Repaso primer parcial

2019/02/16 - 15:04

Definición de un Sistema Distribuido:

- **Conjunto de programas**
  - → se ejecutan en procesos diferentes
    - ⇒ en la misma o diferentes máquinas
      - en el mismo sitio o interconectadas
  - → se comunican entre sí
  - → ***contribuyen a la ejecución de una tarea común***
- **Conjunto de computadoras autónomas**
  - → da la ilusión de ser un sistema homogéneo
  - → ligadas por una infraestructura de comunicación
  - → explota los recursos disponibles (locales o remotos)
  - → ***contribuyen a la ejecución de una tarea común***

Definición de *sistema*:
Entidad compuesta de elementos que interactúan para realizar una función en un ambiente dado.

Un **sistema informático** puede ofrecer las siguientes funciones:
- Transformación de interfaz
  Ofrecer una máquina virtual que oculte la complejidad del hardware subyacente
- Gestión de recursos físicos y lógicos
- Servicios genéricos y herramientas para la construcción de servicios (middleware)
- Garantía de *calidad de servicio*

Un sistema:
⇒ *analiza* la **interacción con otros sistemas**, qué puede aprovechar y qué puede requerir de recursos externos para la solución de un problema
⇒ *analiza* su **arquitectura y composición** administrando los módulos de manera que sean aprovechados
⇒ *se interesa* por la **evaluación**, es decir, qué tan efectivo está siendo el software, cómo puede mejorar la eficiencia para no perder a su clientela
⇒ *tiene que procurar* **evolucionar y adaptarse** a juzgar por las condiciones ambientales

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Multiprocesadores vs. multicomputadoras

/	**Multiprocesadores**	**Multicomputadoras**
Memoria	Compartida	Independiente
Sistema operativo	SO Distribuido	SO de Red/Basado en Middleware
Aplicaciones	Paralelas	Distribuidas

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Hardware

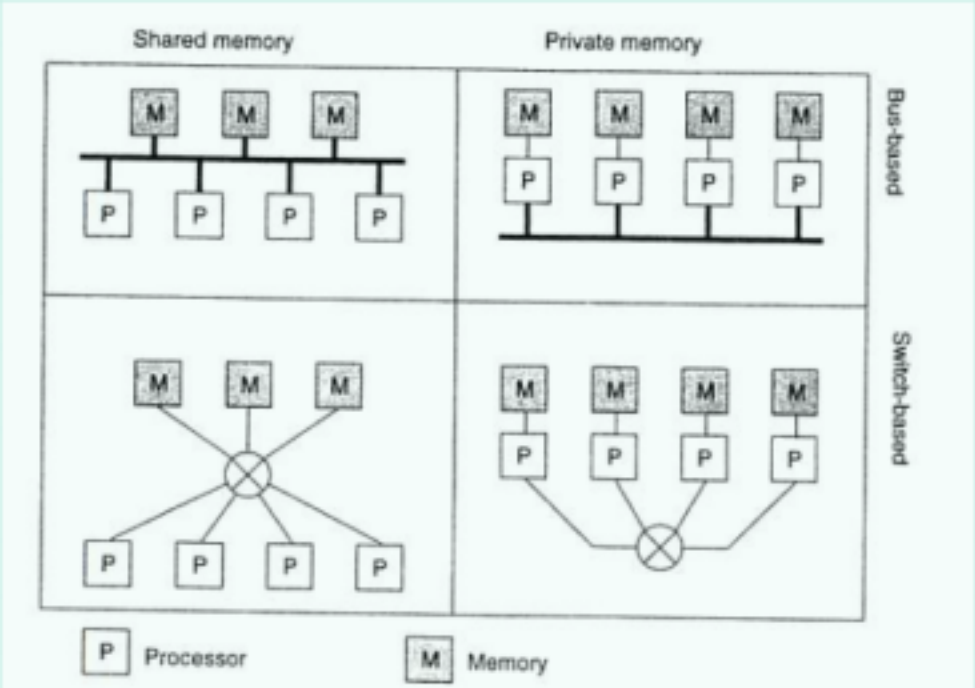
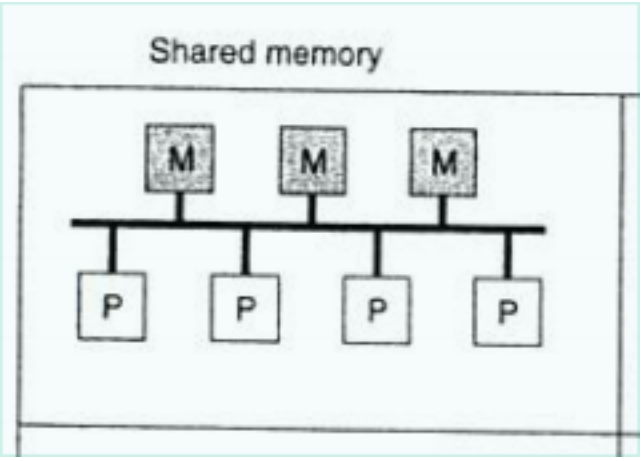


Figure 1-6. Different basic organizations of processors and memories in distributed computer systems.

