modelado de tareas, > reactividad, mejor desempeño, mejor para ambientes Modernos. Tipos: Multiprogramación
dades. 2 o más pro se alojan en MP y se ejecutan c) Tiempo compartido (Comparte c un recurso; inclusión de
s). Multiproceso (Comparten MP y periféricos). Paralelismo (Divide problemas grandes en chuquitos).
Sincronización: Exclusión mutua (evita más de un proceso a la vez en una sección crítica) SYNCHRONIZED().
miento (concepto de wait(), notify y notifyAll()). Métodos synchronized, apropiación del objeto. Semáforos (Sirven para
damiento y exclusión mutua). P duerme, v despierta. Adjetivo volatil marca una variable como almacenada en MP

Otros niveles de concurrencia
Blocking queue
Concurrent Hash Map
Copy On Write Arraylist
Cyclic Barrier

Problemática de la concurrencia
- No determinismo
- Condición de carrera
- Espera activa
- Interbloqueos
- Inanición
- Bloqueos activos

Entonces
El SO se encarga de repartir CPU, MP, archivos y dispositivos E/S

PCB
- Estado del proceso
- Contador del programa
- Registros
- Recursos / Memoria asignada.

Concurrencia pThreads
Threads del lenguaje
- código
- estática — datos que no cambian
- dinámica — se pide en t de ejecución
- M. libre
- autom. pila

Exclusion mut.
Candados, se cede el candado y hay apropiación.

Encuentros
Pthreads: join
Barrier.

Procesos
Registros, pila, Id, datos
Ejecución de un programa, código, estado
CREACIÓN Hijo recibe copia del padre. Son independientes y comparten memoria. El hijo se queda con id 0.

COMUNICACIÓN Física ó lógicamente
1. Variables compartidas ó Intercambio de mensajes por nombre del proceso, BUFFER o canal.
2. Comunicación directa-indirecta; sincrónica ó asincrónica.
3. Buffering cero, limitado o ilimitado.

RENDEZ VOUS
El proceso que solicita el encuentro, espera hasta que llegue el otro.

CONCURRENCIA COOPERATIVA
No se lucha por procesador, se cede.

MANEJO DE THREADS
Espacio del usuario → Espacio del Kernel
Mixto

Programación asincrónica
- Problemática E/S
  - Comunicación (Direccionamiento)
  - Transferencia (Procesador dispositivo) — Programada ó por DMA (sin intermediario)
  - Interrupciones son asincrónicas

Concurrencia en servidores
SOCKETS
Un servidor concurrente crea servidores delegados.

POSIBLES SERVIDORES
1. Iterativo — no hay concurrencia
2. Por conexión — Se pierde tiempo creando cuando hay conex.
3. Manejo Pools
   - concurrencia
   - control sobre recursos
   - no gasta tiempo creando
   - Balancea carga

ICP
- Conexión
- Orden
- Control Flujo
- Congestión

RCP
Remotha Procedure call.

RMI
Remote Method Invocation

WEB SERVICES
Analogo RCP

COSTOS
Manejo cola
No hay prioridad

CUANDO USAR SERVIDORES CONCURRENTES

# Procesadores	Alto
% Uso del procesador	Bajo
Tiempo admon. Threads	Bajo
Carga	Alto
Duración tareas	Alto

Virtualización
Proceso → Pedido → Llamado sistema → Virtualización

VIRTUALIZACIÓN PROCESADOR
Nos turnamos el procesador.

VIRTUALIZACIÓN MEMORIA
- Cargamos solo la parte del programa que nos interesa.
- MV > MR
- En MV tenemos páginas, en MR tenemos marcos de página
- Falla de página: No esta en MP y toca traerla.
- TAE = [(1-P) * accesoAMemoria] + P * servicioFallePag.
- TLB guardamos info de direcciones recientemente usadas.
- Podemos compartir recursos y no duplicar código.

! Si dos pag apuntan a la misma dirección, hablamos de archivos proyectados en memoria.

ALGORITMOS
- FIFO
- LRU
- BIT REF.

! Si cargamos muy pocas páginas < Espacio de trabajo, nos tiramos el desempeño.

