Multiprocesorski sistemi

Directory protokoli

Marko Mišić, Milo Tomašević

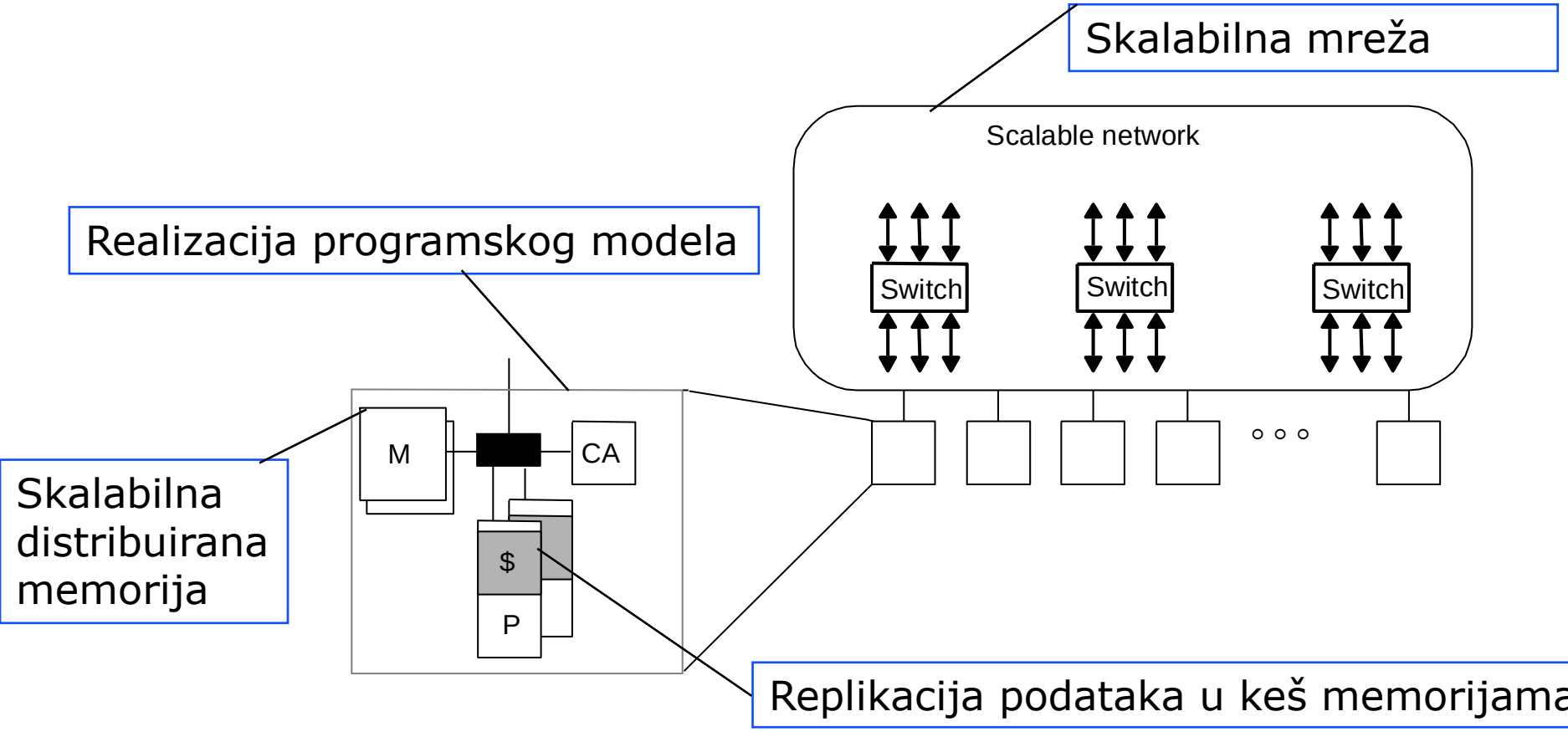
13S114MUPS, 13E114MUPS, 13M114MUPS

2024/2025.

Skalabilni sistemi (1)

- Sistemi sa zajedničkom magistralom
  - Ograničen propusni opseg
  - Ograničen broj procesora
  - Primerena upotreba osluškujućih protokola
- Skalabilni sistemi sa više čvorova
  - Značajno više čvorova
  - Memorija distribuirana po čvorovima
    - Replikacija podataka u lokalnim keš memorijama
  - Skalabilana interkonekciona mreža
    - Omogućava više istovremenih transakcija
  - *Communication Assist* (CA) – komunikaciona podrška
    - Kroz protokole realizuje programski model interpretacijom transakcija na mreži
- Zahtevi koherentnog sistema?

Skalabilni sistemi (2)



Zahtevi koherentnog sistema

- Skup stanja, dijagram prelaza i akcije
- Akcije:
  - (a) Odluka kada pozvati akcije protokola
  - (b) Pronaći informaciju o stanju drugih kopija podatka
    - Da li je potrebna komunikacija?
  - (c) Locirati druge kopije
  - (d) Komunikacija sa drugim kopijama (*inv/upd*)
- (a) slično na svim sistemima
  - Na osnovu lokalnog stanja kopije u keš memoriji
  - Akcije protokola se pozivaju ako je ugrožena koherencija
- Različite strategije za (b) - (d)

Zahtevi skalabilnog koherentnog sistema

- U sistemu sa zajedničkom magistralom akcije (b) - (d) zasnovane na
  - "*broadcast*"-u (jeftina operacija!)
  - "*snooping*"-u (najčešće ne rezultira u akciji – *spam*!)
- U skalabilnim sistemima koji nisu zasnovani na zajedničkoj magistrali (CC-NUMA)
  - "*Broadcast*" je skupa operacija (često *n* zasebnih poruka)
  - Neskalabilno rešenje njegova upotreba
- Skalabilni protokol
  - Ista ili slična stanja i dijagram prelaza
  - ... ali različita organizacija informacija i implementacija akcija
- Rešenje zasnovano na katalozima (*directory*)

Directory protokoli – principi (1)

- Katalog čuva informacije o koherenciji
  - Čuvaju se objedinjene informacije o stanju svakog keširanog bloka
- Akcije zasnovane na sledećim principima:
  - (b) Eksplicitna informacija se nalazi u memoriji
  - (c) Lociranje drugih kopija iz ulaza kataloga
    - Direktno ili indirektno u više koraka u zavisnosti od organizacije kataloga
  - (d) Komunikacija se ostvaruje "*unicast*" porukama na poznati skup odredišta
    - Poruku dobija samo onaj keš kontroler koga se to tiče
    - Skalabilno rešenje (bez spama!)
    - Ne koriste se niti *broadcast*, niti *snooping* tehnika

