Multiprocesorski sistemi

Hijerarhijski protokoli

Marko Mišić, Milo Tomašević

13S114MUPS, 13E114MUPS, 13M114MUPS

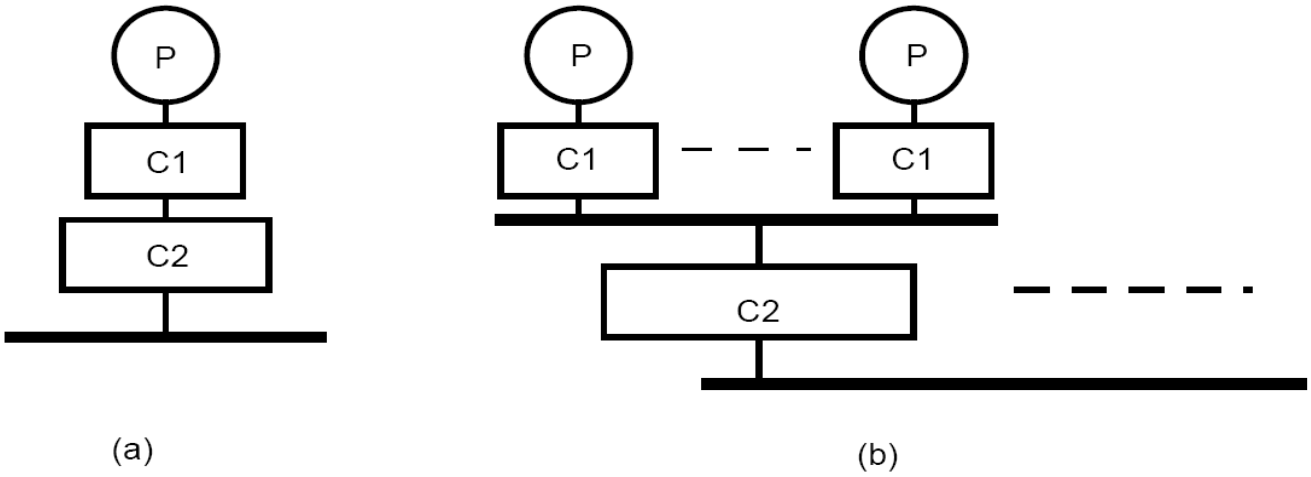
2024/2025.

Hijerarhije keš memorija

- Dva kontradiktorna zahteva za keš memorije
  - Veća brzina -> manje keš memorije
  - Veći faktor pogotka -> veće keš memorije
- Rešenje – hijerarhija keš memorija
  - Brzina se postiže na nižim nivoima
  - Faktor pogotka se postiže na višim nivoima
- Organizacija keš hijerahija
  - Privatna hijerarhija (na čipu i izvan njega)
  - Deljena hijerarhija
- Privatna hijerarhija
  - Prostiji protokol između dva nivoa
  - Manje ekonomična