① Bits para desp → Pags de 2^{12} pos (12) → Bits para desplaza
② Cant páginas $\frac{2^{32}}{2^{12}}$ → páginas de 2^{12} posiciones
③ Tam tabla — cant páginas * 4B ~ 32 bits

Noticas extra
```
P(S) {
  S.contador--
  if(S.contador < 0) {
    duerma }}
V(S) {
  S.contador++
  if(S.contador <= 0) {
    despierta }}
```

Datos importantes
1. El tamaño de la memoria real influye en la concurrencia porque limita el número de procesos que se pueden correr de forma concurrente y la porción de procesos que se pueden cargar. Más fallos de página.
2. Se desarrolla memoria virtual para arreglar problemas que un programa estuviera atado a una máquina.
3. El estado en Espera es importante porque dependen de eso para continuar.
4. # de Procesadores, Tam memoria, intensidad en el uso de CPU - Carga del sistema.
5. Concurrencia por conexión: Servidor crea un canal donde el cliente hace varios pedidos. Concurrencia por solicitud: Se crea un canal por solicitud
6. Es bueno tener memoria virtual para manejar programas grandes, compartir librerías y por la independencia de las direcciones

④ Si necesitamos TP2 → eficiencia

7. VirtualBox es multiplataforma y multihuespedes. Es de tipo 2 donde los usuarios pueden cargar multiples ssoo invitados en un solo SO anfitrión.

Principales problemas

- Espionaje A → escucha → B
- Suplantación A → seHacePasarX → B
- Adulteración A → 3ero → B
- Repudio "Yo no mandé eso"

SE BUSCA CONFIDENCIALIDAD e INTEGRIDAD

Criptografía

↗ Acordar la llave y almacenarla. → DIFFIE-HELLMAN →

A y B escogen un primo p y un g ($g<p$). C/u escoge un $X = p-1$. C/u Genera y comunica y: $y = g^x \bmod p$. C/u genera la llave z: $z = y^x \bmod p$.

! El protocolo no autentica (man in the middle)

1. CIFRADO SIMÉTRICO

- Misma llave para C y DC
- Algoritmo público pero llave secreta
- Espacio de búsqueda 2^n

Cifrado por bloques

- ECB
- CBC: El bloque actual depende de los bloques que han pasado antes. IV → XOR → Encrypt → Nuevo IV.

DES
- Permutación
- 16 Iteración
- Permutación
- Llaves de 56 bits

Bloques 64 bits

3DES
- 168 bits
- Usa 3 llaves

CIFRADO EN FLUJO
- Usa XOR. A partir de una llave K se genera una cadena con la que se hace el XOR.

2. CIFRADO ASIMÉTRICO

Llave pública K^+ (Everyone)
Llave privada K^- (Just me)
Llaves diferentes para C y DC

- **RSA**

C: $M' = M^E \bmod N$ Llave = (E,N) ← Pública
DC: $M = M'^D \bmod N$ Llave = (D,N) ← Privada

¿DC dónde salen las llaves?
Escoge p y q → $n = p*q$
Si $\phi(n) = (p-1)*(q-1)$, escoja e tal que e es primo relativo de $\phi(n)$
$d = e \bmod \phi(n) = 1$

COMPARACIÓN	SIMÉTRICA	ASIMÉTRICA
	Rapido.	Lento.
	Requiere secret.	No requiere secretos
	Req. acordar la llave	Req. obtener llave pública.

RESUMEN DIGITAL (DIGEST)

Toma un mensaje y calcula un número

- Es fácil calcular H(m) pero un m de un H(m).
- Usado en Integridad de datos: M + H(M) → H(M) recibido = H(M) calculado?

CODIGOS DE AUTENTICACIÓN

Me asegura que sí hablo con quien creo hablar.

- MAC: String de bits que depende del mensaje y una llave secreta.
- HMAC: Convierte funciones de Hash en MACs. Gets a digests of 2 separate inputs.
- SOBRE DIGITAL: Cifro con pública del receptor (Puedo enviar una llave de sesión simétrica y enviarla)

FIRMA DIGITAL

integridad - autenticación & no - repudio

Cifro el mensaje (resumen) con llave privada para obtener la firma y envía. Descifro con la pública del emisor y verifico que corresponda.

- Es posible probar la identidad del emisor → Fundamental proteger la privada
- Se firma el resumen, no el mensaje, por eficiencia. → Si el documento cambia, la firma cambia. (Por eso verifica también la integridad)

Info puede ser espiada porque no hay sobre
Tiempo ~ Tamaño llave.
¿Cuándo firmaron el documento?

ESTAMPILLAS CRONOLOGICAS

Mando un Hash o resumen a certicámara, ellos ponen el sello y devuelven

CERTIFICADOS DIGITALES

A — Solicitud → B
A ← certificado? — B
A — FIRMA EC IDY OTROS + PUBLICA(A) → B

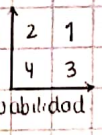
Problema con la revocatoria
Al conocer el CD, se pueden enviar sobres digitales.