La categoría de interconexión de red por **Bus** significa que hay una sola red o medio por el cual todas las máquinas se conectan
La categoría de **Switch**, por su parte, consiste en cables individuales máquina-máquina con muchos diferentes patrones de cableado. El mensaje se mueve entre los cables con una explícita decisión de *switcheo* a cada paso de la ruta que el mensaje debe de seguir

Multiprocesadores



Provee un sistema coherente y extremadamente veloz, sin embargo, pierde rend
son agregados

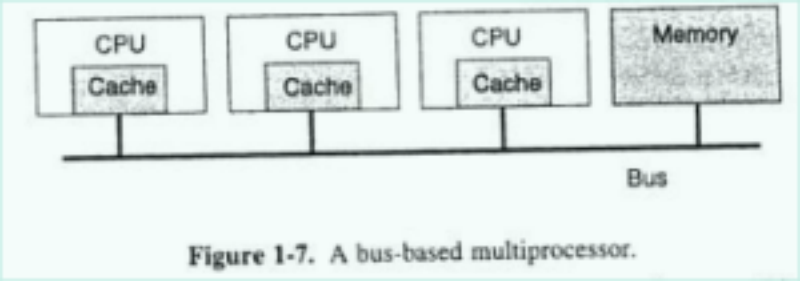


Figure 1-7. A bus-based multiprocessor.

Este esquema es una inmediata solución al problema anterior ya q
determinado CPU no es necesaria una llamada al Bus, además de que si el *hit-rate* (posibilidad de que la información buscada esté en el caché) es lo suficientemente alto, el tráfico del bus será disminuido radicalmente. No obstante, tiene su propio problema: se vuelve incoherente si dos procesadores deciden escribir en dos instancias distintas de la misma información.

En general, el problema con los multiprocesadores basados en bus es su limitada **escalabilidad**.

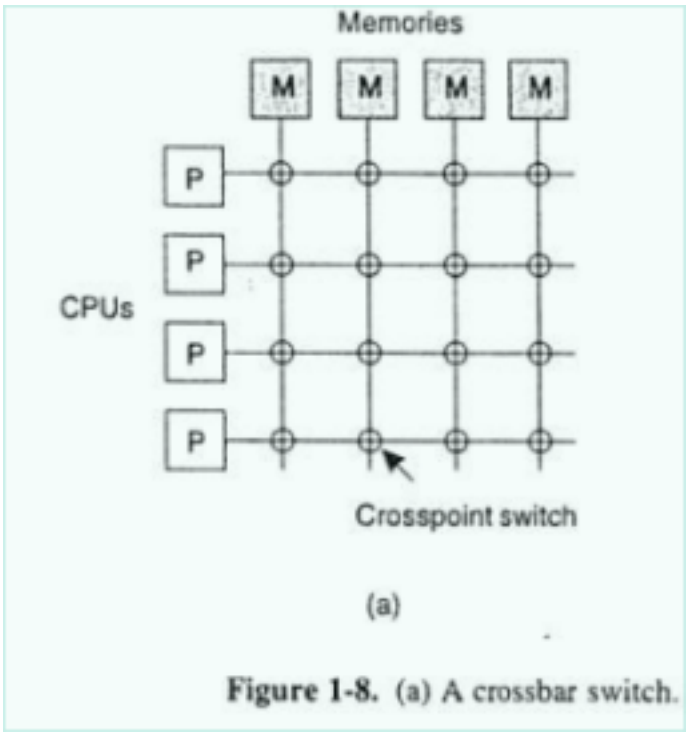
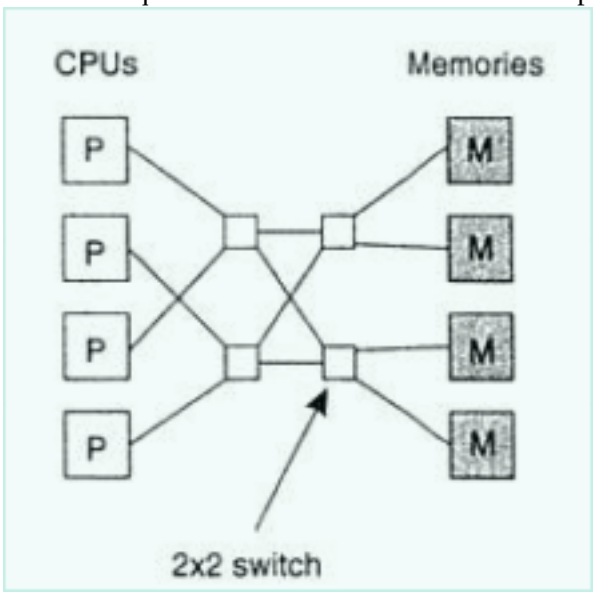


