Vilniaus universitetas
Duomenų mokslo ir skaitmeninių technologijų institutas
L I E T U V A

INFORMATIKOS INŽINERIJA (07 T)

DIDELIO DAŽNIO KOMPIUTERIZUOTŲ PREKYBOS STRATEGIJŲ INŽINERIJA FINANSINĖSE RINKOSE

Mantas Vaitonis

2018 m. spalis

Mokslinė ataskaita DMSTI-DS-07T-18-3

VU Duomenų mokslo ir skaitmeninių technologijų institutas Akademijos g. 4,
LT-04812 Vilnius
www.mii.lt

Santrauka

Šiandien žmogus tampa praktiškai nereiklaingas atlikti prekybą elektroninėse biržose, už jį tai atlieka prekybos algoritmai. Kadangi prekyba vykdoma kompiuterių pagalba, tai užklausų pateikimo greitis išaugo nuo dienų iki milisekundinių ir net nanosekundinių sandorių. Dėl šios priežastie greitis ir būti pirmam prekybos algoitmams tampa vienu svarbiausių faktorių norint išlikti efektyviems. Naujausi technologiniai sprendimai, spartinantys skaičiavimus elektroninėse biržose ir augantis poreikis greitam užklausų vykdymui, stumia algoritminės prekybos rinkos kūrimo ir kainų paieškos strategijas į didelio dažnio prekybą ir šiuo metu didelio dažnio prekyba yra atsakinga už du trečdalius visų sandorių elektroninėse biržose. Tam, kad aiškiau išsiaiškinti, kaip veikia didelio dažnio prekyba elektroninėse biržose, darbe nagrinėta jų šaltiniai ir mokslo raida. Ištyrinėjus įvairius šaltinius paaiškėjo, jog norint dirbti su didelio dažnio duomenimis reikalingi pasitelkti paralelinius ir didelio našumo skaičiavimus. Toliau darbe aiškinamasi ar didesnio dažnio duomenys padeda prekybos sstrategijoms būti efektyvesnėms. Nustatčius, jog nanosekundiniai duomenys padeda algoritminėj prekyboj būti geresniems už kitus, prekybos strategijos perkeliamos į GPU, kuris pritaikytas vykdyti tūkstančius skaičiavimų vienu metu. Algoritmai turi būti adaptuojami kad dirbtų su GPU ir tam pasitelkiams linijinos metodas, kuris leidžia didelį daly skaičiavimų paralelizuoti perkialiant skaičiavimus į GPU. Taip pat ataskaitoje pateikiamos galimos didelio dažnio prekybos strategijų klasifikacijos, kurios dabar yra taikomos elektroninėse biržose, bei pateikiami galimi technologiniai sprendimai leidžiantys apdoroti didelio dažnio duomenis.

Reikšminiai žodžiai: didelio našumo skaičiavimai, didelio dažnio prekyba, algoritminė prekyba, prekybos strategijų inžinerija, GPU.

Turinys

1 Įvadas

Skaičiavimų pajėgumo reikalavimai nuolatos auga kompiuterių mokslo sirtyse kaip fizika, finansai ir kita. Viena iš sričių, kur reikalavimai atlikti daug skaičiavimų per kiek galima trumpesnį laiką yra didelio dažnio prekyba, kurios tikslas kompiuterių pagalba priimti rinkos sprendimus. Visi sprendimai susija su finansinių instrumentų pirkimu ar pardavimu yra atliekama kompiuterio algoritmų pagalba su minimaliu arba jokio žmogaus įsikišimo. Šie algoritmai analizuoja gaunamą informaciją iš elektroninių biržų ir ją apdoroja. Informacija iš šių biržų gali būti susijusi su naujais sandoriais, jų kiekiais, kainomis ir ar sandoriai buvo pateikti, pakeisti ar atšaukti. Kai algoritmas nusprendžia pateikti finansinio instrumento pirkimą ar pardavimą prekybos sistemai, tada per kelias milisekundes ši informacija nusiunčiama į elektroninę biržą, kurioje prekiaujama ir laukiama patvirtinimo signalo [6].