SUPERFICIE DE ATAQUE

Union de todos los puntos de entrada
Hardening: Proceso de asegurar un sistema
MINIMIZAR: Services, cuentas admin, cuentas de user, permisos de ejecución.
ACTUALIZAR - PARCHES: Sistema operacional, aplicaciones.
MANTENER NIVELES DE DEFENSA: Firewall, IDS (host, red), Antivirus, antispam.
AISLAR: Services, unidades organizacionales, intranet e internet
CONFIGURAR: Sistemas operativos y services.
MONITOREAR: Actividades (administradores, usuarios regulares, programas), tráfico de red, mantener y revisar log
ASIGNAR RESPONSABILIDADES
EVALUAR: Pruebas de penetración.
Redundancia de services críticos, seguridad física, políticas de respaldo / recuperación y continuidad.

ADMINISTRACIÓN DEL RIESGO

VULNERABILIDAD: Imperfección en un proceso
AMENAZA: Circunstancia con potencial de afectar
Riesgo = probOcurrencia * nivel de impacto.



RESPUESTA AL RIESGO

Aceptar - Evitar - Mitigar - Transferir.

PROTECCIÓN

OBJETO PERMISOS
OBJETO PERMISOS

CONTROL DE ACCESO
DOMINIO DE PROTECCIÓN
MATRIZ DE CONTROL DE ACCESO ← MUY GRANDE

	Archivo1	Archivo2	Archivo3	Archivo4
Proceso1	read	write	r,w,own	write

LISTA DE CAPACIDADES.
Proceso1 → o1,p1 — o2,p2 — o3,p2

LISTA DE ACCESO
Archivo1 → S1, p1; S2, p3 — S3, p1

Norma

AUTENTICACIÓN

Algo que se tiene
Algo que se es
Algo que se sabe
Definicion de permisos
Quién - qué - cómo

UNIX

login/clave
Función hash f. f(clave).

UNIX/LINUX

Mangan versiones simplificadas de la lista de accesos.
Cada usuario tiene UID/GID
Cada archivo tiene ID propietario/GID propietario.
los procesos se crean con UID del usuario
los hijos heredan UID del padre.

AUTENTICACIÓN VS. AUTORIZACIÓN

Establece identidad del usuario — Da los permisos

Esc
lec → 1 1 1 → Ejecución

Permisos dueño | Permisos grupo | Permisos otros | Dueño | Grupo

Se pueden validar permisos cada vez que se ejecuta un proceso relevante o solo la primera vez.

El tiempo promedio no representa el comportamiento del sistema

tamaño llave
! 2

Para validar un cert
certif, firma, fechas

CBC = MAC

Nivel de seguridad

Bloques ECB
Bits de longitud fija. tomando bloque de txt plano y produce uno =. Como se usa la misma llave es fácil reconocer los patrones con los que se cifra. No muy seguro.

CBC
Cada bloque depende de un XOR y uno que ya se cifró. Difícil encontrar el patrón.

Usos de sesion
Preparacion
Valoracion
Comunicar resultado
Mantenimiento proceso

Autenticidad → Cifrado + hash, Mac, Hmac
No repudio → Cifrado + hash
Integridad → Hash (digest)
Confidencialidad → Cifrado

C — SSL Hello, posibles algoritmos, → S
semilla A
C ← certificado, algoritmo sel, — S
semilla B
C — $CK_S + (semilla_c)$, $HMAC_{K1c-s}(X1)$ → S
C ← $HMAC_{K1s-c}(X2)$ ——— S

De aquí en adelante se cifra con llaves de sesión

— $KC1s-c$, $K1s-c$ →
← $KC1c-s$ $K1c-s$ ——

Dimensionamiento de la capacidad en TI

1. Definir los requerimientos: caracterizar el servicio prestado y determinar las expectativas de los usuarios.
   - Definir el tipo de carga (quien lo hace, qué y como).
   - Definir la unidad de carga
   - Establecer niveles de servicio.
   - Acuerdo entre proveedor y cliente.
2. Determinar componentes informáticos.
3. Determinar requerimientos de infraestructura:
   Procesamiento (Capacidad, escalabilidad, desempeño)
   Seguridad (Confidencialidad, integridad, disponibilidad)
   Otros
4. Hacer prospectiva.
   Elaborar pronóstico de necesidades de procesamiento.
   Elaborar modelo: niveles de servicio = f(carga, recursos).
   Definir acciones jugandole a esos valores.