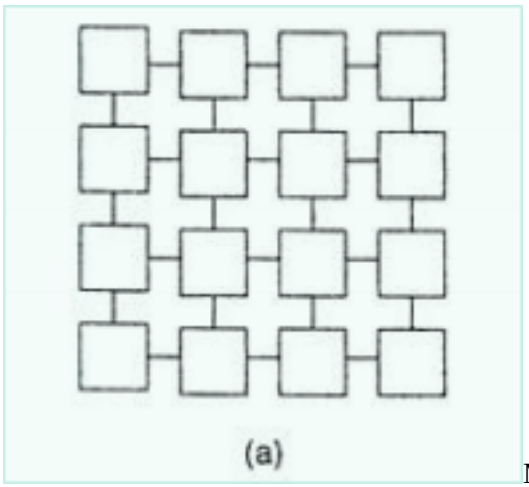
Figure 1-8. (a) A crossbar switch.

Una posibilidad para obrepasar esa limitada escalabilidad es implementar u velocidad de conexión entre CPU-memoria, pero su desventaja es que para cierto número de CPUs y memorias el número de switches requeridos es demasiado elevado como para ser asequible

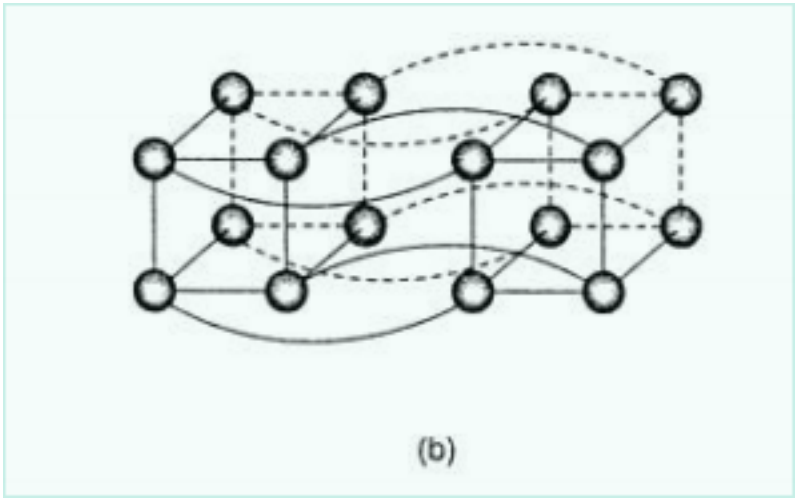


Como solución se sugiere un modelo Omega que utiliza menos switches que tiene n cantidad de switches requerida, mas debido a la posibilidad de que varios saltos sean requeridos, es posible que la latencia entre CPU-Memoria sea alta

Multicomputadoras:
Las topologías más comunes de interconexión de multicomputadoras son:



Mallas: Fáciles de entender e implementar. Útiles para problemas de naturaleza inherente d



Hipercubo: No hay realmente una descripción de su utilidad

Procesadores masivamente paralelos:
• Cientos de CPUs
• Red de interconexión de alto rendimiento
• Tolerancia a fallas
Clusters de estaciones de trabajo:
• PCs estándares
• Redes estándares
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Software

Sistemas Operativos Distribuidos (DOS)
Sistema operativo **estrechamente acoplado**
Objetivo principal: *esconder y administrar recursos de hardware*
Utilizado para:
→ Multiprocesadores
→ Multicomputadoras homogeneas
▪ No es posible simplemente colocar los archivos en la memoria física compartida
▪ La única manera de comunicación es a través de envío demensajes

Sistemas Operativos de Redes (NOS)
Sistema operativo **holgadamente acoplado**
Objetivo principal: *Ofrecer servicios locales a clientes remotos*
Utilizado para:
→ Multicomputadoras heterogéneas
▪ No asume que las computadoras son homogéneas
▪ Cada computadora tiene su propio SO pero todas están conectadas a una red
▪ Un usuario puede iniciar una sesión en una computadora distinta
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Middleware
Sistema operativo que trata de combinar lo mejor de los dos anteriores
Objetivo principal: ofrecer transparencia en la distribución
Utilizado para:
→ Cualquier sistema que soporte NOS