Prekybos sandorių atlikimo greitis augo nuo dieninių iki milisekundinių ir dabar iki nanosekundinių sandorių. Padidėjus greičiui padaugėjo pateikiamų ir atšaukiamų sandorių kiekis. Didelio dažnio prekybos algoritmai yra labai jautrūs laikui ir informacijos uždelisimui, todėl mažiausias vėlinimas informacijos ir sandorių pateikimo yra vienas svarbiausių dalykų. Dėl šios priežasties didelio dažnio prekybos organizacijos daug investuoja į techninę įrangą, didelio greičio prisijungimus ir stato savo prekybos platformas kiek galima arčiau elektroninių biržų. Viena iš techninkų įrangų į kurias investuoja didelio dažnio prekybos organizacijos yra GPU. GPU architektūra yra ekenomiškai efektyvesni sprendimai lyginant su lygegerčių skaičiavimų CPU. Paralelių skaičiavimų perkėlimas iš CPU į GPU atvėrė naujas duris dideliems skaičiavimams, leisdamos išnaudoti didelio našumo kompiuterių galimybes ir standartiniuose kompiuteriuose naudojančiuose GPU paralelizuotus skaičiavimus.[20]

Grafikos svarba įvariose aplikacijose lėmė sparčią GPU raidą ir visus skaičiavimus susijusius su grafika GPU galėjo atlikti atskirai ir leisdama CPU dirbti su kitais skaičiavimais. GPU architektūra tapo efektyvi paraleliems skaičiavimas ir perėmė dalį funkcijų iš CPU. Šiomis dienomis GPU gali atlikti nuo šimtų iki šimtū tūkstančių skaičiavimų vienu metu, nes jie turi šimtus ar tūkstančius skaičiavimų branduolių, kurie gali būti naudojami lygiagretiems skaičiavimams. Reikia pastebėti, kad GPU neturi tiesioginės prieigos prie pagrindinės CPU atminties, dėl šios priežasties, teisingas atminties naudojimas paraleliose sistemose tampa vienu iš

kritinių aspektų. [7] Vis didėjant lygiagrečių skaičiavimų reikalingumui ir besiplečiant lygiagrečios architektūros techniniai įrangai kaip daugiabranduoliniai CPU ir GPU, paralelus programavimas tampa nebe alternatyva, bet būtinybe, didinant šiuolaikinių programų našumą. [2]

Grafikos procesoriai (GPU) suteikia naujas galimybes padidinti didelio kiekio duomenų skaičiavimo greičiams naudojančioms pasikartojančius skaičiavimus neprarandat tikslumo. GPU yra galingas įrankis, kuris gali apdoroti tūkstančius procesų vienu metu ir naudojantis didelį atminties pralaidumą. Lyginant su CPU, GPU yra sukurtas su daugiau tranzistorių skirtų duomenų apdorojimui, o ne duomenų saugojimui ir kontrolei. [5]

Šio tyrimo tiklas yra parodyti kaip GPU naudojimas ir algoritmo perkėlimas naudojant linijinį metodą gali paspartinti finansinius skaičiavimus. Tiksliau kaip paspartinti didelio dažnio prekybą naudojant statistinio arbitražo prekybos strategijas. Skirtos kompanijos siųlo savo įrankius, skirtus darbui su GPU ir pritaikant jį sistemų kūrimui. Kiekvienas iš sprendimų naudoja naujus skaičiavimų modelius, todėl reikalinga perkurti visus algoritmus, kad šie galėtų pilnai išnaudoti GPU paralelizavimo galimybes. Šiame darbe visi skaičiavimai atliekami naudojant MATLAB programinę įrangą, kuri leidžia efektyviau ir paprasčiau išnaudoti GPU galimybes, nei lyginant su C ar Fortana. MATLAP programavimo kalba galima lengvai išnaudoti visus CUDA skaičiavimo technologijos privalumus labai nesigilinant į GPU architektūrą ir nenaudojant žemesnio lygio GPU skaičiavimo bibliotekų.

