RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte

Datorvadības, automātikas un datortehnikas institūts

STUDIJAS DARBS

priekšmeta"" Mūsdienu datoru arhitektūra"

DAUDZ KODOLU ATTĪSTĪBAS

Izstrādājis: Igors Šemels
1. kurss, 1. grupa
091RDB344

2012./2013.m.g.

Saturs

1. NĀKAMAS PAAUDZES DAUDZ PLŪSMU MASVEIDA ARHITEKTŪRU PROJEKTĒŠANAS NEREGULĀRAM LIETOJUMPROGRAMMĀM.

Masveida daudz plūsmu arhitektūrā tieši tāpat, ka Cray XMT adrese vairāk ir vajadzīgi intensīvi datu lietojumprogrammas, nekā patēriņa grupas. Ir piedāvāta evolūcija XMT ar iebūvētiem daudz kodolu procesoriem un nākamas paaudzes savienojumiem kopā ar atminas atsauces vienību, lai optimizētu tīkla utilizāciju.

Tekošas augstas izpildes aprēķināšanas sistēmas ir projektētas lai efektīvi izpildīt peldēšanas intensitātes punktu darba slodzi. HPC sistēmas galvenokārt tiek veidotas priekš zinātniskas modelēšanas, kuras tiek raksturotas ar augsto aprēķināšanas blīvumu un novietojumu, un regulāru, dalošu datu struktūru. Prasības šai lietojumprogrammai ir procesora dizaina vadība attiecībā pret posteni SIMD(vienīgā instrukcija, daudzkārtēji dati) aritmētiskas vienības un dziļas ķešas hierarhijas, lai samazināt piekļūšanas pārklājumus.

Sistēmas līmenī, atmiņās un platums joslas savienojumi palielinās daudz lēnāk nekā skaitļošanas veiktspējas maksimums bet regularitāte un novietojums mazina šis problēmas ietekmi. Tajā pašā laikā jaunās procesora arhitektūras stum lietojumprogrammu attīstību attiecībā uz realizāciju, kuras var izmantot to īpatnības.

Tomēr, lietojumprogrammas no jaunām jomam, piemēram, bioinformātika, kopienas noteikšana, kompleksi tīkli, semantiskas datu bāzes, zināšanas atklājumi, dabiska valodas apstrāde, tēlu atpazīšana, un sociāla tīkla analīze, ir neregulāra rakstura. Tie parasti izmanto rādītāja bāzes datu struktūras, piemēram, nesabalansētus kokus, nestrukturētos režģus, un grafikus, kuri ir masveidā paralēli, bet satur sliktas telpas un uz laiku vietas novietojumu. Efektīva sadalīšana šiem datu struktūrām ir izaicinājums. Turklāt, datu struktūras bieži izmaiņā dinamisku veiktspēju, piemēram, pieliekot un pārvietojot savienojumus grāfiem.

Kompleksas ķešas hierarhijas ir neefektīvas ar tādam nepareizam lietojumprogrammām. Off - chip caurlaides spēja, kura ir pieejama sistēmas atmiņai, lai piekļūtu lokāliem datiem un tiklam, lai piekļūtu datiem uz citiem mezgliem, galvenokārt nosaka sistēmas veiktspēju. Saskaņā ar šiem nosacījumiem, viens kontrole pavediens parasti nepiedāvā pietiekamu vienlaicīgumu, lai izmantot visu pieejamo joslas platumu. Tāpēc, daudz plūsmu arhitektūras parasti cenšas paciest, nevis samazināt atmiņas piekļuves gaidīšanas laiku, pārslēdzoties starp vairākiem pavedieniem, nepārtraukti radot atmiņas atsauces un maksimizēt joslas platuma izmantošanu.

2. PĒTĪJUMA MOTIVĀCIJA

Cray XMT ir izstrādāts superdators ar daudziem mezgliem, kas paredzēts attīstībai un neregulāras programmas izpildei. Tās arhitektūra ir balstīta uz trim "stabiem": globālā adrešu telpa, detalizēti sadalīta sinhronizācija, un daudz plūsmas.

XMT ir izplatīta kopīga atmiņa (DSM) sistēma, kur kopīga globāla telpa vienmērīgi sadalīta uz ļoti smalkam detalizācijām, kas atrodas uz dažādam mezglu atminam. Katrs mezgls apvieno parasto ThreadStorm procesoru, kas pārslēdz, ciklu pa ciklu, starp daudziem aparatūras vītnēm. Tas ļauj sistēmas latentumam piekļūt lokālai atmiņai uz mezgliem, un tīkla latentuma piekļuvi atmiņai attālinātos mezglos.

Atšķirībā no jaunākajām HPC sistēmām, XMT nodrošina sistēmas mēroga plānošanas modeli, kas atvieglo programmu īstenošanu ar lielu atmiņas pēdu bez optimizācijas vides. Pat, ja mūsdienu HCP sistēmām integrēt daudz plūsmu arhitektūras, piemēram, grafikas apstrādes vienību (GPU), tie būtu labāk piemēroti regulārām programmām. Līdz šim tie nav paredzēti, lai pieļaut pārklājumam piekļūt atmiņai uz dažādiem mezgliem. Daudzos gadījumos viņi pat nevar piekļūt citu procesoru pārklājuma atmiņai vienā mezglā. Turklāt, koordinējot atmiņas piekļūšanu un maksimāli joslas platuma izmantošanu, prasa datu sadalīšanu un ievērojamas programmēšanas centienus.

Pacific Northwest valsts laboratorija ir CASS - MT projekts (http://cass-mt.pnnl.gov) pašlaik pēta masveidā daudz plūsmu arhitektūras nesimetriskas programmas. Šeit, mēs piedāvājam daudz plūsmu arhitektūras klasifikāciju un apspriežam to saistībā ar Cray XMT. Mēs pēc tam piedāvāsim priekšlikumus, kā attīstīt šīs arhitektūras un novērtēsim iespējamo nākotni XMT dizaina ar integrētiem vairākiem kodoliem vienā mezglā un nākamās paaudzes tīkla savienojumus. Visbeidzot, mēs parādīsim, kā integrācija aparatūras mehānisms tālvadības programmām var optimizēt tīkla izmantošanu.

3. DAUDZ PLŪSMU ARHITEKTŪRAS

Daudz plūsmu procesors vienlaikus izpilda instrukcijas no dažādiem kontroles pavedieniem, kuri ir vienā cauruļvadā. Ir divi pamata veidi daudzu plūsmu procesoriem: tie, kas izdod instrukcijas tikai no viena pavediena cikla, un tie, kas izdod instrukcijas no vairākiem pavedieniem vienā ciklā.

Daudzi uzlaboti ārpus kārtas superskalārie procesori, piemēram, Power6 IBM un Power7 vai jaunāka Intel arhitektūra, Nehalem un Sandy Bridge, atbalsta vienlaicīgu daudz plūsmu (SMT) tehniku. SMT tur vairākus pavedienus aktīvi katrā kodolā- procesors identificē neatkarīgas instrukcijas un vienlaikus emitē tos dažādām vienībām izpildei, tādējādi atbalstot procesora resursu augstu utilizāciju.

Daudz plūsmu procesoru, kas dot instrukcijas no viena pavediena katrā ciklā, pazīstami vēl, kā īslaicīgi daudzu plūsmu procesori, pārslēdzas starp dažādiem pavedieniem, lai turēt cauruļvadu piepildītu un izvārīties no stendiem. Pagaidu daudz plūsmu var būt rupj šķiedrains (bloka daudz plūsmu) vai smalkgraudains (instrukcija / cikls starplikām).

Bloku daudz plūsmu pārslēdzas no viena pavediena uz otru tikai tad, kad instrukcija ģenerē gara pārklājuma stendu, piemēram, buferatmiņas trūkums, kas prasa piekļuvi offchip atmiņai. Intel Montecito izmanto daudz plūsmu bloku.