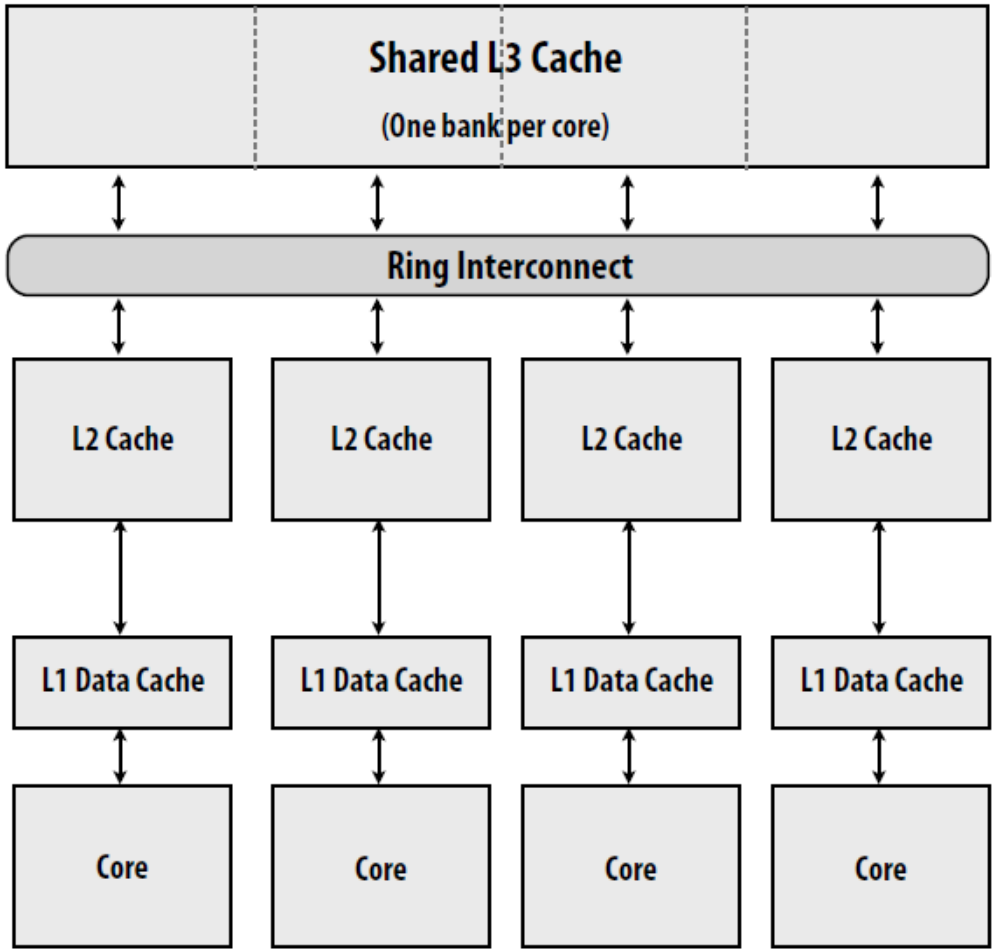
Hijerarhije keš memorija

- Deljena hijerarhija
  - Ekonomičnija i fleksibilnija
  - Protokol na obe magistrale



- Problem koherencije se usložnjava!

Hijerarhije keš memorija



L3: (per chip)
8 MB, inclusive
16-way set associative
32B / clock per bank
26-31 cycle latency

L2: (private per core)
256 KB
8-way set associative, write back
32B / clock, 12 cycle latency
Up to 16 outstanding misses

L1: (private per core)
32 KB
8-way set associative, write back
2 x 16B loads + 1 x 16B store per clock
4-6 cycle latency
Up to 10 outstanding misses

Intel Core i7

64-bajtni blok

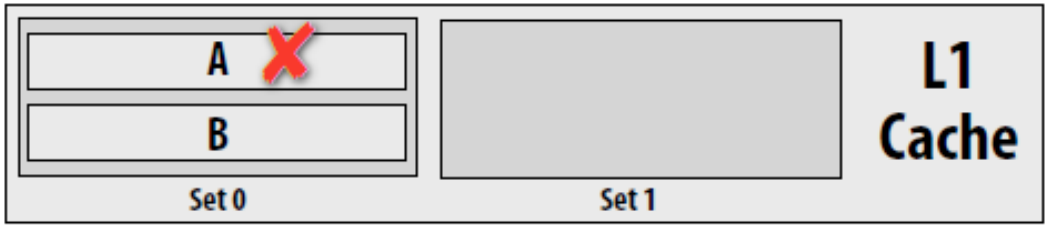
directory protokol (katalog u L3)

