Sistemas Distribuidos
Introducción

Sergio Yovine

Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina

Sistemas Operativos, primer cuatrimestre de 2017

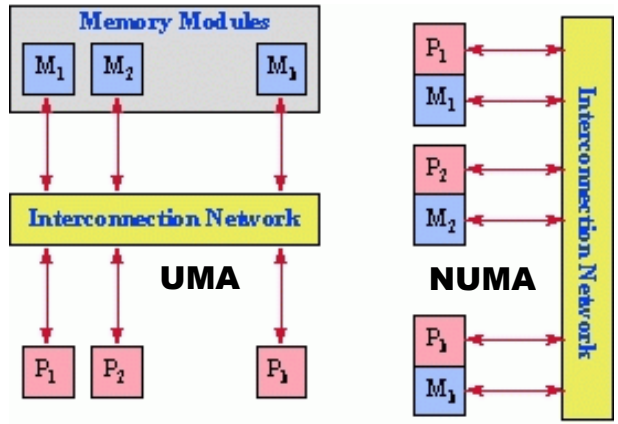
(2) Sistemas distribuidos

- Conjunto de recursos conectados que interactúan
  - Varias máquinas conectadas en red
  - Un procesador con varias memorias
  - Varios procesadores que comparten una (o más) memoria(s)
- Fortalezas
  - Paralelismo
  - Replicación
  - Descentralización
- Debilidades
  - Sincronización
  - Coherencia
  - Información parcial

(3) Sistemas distribuidos: Memoria compartida

Hardware

- Uniform Memory Access (UMA)
- Non-Uniform Memory Access (NUMA)
- Híbrida



Software

- Estructurada
  - Memoria asociativa: Linda, JavaSpaces
  - Distributed arrays: Fortran, X10, Chapel
- No estructurada
  - Memoria virtual global
  - Memoria virtual particionada por localidad

(4) Sistemas distribuidos: Mensajes

Comunicación sincrónica

- Remote Procedure Call (RPC)
  - Java Remote Method Invocation (RMI)
  - JavaScript Object Notation (JSON-RPC)
  - Simple Object Access Protocol (SOAP)

Comunicación asincrónica

- RPC asincrónico
  - Promises (B. Liskov, 1988)
  - Futures (Walker, 1990) – Java
  - Windows Asynchronous RPC
- *send* / *receive*
  - Mailbox
  - Pipe
  - Message Passing Interface (MPI) para C/C++
  - Scala actors: send, receive/react

(5) Sistemas distribuidos: Sistema de archivos

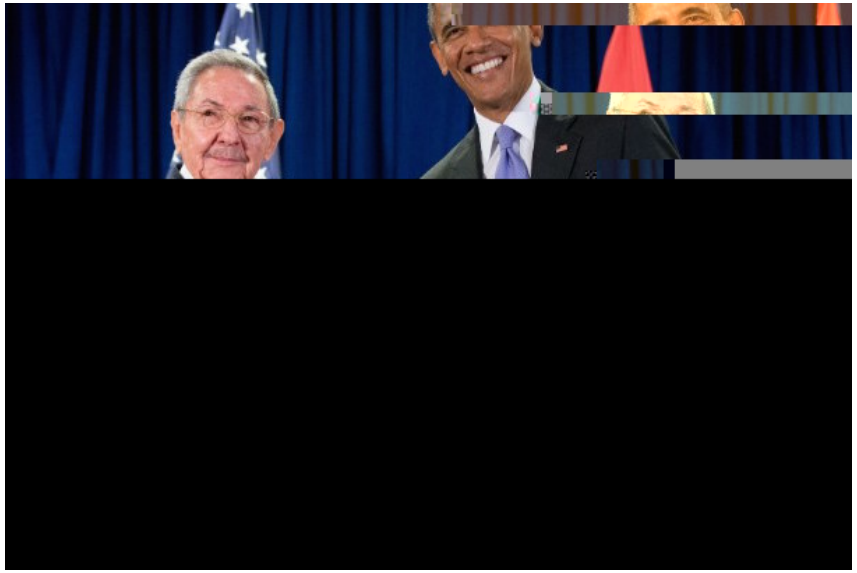
- Propiedades
  - Transparencia
    - Acceso
    - Ubicación
  - Escalabilidad
    - Concurrencia
    - Tamaño
  - Tolerancia a fallas
    - Replicación
    - Migración
  - Heterogeneidad de plataformas de SW y HW
- Ejemplos
  - Hadoop Distributed File System (HDFS)
  - Red Hat Gluster Storage