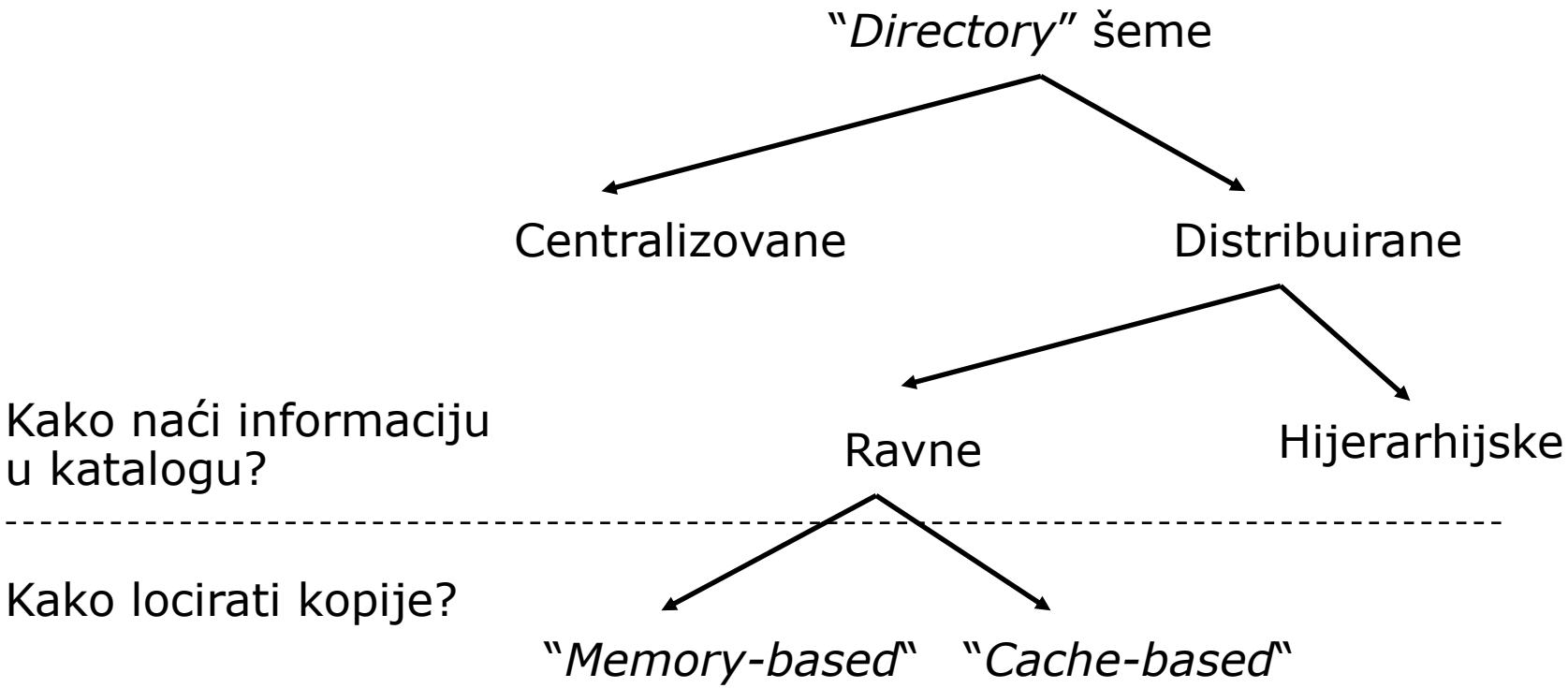
Directory protokoli – principi (2)

- Primer scenarija za RM:
  - Desi se promašaj pri čitanju u lokalnoj keš memoriji
  - Ode se u katalog i sazna gde se nalazi ažurna kopija
    - Može biti u memoriji ili kod nekog drugog vlasnika
  - Dobavlja se ažurna kopija od vlasnika
- Primer scenarija za WH:
  - Ode se u katalog i sazna da je kopija u deljenom stanju
  - Sazna lokacija deljenih kopija u odgovarajućim keševima
    - Vraća se informacija onome ko želi da upisuje
  - Šalju se poruke za invalidaciju ili ažuriranje deljenih kopija
    - U zavisnosti od izabrane strategije

Directory protokoli – principi (3)

- Odgovornost za održavanje koherencije:
  - Raspodeljena odgovornost na više aktera
  - Na centralizovanom kontroleru uz katalog
    - Vrši i održavanje kataloga
  - ... ali i na lokalnim kontrolerima (obično CA)
    - Ponekad čitav procesor zadužen za sprovođenje protokola
- Različite projektne odluke
  - Organizacija kataloga
  - Strategija protokola

Organizacija kataloga (1)



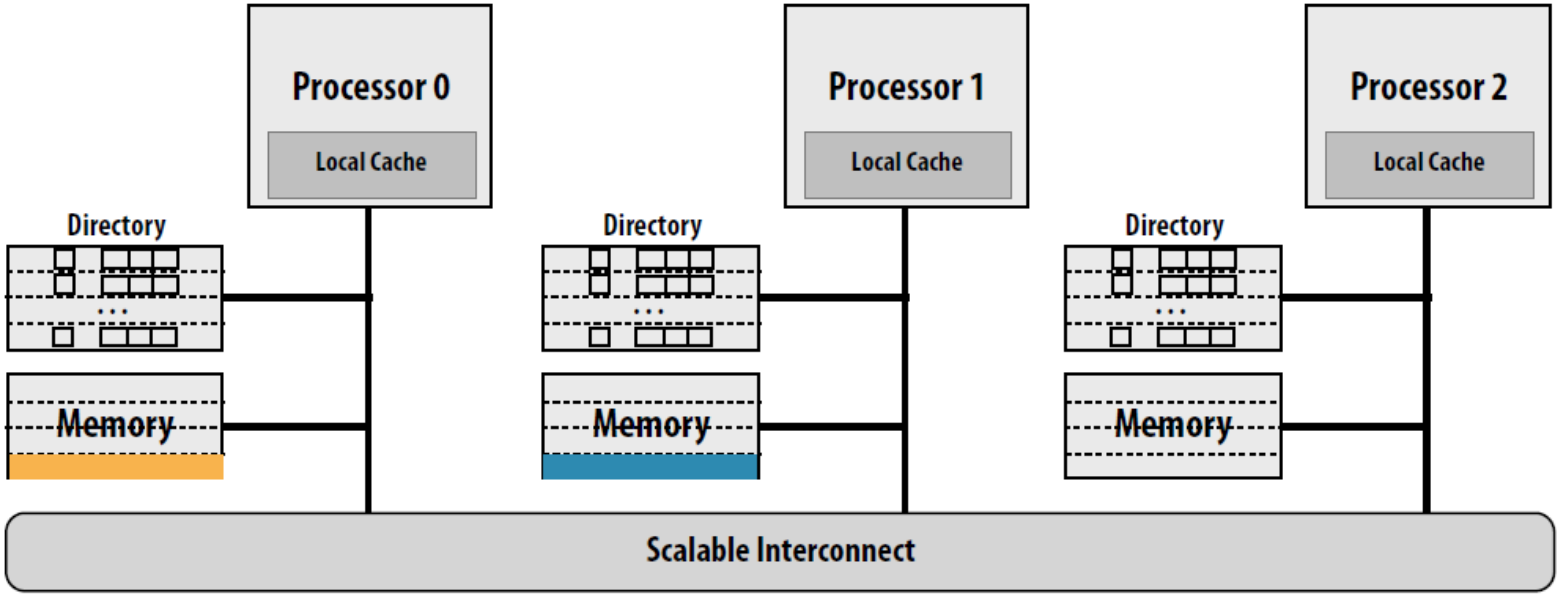
Kako naći informaciju u katalogu?

Kako locirati kopije?

Organizacija kataloga (2)

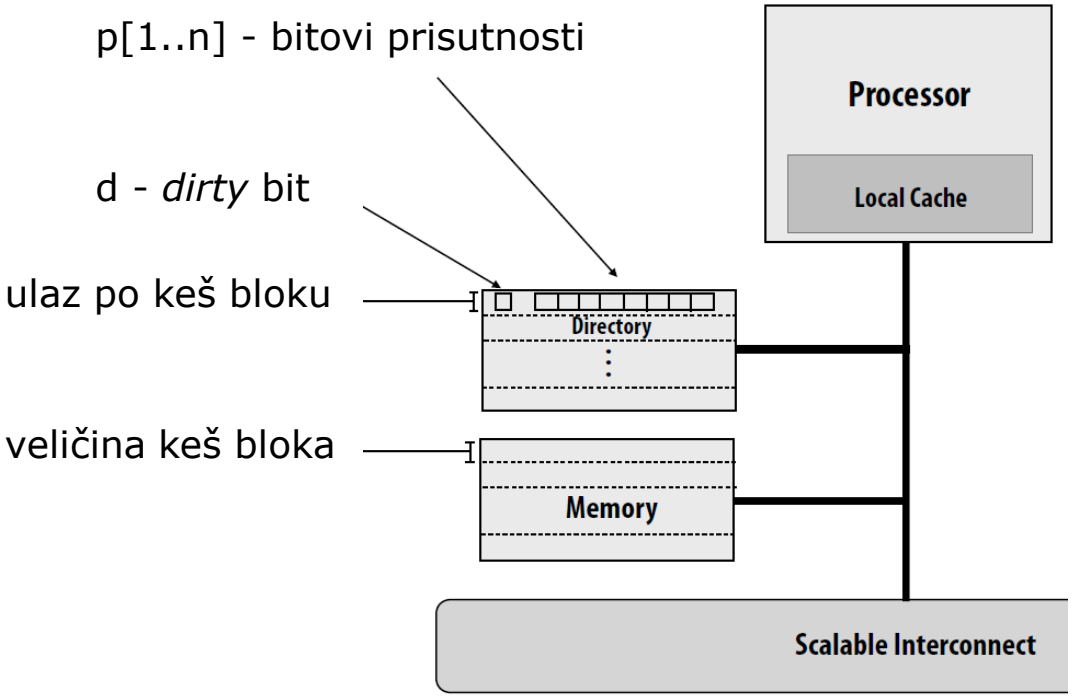
- Katalog - centralizovana informacija o stanju bloka
  - Jedan ulaz odgovara jednom keš bloku
  - Sadrži lokacije i stanje keširanih kopija tog bloka
- Centralizovana memorija i katalog
  - Lako pronalaženje, lakša serijalizacija
  - ... ali loša skalabilnost
- Distribuirana memorija i katalog
  - Ravne šeme
  - Hijerarhijske šeme
- Ravne šeme
  - Razlikuju se po performansama i zauzeću memorije
  - Ulazi kataloga samo u memoriji (*memory-based*)
  - Ulazi kataloga u memoriji i keševima (*cache-based*)