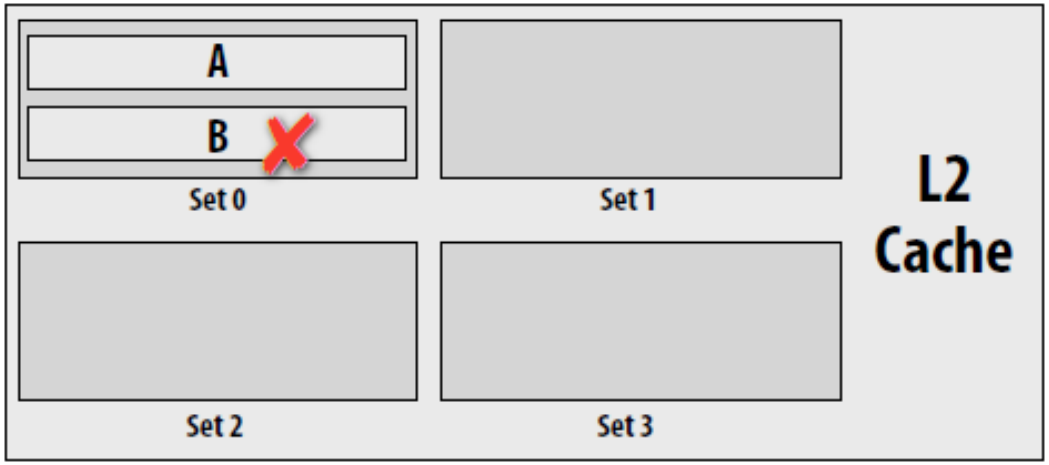
Inkluzija

- Zahtevi za MLI (*multilevel inclusion*)
  - Sadržaj L1 keša mora biti podskup od sadržaja L2
  - Modifikovan blok u L1 mora imati isto stanje i u L2
- Problemi kod održavanja inkluzije
  - Set-asocijativni keševi sa LRU algoritmom zamene
  - Više keševa (I, D) na nižem nivou
  - Različite veličine bloka
- Potrebni uslovi (nije ih teško postići)
  - Broj L2 setova >= broj L1 setova
  - Asocijativnost L2 >= Asocijativnost L1
- Olakšava održavanje koherencije
  - Samo L2 nadgleda magistralu (*snooping*)
  - Bit inkluzije u L2 (i) za svaki blok

Inkluzija

- Primer
  - L2 duplo veći, ista veličina bloka, 2 bloka po setu, LRU
  - A, B i C se mapiraju u isti L1 set





Primer:

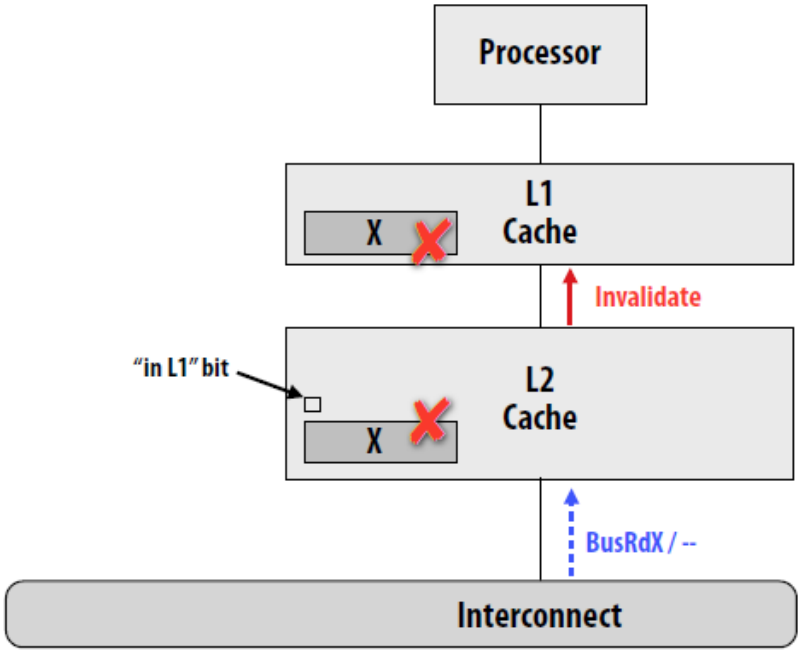
- Pristup A (L1+L2 *miss*)
- Pristup B (L1+L2 *miss*)
- Više pristupa A (L1 *hit*)
- Pristup C (L1+L2 *miss*)
- L1 zamenjuje B, a L2 zamenjuje A (zbog LRU)
- Inkluzija narušena!

Inkluzija

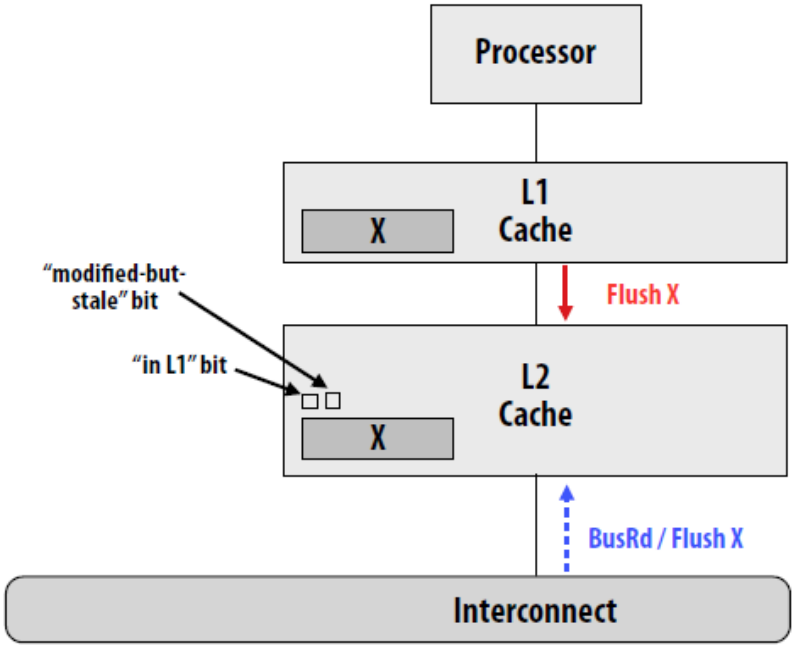
- Propagacija modifikovanog stanja iz L1 u L2
  - Za WT u L1 – automatski
  - Za WB u L1 – *inv* (potrebno stanje *dirty-invalid*)
- Transakcije između nivoa
  - L1 -> L2
    - Promašaji pri upisu i čitanju
    - Upis u deljeni blok
    - Zamena
  - L2 -> L1 (samo ako je i = 1)
    - Invalidacije
    - *Write-back*
- Prednosti
  - Filtriranje saobraćaja prema L1
  - Mnogo manja potreba za duplim tagovima