(6) Sistemas distribuidos: Interacción entre procesos

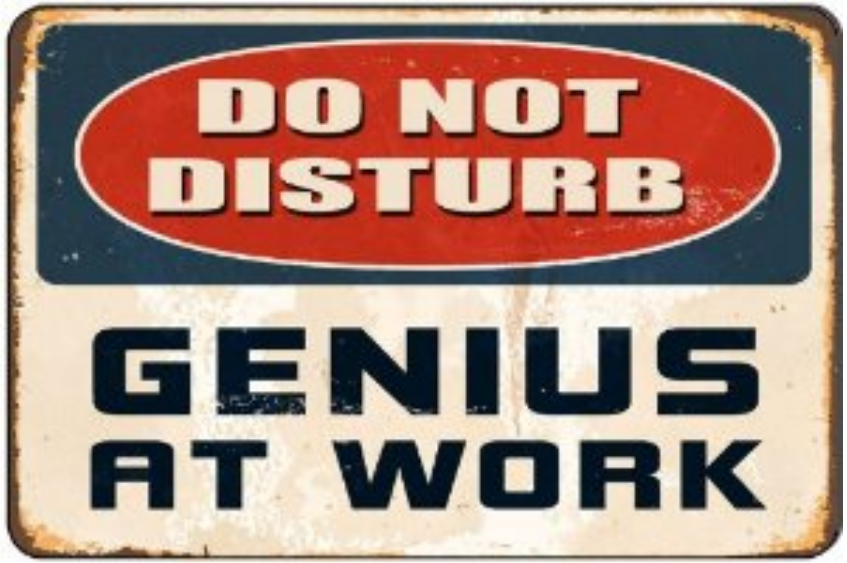
Scheduling

Coordinación

Sincronización



Uso exclusivo



(7) Sistemas distribuidos: Scheduling

- Dos niveles
  - Local: dar el procesador a un proceso listo
  - Global: asignar un proceso a un procesador (*mapping*)
- Global: *compartir* la carga entre los procesadores
  - Estática: en el momento de la creación del proceso (*affinity*)
  - Dinámica: la asignación varía durante la ejecución (*migration*)
- Compartir vs *balancear*
  - Balancear: repartir equitativamente
  - Evaluar costo-beneficio

(8) Sistemas distribuidos: Scheduling

- Migración
  - Iniciada por el procesador sobrecargado (*sender initiated*)
  - Iniciada por el procesador libre (*receiver initiated / work stealing*)

- Política de scheduling
  - Transferencia: cuándo hay que migrar un proceso
  - Selección: qué proceso hay que migrar
  - Ubicación: a dónde hay que enviar el proceso
  - Información: cómo se difunde el estado

(9) Sistemas distribuidos: Sincronización

Modelos de comunicación

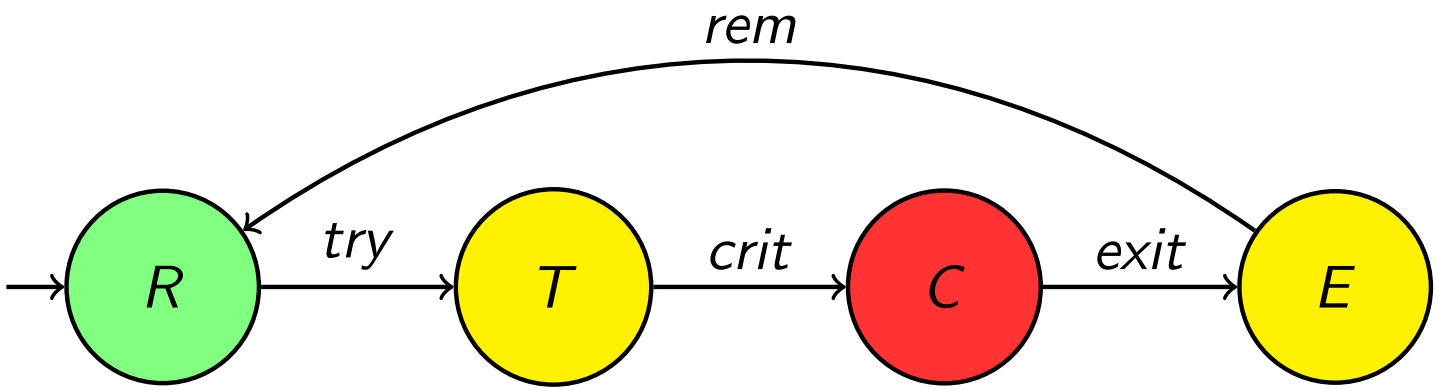
- Envío asincrónico de mensajes
- Memoria compartida global direccionable