Directory protokoli - principi



- Ravne šeme
  - Katalog uz memoriju, obično u matičnom (*home*) čvoru
  - Pristupa mu se na osnovu adrese bloka (npr., heširanjem)
  - Poruka se šalje direktno matičnom čvoru (ako je udaljen)

Directory protokoli - organizacija



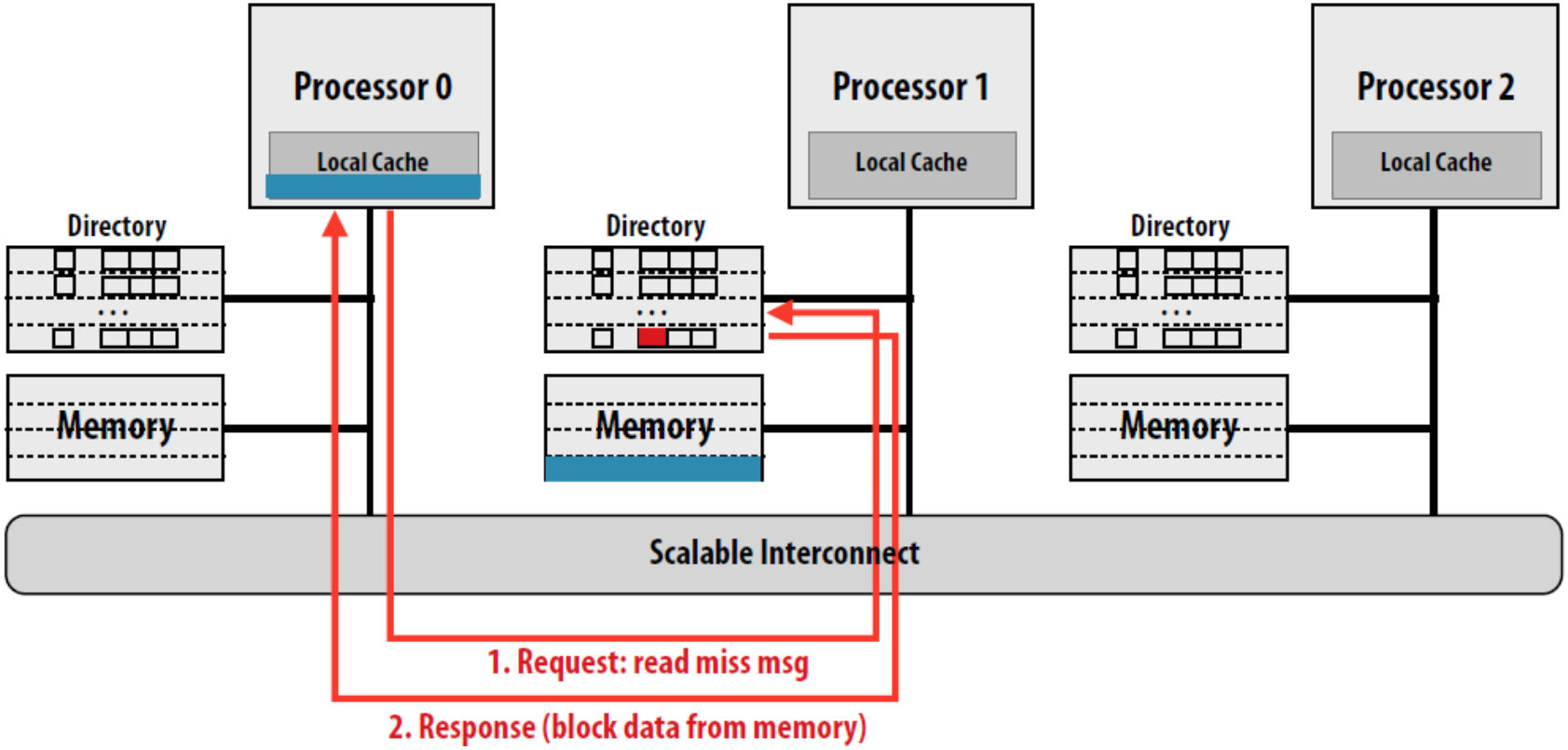
- Različita organizacija informacije o koherenciji, ali ista stanja
  - Ulaz "*full-map*" kataloga – vektor prisutnosti
    - *n+1* bit (*n* bitova prisutnosti + *dirty* bit) - *p[1..n] + d*
  - Dva bita (*valid* + *dirty*) uz blok u lokalnoj keš memoriji

Directory protokoli – operacije (1)

- *Pi*: RH B
  - Ako je *v = 1*, nema izlaženja na magistralu
  - Vrati se podatak procesoru
- *Pi*: RM B
  - Ako je *d = 0* u matičnom čvoru koji je u lokalnoj memoriji *Pi*
    - Izvrši se čitanje i dostavi blok
  - Ako je *d = 0* i ne nalazi se u lokalnoj memoriji *Pi*
    - Traži se matični čvor
    - M: send B -> Pi; *p[i] = 1*
  - Ako je *d = 1* i *p[j] = 1*
    - *j* (identitet vlasnika) -> *Pi* ; *Pi*:read *B* -> *Pj* ;
    - *Pj*: send *B* -> *Pi, M* ; *Mupd, p[i] = 1, d = 0* ; lokalna stanja *v = 1, d = 0*
    - Semantika deljenog stanja