Inkluzija

invalidacija



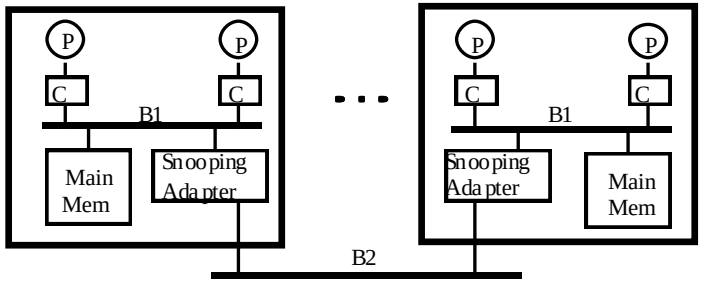
write hit



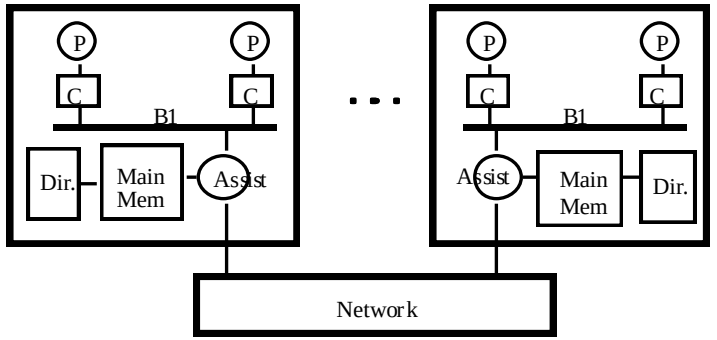
Dvonivoske hijerarhije

- Vrlo široko primenjivan pristup
- Čvorovi – multiprocesori manjeg obima
  - Čak i kao CMP
- Unutrašnji i spoljašnji protokol ortogonalni
  - Četiri moguće topologije
- Spoljašnji protolol obično "*directory-based*"
  - na nivou čvorova (ne procesora!)
- Unutrašnji protolol obično "*snoop*y-based"
- Primeri:
  - Convex Exemplar - *directory-directory*
  - Sequent, Data General, HAL - *directory-snoopy*
  - Encore Gigamax – *snoppy-snoopy*
  - Stanford Dash - *snoopy-directory*