Problemas

- Orden de ocurrencia de los eventos
- Exclusión mutua
- Consenso

(10) Modelo de proceso

- N. Lynch, Distributed Algorithms, 1996 (Cap. 10)



- Estado: $\sigma : [0 \ldots N-1] \mapsto \{R, T, C, E\}$
- Transición: $\sigma \xrightarrow{\ell} \sigma'$, $\ell \in \{rem, try, crit, exit\}$
- Ejecución: $\tau = \tau_0 \xrightarrow{\ell} \tau_1 \ldots$
- Garantizar *PROP*: Toda ejecución satisface *PROP*
- Notación: $\#S$ = cantidad de elementos del conjunto S

Exclusión mutua (EXCL)

Para toda ejecución τ y estado τ_k, no puede haber más de un proceso i tal que $\tau_k(i) = C$.

$$\Box \#CRIT \leq 1$$

Progreso (PROG) (*lock-free*)

Para toda ejecución τ y estado τ_k,
si en τ_k hay un proceso i en T y ningún i' en C
entonces $\exists j > k$, t. q. en el estado τ_j algún proceso i' está en C.

$$\Box(\#TRY \geq 1 \wedge \#CRIT = 0 \implies \Diamond \#CRIT > 0)$$

Progreso global absoluto (WAIT-FREE)

Para toda ejecución τ, estado τ_k y todo proceso i,
si $\tau_k(i) = T$
entonces $\exists j > k$, tal que $\tau_j(i) = C$.

$$IN(i) \equiv i \in TRY \implies \diamondsuit\, i \in CRIT$$

$$\forall i. \Box\, IN(i)$$

Progreso global dependiente (G-PROG)

(*deadlock-, lockout-, o starvation-free*)

Para toda ejecución τ,
si para todo estado τ_k y proceso i tal que $\tau_k(i) = C$,
$\exists j > k$, tal que $\tau_j(i) = R$
entonces para todo estado $\tau_{k'}$ y todo proceso i',
si $\tau_{k'}(i') = T$
entonces $\exists j' > k'$, tal que $\tau_{j'}(i') = C$.

$$OUT(i) \equiv i \in CRIT \implies \diamondsuit\, i \in REM$$

$$\forall i.\Box\, OUT(i) \implies \forall i.\Box\, IN(i)$$

Justicia (FAIR) (*fairness*)

Para toda ejecución τ y todo proceso i,
si i puede hacer una transición ℓ_i en una cantidad infinita de estados de τ
entonces existe un k tal que $\tau_k \xrightarrow{\ell_i} \tau_{k+1}$.

(16) Bibliografía extra

- Nicholas Carriero, David Gelernter: Linda in Context. CACM, 32(4), 1989. http://goo.gl/gfgbsQ
- Andrew D. Birrell and Bruce Jay Nelson. 1984. Implementing remote procedure calls. ACM Trans. Comput. Syst. 2, 1 (February 1984), 39-59. http://goo.gl/3eIskN
- A. L. Ananda, E. K. Koh. A survey of asynchrounous RPC. http://goo.gl/t96vFg
- Andrew S. Tanenbaum. RPC. http://goo.gl/9N3zKz
- T.L. Casavant, J.G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. IEEE TSE 14(2):141-154, 1988.
- M. Singhal and N. G. Shivaratri. Advanced Concepts in Operating Systems. McGraw Hill, 1994.