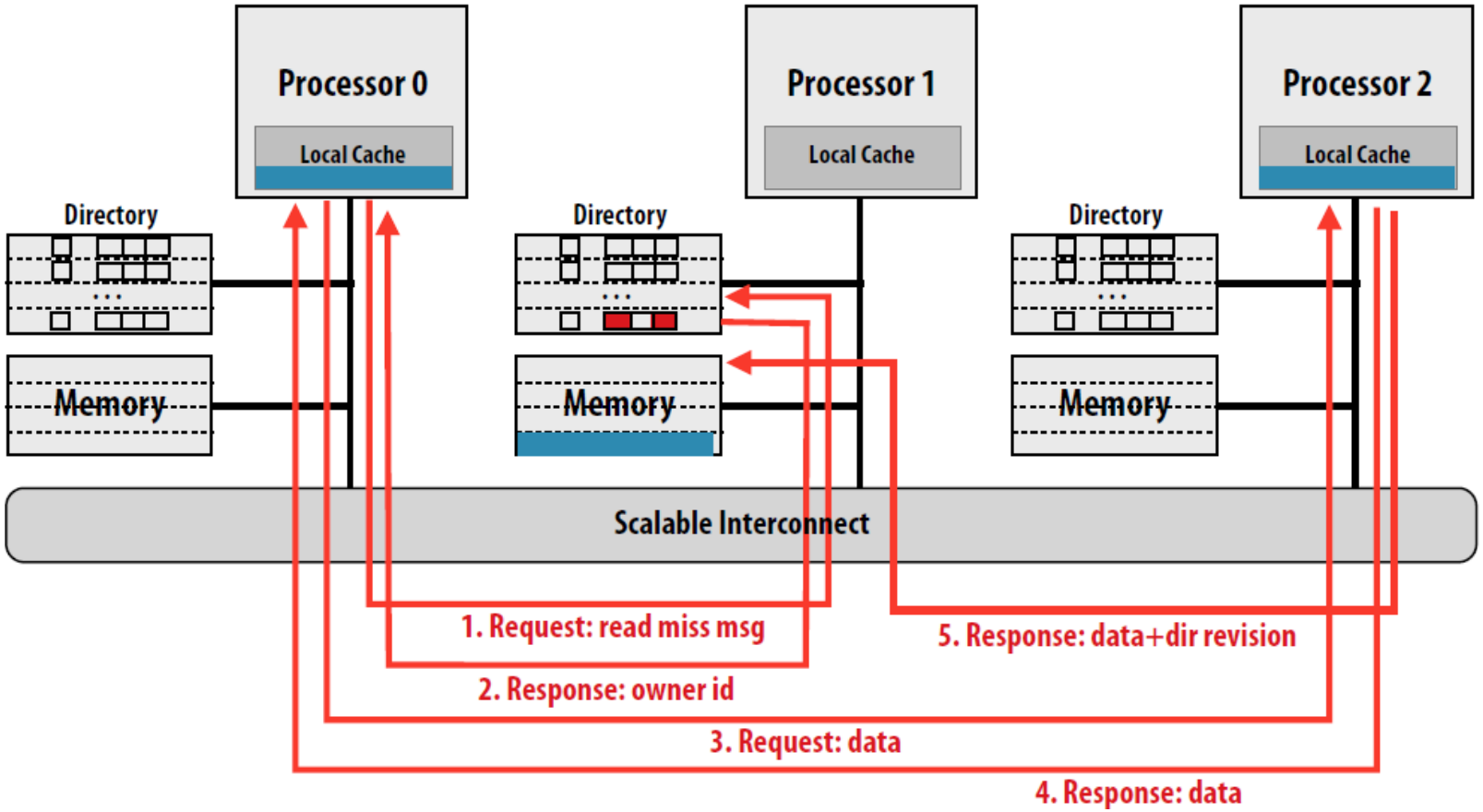
Directory protokoli – RM (1)

Promašaj pri čitanju, kopija u memoriji ažurna (d = 0)



Directory protokoli – RM (2)

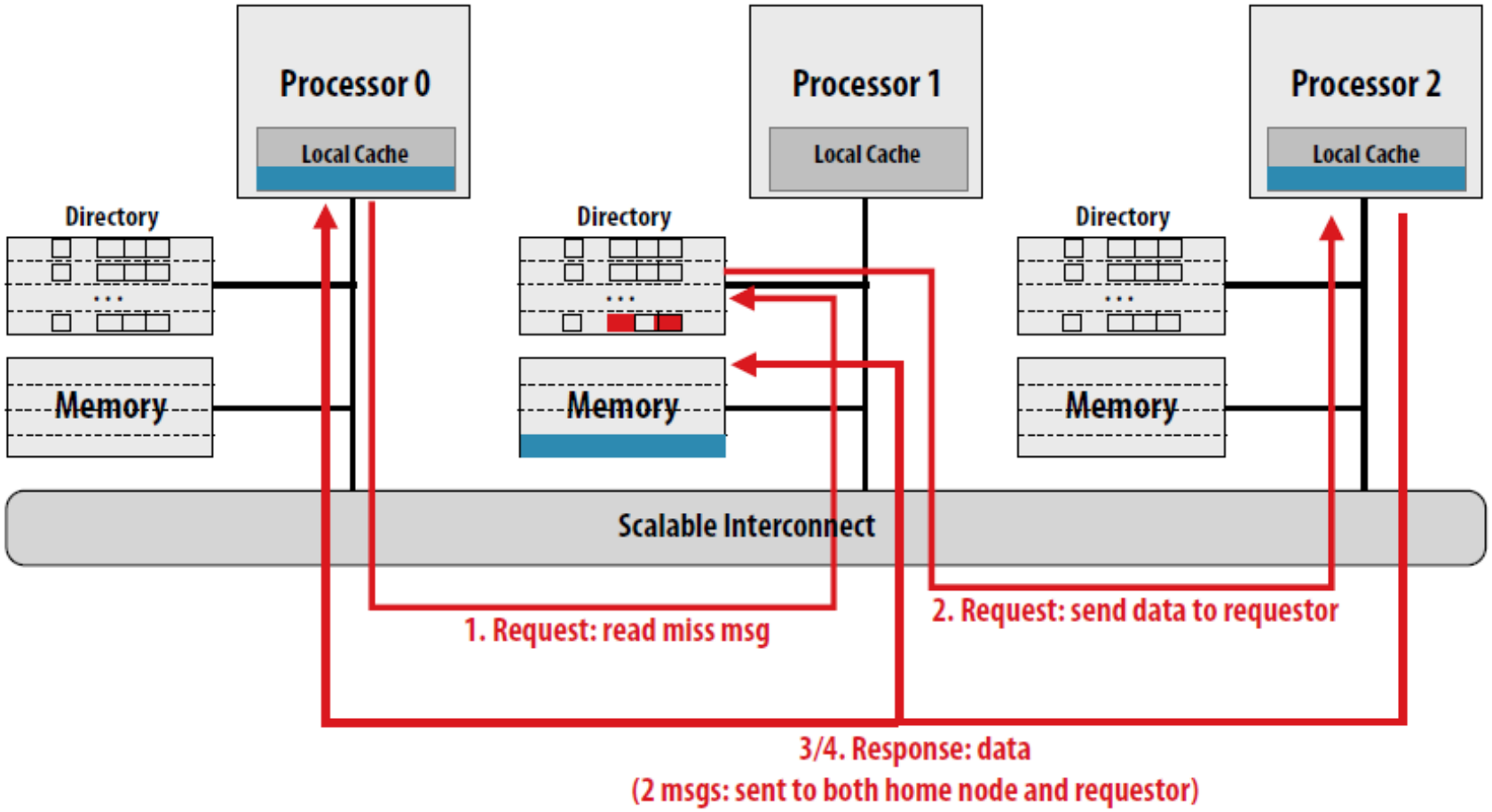
Promašaj pri čitanju, kopija u memoriji neažurna (d = 1)



Directory protokoli – RM (3)

- Performanse ove operacije
  - Broj poruka
  - Latencija – broj poruka na kritičnom putu (sekvenca zavisnih operacija da bi se opslužio zahtev)
- Od 5 poruka, 4 su na kritičnom putu
  - Poruke 4 i 5 mogu u paraleli
- Optimizacija
  - Prosleđivanje zahteva (*request forwarding*)
    - Direktno slanje poruke iz matičnog čvora vlasniku
  - Smanjuje broj poruka i latenciju
  - Od 4 poruka, 3 su na kritičnom putu (Poruke 3 i 4 mogu u paraleli)

Directory protokoli – RM (4)

