Teoria-Tema-3.pdf

BlackTyson

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Resumen T3 Arquitectura de computadores:
Wuolah: **@BlackTyson**

Lección 7:

1. **Clasificación y estructura de arquitecturas con TLP explícito**
    a. **Tipos:**
        i. Multiprocesador: ejecutan **varios threads en un computador con varios cores.** Encontramos diversos niveles de empaquetamiento
        ii. Multicore: ejecutan **varios threads en paralelo** en un chip de procesamiento multicore.
        iii. Core multi thread: core que modifica su arquitectura para ejecutar **threads concurrentemente.**
2. **Multiprocesadores**
    a. Criterio: nivel de empaquetamiento:
        i. Sistema
        ii. Armario
        iii. Placa
        iv. Chip
    b. Criterio clasificación de memoria:
        i. **Centralizada (UMA): mayor latencia, poca escalabilidad.**
        ii. Numa
    c. Evolución UMA -> NUMA
        i. Pasamos de un controlador de memoria en chipset a uno en un chip del procesador.
        ii. Pasamos de usar bus a usar enlaces y conmutadores.

Lección 8

1. **Sistema de memoria en multiprocesadores**
    a. Incluye:
        i. Caches en todos los nodos
        ii. Memoria principal
        iii. Controladores.
        iv. Buffers: (escritura/almacenamiento)
        v. Medio de comunicación de todos estos componentes
    b. Comunicación de datos realizada por el sistema de memoria
2. **Concepto de coherencia en el sistema de memoria:**
    a. **Métodos de actualización de memoria principal implementados en caché:**
        i. **Inmediata** (write-through): **cada vez que se escribe** un dato en caché **se actualiza en memoria.**
        ii. **Posescritura**(write-back): se **espera a completar todo el bloque** y entonces **se escribe todo** el bloque en memoria.
    b. **Métodos para propagar escrituras entre cachés:**
        i. **Actualización** (write-update): **al escribir** en una caché, **se escribe en todas las demás.**
        ii. **Escritura con invalidación**(write-invalide): **al escribir** en una caché, **se invalidan las demás.**

c. **Requisitos del sistema de memoria para evitar problemas de incoherencia:**
    i. **Propagar las escrituras en una dirección:** debe hacerse en un **tiempo finito** a otros procesadores y los componentes **tienen que estar conectados con un bus.**
    ii. **Serializar las escrituras en una dirección**. Deben verse en el **mismo orden por todos los procesadores**. El orden en el que los paquetes aparecen en el bus determina el orden en que se ven por todos los nodos.
d. **Requisitos cuando la red no es un bus:**
    i. **Propagar escrituras en una dirección**: se usa **difusión**, enviando paquetes de actualización a todas las cachés. Para conseguir **mayor escalabilidad se envía dichos paquetes solo a cachés que tengan la copia del bloque.**
    ii. **Serializar escrituras en una dirección.** El **orden** en que las **peticiones de escritura** llegan a su home o **directorio centralizado** sirve para **serializar el orden de las transferencias entre dos puntos**
e. **Alternativas para implementar el directorio:**
    i. **Centralizado: compartido por todos los nodos**. Contiene información de todos los módulos de memoria.
    ii. **Distribuido: las filas se distribuyen entre los nodos**. El directorio de un nodo contiene información de los bloques

3. **Protocolos mantenimiento de coherencia: clasificación y diseño.**
    a. **Clasificación:**
        i. **Snoopy**: Para buses y sistemas con una **difusión eficiente**
        ii. **Protocolos basados en directorios:** para r**edes sin difusión** o escalables.
        iii. **Esquemas jerárquicos: para redes jerárquicas.**
    b. **Facetas de diseño lógico:**
        i. **Política de actualización de MP** (inmediata, posescritura, mixta)
        ii. **Política de coherencia entre cachés**(invalidación, actualización, mixta)
        iii. **Describir comportamiento:**
            1. Definir **posibles estados de los bloques en caché y memoria.**
            2. **Definir transferencias a generar** ante lecturas/escrituras o llegada de paquetes de otros nodos
            3. **Definir transiciones de estados** para un bloque en caché y en memoria.

