

Sistemas distribuidos

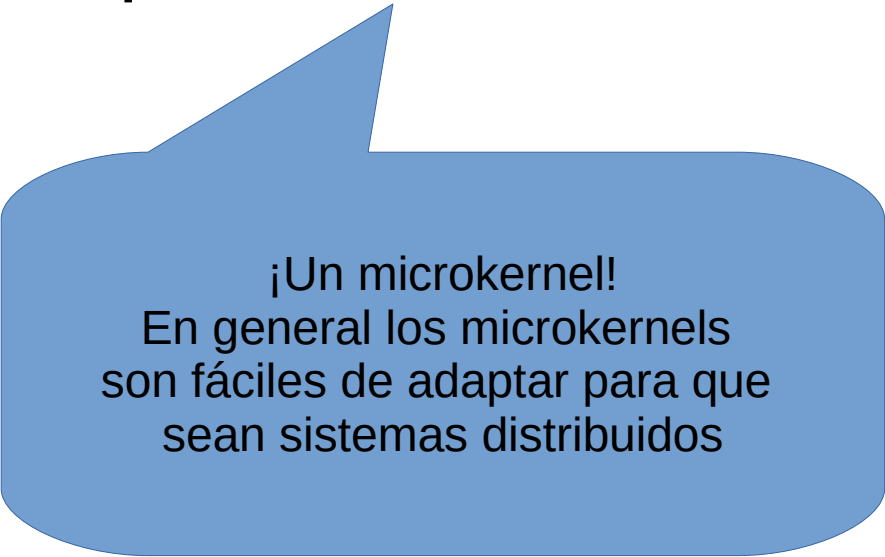
¿qué es un sistema distribuido?

Un sistema que utiliza más de una CPU para realizar su tarea (y una red de comunicación)

Implicación importante: no hay memoria compartida

Sistema distribuido VS Sistema Operativo Distribuido

- Un sistema operativo distribuido es un sistema distribuido que realiza las tareas de un sistema operativo (abstracciones, gestión de recursos, etc...)
- Ejemplos: Proyecto para Minix distribuido, Plan 9, Inferno



¡Un microkernel!
En general los microkernels
son fáciles de adaptar para que
sean sistemas distribuidos

S. Distribuidos en general: ventajas

- **Costo/beneficio**: ¿cómo duplicar el poder de cómputo de una CPU? Conseguir una CPU el doble de rápida y con el doble de RAM puede ser muy costoso. Agregar una CPU igual es barato.
- Aplicaciones **inherentemente distribuidas**: juegos online, reserva de pasajes, ...
- **Alta disponibilidad**: replicando recursos puede lograrse que el sistema siga funcionando incluso si algún recurso falla.
- **Crecimiento incremental**: si un sistema crece más de lo esperado y se necesitan más recursos se pueden agregar más nodos al sistema.

S. Distribuidos en general: desventajas

Software complejo: habiendo varios nodos realizando procesamiento hay problemas de concurrencia, hay que manejar replicaciones, fallos de red y de nodos, no se puede confiar en un reloj global, no hay memoria compartida, la seguridad se vuelve más compleja, etc...

S. Distribuidos en general: transparencia

- En general son deseables algunos de estos tipos de transparencia:
 - De ubicación: el usuario no puede saber dónde están los recursos.
 - De migración: los recursos pueden moverse sin afectar su nombre
 - De replicación: no se puede saber cuántas copias hay
 - De concurrencia: los recursos se pueden utilizar si necesidad de coordinar con otros usuarios
 - De paralelismo: una tarea puede resolverse en forma paralela utilizando múltiples nodos

Otra característica deseable

- No debería haber un “servidor” ejecutando en un único nodo que centralice una tarea, pues sería un **único punto de falla** y si ese nodo falla todo el sistema falla. Además se puede convertir en un cuello de botella.
- Se priorizan **algoritmos distribuidos**
- Si no se consigue un algoritmo distribuido tiene que haber alguna forma de **elegir un reemplazo** si el “servidor” se cae (algoritmos de elección de líder)

Ejemplo: algoritmo centralizado en un SO Distribuido para exclusión mutua

- Existe un proceso o nodo coordinador.
- Cuando un proceso necesita ingresar a una sección crítica envía un mensaje al coordinador: Soy $\langle n \rangle$ quiero entrar a la sección crítica $\langle m \rangle$
- El coordinador responde OK si no hay otro proceso en la sección crítica
- Cuando el proceso termina de usar la sección crítica le avisa al coordinador

Ejemplo: algoritmo centralizado en un SO Distribuido para exclusión mutua

- ¿qué pasa si llega un pedido para un recurso actualmente en uso? El coordinador puede:
 - 1) Responder “denied” (el proceso deberá reintentar)
 - 2) Demorar la respuesta hasta que se libere el recurso
- Funciona si no falla la red ni los nodos
- Problemas:
 - Único punto de falla
 - Cuello de botella
 - Si se usa “2)” y el coordinador se cae un proceso que está esperando un recurso quedará esperando por siempre

Ejemplo: algoritmo distribuido en un SO Distribuido para exclusión mutua

- Para solicitar acceso a una sección crítica un proceso envía la tupla $\langle n, r, t \rangle$ a todos los otros procesos
 - $n \Rightarrow$ identificador de este proceso
 - $r \Rightarrow$ recurso que quiero acceder
 - $t \Rightarrow$ hora actual
- Debe esperar el OK del resto de los procesos

Ejemplo: algoritmo distribuido en un SO Distribuido para exclusión mutua

- Cuando un proceso recibe $\langle n, r, t_1 \rangle$:
 - Si no está en “r” ni necesita por ahora entrar responde OK
 - Si está en “r” demora la respuesta
 - Si quiere entrar (i.e.: ya mandó al menos un mensaje indicando $\langle m, r, t_2 \rangle$) el pedido con el menor “t” gana: si $t_1 < t_2$ responde OK (si no demora la respuesta).

Ejemplo: algoritmo distribuído en un SO Distribuído para exclusión mutua

- Ventaja: no hay un coordinador
- Desventajas:
 - ¡¡ahora hay “n” puntos de falla (n=número de procesos, si un nodo falla el algoritmo falla)!!
 - Número de mensajes: $2*(n-1)$ para ingresar a una sección crítica . Se puede mejorar por ejemplo usando mayoría simple en vez de mayoría absoluta.



En este caso conviene usar un algoritmo centralizado