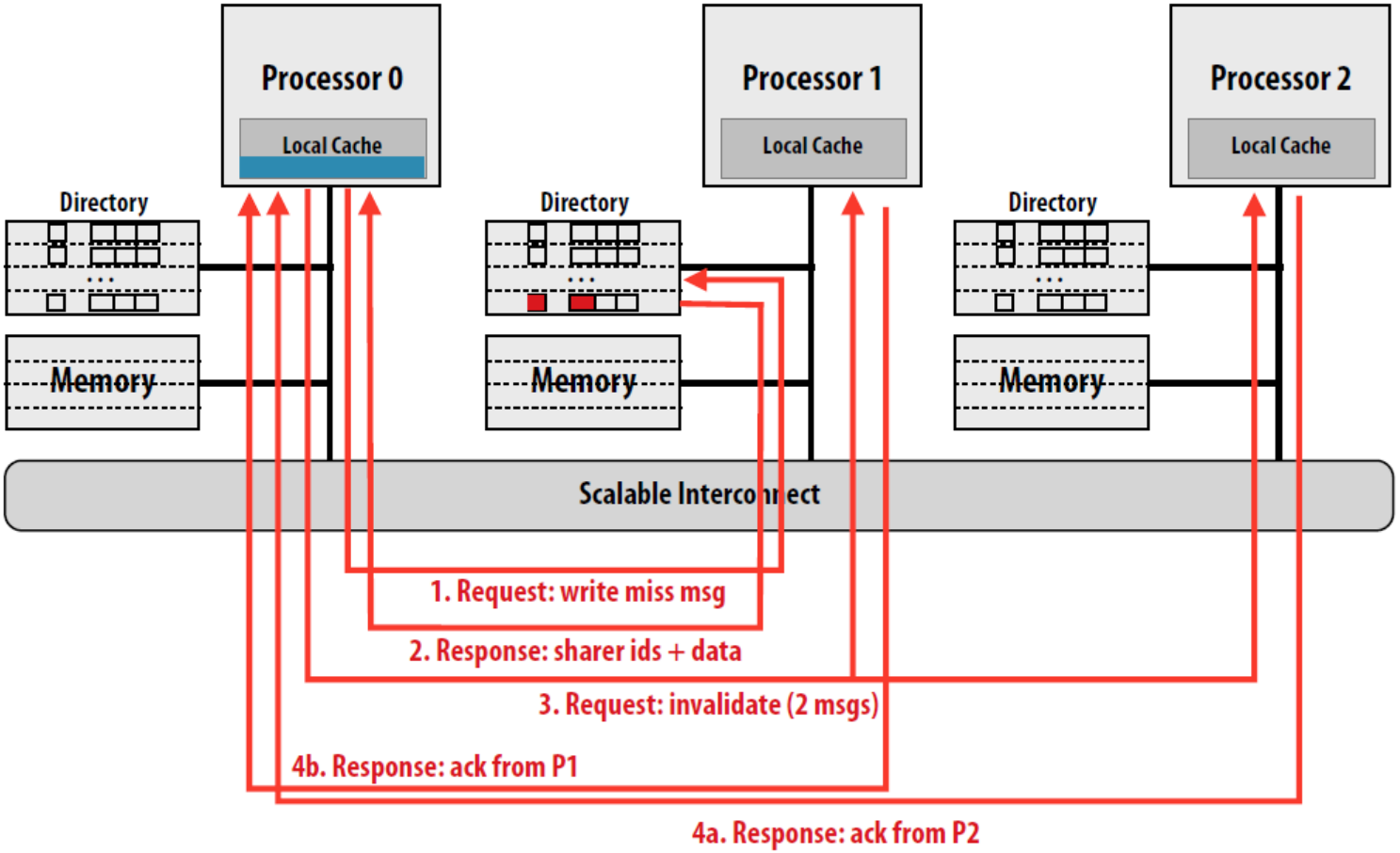


Directory protokoli – operacije (2)

- *Pi*: WM B
  - Ako je *d = 0* u matičnom čvoru
    - *M*: send *B , p -> Pi*; *p[1..n] = 0* osim *p[i] = 1 ; d = 1* ;
    - *Pi*: *inv -> Pk*, *p[k] = 1*; *Pk*: *v = 0*, *ack -> Pi*
    - *Pi*: write *B*, *v = 1*, *d = 1*
  - Ako je *d = 1* i *p[j] = 1*
    - Slično, samo *Pi* traži i dobija blok od *Pj*, koji ga zatim invaliduje
- *Pi*: WH B
  - Slično WM, ali se ne šalje blok (*upgrade*)

Directory protokoli – WM

Promašaj pri upisu, kopije u memoriji ($d = 0$), P1 i P2 ažurne



Directory protokoli – operacije (3)

- Zamena kod Pi
  - Ako je lokalni *d = 1*
    - Ažuriranje memorije, *p[i] = 0 ; d = 0*
  - Ako je lokalni *d = 0*
    - Može se (ali ne mora) obavestiti memorija (konzervativno!)
    - \+ Nema nepotrebnih *inv* poruka, oslobađa pointere u katalogu
    - \- Nepotrebno za podatke koji se samo čitaju, dodatni saobraćaj
- Implementacija složena
  - Serijalizacija mnogo složenija
  - Dosta prelaznih stanja (desetine ... , česti prelazi)
  - Mogući problemi "trke" kada informacija u katalogu nije sasvim ažurna (npr., pri zameni)
    - Vlasnik izbacuje blok, drugi traži dozvolu za upis
    - Rešava se odbijanjem (*NACK*)

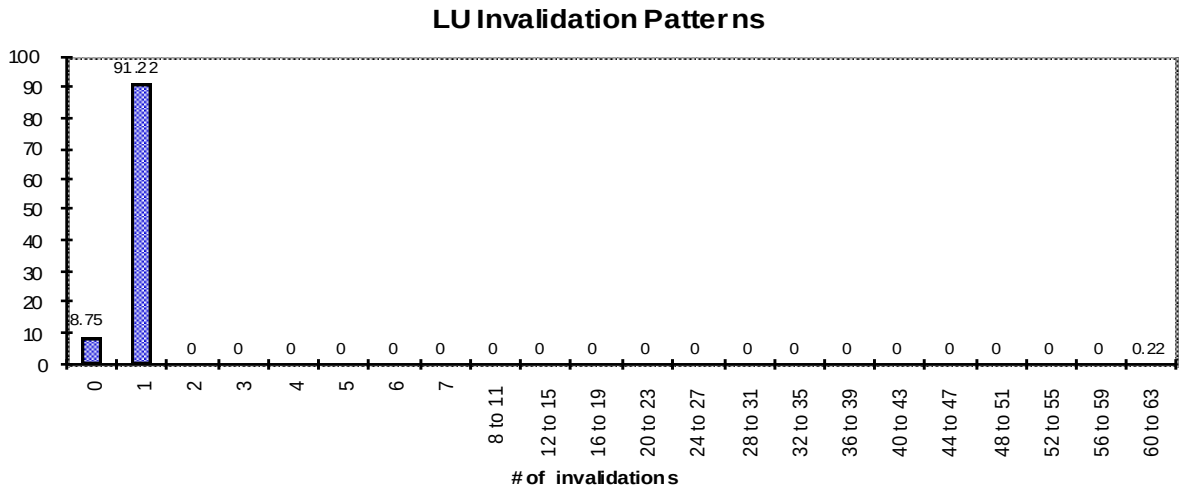
Directory protokoli - skaliranje

- Skaliranje propusnog opsega memorije i kataloga
  - Centralizovani katalog – usko grlo!
  - Kako distribuirati informaciju iz kataloga?
- Skaliranje karakteristika performanse
  - *Saobraćaj*: broj transakcija u mreži koje generiše protokol
  - *Latencija*: broj transakcija koje su na kritičnom putu
  - Neke transakcije mogu da se odvijaju u paraleli!
- Skaliranje prostora potrebnog za katalog
  - Broj bitova raste linearno sa brojem procesora
  - ... ali i veličina memorije
- Organizacija kataloga
  suštinska za skalabilnost protokola!