Starplikas daudz plūsmu pārslēdzas no viena pavediena uz otru cikla pamatā. Ar ThreadStorm procesors no Cray XMT, kā arī viņu priekšgājēji no Tera MTA un Cray MTA -2, izmanto starpliku daudz plūsmu. SPARC kodoli no Sun UltraSPARC T1,T2 un T3 SPARC arī izmanto starpliku daudz plūsmu formu.

UltraSPARC T1 sastāv no astoņiem kodoliem, katrā ir četri pavedieni. T2 arī ir astoņi kodoli, bet divkāršojas izpildes vienības un diegu skaits, ļaujot kopīgi dot noradījumus no diviem diegiem, skaitot astoņus katrā pulksteņu ciklā. SPARC T3 divkāršo kodola skaitu attiecībā uz UltraSPARC T2. Šie kodoli dot komandas no citiem pavedieniem katrā ciklā. Kad garš pārklājuma notiek, tās ņem pavedienu, kurš izvēlēts no saraksta, un tas turpinās tik ilgi, kamēr notikums beidzas.

GPU arī integrē vairākus plānošanas blokus (deformācija vai viļņu fronšu), kuri var tikti efektīvi pārslēgti uz SIMD izpildi, lai uzturētu ilgas pārklājuma atmiņas darbības. GPU iezīme simtiem peldošo vienību un masveida atmiņas joslas, un tam ir onchip atmiņas, kas optimizē piekļūšanu pie bieži izmantotiem datiem. Tomēr, GPU pašlaik projektēts, kā paātrinātājs ar savu privāto atmiņu un vairāk pakļaujas pie regulāras darba slodzes.

Keš bāzētie procesori parasti rada lielu skaitu kešatmiņas netrāpījumus neregulārajās programmās neprognozējamās atmiņas piekļūšanas dēļ. Pārejošā daudz plūsmu arhitektūra kopumā ir labāk piemērota šīm programmām, jo tā var uzturēt ilgākas pārklājumu atmiņas piekļūšanu, pārejot uz citiem gataviem diegiem, kamēr atmiņas apakšsistēma lāde vai raksta datus, tādējādi nav vajadzības pieprasīt kešatmiņu lai samazināt piekļuves pārklājumu.

4. CRAY XMT ARHITEKTŪRA

Cray XMT sastāv no divējādas ligzdas Opteron AMD apkalpošanas mezgla un no uz pasūtījuma izstrādāta daudz plūsmu izskaitļotu mezglu ar vienu ThreadStorm procesoru. Sistēma var izmērīt līdz 8192 aprēķinātu mezglu ar 128 terabaitiem kopīgu atmiņu. Tomēr lielākā sistēma veidota līdz šim ar 512 aprēķinātiem mezgliem.

Katrs ThreadStorm ir 64 bitu VLIW (ļoti garš instrukcijas vārds) procesors, kas satur atmiņas vienību, aritmētisko vienību, un kontroles vienību. Tas pārslēdzas ciklu pa ciklu starp 128 smalkgraudaina aparatūras plūsmām, kas rada atmiņas pieejas.

Izpildīšanas laikā, sistēma programmatūras kartes pavedienu aparatūras plūsma, kas ietver programmas skaitītāju, statusa vārdu, mērķa un izņēmuma reģistru kopumu, un 32 universālus reģistrus. Cauruļvads ir 21 posmu garš visam komandām. Pēc projekta, procesors neizdot jaunu komandu no pašas plūsmas, līdz iepriekšējais komanda ir, izieta cauri cauruļvadam.

Tā ka atmiņas operācijai ir nepieciešams vairāk, ne kā 21 cikls, tad procesors atbalsta īpašu piekļuvi neatkarīgam komandām, lai izdot ik pēc 21 cikliem. Kompilators identificē skatu uz priekšu katrai komandai, kura var sastāvēt līdz astoņām komandām. Katra aparatūras plūsma var būt līdz astoņām izskatītam operācijām, tajā pašā laikā, kā rezultātā maksimāli 1024 atmiņas darbības izskata pilnais procesors. Atmiņas operācijas var tikt pabeigtas bez rindas.

ThreadStorm ir 64 KB, četru veidu asociatīvā komandas kešatmiņa, kas izmanto kodu apvidū un strādā pie 500 MHz frekvences.

Procesors ir savienots ar starp savienojumu tīklu caur point-to-point HyperTransport kanālu. Tīkla interfeisa kontrolleris (NIC) neveic visas apkopošanas un vienota tīkla pakete apkopo katru atmiņas darbību. Tīkla apakšsistēma ir bāzēta uz Cray SeaStar2 starp savienojumu, 5 un uz topoloģijas 3D torus.

Katrs ThreadStorm's atmiņas kontrolieris vada līdz 8Gb 128 bitu platu DDR (double data rate) RAM un tam ir 128-Kbyte četru veidu asociatīvā kešatmiņa, kas palīdz mazināt piekļuves pārklājumu. Programma piekļūst sistēmas atmiņai, izmantojot koplietoto atmiņas abstrakciju: sistēma var tieši ielikt un glabāt operāciju uz jebkuras fiziskās atmiņas vietas no jebkura ThreadStorm procesora, kas pievienots tīklam.

Sistēmas atmiņa ar 64 baitu detalizāciju, lai piešķirtu loģiski secīgus datus fiziskai atmiņai, piesaistīta dažādiem procesoriem. Dati ir sadalīti, gandrīz vienādi par atmiņas pilniem sistēmas mezgliem, neatkarīgi no tā, cik lietojumprogramma izmanto procesorus.

Pilnība iztukšots bits, kurš strādā, ka slēdzene;

Rādītāju-nosūtīšanas bits, kurš dot signālus par atmiņas atrašanas vietām, kuri satur vairāk rādītājus, nevis datus, un kuri atļauj automātisko ģenerēšanu jaunas atmiņas atsaucei; un

Divu slazdu biti, kuri var radīt signālu, kad atrašanas vieta ir saglabāta vai ielādēta.

ThreadStorm ģenerē līdz 500 miljoniem sekundē atmiņas atsauces (Mref s), t.i., viena reference vienā pulksteņa ciklā. Tomēr MC var uzturēt līdz 100 Mref/s DDR RAM, kamēr SeaStar2 Nika sasniedz 140 Mref/sec. Atmiņas operācijām pārklājums svārstās no ~ 68 cikliem. Operācijām, kuras atmiņas pārklājums svārstās no ~ 68 cikli (vietējā atmiņas kontrolieris) līdz~ 1200 cikliem (attālās atmiņas kontrolieris) 128 mezglu sistēmām.

Procesora arhitektūra nav mainījusies ar nesen ieviesto Cray XMT 2. Vienīga nozīmīga modifikācija ir augstākais vietējās atmiņas joslas platums, iegūts, izmantojot DDR2 atmiņu un papildus atmiņas kanālu, tas ir līdzeklis, kas uzliek lielāku pārklājumu (aptuveni divreiz) par vietējas atmiņas piekļuvi.

Salīdzinot ar citām daudz plūsmu arhitektūrā, XMT ir elastīgāka priekš neregulāro programmu izstrādāšanas un izpildīšanas .

4.1. SALĪDZINĀJUMS AR CITĀM ARHITEKTŪRAM

Attiecībā ar citām daudz plūsmu arhitektūrā, piemēram, kā GPUs un UltraSPARC procesori, XMT ir elastīgāka, izstrādājot un izpildot neregulāras programmas ar lielu atmiņas pēdu, un ir tipiskā datu analīzē.

5. GPUS AND ULTRASPARC PROCESORI

Lai papildus darbotos desmitiem tūkstošu pavedienu, GPUs ir liels skaits atmiņas joslu platumu. Tomēr, lai palielinātu atmiņas izmantošanu, programmām jāpiekļūst atmiņas regulārajiem modeļiem. Lai atļautu MC saplūdināt daudzas atmiņas darbības vienā lielā atmiņas darbībā, pavedieni, darbojas tās pašas SIMD vienībās, faktiski, piekļūt secīgas atmiņas vietām vai vismaz uzturēties vietā tā paša atmiņas segmentā. Ja atmiņas operācijas piekļūst dažādiem atmiņas segmentiem, viņi pieprasa vairākus operācijas (sliktākajā gadījumā, viens par katru operāciju), novājinot atmiņas joslas platumu. Citiem vārdiem sakot, adreses joslas ir ievērojami lēnākas, nekā datu joslas.