- Busca brindar la escalabilidad y apertura de los NOS y la transparencia y facilidad de uso de los DOS
- Consiste en una capa de software sobre un NOS
- Esconde la heterogeneidad de las plataformas
- Mejora la distribución de transparencia
- Asume una postura ligeramente menos estricta a la que UNIX propone: todo es manejado como archivos
- Utiliza RPC (Llamadas Remotas Procedurales)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Comparacion entre sistemas operativos

Item	Distributed OS		Network OS	Middleware-based DS
	Multiproc.	Multicomp.		
Degree of transparency	Very high	High	Low	High
Same OS on all nodes?	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

Figure 1-24. A comparison between multiprocessor operating systems, multicomputer operating systems, network operating systems, and middleware-based distributed systems.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Sistemas distribuidos de información
Está conformado por cuatro capas:

- ◇ Base de datos:
- ◇ Administración de datos:
- ◇ Aplicación:
- ◇ Interfaz de usuario

Requerimientos mínimos

- ◇ Lo que nosotros **nombraremos** es a los servicios.
- ◇ **Estado global y recuperación:** cada proceso tiene su propia línea de tiempo, a pesar de eso hay que ser capaces de verlo como un solo sistema
- ◇ **Tolerancia a fallas:** Los sistemas deben ser capaces de superar adversidades
- ◇ **Seguridad:**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Aplicaciones distribuidas

La razón por la que es importante implementar aplicaciones especiales para ambientes distribuidos radica en la misma naturaleza de los sistemas distribuidos, pues es necesario contar con aplicaciones que puedan comunicarse y cooperar con recursos distantes

Un [aplicación para] sistema distribuido debería:
- Fácilmente **conectar a los usuarios con los recursos**
- **Ocultar** el hecho de que **los recursos están distribuidos**
- Ser **abierto**
- Ser **escalable**

Transparencia

Si un sistema es capaz de mostrarse a sí mismo ante los usuarios y aplicaciones **como si fuera una sola computadora** se dice que el sistema es *transparente*

Types of transparency:

Access: Hide differences in **data representation** and **how a resource is accesed**
Deals with data representation and the way resources can be accessed by users.

Location: Hide where a resource is **located**
Users shouldn't be able to tell where a resource is phisically located

Migration: Hide that **a resource may move** to another location
Users should be able to access a resource in the same way even if it has been moved

Relocation: Hide that a **resource may be moved** to another location **while in use**
Users shouldn't be able to notice that a resource they're accessing is being moved.

Replication: Hide that a resource is **replicated**
Deals with the fact that several copies of a resource exist

Concurrency: Hide that **a resource may be shared** by several competitive users
Users shouldn't be able to notice that other user is making use of the same resource

Failure: Hide a **failure and recovery** of a resource
Users do not notice that a resource fails to work properly AND the system recovers itself from that failure

Persistance: Hide whether a resource is in **memory or on disk**
Deals with masking whether a resource is in volatile memory or perhaps somewhere on a disk

Openness

Un sistema es considerado abierto si ofrece **servicios acordes a las reglas estándar que describen la semántica y la sintaxis** de esos servicios
Son especificados a través de Interfaces en IDL (Interface Definition Language))

Interface definitions written in IDL nearly always capture only the syntax of services. [names of the functions, types of the parameters, return values, possible exceptions. These allow:
- An arbitrary process that needs a certain interface to communicate to another process that provides that interface.
- Two independent parties to build completely different implementations of those interfaces, leading to two separated DS that operate in the exactly same way.

Proper specifications:
- ◇ **Complete:** everything that is necessary to make an implementation has, indeed, been specified
- ◇ **Neutral:** specification don't prescribe what an implementation should look like

Are important for:

Interoperability: two implementations of systems from different sources can co-exist and work together since both rely on each other's services pre-specified by a common standard.

Portability: an application developed for a DS A can be executed, without modification, on a DS B that implements the same interfaces as A

Another goals:

• **Flexible:** easy to configure the system out of different components possibly from different developers
• **Extensible:** easy to add new components or replace existing ones without affecting those components that stay in place

Scalability

Se desea que no se degrade el rendimiento cuando aumenta:

- El número de sitios
- El número de usuarios
- El volumen de datos
- La frecuencia de interacciones

Es recomendable:

- Evitar la centralización de Datos
- Evitar algoritmos que requieren la participación de todos los nodos
- Descentralizar y delegar las decisiones