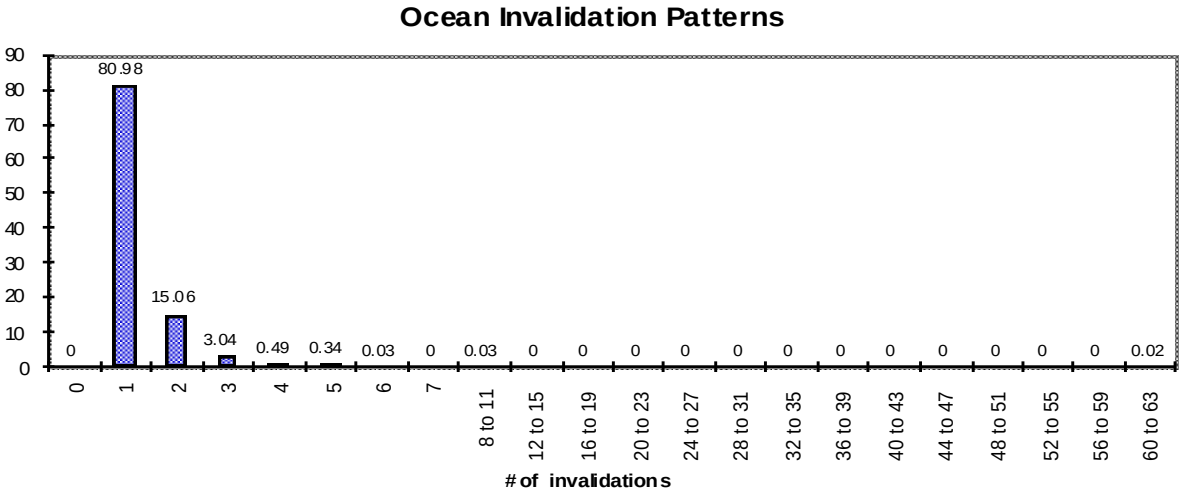
Directory protokoli - skaliranje

- Koliko se dobija izbegavanjem "*broadcast*"-a?
- Bitne inherentne karakteristike aplikacija
  - Frekvencija operacija kada su potrebne invalidacije
  - Prosečan broj invalidacija po takvoj operaciji
  - Kako ova dva indikatora skaliraju?
- Ukazuju na to kako organizovati katalog
- Simulacija sa aplikacijama iz SPLASH-2
  - Keš memorije beskonačne veličine
  - Sistemi sa 64 procesora
- Važan zaključak
  - Broj aktivno deljenih kopija obično dosta mali!
  - Sporo raste sa povećanjem broja procesora

Načini deljenja u aplikaciji (1)

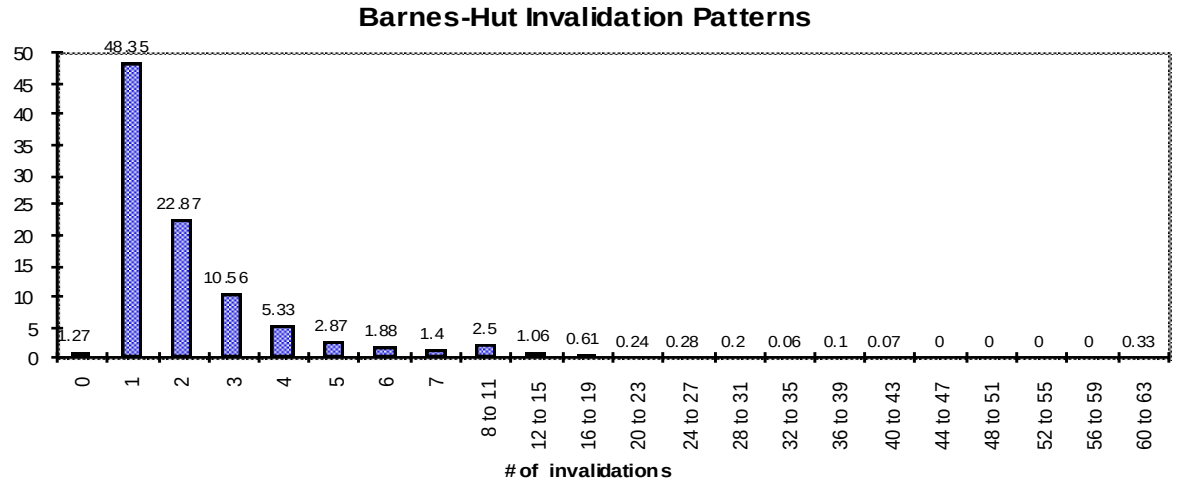


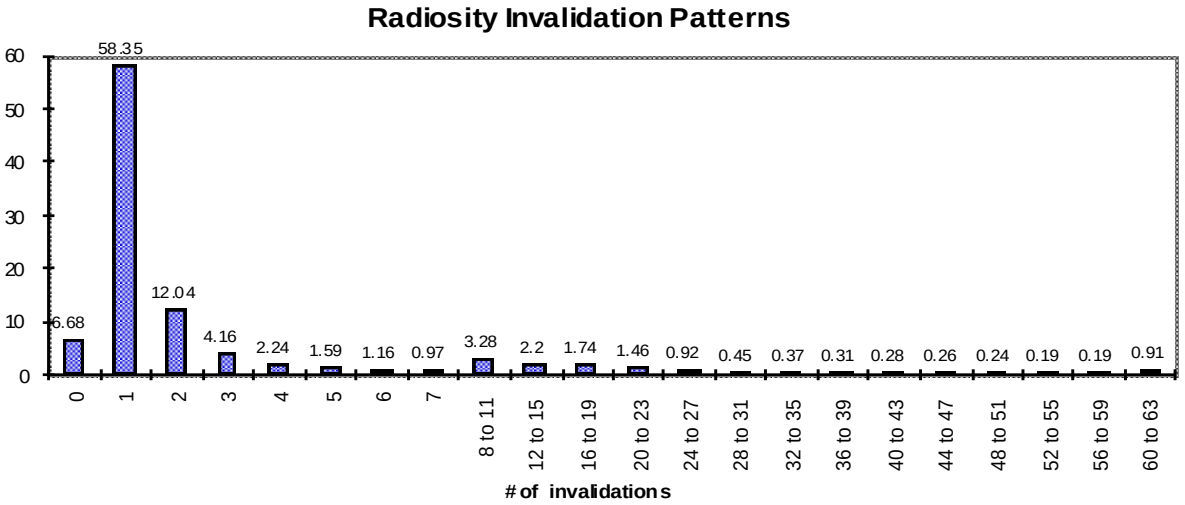
Migratorna deljivost



Nema širokog deljenja podataka

Načini deljenja u aplikaciji (2)





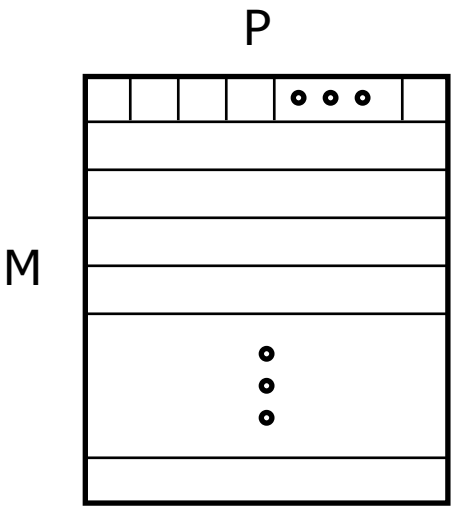
Nema širokog deljenja podataka, niti prevelikog broja invalidacija

Načini deljenja u aplikaciji

- Načini deljenja sa aspekta broja invalidacija
  - Kod i "*read-only*" objekti – nema invalidacija
  - Migratorni objekti – 1-2 invalidacije po upisu
  - "*Mostly-read*" objekti – dosta invalidacija, ali retko
  - Često upisivani i čitani objekti – invalidacije česte, ali malo deljenih kopija može da se napravi između njih
  - Sinhronizacioni objekti – ako je velika kontencija za njih, neophodna posebna SW ili HW podrška
- U multiprogramskim serverima problem još lakši
  - Migracija procesa nameće samo jednu invalidaciju
- Zaključci analize
  - Skaliranje saobraćaja i latencije nije loše
  - Moguće smanjiti veličinu kataloga