Dažos gadījumos ir iespējams izmantot GPUs' faktūras vienības — slodzes laikā grafikas renderēšanas pikseļu krāsu datus atmiņas vienības — lai aprēķinātu vispārīgu datu slodzi. Šī pieeja var samazināt kļūdas, kas saistītas ar nesakārtotām atmiņas operācijām. Tomēr, sakarā ar tās grafisko sākumpunktu, tekstūras atmiņa ir, galvenokārt, efektīva tad, kad datiem ir laba atrāšanas vieta un programma labāk izmantojama, kad pievienota tekstūru kešatmiņai.

GPUs arī sastāv no ātras piezīmju bloka mikroshēmas atmiņās, kura var samazināt nelielu daļu ar augsto apvidus datu piekļuves pārklājumu. Neregulāro programmu izstrādātāji var dažreiz izmantot piezīmju bloku, lai īstenotu plānošanas metodes, uzkrājot datus un veicot atmiņas mikroshēmu piekļūšanu vairāk regulāru. Tomēr, vajadzības gadījumā šī pieeja prasa lielas pūles.

Jaunākās paaudzes GPUs arī iekļauj sevī datu kešatmiņas, bet, ja programmas izmanto ļoti neregulāras piekļuves veidus , tie var nebūt efektīvi sakarā ar lielo skaitu notikušam kļūdām. Kad programmu neregularitāte atrodas plūsma kontrolē, un pavedieni darbojas uz vienas SIMD vienības GPU novirzes, tad notiek kļūdas veikšana.

XMT dalas ar atmiņas abstrakciju, plakanas atmiņas hierarhiju un smalkgraudaina sinhronizācija nodrošina visu mašīnas vienkāršu programmēšanas modeli.

Neregulāram programmām ar lielu datu kopu, mērogojamību uz vairākām GPUs joprojām ir izaicinājums. Pašlaik lielākais GPU plates atmiņas lielums ir 6 Gigabaiti. Kopš tā laika, kad nav koplietojamo atmiņu starp dažādām GPUs, lietojumprogrammu izstrādātājiem manuāli jāsadala datus, un saziņa starp PCI Express vadu varētu kļūt par lieku algoritmiem, kas uztur kopīgu statu. Pat ar peer-to-peer un tiešas piekļuves pieejamajiem jaunākiem GPUs joprojām ir nepieciešams dalīt datus pāri GPUs. Klastera līmenī, kad ir nepieciešams šķērsot tīklu, izvirzītajiem mērķiem, dalīšana kļūst vēl vairāk sarežģīta un komunikācijas pieskaitāmās izmaksas palielinās.

UltraSPARC procesori ietver sevī vienkāršas sacensības problēmu mērogojamībā pa punktiem. Mezglu iekšpusē, viņi pakļauj koplietojamas atmiņas ieguvi. Pāri mezgliem, tomēr, tie joprojām izstāda parastus sadalītus atmiņas iestatījumus un prasa izplatīt programmēšanas pieejas atmiņas. UltraSPARC procesori ir paredzēti, lai

paslēptu pārklājumu tikai mezglu līmenī, un neveic nekādas slodzes, lai tiktu gala ar clusterwide tīkla pārklājumu.

5.1. XMT PRIEKŠROCĪBAS

XMT ir speciāli izstrādāts, lai mērītu attālumu starp mezgliem. Koplietojamo atmiņas ieguvei, līdzenums atmiņas hierarhija un detalizēta sinhronizācija nodrošina visu mašīnas vienkāršu programmēšanas modeli. Programma ar lielu atmiņu var izmantot visu XMT pieejamo atmiņu, neprasot programmētājām, lai pārstrukturētu kodu.

ThreadStorm procesoru lokālie rokturi un attālie (ar citiem mezgliem) atmiņas operācijas viena vārdā detalizācijā. Atmiņas kodēšanas mērķis ir izplatīt atmiņu un tīkla piekļuves veidus, samazinot karstos punktus. Saistībā ar lielo skaitu pavedienu ar tās ierobežotajiem resursiem, ThreadStorm var pieļaut starp savienojumu tīkla pārklājumu. Šie līdzekļi izmantojami datu joslas samazināšanai, bet arī noņem sankcijas attiecībā uz izpildi un plānošanu, piekļūstot atmiņai ar ļoti neregulāro veidu.

Kopumā, papildus vienkāršākas mašīnas visā programmēšanas modelī, XMT ir efektīvāka nekā citas arhitektūras, kad dati nesatilpst vienā mezgla atmiņā — vai vienā platē GPUs gadījumā — un, kad ļoti neregulāras piekļuves modeļi veido kešatmiņu un koordinē atmiņas neefektīvu piekļuvi. Piemēram, programmētājs var ieviest vienkāršu meklēšanas platumu pirmā (BFS) algoritmā bez bažām par datu pārvietošanu un atmiņas modeļa optimizāciju, un iegūt labu veiktspēju un mērogojamību.

Vēl viens piemērs ir Aho Corasick virkne saskaņošana ar ļoti lielam vārdnīcām. Procesoros, kas balstīti uz kešatmiņas, ja modeļi ir kešatmiņa, tad saskaņošana procedūra ir ātra. Ja viņi tur neatrodas, tad procedūra kļūst lēnāka, jo procesoram ir nepieciešams ielādēt modeli no atmiņas. Piekļuve modeļa struktūrai ir neprognozējama, tādējādi viņiem sapludināšana ir grūta.

Ja algoritms salīdzina vārdnīcu ar sevi (sliktākais gadījums), tas rūpīgi pēta datu modeli, un arhitektūras optimizējas regulārajai piekļuvei veiktspējas samazināšanai. No otras puses, kad algoritms salīdzina tikai dažus modeļus un var izmantot kešatmiņu (labākais gadījums), XMT nebūt nesasniedz vienu un to pašu veiktspējas virsotni. Tomēr, sistēmas plates atmiņas hierarhija nodrošina veiktspējas stabilitāti un sanāk, ka labāka gadījuma veiktspēja ne tik daudz atšķiras no sliktā gadījuma veiktspējas.

5.2. XMT SIMULĒŠANA

Lai atbalstītu mūsu pētniecību par masveidā daudzkanālu sistēmām, mēs izstrādājām jaunu paralēlo pilnas sistēmas simulatoru, kas var atkārtot 128 procesorus uz XMT 48 kodolu (četru ligzdu Opteron 6176SE) ar ātrumu līdz 250 kilocycles sekundē un ar precizitātes kļūdas likmi zem 10 procentiem priekš neregulāras lietojumprogrammu kopas lielām tipiskiem iestāstījumiem. Simulators vada nemodificētas XMT programmas, atbalstot visas sistēmas arhitektūras iezīmes, un saskaņo ar parametru modeli, kas ātri, bet precīzi apraksta tīkla un atmiņas sakaru notikumus, ieskaitot