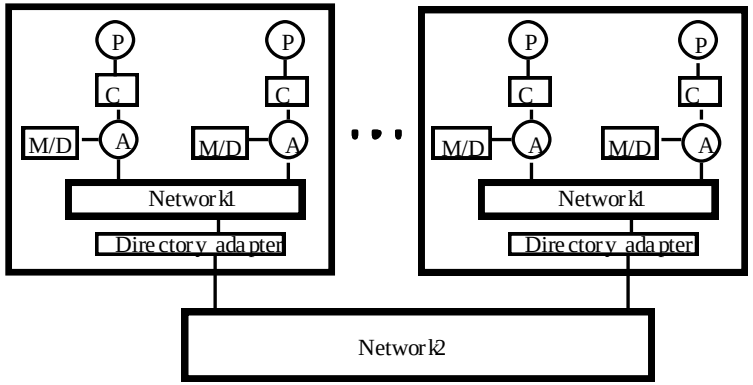
Dvonivoske hijerarhije



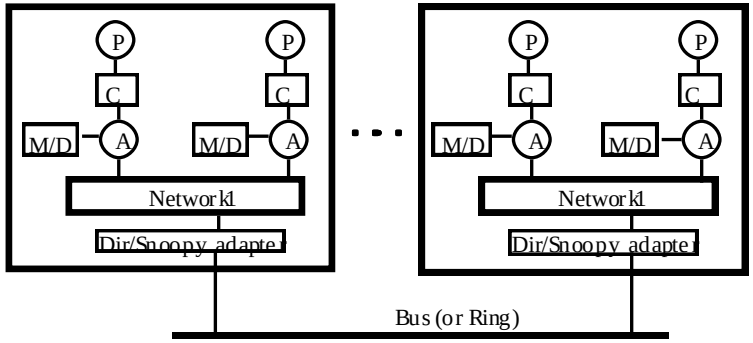
(a) Snooping-snooping



(b) Snooping-directory



(c) Directory-directory



(d) Directory-snooping

Dvonivoske hijerarhije

- Prednosti multiprocesnih čvorova
  - Amortizacija fiksne cene resursa čvora
  - Mogu se koristiti raspoloživi SMP-ovi
  - Manji potreban prostor za katalog
  - Dosta komunikacije lokalizovano u čvoru
  - Manje "udaljenih" promašaja ("*prefetch*" efekt u čvoru)
  - Kombinovanje zahteva
  - Moguće deliti keševe (preklapanje radnih skupova)
  - Dobici zavise od načina deljenja i mapiranja
    - dobro za intenzivno deljenje za čitanje
    - dobro za deljenje tipa "najbliži sused"
    - ne tako dobro za komunikaciju "svi sa svima"

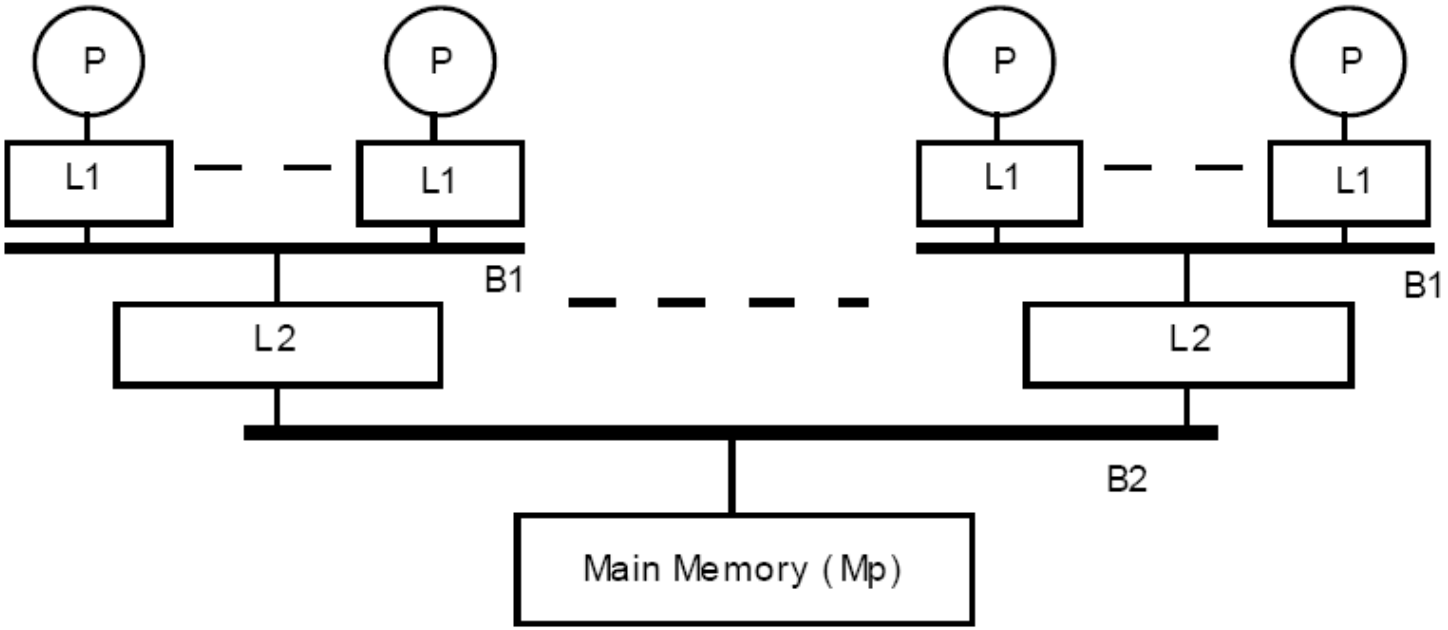
Dvonivoske hijerarhije

- Nedostaci multiprocesnih čvorova
  - Propusni opseg se deli u okviru čvora (npr. komunikacija “svi sa svima”)
  - Magistrala povećava latenciju ka lokalnoj memoriji
  - Čeka se na rezultat nadgledanja lokalne magistrale, pa se tek onda šalje zahtev za udaljenim pristupom
  - Magistrala u udaljenom čvoru povećava latenciju pristupa i smanjuje propusni opseg
  - Ako podaci nisu mapirani na čvorove u skladu sa načinom deljenja – degradacija perfromansi

