máquina. En este ejemplo, el emisor toma la iniciativa para localizar más ciclos del CPU. Por el contrario, en la figura 4-16(b), una máquina inactiva o subcargada anuncia a las demás que tiene poco trabajo y está preparada para más. Su objetivo es localizar una máquina dispuesta a darle trabajo.

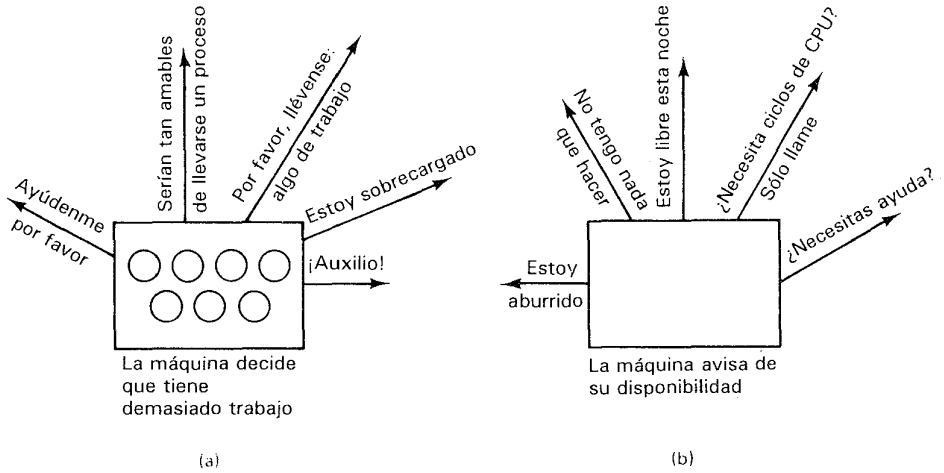


**Figura 4-16.** (a) Un emisor en búsqueda de una máquina inactiva. (b) Un receptor en búsqueda de trabajo por realizar.

Para ambos casos, el de los procesos iniciados por el emisor o por el receptor, los distintos algoritmos tienen estrategias diversas para decidir a quién examinar, el tiempo que durará dicho examen y qué hacer con los resultados. Sin embargo, en estos momentos debe quedar clara la diferencia entre ambos enfoques.

### Ejemplo de algoritmos de asignación de procesadores

#### ***Un algoritmo determinista según la teoría de gráficas***

Una clase de algoritmos analizada con amplitud es para los sistemas que constan de procesos con requerimientos conocidos de CPU y memoria, además de una matriz

conocida con el tráfico promedio entre cada pareja de procesos. Si el número k de CPU es menor que el número de procesos, habrá que asignar varios procesos al mismo CPU. La idea es llevar a cabo esta asignación de forma que se minimice el tráfico en la red.
El sistema se puede representar como gráfica con pesos, donde cada nodo es un proceso y cada arco representa el flujo de mensajes entre dos procesos. Desde el punto de vista matemático, el problema se reduce entonces a encontrar una forma de partir (es decir, cortar) la gráfica en dos subgráficas ajenas, sujetas a ciertas restricciones (por ejemplo, el total de requerimientos de CPU y memoria deberá estar por debajo de ciertos límites para cada subgráfica). Para cada solución que cumpla las restricciones, los arcos contenidos por completo dentro de una subgráfica representan la comunicación entre las máquinas y se puede ignorar. Los arcos que van de una subgráfica a la otra representan el tráfico en la red. El objetivo es entonces encontrar una partición que minimice el tráfico en la red, a la vez que satisfaga todas las restricciones. La figura 4-17 muestra dos formas de partir la misma gráfica, lo cual produce dos cargas distintas en la red.