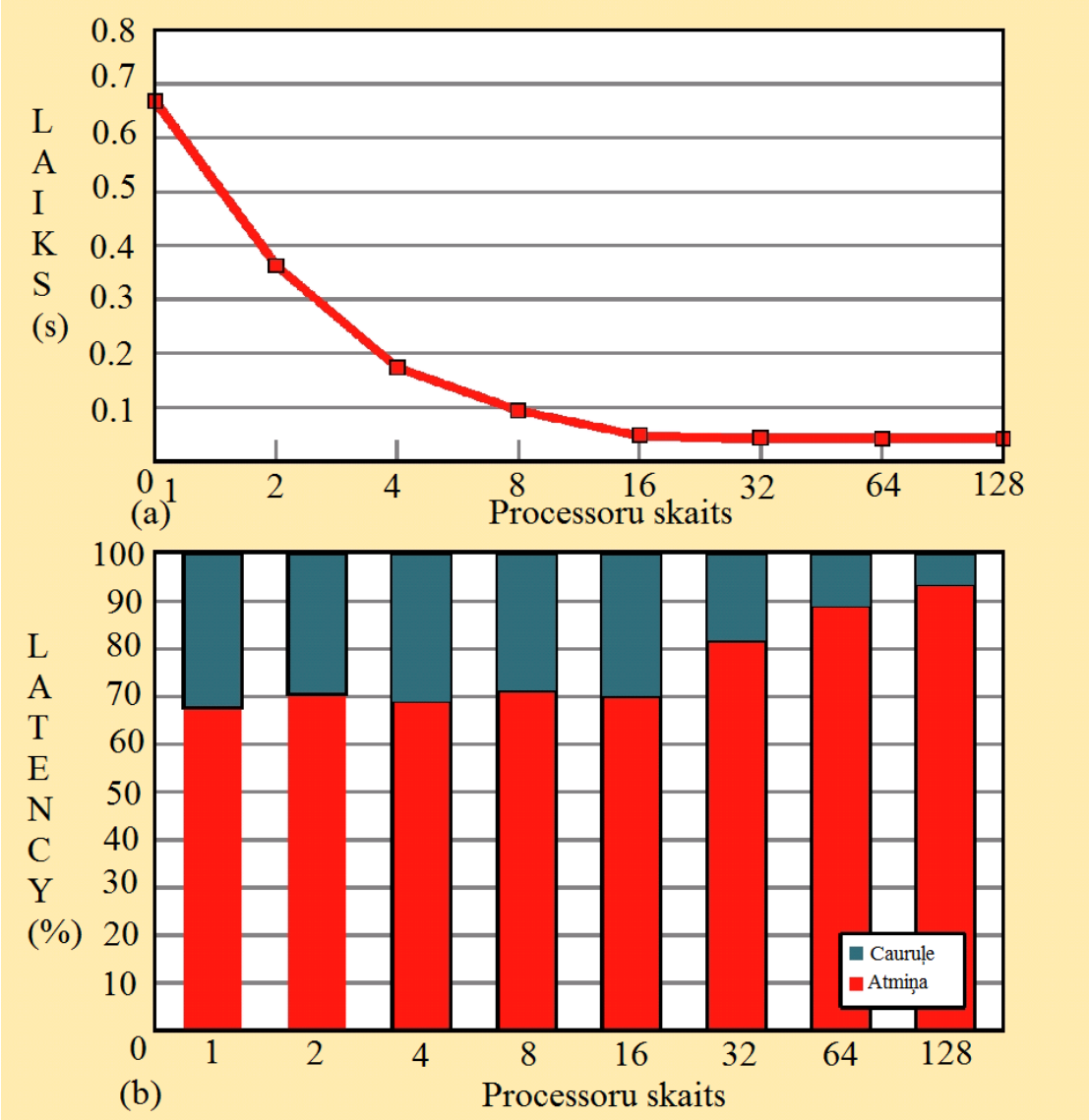
apgalvojuma efektu. Mēs izmantojām šo simulatoru, lai identificētu nepilnības pašreizējā XMT arhitektūrā un ierosināt iespējamos vairāk kodolu balstīto attīstību, lai pārvarētu šos ierobežojumus.

Mēs sākotnēji paredzam atmiņas ietekmi uz 128 procesora XMT darbības kopējo izpildes laiku ar aprīkotu stimulatoru, kas atjauninās ikreiz, kad vadu stendi gaida pieprasījumu no atmiņas skaitītāja. Attēlā 1a paradīts simulēts izpildes ilgums uz Aho Corasick rindu saskaņošanas algoritma un vārdnīcu veido visbiežāk 20.000 angļu vārdu un ievades komplekts no nejaušiem simboliem ar vienādu sadalījumu no ASCII alfabēta kopas un vidējais garums 8,5 biti.

Pat tad, kad sistēma neizmanto visus procesorus atmiņas operācijām, tas joprojām sajauc visu adrešu vietu uz zariem. Attēls 1b parāda laika procentus, kuros sistēma gaida atmiņas piekļuvi un saglabājas stabilā līmenī ap 70 procentiem līdz 16 procesori kļūst aktīvi. Kad vairāk, nekā 16 procesori ir aktīvi, tomēr, kopējais izpildes laiks apstājas, samazinās un laiku sistēma pavada, gaidot atmiņas piekļuves palielināšanos, pārsniedz 90 procentus 128 aktīvo procesoru.

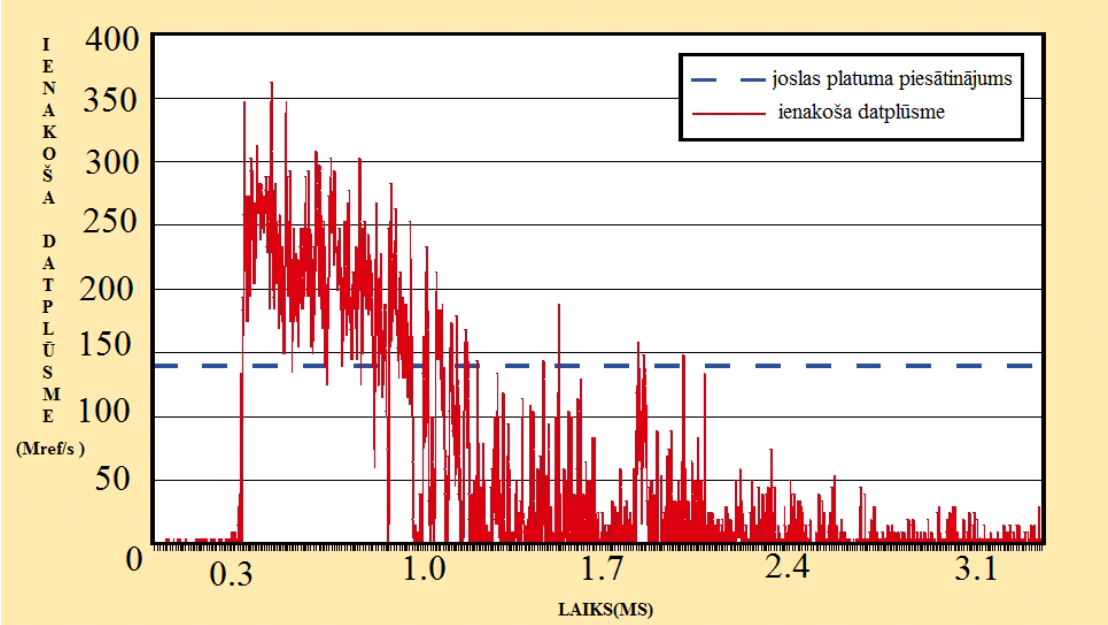
Lai noteiktu šīs problēmas cēloni, mēs izmantojām stimulatoru, kurš uzrauga atmiņas atsauces satiksmi uz NIC un MCs. 2. attēls parāda ienākošo trafiku uz vienu NIC tīklu, kad visi 128 procesori ir aktīvi. Nepārtraukta līnija apzīmē atsauces ienākošo skaitu, kas pieprasa piekļuvi NIC, kamēr raustītā līnija norāda atsauces maksimālo likmi vienā sekundē, kad NIC var kalpot (140 Mref/s divos virzienos). Attēlā redzams, ka daudzos programmu segmentos pieprasītas injekcijas ātrums pārsniedz NIC joslas pieejamo, strupceļa attālās atmiņas atsauci un aizkavētu vispārējo izpildi.

Runājot par MC interfeisiem, mēs atklājām, ka joslas piesātinājuma efekti ir mazāk stingri, nekā tiem NIC. Lai gan tas var šķist kontrastā ar ilgstošu atsauces likmi, tas diktē vienotu adrešu kodēšanu. Piemēram, 128 mezglu konfigurācija, sakarā ar kodēšanu, procesors rada vidēji vienu vietējo atsauci un 127 attālās atsauces no 128 iespējamo mērķa mezglu. Tikai vietēja atsauce izmantos MC kanālu, kamēr tālvadībā tie izmantos NIC kanālu. Tomēr, ienākoša atsauce no citiem mezgliem plūst gan caur NIC, gan arī caur MC kanāliem. Apvienojot ienākošas un izejošas atsauces, NIC kanāls ir pakļauts lielākām trafikam, nekā MC kanāls.