Darbo objektas - didelio dažnio duomenys elektroninėse finansų biržose.

Darbo tikslas - sukurti didelio dažnio likvidumą didinančių prekybos strategijų elektroninėse finansų biržose apdorojimo metodą, taikant didelės apimties skaičiavimus, leidžiančius apdoroti duomenis mikro/nanosekundžių tiklsumu.

Siekiant užsibrėžto tikslo iškelti tokie **darbo uždaviniai**:

1. Apžvelgti didelio dažnio duomenų algoritimes strategijas, didelio dažnio duomenų apdorojimo metodus, jų įrankius bei technologijas.
2. Sukurti ir ištirti didelio dažnio duomenimis paremtų algoritminių strategijų elektroninėse finansų biržose apdorojimo metodą, dirbantį su didelės apimties duomenimis.

3. Sukurti algoritminių strategijų apdorojimo prototipą, integruojant pasiūlytą metodą, skirtą didelio dažnio ir apimties duomenų elektroninėse finansų biržose apdorojimui.

4. Aprašyti gautus rezultatus ir pastebėjimus, įvertinti sukurtą algoritminių strategijų prototipą.

Rengiant darbą remtasi moksline literatūra, moksliniais straipsniais, interneto šaltiniais.

Darbe naudojami **metodai:** mokslinės literatūros analizė ir apibendrinimas, stebėjimas, skaitmeninis modeliavimas, atrankos metodas, sintezė, eksperimentai, koreliacinė analizė, statistinė analizė, kompiuterinis duomenų apdorojimas.

Darbo rezultatų teorinė ir praktinė reikšmė:

Teorinė reikšmė – statistinio arbitražo prekybos strategija gali būti naudojama ateities sandorių rinkos prekybai. Didelio dažnio duomenų panaudojimas padeda gauti didesnį pelningumą taikant porų prekybos strategijas.

Praktinė reikšmė – atliktų tyrimų pagrindu gali būti kuriama sistema, atliekanti didelio dažnio skaičiavimus linijiniu metodu GPU pagalba. Šie skaičiavimai gali būti pritaikomi didelio dažnio prekyboje siekti pelno arba didinti likvidumą elektroninėse biržose.

Darbo apribojimai ir sunkumai:

Didelio dažnio duomenys yra brangūs, todėl sudėtinga juos gauti. Šių duomenų normalizavimas užima daug laiko ir apsunkina strategijų testavimą. Pateikiamų strategijų matematinių modelių perteikimas į tiesinius algoritminius modelius, aprašant programiniu kodu yra sudėtingas ir ne visada pilnai įgyvendinamas. Didelio dažnio duomenys susideda iš milijonų eilučių, todėl strategijų testavimas ir naudojamų algoritmų efektyvumui parodyti reikalinga galinga techninė įranga, kuri leistų pilnai išnaudoti lygegrečių skaičiavimų galimybes.

1.1 Kas yra algoritminė prekybą?

Algoritminė prekyba (automatinė prekyba, juodosios dėžės prekyba) yra procesas, kurio metu pasitelkiami kompiuteriai su įdiegtom specifinėm programom, kurių tikslas prekiauti skirtingos rinkose tokiu greičiu, kuris žmogui yra neįmanomas. Sukurtas algoritmas remiasi laiku, finansinių instrumentų kaina, kiekiu arba

matematiniais modeliais. Tokio tipo sistemos yra nuoseklesnės ir išvengiama žmogiškojo faktoriaus, kai prekiaujat remiamasi emocijomis, o ne sudarytomis taisyklėmis.[12]

Pavyzdys:

Investuotojui reikia vadovautis tokiomis taisyklėmis:

• Nupirkti 50 akcijų, kai jų penkiasdešimties dienų kainų vidurkis viršija dviejų šimtų dienų kainų vidurkį;

• Parduoti 50 akcijų, kai jų penkiasdešimties dienų kainų vidurkis yra žemesnis už dviejų šimtų dienų kainų vidurkį.