Ravne "*memory-based*" šeme - skaliranje

- Informacija o keširanim kopijama u matičnom čvoru
  - Slično centralizovanim šemama, ali distribuirana
- Skaliranje performanse
  - Saobraćaj pri upisu: srazmeran broju deljenih kopija
  - Latencija upisa: invalidacije mogu da se šalju u paraleli
  - Performanse najbolje
- Skaliranje prostora u "*full-map*" šemi
  - Prostorni "*overhe*ad" za blok od 64 bajta:
    - P = 64 -> 12.5% (64 bita za pointere, 8 bajtova)
    - P = 256 -> 50% (256 bita za pointere, 32 bajta)
    - P = 1024 -> 200%
      (1024 bita za pointere, 128 bajta!)
  - Prostorni "*overhe*ad" – O(P*M)
  - Loša skalabilnost

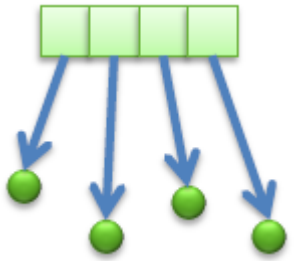


Smanjivanje prostora za katalog (1)

- Kako smanjiti "visinu" M i "širinu" P ?
  - M – broj blokova, P – broj procesora
- Optimizacije za osnovnu "*full-map*" tehniku
  - Povećanje veličine bloka smanjuje broj ulaza, ali povećava lažno deljenje
  - Multiprocesorski čvorovi (1 bit/MP čvor, ne 1 bit/PE!)
  - Opet skalira kao P*M, ali ipak prihvatljivo za velike mašine
    - P = 256, 4 PE čvor, 128 B blok -> 6.25% overhead
- Smanjivanje "visine" kataloga M
  - Broj memorijskih blokova >> broj keširanih blokova
  - Mnogi ulazi kataloga se ne koriste u datom vremenu
  - Umesto rezervisanja ulaza za svaki memorijski blok, organizovati katalog na principu keš memorije

Smanjivanje prostora za katalog (2)

- Smanjivanje "širine" kataloga P
  - Za većinu blokova samo mali broj keširanih kopija (često do 5)
  - Umesto posebnog bita za svaki PE, ulaz kataloga sadrži manji broj pokazivača (svaki log P bita) na deljene kopije
  - Npr., P=1024 => 10-bitni pointeri,
    čak i 100 pokazivača malo štedi prostor
  - Analize načina deljenosti ukazuju
    da je nekoliko pokazivača dovoljno (često do 5)
  - Neophodna strategija prekoračenja
    kada ima više deljenih kopija nego pokazivača



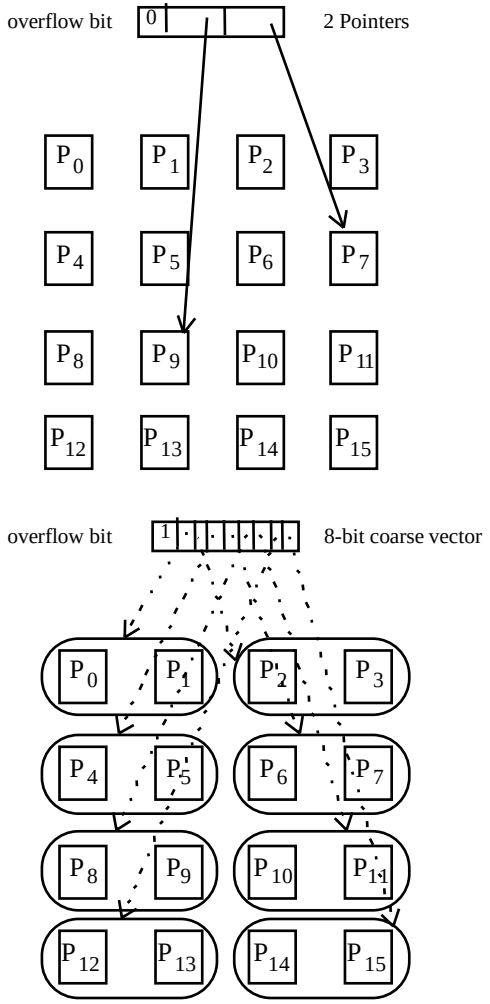
- *Dir$_i$ X* šeme sa ograničenim brojem pokazivača
  - Ograničen broj pointera *i* po ulazu (obično *i* << P)
  - *X* - strategija prekoračenja (*pointer overflow*)
  - Skalabilne po veličini kataloga – O(M*log P)
  - Degradacija performansi kod intenzivne deljenosti

Smanjivanje širine kataloga (1)

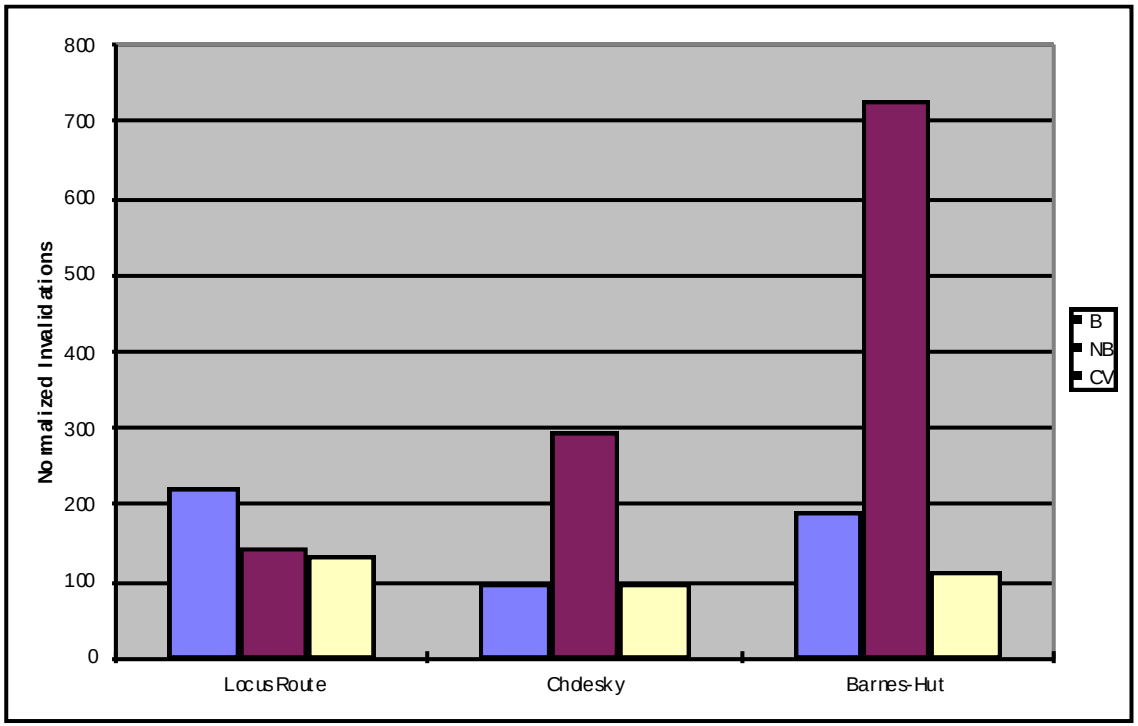
- Dir_i *NB* (Agarwal et al.) – *non-broadcast*
  - Pri prekoračenju pointera, oslobađa jedan pointer invalidacijom jedne deljene kopije (RR, LRU ili sl. izborom)
  - Izbegava "*broadcast*"
  - Ograničava broj deljenih kopija, čak i za čitanje
  - Degradacija performansi kod intenzivne deljenosti
- Dir_i *B* (Agarwal et al.) – *broadcast*
  - "*Broadcast*" bit (b) se postavlja pri prekoračenju pointera
  - Kada je b=1, naredni upis rezultuje "*broadcast*"-om invalidacija svim procesorima (mnoge nepotrebne!)
  - Ne ograničava broj deljenih kopija za čitanje
  - Jednostavna šema, ali ...
    - povećava saobraćaj u mreži
    - povećava latenciju upisa

Smanjivanje širine kataloga (2)