Attēls 1

Atmiņas darbības vispārējais izpildes laiks uz Aho Corasick rindu saskaņošanas algoritma pret aktīviem procesoriem, uz modelētiem 128 mezglu Cray XMT ietekmes: (a) kopējais programmas izpildes laiks un (b) procentos no laika pavadītas atmiņas operācijām. Globāli koplietojamā adrešu telpa vienmēr vienādi pakļaujas 128 fiziski dalītām atmiņām.

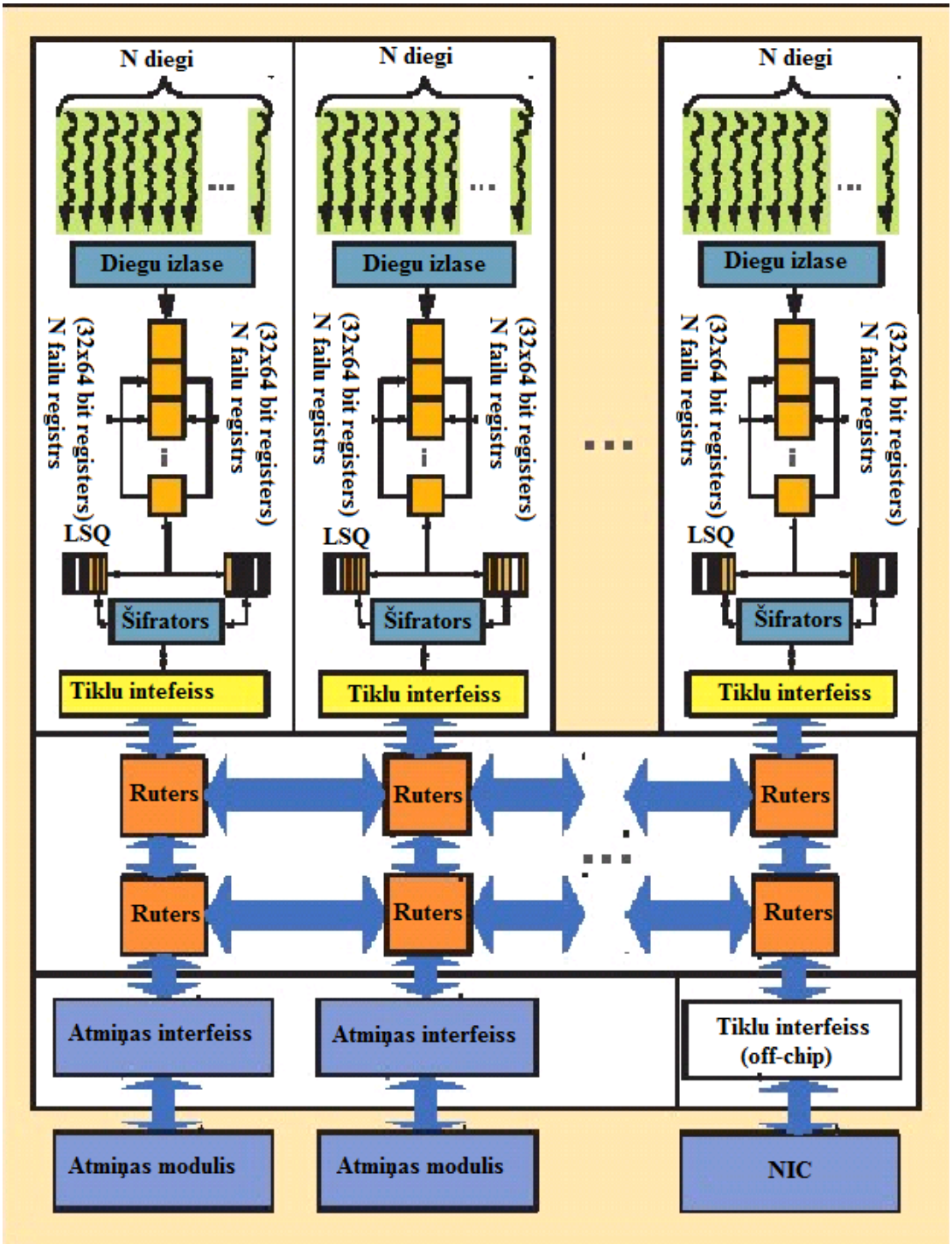


Attēls 1

Viena tīkla interfeisa kanāla (NIC) joslas piesātinājuma modeļi Aho Corasick rindu saskaņošanas algoritmam uz 128 mezglu XMT līdzīgas sistēmas, kad visi procesori ir aktīvi. Nepārtraukta līnija attēlo ienākošas atsauces pieprasītu piekļuvi NIC, bet pārtrauktas līnijas apzīmē maksimālo atsauces likmi vienā sekundē, ar ko var tik galā NIC (140 Mref/s divvirzienu). Daudzu programmu segmentos, pieprasītas injekcijas kurss pārsniedz pieejamo NIC joslu, attālās atmiņas atsauci un aizkavē vispārējo izpildi.

6. DAUDZKODOLU PROCESORU DIZAINS EVOLŪCIJA

Pamatojoties uz šiem rezultātiem, mēs izstrādājām iespējamo XMT balstītu daudz kodolu sistēmas arhitektūru. Mēs izmantojām stimulatoru, lai noteiktu labāko tirdzniecības kodolu skaitā un MCs, kas uzlabo sistēmas veiktspēju kopumā.



Attēls 2

6.1. SISTĒMAS DIZAINS

Kā redzams 3. attēlā, dizains ietver vairākus ThreadStorm līdzīgus kodolus, kurā katram aparatūras pavedienam ir īpašais faila reģistrs un slodzes rinda (LSQ). LSQs visu interfeisa kodēšanas modulis, kas vienmērīgi sadala atmiņas darbību visās sistēmas atmiņas moduļos. Katrs procesors ietver arī vairākas neatkarīgas MCs, katrs saskarnē ar dažādam mikroshēmām pie atmiņas moduļiem. Visas sastāvdaļas tiek savienotas, izmantojot pakešu komutācijas tīklu on-chip (NoC) konfigurēta, kā 2D acs, un katrai no tām ir savs īpašais maršrutētais. Iebūvētais NIC savienots ar citiem mezgliem procesoru sistēmā. NoC pārklājums veido mazāk, kā 2 procentus no pārklājuma atsauces vidējas sistēmas atmiņas.

6.2. VEIKTSPĒJAS NOVĒRTĒJUMS

Mēs izmantojām simulatora XMT līdzīgo sistēmu, lai novērtētu šo projektu, ko veido 32 mezgli, palielinot neatkarīgu MCs skaitu no 1 līdz 8 procesoriem un par vienu procesora kodolu no 1 līdz 16 . Mēs uzstādījām joslas platumu Nika 670 Mref/s (piecas reizes, ka pašreizējā XMT SeaStar2 tīkls 5 un līdzīgi kā Cray Gemini starp savienojums) un joslas platumu priekš MC 200 Mref/s (divreiz, ka pašreizējā XMT DDR atmiņa kanāls). Gan pārklājumu, gan joslas platuma parametrus, ko izmanto simulācijā, jāsaskaņo ar ceļu tīkla karti 1 un DDR3 atmiņas specifikāciju.

4. Attēla parāda vērtēšanas rezultātus, izmantojot trīs kritērijus: pats Aho Corasick rindu saskaņošanas programma, kuru izmantojam iepriekšējos izmēģinājumos; BFS kopējā kodola lietojumprogramma, kas izmanto grafa balstītu datu struktūru; un matricu reizināšana, tipiskais regulārais kodols, kuru mēs iekļāvām, lai novērtētu mūsu dizaina ietekmi par kopēju un mazāk ierobežoto programmu. Nepārtraukto līniju "ideāls" skaitļos norāda sasniegto veiktspēju, kad tīkla un atmiņas interfeisi ir brīvi no apgalvojuma un piesātinājums nekad nenotiek. Šīs rindas parāda, ka kritērijam ir pietiekami daudz analoģiju ja palielina kodolu un procesoru skaitu.