Hijerarhijski *snoopy* protokoli

- Hijerarhija magistrala i keš memorija
  - Prirodan način da se reši ograničena skalabilnost jednonivoskog sistema
  - Procesori (sa keš memorijama) samo na prvom nivou
  - Na višim nivoima keš memorije i/ili katalozi
- Dve vrste sistema
  - Sa centralizovanom globalnom memorijom
  - Sa distribuiranim lokalnim memorijama

Hijerarhije sa globalnom memorijom



- Prvi nivo
  - Obično SRAM keševi najboljih performansi
  - Na B1 standardni "*snoopy*" protokol

Hijerarhije sa globalnom memorijom

- Drugi nivo
  - Mnogo veće set-asocijativne keš memorije (može DRAM)
  - Mora da se održava inkluzija u hijerarhiji
  - I = 1 ako bilo koji L1 ima validnu kopiju bloka
  - Filtrira saobraćaj ka prvom nivou

I	D	Akcija na L2	Akcija na L1
0	-	*	-
1	0	*read*	-
1	0	*write* (ili *inv*)	*inv*
1	0	zamena	*inv*
1	1	*	*write-back*

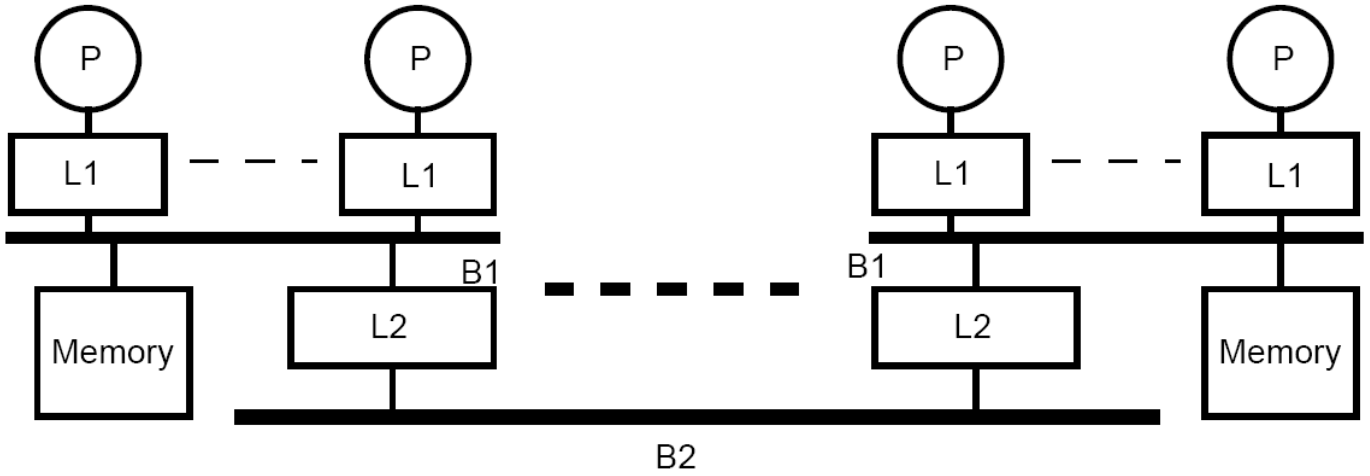
Primer dvonivoskog protokola

- Karakteristike
  - Promašaji zahtevaju prolazak hijerarhijom do glavne memorije (čak i za lokalne podatke)
  - Manje osetljivo na raspored podataka
  - Lakše programiranje
  - Memorija mora biti preklopljena zbog propusnog opsega
- Primer dvonivoskog protokola
  - Računa se na "geografsku" lokalnost (procesori koji intezivno dele podatke - na istoj magistrali)
  - Intenzivnije deljenje -> ažurirajući protokol na B1 (npr. Dragon + *invalid* stanje)
  - Slabije deljenje -> invalidacioni protokol na B2 (npr. MESI)
  - Interfejs između nivoa – *write-back* i invalidacija

