SPM.21 Cap.6, slide 65: Protocolul Write-Update snoopy cache

Face actualizări ale tuturor copiilor blocului ce a suferit modif.

• Read Hit:

Dacă se citesc, se modifică ...

• Read Miss:

Un µP îți să citească un bloc ce nu se află în mem. cache → → blocul e adus din mem. comună în mem. cache a µP respectiv, e sg. bloc din mem. cache ale µPs sistemului, și s.n. single consistent.

De. ∃ m. multe copii ale blocului în diverse mem. cache → aceste copii trebuiau să fie Multiple consistent.

De un µP în tentativa de a citi un bloc se află în starea Read Miss & ∃ un bloc single inconsistent atunci info. din blocul respectiv e ---

Blocul respectiv e adus în mem. cache a µP care a cerut info., iar blocul respectiv și blocul din cache devin multiple inconsistent.

• Write Hit:

→ mem. cache a µP care cere inscrierea are blocul;

→ bloc în single inconsistent → se actualiz. info și își păstrează starea de single inconsistent;

→ dc. blocul era în starea multiple consistent, blocul e actualizat, info. e utiliz. pt. a actualiza și celelalte blocuri identice → la sf. toate blocurile sunt multiple consistent;

• Write miss:

→ vrem să scriu un bloc care ∄;

→ îl aduc din mem., e modif., și devine single inconsistent;

dc. ∄ blocul în mem. cache a niciunui µP

→ dc. blocul ∃ în altă mem. cache și e single consist. → se aduce în mem. cache a blocului care a cerut info., se inscrie, se actualiz. cont. blocului care era single consist. (mem. cache a µP1), iar apoi ambele blocuri sunt multiple consistent;

→ dc. erau mai multe blocuri single inconsistent → ---

→ dc. ∄ copii multiple consistent → µP aduce copia în cache, o modif., apoi modif. tuturor blocurilor multiple consist. anterioare, iar apoi toate blocurile intră în multiple consistent.

Protocolul write invalidate e mai rapid pt că nu necesită transfer de dte. pe mag., invalidarea făcându-se printr-o sg. comandă.

Protocoale pe directoare:

∄ unor facilități care să stocheze info. în leg. cu sistemului → e necesar un hw. suplim. a.î. să adauge fiecărui bloc o serie de biți care dau starea blocului respectiv în mem. cache ale µPs. ce form. sist.

ex.: pg. 73 : *3 µPs → 3 biți suplim. numiți biți 'present' → spun dacă blocul de dte. e prezent în mem. cache a unui µP (1 = e prezent)

* bitul single inconsistent → dc. e 1 (dacă sist.) bloc în cauză e single inconsistent (între un bloc x mem. comună ∄ diferențe)

* la mem. cache ale µPs, fiecărui bloc de dte. i se asociază 2 biți: V = valid
P = ~~present~~ private

Nu sunt ∄ bune pt. că tb. să avem atâția biți câte µP-uri dc. mai adaug un µP tb. să mai adaug un bit

Limited directory protocol:

Are doar un număr max. de biți care dau starea blocului în mem cache ale µPs. → dc. am N biți, o să țină minte starea blocului în mem. cache ale ultimelor cele mai utilizate N µPs.

Sisteme bzte. pe utilizatori:

∄ dte. care se pot scrie în cache
care nu se pot —

Hotărârea dc. dte. sunt cacheable sau noncacheable e luată de compilator → compl. tb. să fie complex.

Multicomputers:

→ ansamblu μP + mem. proprie μP + porturi I/O proprii μP;
→ struct. poate fi omogenă sau heterogenă;
→ dezavantaje
→ sist. distribuite → comunică prin message passing;

- pachetele tb. să aibă un format standard;
- (↑) pachet are un antet ce conține info. despre sursă, dest, conținut, ~~...~~ check sum;
- la sf. pachetului se pune suma ciclică de control;
- pachetele au o dim. max.;

Multiprocesor vs. Multicomputer:

→ depinde de tipul pb. și de cadrul în care e utiliz. sist.;
→ ex.: sist. ptr. controlul frontierelor unei țări → avem multe pete. de verificare la frontiere
→ ex.: gestiunea unei BD enorme.