4. **Protocolo MSI espionaje:**
    a. **Estados de un bloque en caché:**
        i. Modificado (M): única copia válida en todo el sistema
        ii. Compartido(C,S): está válido, pero también puede estarlo en memoria u otras cachés.
        iii. Inválido(I): invalidado o no está físicamente.
    b. **Estados en un bloque en caché:**
        i. Válido(puede haber copia válida)
        ii. Inválido.
    c. **Tipos de Transferencias generadas por un nodo con caché:**
        i. Petición lectura de un bloque**(PtLec)**: **por lectura con fallo de caché del procesador (PrLec)**
        ii. Petición de acceso exclusivo**(PtLecEx): por escritura del procesador (PrEsc) en bloque inválido**
        iii. Petición de posescritura**(PtPEsc): por reemplazo del bloque modificado**
        iv. Respuesta con bloque**(RpBloque): al tener en estado modificado el bloque solicitado por una PtLec o PtLecEx**
    d. **Tabla de descripción:**
        i. Modificado
            1. PrLec/PrEsc -> Modificado
            2. PtLec(genera RpBloque) -> compartido
            3. PtLecEx(genera RpBloque e invalida copia local) -> inválido
            4. Reemplazo(genera PtPEsc) -> inválido
        ii. Compartido
            1. PrLec -> compartido
            2. PrEsc(genera PtLecEx) -> modificado
            3. PtLec -> compartido
            4. PtLecEx(invalida copia local) -> inválido
        iii. Inválido:
            1. PrLec(genera PtLec) -> compartido
            2. PrEsc(genera PtLecEx) -> modificado
            3. PtLec/PtLecEx) -> inválido
5. **Protocolo MESI de espionaje**
    a. **Estados de un bloque en caché:**
        i. Modificado(M)
        ii. **Exclusivo(E): única copia de bloque válida** en cachés, estando actualizada en memoria.
        iii. **Compartido(C,shared):** válido pero también lo es en memoria y en **al menos otra caché**
        iv. Inválido(I)
    b. **Estados de un bloque en memoria:**
        i. Válido
        ii. Inválido.

6. Protocolo MSI basado en directorios con o sin difusión:

a. Sin difusión:

i. Estados de un bloque caché (M,C,I)

ii. Estados de un bloque MP (V e I)

iii. Tipos de transferencias

1. **Tipos de nodos**: Solicitante, origen, modificado, propietario y compartidor.
2. Petición de nodo S a O: PtLec, PtLecEx, PtEx, PtPesc
3. Reenío de petición de nodo O a nodo copia(P,M,C): RvInv, RvLec, RvLec,Ex
4. Respuesta de nodo P a O o respuesta de nodo O a S: RpBloque, RpInv, RpBloqueInv)

iv. Tabla de descripción:

1. Fallo de lectura:
   - Inválido(D) -> Válido
   - Inválido(S) -> Compartido
   - Modificado (P) -> Compartido
2. Fallo de escritura:
   - Válido (D) -> Inválido (D)
   - Inválido(S) -> Modificado
   - Compartido(P) -> inválido

b. Con difusión:

i. Estados en cache: M,C,I

ii. Estados en MP: V, I

iii. Tipos de transferencias:

1. Tipos de nodos: Solicitante(S), origen(O), modificado(M), propietario(P), compartidor(C)
2. Difusión de petición S -> O y P: PtLEc, PtLecEx, PtEx
3. Difusión de petición S-> O: PtPEsc
4. Respuesta P -> O: RpBloque, RpInv
5. Respuesta O -> S: RpBloque, RpInv, RpBloqueInv

iv. Tabla de descripción:

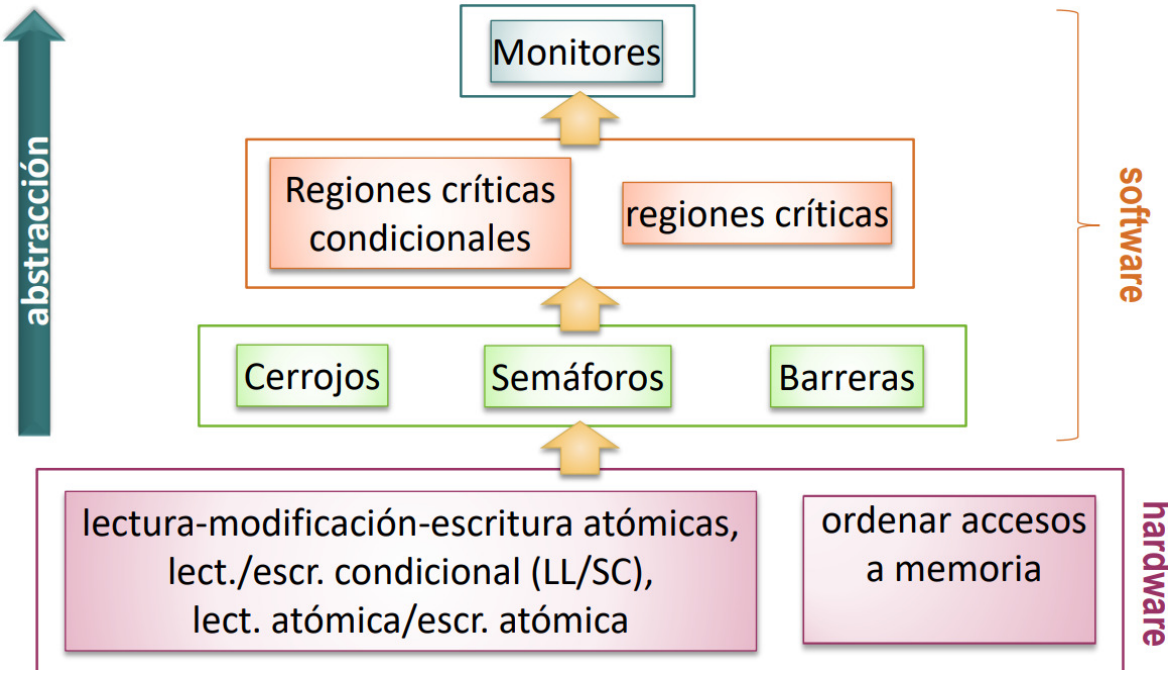
1. Fallo de lectura:
   - Inválido(D) -> Válido
   - Inválido(S) -> Compartido
   - Modificado(P) -> Compartido
2. Fallo de escritura:
   - Válido(D) -> Inválido
   - Inválido (S) -> Modificado
   - Compartido(P) -> Inválido

Lección 9

1. **Consistencia de memoria**
    a. Especifica el **orden** en el que las **operaciones de memoria** deben p**arecer haberse realizado.** La coherencia abarca sólo **operaciones realizadas por múltiples componentes en una misma dirección.**
2. **Consistencia secuencial(SC)**
    a. Es el modelo de consistencia que se espera de herramientas de **alto nivel.** Requiere que todas las **operaciones de un procesador parezcan ejecutarse en el orden descrito por el programa.** Todas las operaciones en memoria deben ser ejecutadas **una cada vez**
    b. Presenta el sistema de memoria como **memoria global** conectada a todos los procesadores a través de un conmutador.
3. **Modelos de consistencia relajados**
    a. **Requisitos para garantizar SC:**
        i. Orden del programa
        ii. Atomicidad
    b. **Modelos relajados comprenden:**
        i. Órdenes de acceso a memoria
        ii. Mecanismos ofrecidos por el hardware para garantizar el orden

Lección 10

1. **Comunicación en multiprocesadores y necesidad de sincronización.**
    Uno-a uno:
        i. Se debe garantizar que el **proceso receptor lea la variable compartida** cuando el **enviador haya escrito** el dato en dicha variable.
        ii. **Si se reutiliza dicha variable,** se debe garantizar que n**o se envíe un nuevo dato** hasta que no se haya leído
2. **Soporte Software y hardware para sincronización:**



3. **Cerrojos**
    a. **Simples:**
        i. Permiten sincronizar mediante **dos operaciones:**
            1. **Lock(K):** adquirir el derecho de **acceso** a una sección crítica. Si varios procesos lo intentan a la vez, solo uno lo consigue y el resto esperan.
            2. **Unlock(K)**: **libera** a un threads que espera el acceso. Si no hay threads en espera, permitirá que el siguiente que llegue pueda hacer lock
        ii. Componentes de un código para sincronización:
            1. Adquisición.
            2. Espera
            3. Liberación
    b. **Con etiqueta:**
        i. Fijan un orden FIFO en la adquisición del cerrojo
    c. **OpenMP:**
        i. Inician, destruyen cerrojo, cierran cerrojo, abren cerrojo.
4. **Apoyo Hardware a primitivas software:**
    a. **Test&Set(x):** lock(k) while **test&set == 1**
    b. **Fetch&or(x,a)**: lock(k) while **fetch&or(k,1) == 1**
    c. **Compare&swap(a,b,x):** lock(k) {b = 1 do **compare&swap(0,b,k) while (b==1)**