Alternativa – mreža magistrala

- Wisconsin Multicube
  - 2-D mreža magistrala (redovi i kolone)
  - Memorijski moduli vezani za kolone
  - Na svakom preseku - PE + L1 keš + L2 keš
  - Kontroler L2 keša nadgleda obe magistrale u preseku
  - Tabela modifikovanih blokova za svaku kolonu
- Akcije protokola
  - RM se preusmerava ka modulu na matičnoj koloni ili koloni na kojoj se nalazi modifikovana kopija
  - WM se takođe preusmerava
    - na matičnu kolonu za nemodifikovan blok (inv po toj koloni se zatim preusmeravaju na vrste)
    - na kolonu sa modifikovanom kopijom (podaci se vraćaju preko vrste)
  - Ažuriranje tabele kod WM i WH, kao i pri zameni

Hijerarhije sa distribuiranom memorijom

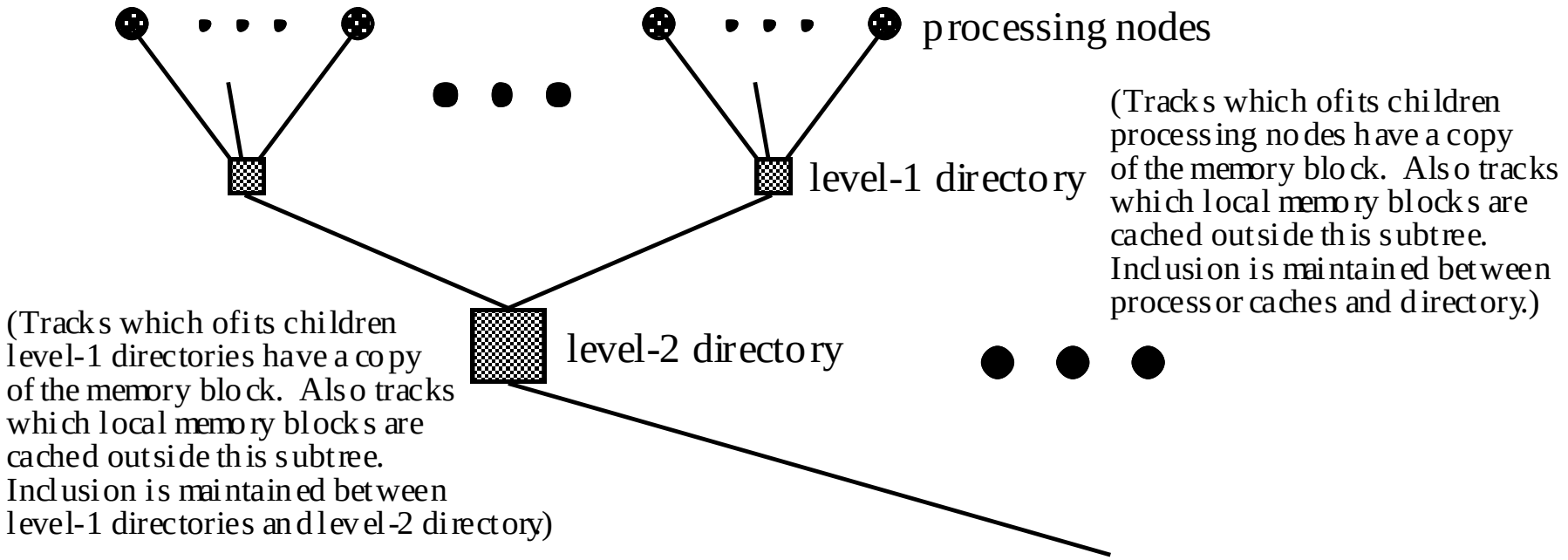


- Memorija raspodeljena između čvorova
  - Smanjuje saobraćaj na B2
  - Smanjuje latenciju (manja kontencija i brži lokalni pristup)
  - Umesto L2 može biti samo kontroler i katalog (*mapping* MLI)
    - Za podatke alocirane unutra, a keširane spolja
    - Za podatke alocirane spolja, a keširane unutra