Kā parāda attēls 4a, rindu saskaņošanā MCs skaits viena procesorā būtiski neietekmē kopējos rādītājus. Tās ir sekas atmiņas organizācijai un programmu pilnveidei, lai samazināt karsto smērēšanu. Speedup ir gandrīz lineārs līdz četriem kodoliem vienā procesorā, un pēc tam sasniedz plato, sakarā ar NIC piesātinājumu.

Rezultāti ir diezgan atšķirīgi priekš BFS, kā to parāda attēls 4b. BFS ir sinhronizācijas intensīvais etalons (tas izmanto katru virsotni apmeklējamai slēdzenei) un NIC jau piesātināts ar atkārtotas darbības pamata konfigurāciju. Tādējādi programmai nav skalas.

Matricu reizināšana, kā rāda 4c. attēls, izmantojot vairāk MCs (ne vairāk kā divas) uzlabo veiktspēju, jo tā samazina karsto smērēšanu dažādos atmiņas interfeisos. Ar vairāk nekā diviem kodoliem un divām MCs uz vienu procesoru, par vājo vietu atkal

kļūst NIC un neatkarīgi no tā, cik kodolu vai MCs ir, tas stabilizē lietojumprogrammu veiktspēju.

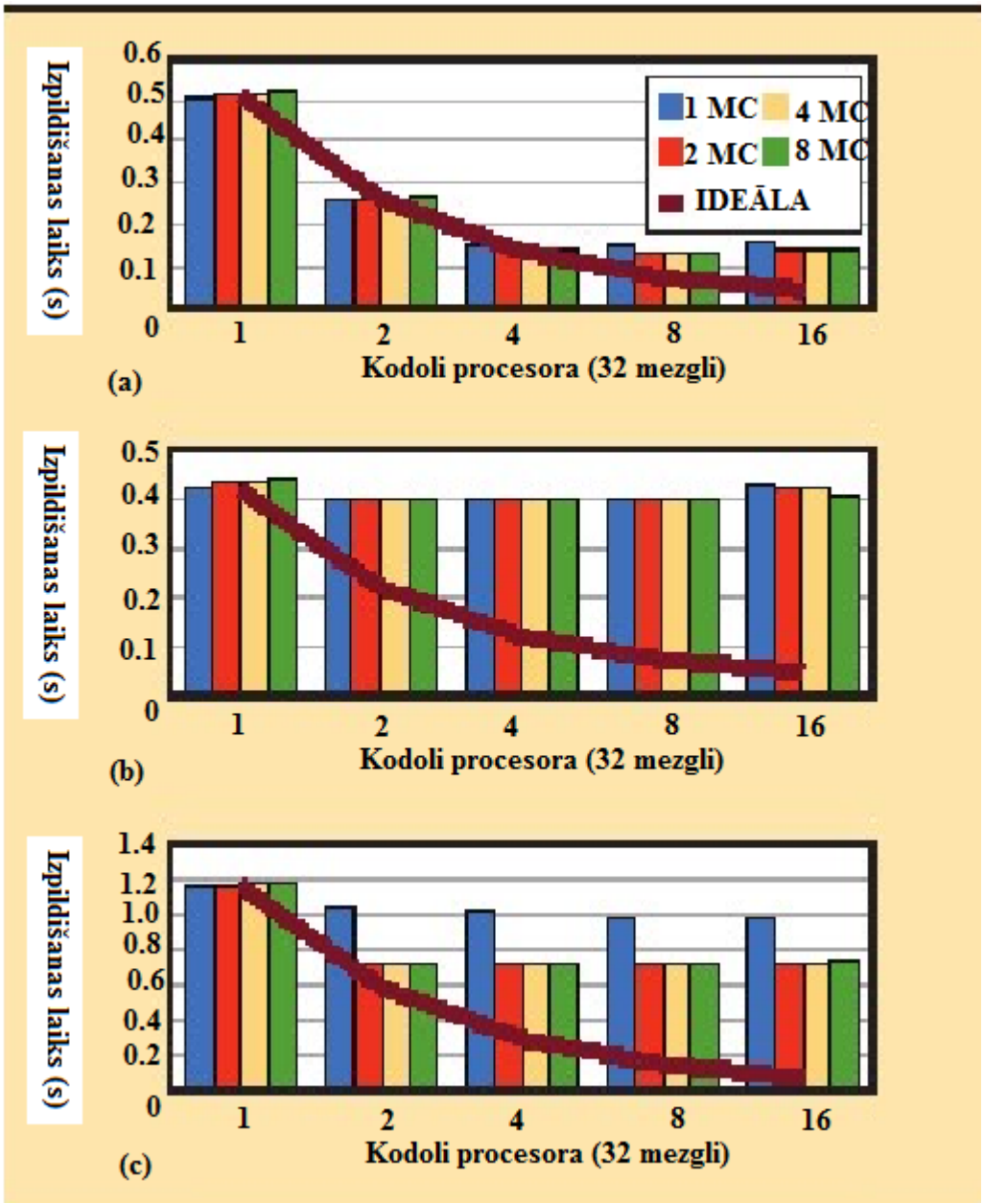
Šie eksperimenti liecina, kā vairākkodolu pieeja ļautu izmantot nākamās paaudzes savienojumus neregulārām programmām. Patiesībā, ar tīklu, kas var uzturēt joslu (ar nelielu ziņojumu) gandrīz piecas reizes augstāka par pašreizējo SeaStar2, šāda sistēma varētu palielināt veiktspēju līdz četriem kodoliem vienā zarā, augstākas injekcijas kursā. Regulāras programmas arī parādā gūt labumu no lielākas vietējas atmiņas joslas platuma, kas nodrošina vairākus MCs. uz vienu procesoru.

7. ATMIŅAS ATSAUCES APKOPOŠANA

XMT un mūsu vairāk kodolu procesoru dizainā, nelielie tīkla paketes apkopo katras attālās atmiņas darbību, radot smalkgraudainu satiksmi. Mazi ziņojumi parasti rodas augsto izmaksu dēļ.

Lai mazinātu šo ietekmi, dažas starp savienojumu sistēmas apkopo mazas paketes. Tomēr, XMT, kā mašīnas, kur katra slodzes – krātuves ir tieši novietota tīklā atsaucē bez jebkādas zem programmatūras slāņa, otro pieeju ir grūti saprast.

Lai samazinātu tīkla vadāmību nelieliem ziņojumiem bez dārga pielāgota tīkla projektēšanas, mēs integrējām izmaiņas mūsu vairāk kodolu procesora dizainā, kas izpilda tīkla atsauces apkopošanu procesoru līmenī.