$$\text{Productividad} = \frac{\#\text{ tareas terminadas}}{t}$$

$$\text{Disponibilidad} = \frac{t_{disponible}}{(t_{disponible} + t_{no\,disp})} = \frac{MTBF}{(MTBF + MTTR)}$$

$$\text{Rendimiento} = \frac{req}{t}$$

$t_{usuario} \equiv t_u$ $t_{sistema} \equiv t_s$ $t_{total\ CPU} = t_u + t_s$

$$\text{Eficiencia} = \frac{(t_u + t_s)}{t_r}$$

Alarmas: total CPU << t_r ($t_s > 10\%$)

< 80% sí
= 80 saturado
> 80 no

MULTIPROCESADORES

$$\text{costo}(p) = T(p) * P \qquad S(p) = \frac{T(1)}{T(p)}$$

$$\text{eficiencia}(p) = \frac{S(p)}{P}$$

DISPONIBILIDAD EN SERIE: $P(S) = \prod A_i$

DISPONIBILIDAD EN PARALELO: $P(A) + P(B) - P(A)P(B)$

LEY DE LITTLE

$N = aR$ — t de rta

Total de req. ↳ Tasa de llegada

LEY DE USO

$R_s/R_a = U = a R_s$ — Tiempo de serv.

Uso del controlador. ↳ Tasa de llegada

Tiempo promedio entre llegadas $R_a = \frac{1}{a}$

$R = R_s / (1 - U)$

Diseño: Clusters

PROPOSITO
- Alta disponibilidad
- Tolerancia a fallas
- Escalabilidad
- Carga
- Desempeño

TIPOS (Escalamiento vertical.)

FAILOVER: El servicio migra automáticamente ante una falla.

BALANCEO DE CARGA. — Horizontal.

ACTIVO/PASIVO ó ACTIVO/ACTIVO.

FAILOVER	BALANCEO DE CARGA.
Recuperación ante fallas	No recuperación ante fallas
Persistencia por almace.	Sin estado.
Grandes servidores con hardware redundante	Servidores sencillos y baratos.
Solución específica.	

! Se pueden mezclar.

! Se puede redundar en balanceadores.

Diseño: Clusters - bases de datos

ALMACENAMIENTO COMPARTIDO
Varios SMBD pegan a un mismo disco.

MULTIMAESTRO
Se pasa por un replicador. (Mayoría escrituras)

MAESTRO - ESCLAVO
Se pasa por un replicador (Se escribe una vez y ese hace commit)

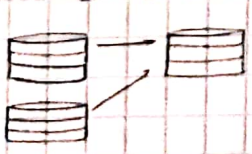
SHARED NOTHING
Lectura escritura para bases distintas

	Solución	Desempeño	Product.	carga.	Disponibilidad	Escalabilidad
	Esc. vertical					
	+ recursos	x	x	x		
	multiproc	x	x	x	x	x
	Cluster					
	computo	x	x			
	fallover.				x	
	balanceo					
	con red	x	x	x	x	x
	sin red.	x	x	x		x
Almacenamiento	Shared disk	x lecturas.	x	x	x	x Limitada por sincr.
	Replicación					
	MM.	x lecturas		x lecturas	x	x lecturas
	ME	x lecturas		x lecturas	x lento	x lecturas.
	Shared nothing	x		x Estático	x	x

Diseño: Almacenamiento

ARCHIVOS Como un arbolito.

LVM



SNAPSHOTS

DISPOSITIVOS

DAS. Direct attached storage.

RAID. Redundant array of independent disk.

Striping, mirroring, paridad. composición

0 Stripping
1 Mirroring
2 Bit striping.
3 Byte striping.
4 Striping con paridad
5 Striping con paridad distribuida
6 Doble paridad distribuida

	Eficiencia/capacidad	Tolerancia Fallos	Disponibilidad	Lect. aleat.	Escrit aleat.	lect secuen.	Escrit. secuen
0	100%	~~4~~	3	1	1	1	1
1	50%	2	2	2	2	3	2
~~2~~	(N-1)/N	~~3~~					
3	"	~~3~~	2	3	5	1	4
4	"	3	2	1	5	1	4
5	"	3	2	1	3	2	3
6	(N-2)/N	1	1	1	4	2	4

NAS (Network attached storage).
Servidor de archivos

SAN (Storage area network)
Interconectar servidores con dispositivos de almacenamiento.
Bajo costos.

Backup.
Full.
Diferencial
Incremental.
Snapshot.

10. Stripe de espejos → Mejor
01 Espejo de stripes.

Errores comunes

La carga de prueba solo muestra el comportamiento típico.
Se ignora el sesgo de demandas
Se ajustan mal los parámetros
Ignorar efectos del
- caché
- buffering
- condiciones iniciales

Ignorar sobrecarga del monitor
Impresiciones por muestreo.
Recoger mucho y analizar poco.