Naudojantis šiomis paprastomis taisyklėmis yra lengva parašyti kompiuterinę programą, kuri automatiškai stebėtų akcijų kainas ir pirktų – parduotų, kai atsiranda signalas sukurtoms taisyklėms. Investuotojui nebereikia pačiam stebėti rinkos ir teikti užsakymus, tai už jį atlieka sukurtas algoritmas.

Algoritminės prekybos privalumai:

• Užsakymai pateikiami esant geriausioms kainoms;

• Greitas ir efektyvus užsakymo pateikimas;

• Sumažinamos išlaidos už komisinius mokesčius;

• Automatiškai tikrina rinka įvykus mažiausiems pasikeitimams;

• Galima ištestuoti algoritmą su turima istorine ir realia informacija;

• Sumažinama rizika klaidingo užsakymo pateikimo, kuris gali įvykti dėl žmogiškojo faktoriaus.[13][24]

Algoritminė prekyba yra naudojama daugybės prekybos ir investavimo organizacijų, kurios užsiima skirtingomis prekybomis:

• Vidutinio ir ilgo laiko investuotojai (pensijų, investiciniai fondai, draudimo kompanijos), kurios perka akcijas dideliais kiekiais ir ilgam laikotarpiui.

• Trumpo laikotarpio investuotojai (rikos kūrėjai, spekuliantai ir arbitražo naudotojai), kurie naudojas automatinės prekybos sistemomis.

• Sistematiniai investuotojai (rinkos sekėjai, porų prekybos naudotojai, rizikos fondai), kurie patys kuria prekybos sistemas ir jas pritaiko skirtingoms rinkoms.[26]

Visos strategijos, kurios vykdo algoritminę prekybą, ieško rinkoje situacijų, kada galima išlošti iš padidėjusios finansinio instrumento kainos ar sumažėjusių išlaidų.[24] Toliau pateikiamos kelios pagrindinės algoritminės prekybos strategijos:

- Rinką sekančios strategijos - dažniausiai naudojama strategija, kuri remiais vidurkine ar labai pakitusia informacija, kuri yra susijusi su finansinio instrumento kaina ar kitu techniniu indikatoriumi. Šio tipo strategijos nesistengia nuspėti ateities kainų judėjimo, bet remiais istorine vidutine informacija ir pagal tai priima sprendimus.[13]

- Arbitražinės strategijos - tokio tipo strategijos stebi tą patį finansinį instrumentą skirtingose rinkose. Atsiradus prekybos signalui perkama toje rinkoje, kur mažesnė kaina ir tuo pačiu metu parduodama kitoje rinkoje, kur yra didesnė kaina.[13]

- Matematiniais modeliais besiremiančios strategijos - daugybė matematinių modelių gali būti pritaikyta algoritminėje prekyboje. Pavyzdžiui yra delta – neutrali strategija, kuri leidžia investuotojui kompensuoti teigiamas ir neigiamas deltas taip, kad bendra portfelio delta būtų suamžinta iki nulio.[13]

- Prekybos intervalo strategijos - šio tipo strategijos remiasi idėja, jog aukščiausios ir žemiausios akcijos kainos yra laikinos, todėl jos periodiškai turi grįžti prie vidutinės kainos. Taikant šią strategija reikia nustatyti kainų intervalą, kurį reikia stebėti ir kai pasiekiamos šio intervalo ribos pradėti prekybą.[13]

- Akcijos kiekio vidutinės kainos strategijos - jos aplenkia sukurtą didelį užsakymą, tam kad pati sukurtų daug smulkių užsakymų, kurių kiekis neviršija nurodyto laikotarpio kiekio. Tikslas skurti užsakymus atsižvelgiant į kainas apskaičiuotas pagal parduotus jų kiekius ir nustatant vidutinę kainą. [13]