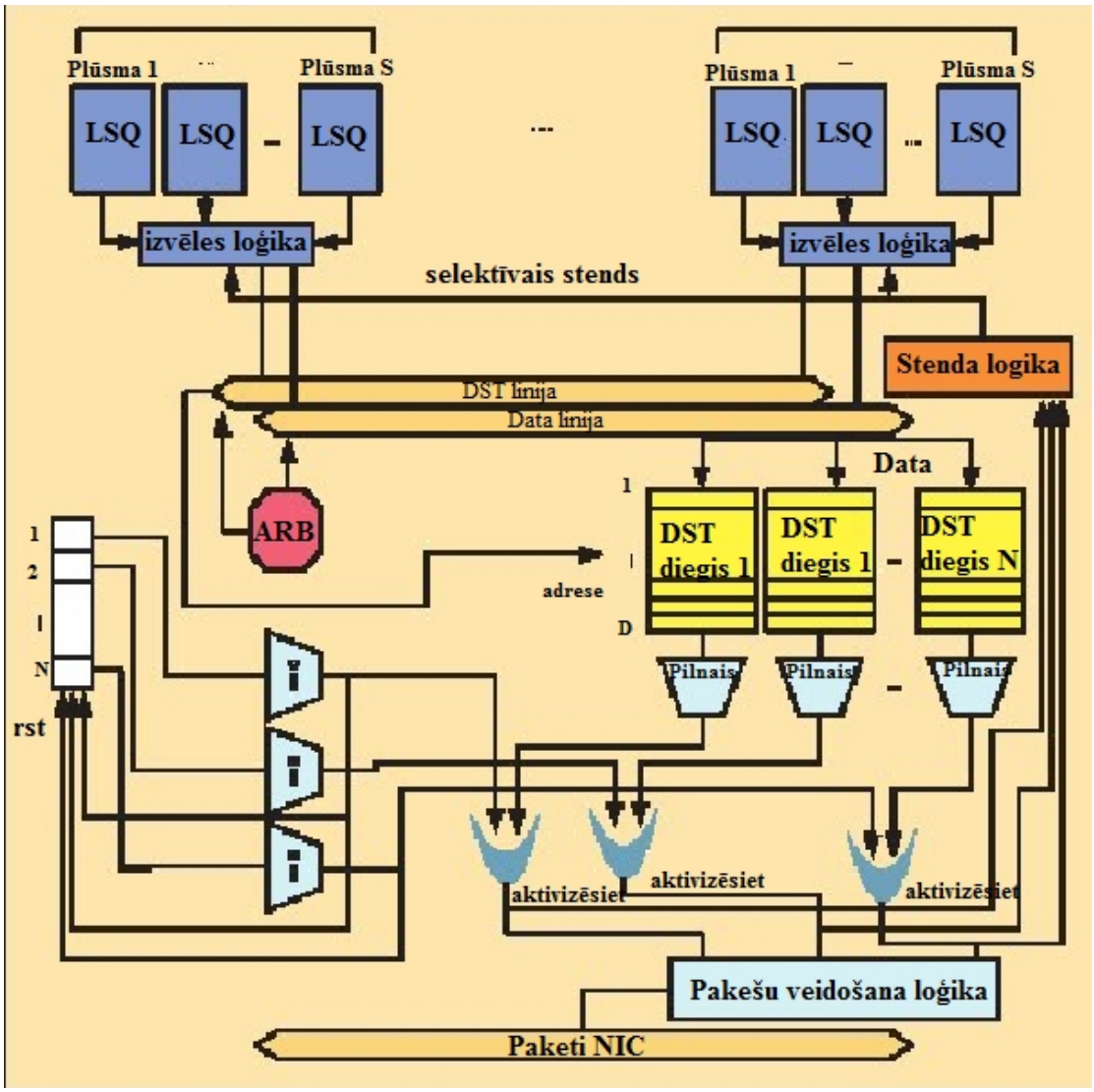


Attēls 3

Izpildes ilgums trīs kritēriju programmas pret kodolu procesoru 32 XMT līdzīgu sistēmu, izmantojot ierosināto vairāk kodolu procesoru dizainu: () Aho Corasick rindu saskaņošana, (b) pirmā platuma meklēšana (400,000 virsotnes, 400 kaimiņi uz vienu virsotni) un (c) matricu reizināšana (2000 x 2000 elementu matricas). Par katru programmu, rezultāti tiek parādīti dažādiem numuriem par vienu procesora atmiņas interfeisu.

8. APARATŪRAS IEVIEŠANA

Summēšanas mehānisms darbojas pēc buferizācijas līdz D atmiņas atsauksmes par vienu un to pašu mērķa mezglu (DST) W ciklu laika logā. Kad D atsauces atmiņa ir sasniegta vai logu laiks ir pagājis, mehānisms nosūta atmiņā atsauces buferi pret DST izmantojot vienotā tīkla pakešu, koplietošanas galveni.



Attēls 4

5. Attēla parāda aparatūras projektēšanu, īstenojot ierosināto mehānismu. LSQs dažādu kodolu interfeisu ar atlases loģiku, kas identificē katru atmiņas atsauci DST. Atlases loģika izmanto DST, lai risinātu vienu no FIFO (pirmais iekšā, otrais ārā) buferi, kas saglabā atmiņā atskaites datus — adresi, vērtību, atmiņas darbību kodus un kontroles biti — summēšanai.

Ir FIFO buferis katrai DST sistēmai un visiem buferiem ir D atsauces. Kad D atsauces ir buferī, paketes izveides loģika izveido tīkla pakešu visu saturu un bufera vienādu galveni. Katrs FIFO buferis ir saistīts arī ar atpakaļ skaitītāju, kas sākas no cikla W. Ja atpakaļ skaitītājs sasniedz 0, paketes izveides loģika tiek nosūtīta, pat ja buferis nav pilns. Ja buferis aizpildās pirms atpakaļ skaitītājs sasniedz 0, modulis nekavējoties

nosūta paketi un restartē atpakaļ skaitītāju uz W. Selektīvās kabīnes loģikas modulis novērš procesoru no papildu norāžu radīšanas ar vienu un to pašu DST, un ja buferis ir pilns, bet tīkla paketes nevar ģenerēt, piemēram, kad summēšanas loģika šobrīd rada vēl vienu paciņu.

Sistēmā, kas nodrošina lielu skaitu mezglu, īsteno FIFO bufera katrām DST, un var pieprasīt lielu daudzumu atmiņas mikroshēmu. Samazinot šo prasību, ir iespējams izmantot daudzpakāpju un hierarhisku apkopošanu shēmās, kur katrs buferis vāc atmiņas atsauces vērstas uz mezglu grupām. Mezglu grupas pēc tam izplata atsauces uz konkrētu galamērķi. Dažādi viedokļi par šādu pieeju par nedaudz augstākas agregācijas loģiku sarežģītību un kavēšanos. Tomēr, vēl vairāk palielinot par vienu, procesora pavedienu skaitu, kas var mazināt aizkavēšanās.