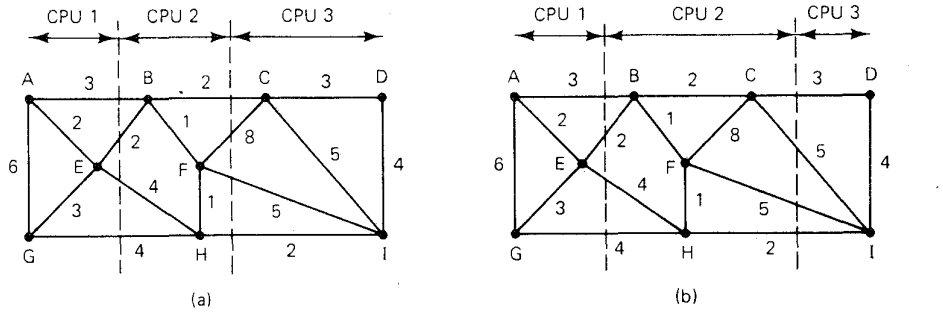


**Figura 4-17.** Dos formas de asignar 9 procesos a 3 procesadores.

En la figura 4-17(a) hemos partido la gráfica con los procesos A, E y G en un procesador; los procesos B, F y H en un segundo; los procesos C, D en el tercero. El tráfico total en la red es la suma de los arcos intersecados por las líneas punteadas, que en este caso es 30 unidades. En la figura 4-17(b) tenemos una partición distinta, la cual sólo tiene 28 unidades de tráfico en la red. Si suponemos que cumple con todas las restricciones de memoria y CPU, ésta es una mejor opción puesto que utiliza menor comunicación.
De manera intuitiva, lo que hacemos es buscar unidades de asignación fuertemente acopladas (flujo intenso de tráfico dentro de las unidades de asignación), pero que interactúen poco con las demás unidades (poco flujo de tráfico entre las unidades).

## ***Un algoritmo centralizado***

Los algoritmos de teoría de gráficas del tipo analizado tienen poca aplicabilidad, puesto que necesitan información completa de antemano, por lo que veremos ahora un algoritmo heurístico que no necesita dicha información. Este algoritmo, llamado arriba-abajo (Mutka y Livny, 1987) es centralizado, en el sentido de que un coordinador mantiene una tabla de uso, con una entrada por cada estación de trabajo personal (es decir, por usuario), con un valor inicial de 0. Cuando ocurren eventos significativos, se pueden enviar mensajes al coordinador para actualizar la tabla. Las decisiones de asignación se basan en esta tabla. Estas decisiones se tornan cuando ocurren eventos de planificación: se realiza una solicitud, se libera un procesador, o bien el reloj hace una marca de tiempo. La parte poco común de este algoritmo y la razón de ser centralizado es que en vez de intentar maximizar el uso del CPU, se preocupa por darle a cada poseedor de una estación de trabajo una parte justa del poder de cómputo. Mientras que

otros algoritmos pueden otorgarle todas las máquinas a un usuario, con la única condición de que las mantenga ocupadas (es decir, lograr un alto uso del CPU), este algoritmo está diseñado para evitar eso precisamente.

Cuando se va a crear un proceso y la máquina donde se crea decide que el proceso se debe ejecutar en otra parte, le pide al coordinador de la tabla de usos que le asigne un procesador. Si existe uno disponible y nadie más lo desea, se otorga el permiso. Si no existen procesadores libres, la solicitud se niega por el momento y se toma nota de ella.

Cuando el poseedor de una estación de trabajo ejecuta procesos en las máquinas de otras personas, acumula puntos de penalización, un número fijo por cada segundo. Estos puntos se añaden a su entrada en la tabla de usos. Cuando tiene solicitudes pendientes no satisfechas, los puntos de penalización se restan de su entrada en la tabla de usos. Si no existen solicitudes pendientes y ningún procesador está en uso. La entrada de la tabla de usos se desplaza un cierto número de puntos hacia el cero, hasta que llega ahí. De esta forma, su puntuación se mueve hacia arriba o hacia abajo; de ahí el nombre del algoritmo.

Las entradas de la tabla de usos pueden ser positivas, cero o negativas. Una puntuación positiva indica que la estación de trabajo es un usuario de los recursos del sistema, mientras que uno negativo significa que necesita recursos. Una puntuación O es neutra.

Podemos dar ahora la heurística utilizada para la asignación de procesadores. Cuando un procesador se libera, gana la solicitud pendiente cuyo poseedor tiene la puntuación más baja. En consecuencia, un usuario que no ocupe procesadores y que tenga pendiente una solicitud durante mucho tiempo siempre vencerá a alguien que utilice muchos procesadores. Esta propiedad es la intención del algoritmo: asignar la capacidad de manera justa. En la práctica, esto quiere decir que si un usuario tiene carga justa y continua en el sistema, pero otro usuario llega y desea iniciar un proceso, el usuario ligero será favorecido, por encima del usuario pesado. Estudios de simulación (Mutka y Livny, 1987) muestran que el algoritmo funciona como se esperaba bajo una variedad de condiciones de carga.