Hijerarhije sa distribuiranom memorijom

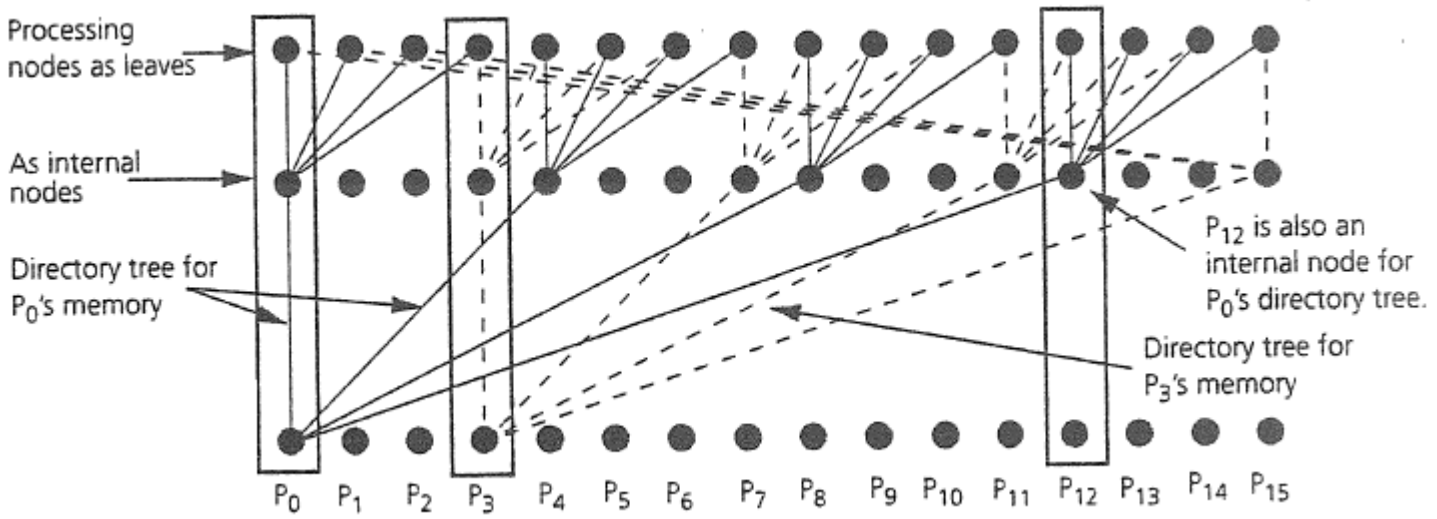
- Zahtev za čitanjem
  - Ako se ne zadovolji na B1, izbacuje se na B2
  - Ako neki L2 ima validnu kopiju, odmah je izbacuje na B2
  - Ako je modifikovana, postaje deljena (objava na B1)
  - Ako je nema u L2, mora da se dobije sa B1
- Upisi
  - Objavljuje se na B1 u svom čvoru
  - Postavlja stanje *dirty-invalid*
  - Ako je alociran ili keširan izvan, objavljuje se i na B2
  - Invalidacija sa B2 na B1 u čvoru koji ima kopije
- Potrebne neblokirajuće keš memorije (*lockup-free*) koje mogu da odrađuju više zahteva istovremeno

Hijerahijski *directory* protokoli



- Hijerarhijska struktura kataloga za svaki blok
  - Listovi - PE + memorija
  - Interni čvorovi – samo informacije iz kataloga
  - Logička hijerarhija, ne mora biti i fizička
    - može se ugraditi u mrežu opšteg tipa

Hijerahijski *directory* protokoli



- Hijerarhijska struktura kataloga za svaki blok
  - Čvor je koren logičkog stabla kataloga za blokove u pridruženom memorijskom modulu
  - ... ali može biti interni čvor ili list stabla za druge blokove
  - Ulaz kataloga govori da li je blok keširan u podstablima, kao i da li su lokalni memorijski blokovi keširani izvan

Hijerahijski *directory* protokoli

- Sadržaj kataloga i akcije
  - Svaki PE je list za neke blokove, ali i interni čvor za druge
  - Katalog se traži porukama prema gore, a ne direktno
  - Kao hijerarhijski "*snooping*", ali sa usmerenim porukama
  - Primer – DDM (Data Diffusion Machine)
- Usmerene poruke od čvora do čvora
  - Kod RM, zahtev ide nagore do prvog čvora koji u nekom podstablu ima kopiju ili matični čvor (a blok nije *dirty*)
  - Kod WM, zahtev ide nagore dok se ne nađe podstablo sa vlasnikom (matični čvor ili *dirty* kopija)
  - Odgovor unazad po istoj putanji

Hijerahijski *directory* protokoli

- Prednosti
  - Pronalaženje bliže kopije, a ne iz udaljenog matičnog čvora
  - Kombinovanje zahteva od različitih čvorova u pretku
- Nedostaci
  - Broj poruka po hijerarhiji povećava potrebni propusni opseg
  - Pretraživanje i održavanje kataloga povećava latenciju
- Dobro skaliranje prostora za katalog - $O(C \log_b P)$
  - Katalozi kao keševi
  - Cena skoro konstantna po nivou (veličina x broj)