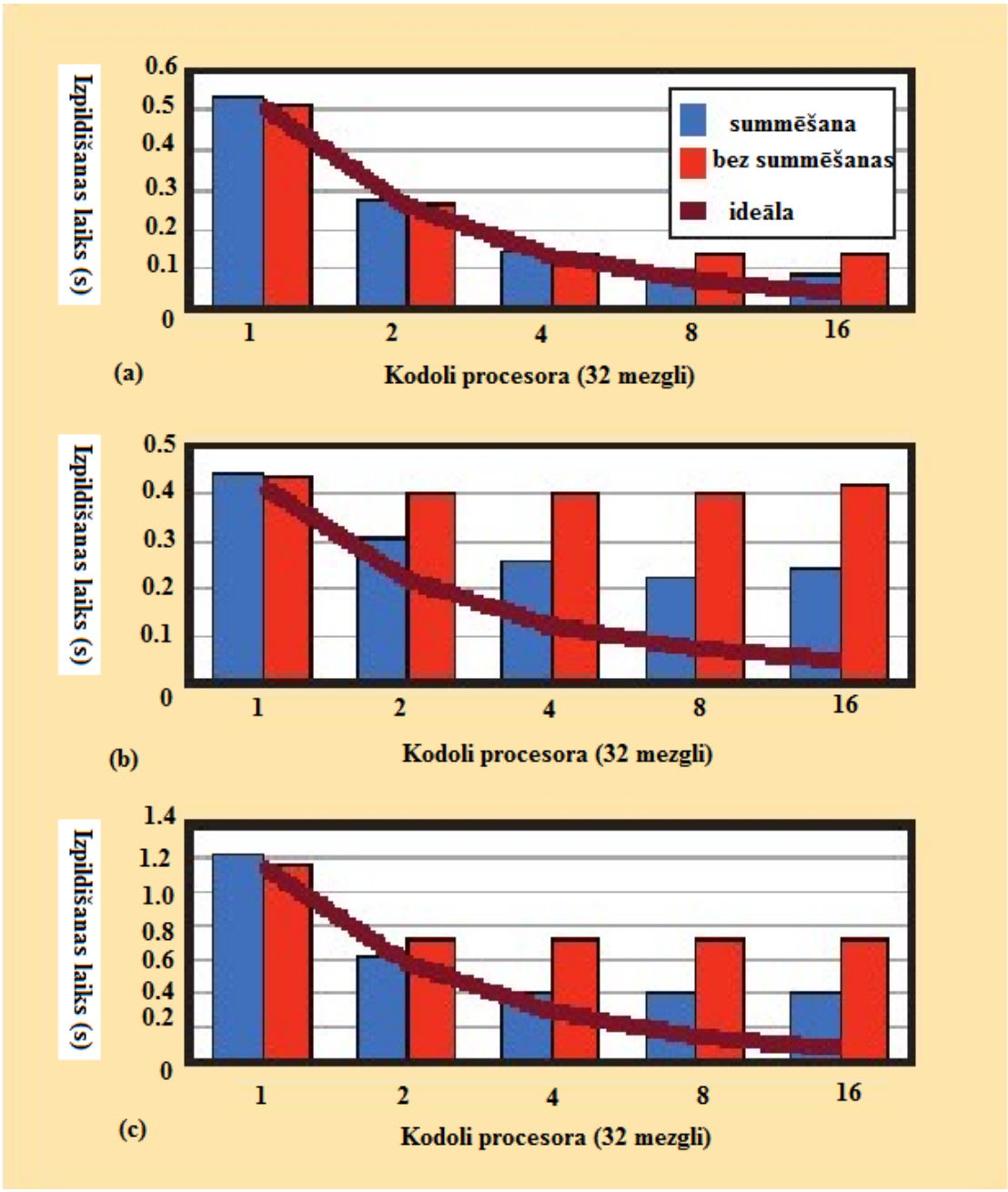
9. MĒROGOJAMĪBAS ANALĪZE

Mēs integrējam mūsu summēšanas mehānismu 32 mezgls XMT simulācijas modelī un sākotnēji novērtējam kompromisus starp bufera lielumu D un logu laiku W. Mēs atradām labākos parametrus D = 16 un W = 32. Šis rezultāts ir acīmredzami atkarīgs no mezglu mašīnu skaita. Laiks, kāds nepieciešams, lai aizpildītu buferi un vidējo tīklu pārklājumu palielinās līdz ar mezglu skaitu.

Mēs pārpildījām mērogojamības analīzi vairāk kodolu procesoru konfigurācijai, kas parādīta iepriekš 4. attēlā, izmantojot vienus un tos pašus trīs kritērijus. 6. Attēls salīdzina rezultātus ar un bez loģikas apkopošanas; šajā gadījumā katram procesoram bija divi MCs, kas mums līks labākais tirdzniecības iepriekš. Veiktspējas uzlabojumus, kas ir nozīmīgi visos kritērijos, tuvojas ideālām mērogošana. Pat, ja summēšanas mehānisms liek klāt vairāk pārklājuma attālās atmiņas operācijām (laika logs plus pieskaitāmās izmaksas pakešu ražošanā), kodolos ir pietiekami daudz pavedienu, lai paciest papildus pieskaitāmās izmaksas.

Augstas veiktspējas skaitļošanas pašreizējā tendence ir uz sistēmām, kuras izmanto procesorus ar papildus kešatmiņas arhitektūrā tēmēta uz intensīviem peldējušo punktu aprēķiniem. Šie risinājumi ir grozāmi lietojumprogrammām, kurās ir augsta blīvuma skaitļošana, augstu apvidū un regulāra datu struktūras. Tomēr, tie nespēj tikt galā ar jauno intensīvu datu apmaiņu neregulāro programmu prasībām, kas rada lielus nestrukturētos datu kopas ar sliktu rajonu, augstu sinhronizācijas intensitāti un atmiņas pēdas krietni pārsniedz lielumu, kas pieejams uz kopējā aprēķina līmeņiem. Šis programmas attaisno atjaunoto interesi par arhitektūru, kas var panest, nevis samazināt pārklājumu cauri masveida grafiku un būt vienkāršāka visā mašīnas plānošanas modelī.

Romāna pieejas, tādas, kā datu apkopošana un saspiešana, būs nepieciešama, lai nodrošinātu labāku starp savienojumu infrastruktūras izmantošanu nākotnē un ievērojamu veiktspējas pieaugumu. Šīs pieejas apstiprināšana arī prasīs jaunus rīkus, piemēram, optimizēt pielāgojamus simulatoriem un prototipu platformām.



Attēls 5

Izpildes laiks trīs kritēriju programmām pret kodolu skaitu uz procesora 32 mezgliem XMT sistēmas, izmantojot ierosināto vairāk kodolu procesoru dizainu ar integrētu summēšanas loģikas mehānismu: (a) Aho Corasick rindu saskaņošana, (b) platums-pirmā meklēšana un (c) matricu reizināšana. Katrai programmai, rezultāti tiek rādīti ar summēšanas loģiku aktivizēšanu un deaktivizēšanu. Procesoros ir divi MCs.

MANS VIEDOKLIS

Nemitīgā kodolu skaita dubultošana jau tuvākajā laikā pārsniegs līmeni, kurā programmatūra pilnībā spēj izmantot sniegto jaudu.

Ik jaunu procesoru paaudzi, kas iznāk apmēram reizi divos gados, serveru uzturētājiem un programmu veidotājiem jāsaskaras ar izaicinājumu kā pilnvērtīgi pielietot sniegto jaudu.

Šodienas serveru programmatūrai ir divu veidu limiti. Viens ir t.s. „hard limit", kas ir programmas izstrādātāja noteiktais teorētiskais kodolu daudzums, savukārt otrs ir „soft limit", kas lielākoties ir mazāks nekā pirmais, un to iespaido dažādi rādītāji programmatūras līmenī.

Procesoru kodolu skaita palielināšana ātri vien novedīs pie situācijas, kad būs grūtības ar pilnvērtīgas programmatūras izstrādi. Iespaids tiks atstāts uz visu – operētājsistēmām, programmām, vizualizācijas rīkiem u.tml. Jāpiezīmē, ka lielākā daļa vizualizācijas rīku šodien nespēj izmantot vairāk par 64 procesoriem.''

ATSAUCES

1. P. Kogge et al., “ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems,” DARPA Information Processing Techniques Office, 2008; www.cse.nd.edu/Reports/2008/TR-2008-13.pdf.

2. J. Feo et al., “ELDORADO,” Proc. 2nd ACM Int’l Conf. Computing Frontiers (CF 05), ACM, 2005, pp. 28-34.

3. J.L. Shin et al., “A 40 nm 16-core 128-Thread CMT SPARC SoC Processor,” Proc. 2010 IEEE Int’l Solid-State Circuits Conf. (ISSCC 10), IEEE, 2010, pp. 98-99.

4. “NVIDIA’s Next Generation CUDA Compute Architecture: Fermi,” white paper, Nvidia, 2009; www.nvidia.com/ content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_ Architecture_Whitepaper.pdf.

5. R. Brightwell et al., “SeaStar Interconnect: Balanced Bandwidth for Scalable Performance,” IEEE Micro, May 2006, pp. 41-57.

6. A. Tumeo, O. Villa, and D.G. Chavarrķa-Miranda, “Aho- Corasick String Matching on Shared and Distributed- Memory Parallel Architectures,” IEEE Trans. Parallel and Distributed Systems, Mar. 2012, pp. 436-443.

7. O. Villa et al., “Fast and Accurate Simulation of the Cray XMT Multithreaded Supercomputer,” preprint, IEEE Trans. Parallel and Distributed Systems, 2012; http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.70.

8. S. Secchi, A. Tumeo, and O. Villa, “Contention Modeling for Multithreaded Distributed Shared Memory Machines: The Cray XMT,” Proc. 11th IEEE/ACM Int’l Symp. Cluster, Cloud and Grid Computing (CCGRID 11), ACM, 2011, pp. 275-284.

9. B. Arimilli et al., “The PERCS High-Performance Interconnect,” Proc. 18th IEEE Symp. High-Performance Interconnects (HOTI 10), IEEE, 2010, pp. 75-82.