- $Dir_i\ CV_r$ (Gupta et al.)
  - Šema sa "grubim" (*coarse*) vektorom
  - Dinamička reorganizacija ulaza
  - Format ulaza zavisi od broja kopija
    - *i* pointera do prekoračenja, a onda
    - ... svaki bit označava grupu od *r* PE
  - $r = p/(i*\log p)$
  - Bit se setuje ako bilo koji PE ima kopiju
  - Invalidacije ograničene na grupu PE (nema "*broadcast*"-a, već *multicast*-a)
  - Neke mogu biti nepotrebne
  - Pogodne za ...
    - Multiprogramska okruženja
    - Aplikacije sa dobrom lokalnošću podataka



Smanjivanje širine kataloga - evaluacija



- 64 PE, 4 pointera
- Broj invalidacija normalizovan prema *full-map* šemi

Smanjivanje širine kataloga (3)

- *Dir$_i$ SW*
  - Pri prekoračenju pointera, SW prekid
  - Pamte se pointeri u lokalnoj memoriji i setuje OW bit
  - Pri upisu, ako je OW = 1 koriste se i zapamćeni pointeri
  - Nema nepotrebnog saobraćaja
  - Cena prekida i SW obrade
  - Povećava zauzetost PE, a time i latenciju i kontenciju
  - Npr., LimitLESS (MIT Alewife)
    - 5 pointera i lokalni bit
    - SW emulacija u heš tabeli
    - Podrška za višenitno izvršavanje
    - Latencija za čitanje iz udaljenog čvora 40-425 ciklusa
    - 84 ciklusa za 5 invalidacija (HW)
    - 707 ciklusa za 6 invalidacija (HW+SW)

Smanjivanje širine kataloga (4)

- *Dir_i DP – dynamic pointers*
  - Vrlo mali broj fiksnih pointera (može i bez njih!)
  - Lista slobodnih dinamičkih pointera u memoriji (*pointer/link store*)
  - Pri prekoračenju se dodaje kružna ulančana lista dinamičkih pointera
  - HW mehanizam + protokol procesor
  - Ako ponestane dinamičkih pointera, jedan se oslobađa
  - Problem pointera na kopije koje su zamenjene
  - Skalabilnost kataloga (P/L prostor srazmeran broju keširanih kopija)
  - Npr., Stanford FLASH

Smanjivanje visine kataloga (1)

- Broj keširanih blokova relativno mali u odnosu na ukupan broj
  - Vrlo mala iskorišćenost ulaza kataloga
- Retki (*sparse*) katalog
  - Katalog kao keš memorija
  - Alokacija ulaza dinamički samo kada ima keširane kopije
  - Manji broj ulaza, može i SRAM tehnologija
- Specifičnosti u odnosu na keš podataka
  - Nema *write-back* (invalidacije pri zameni)
  - Jedan ulaz po bloku (nema prostorne lokalnosti)
  - Obrađuje zahteva od svih procesora
  - Potrebna veća asocijativnost i dovoljna veličina da se smanje verovatnoća kolizije

Smanjivanje visine kataloga (2)

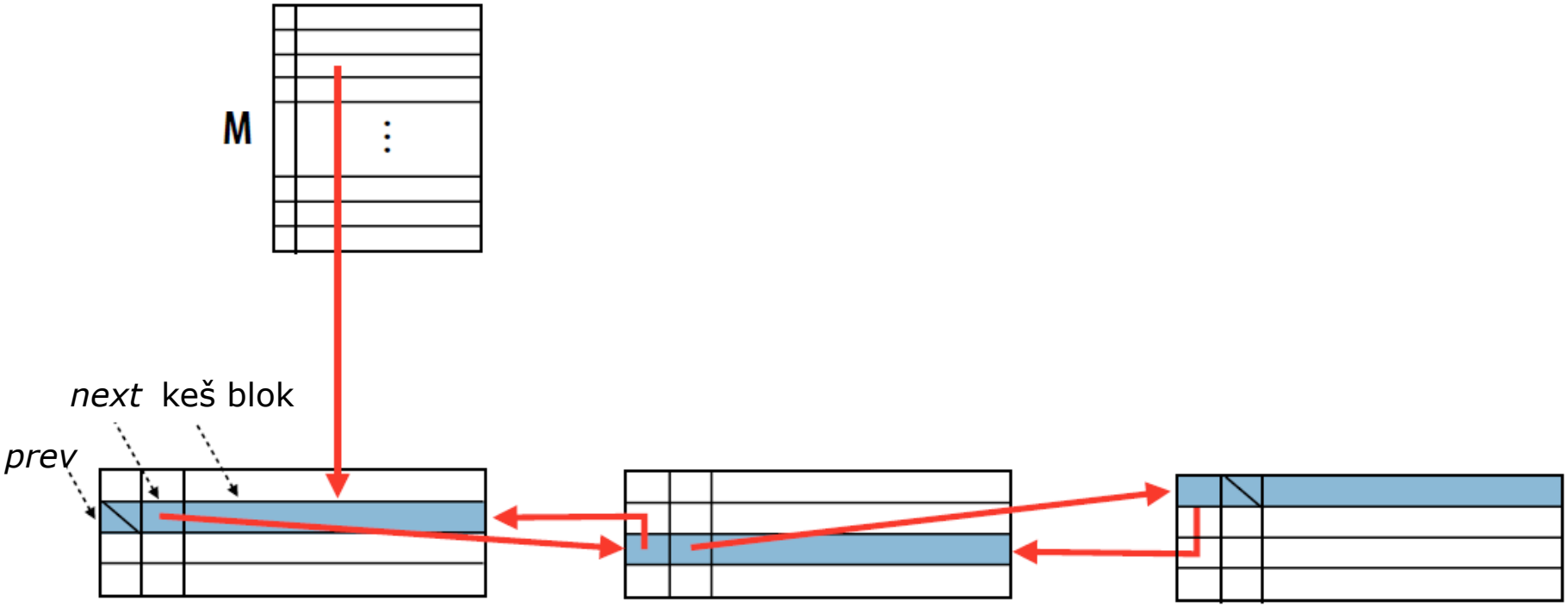
- Performanse
  - Smanjenje veličine kataloga za bar red veličine
  - Kolizije u katalogu donose 15-20% dodatnog saobraćaja
- Tehnike za smanjivanje širine i visine su ortogonalne
  - Mogu se primenjivati zajednički, istovremeno
- Kombinovani katalog
  - Dve asocijativne keš memorije:
    - Veći keš sa ulazima sa ograničenim brojem pointera (D1)
    - Manji keš sa ulazima u vidu vektora prisutnosti (D2)
  - Prvo se alocira ulaz u D1
  - Pri prekoračenju pointera alocira se ulaz u D2, a oslobađa ulaz u D1
  - Performanse mogu biti bliske "*full-map*" šemi

Ravne "*cache-based*" šeme

- Ulaz kataloga za jedan blok distribuiran
  - … u vidu ulančane liste
  - U matičnom čvoru samo zaglavlje liste (prva kopija)
  - Svaka kopija u kešu pokazuje na sledeću
- Čitanje
  - Blok se dostavlja iz memorije ili sa čela liste (ako je d=1)
  - Nova deljena kopija ide na početak liste
- Upis
  - Propagacija invalidacija (ili ažuriranja) kroz listu
  - Upisana kopija ide na početak liste
- Zamena
  - Izlančavanje iz liste (lakše u dvostruko ulančanoj listi)

Ravne "*cache-based*" šeme - primer

- Scalable Coherent Interface (SCI) IEEE Standard
  - Dvostruko ulančane liste (pokazivači *prev* i *next*)



Ravne "*cache-based*" šeme - skaliranje

- Prednosti
  - Manji prostor za katalog (manje pokazivača)
    - Broj mem. blokova * 1 + ukupan broj keš blokova * 2
  - Poredak pristupa poznat
  - "*Overhead*" invalidacija decentralizovan
  - IEEE standard
- Problemi
  - Latencija pri upisu srazmerna broju deljenih kopija
  - Povećana zauzetost
    (čak i promašaj pri čitanju angažuje prvi čvor)
  - Potrebna sinhronizacija pri zameni susednih kopija
  - Kompleksne implementacije (npr., NUMA-Q, Exemplar)