- Akcijos laiko vidutinės kainos strategija - ji aplenkia sukurta didelį užsakymą, tam kad pati sukurtų daug smulkių užsakymų, naudojant vienodai išdalintus laiko intervalus nusirodytam laikotarpiui. Tikslas sukurti, kiek galima daugiau užsakymų artimai vidutiniai kainai, per nurodytą laikotarpį.[13]

- Procentinio kiekio strategijos - iki tol, kol užsakymas yra pilnai įvykdytas, algoritmas siunčia dalinius užsakymus, nurodytu kiekiu ir dažniu, kuris buvo pateiktas investuotojo. Ši „žingsnių“ strategija siunčia vartotojo nurodytus užsakymus, kurie apskaičiuojami pagal procentinį kiekį nuo rinkos,

taip didinant arba mažinant pateikiamų užsakymų kiekį, kol pasiekiama norima kaina.[13]

1.2 Kas yra didelio dažnio prekyba?

Dabar finansų rinkos yra pilnai automatizuotos, susidedančios iš prekybos algortimų, todėl didžioji dalis rinkos yra užvaldyda didelio dažnio prekybos algoritmų (Fox et al., 2015). Didelio dažnio prekyba apima algoritminės prekybos programas, kurios išnaudoja sekundinius pasikeitimus rinkoje vykdyti prekybą. Šio tipo prekyba jau kuris laikas yra pakeitusi tradicinę prekyba, kai būdavo prekiaujama biržoje paties žmogaus (Fox 2015). Vienas iš pagrindinių tiklų didelio dažmio prekyboje yra būti pirmam aplenkent kitus, laiko atžvilgiu t.y. pateikti pirkimą ar pardavimą greičiau už kitus. Didžio dalis šio tipo strategijų sudaro taip vadinamos rinkos kūrimo strategijos, kurių pagalba yra didinamas likvidumas esačiose elektroninėse rinkose. Rinkos kūrimo strategis yra labiausiai reikalingos mažo likvidumo rinkose, kad jos taptų patrauklensės kitiems rinkos dalyviams. Būtent dėl šios priežasties rinkos kūrėjai moka mažus ir kartais jokių komisinių mokesčių. (Herlemont, 2013; Zubulake and Lee, 2011; Brogaard et al., 2013; Jaramillo, 2016). Didžioji dalis ekonomistų ir akademinės finansų bedruomenės didelio dažnio prekybą laiko naudingą rinkai, dėl likvidumo suteikimo ir taip pritraukia kapitalą ir naujos rinkos dalyvius (Jaramillo, 2016).

Reikia paminėti kad didelio dažnio prekyba yra terminas apimantis elektroninę prekybą, kurios metu rinkos dalyviai atidarytas pozicijas laiko labia trumpą laiką, sekunę ar net nanosekundę (Ahmed et al., 2009). Turint omenyje kadprekyba turibūti vykdoma sekindiniu, o kartais ir nanosekundiniu greičiu, visa prekyba gali būti atliekama tik naudojant didelio našumo kompiuterius, kurie leidžia vykdyti prekybą tokiu greičiu. Didelio dažnio prekyba yra automatinė algoritminė prekyba, kuri gali būti charakterizuota taip:

a) Prekyba vykdoma kompiuterių pagalba, su jokiu arba minimaliu žmogaus įsikišimu;

b) Naudojama mažė vėlinomo technologija, kurios pagalba pagretinamas prekybos sprendimų priėmimas ir sumažinamas laikas pateikti prekybos signal elektroninėje biržoje;

c) Naudojamas didelio našumo prisijungimas prie elektroninės biržos;

d) Atliekamas didelis užklausų kiekies(pirkimai, pardavimai, pozicijų atidarymas, uždarymas ir tašaukimas), dėl didelio greičio.

Didelio dažnioprekybos apimtys stipriai augo per pastaruosius dvidešimt metų. 2010 metais didelio dažnio prekybos apimtys nuo 0% 2005 metais pakilo iki 40% visoje Europoje. Pavyzdžiui 2005 metais didelio dažnio prekyba užėmė apie 20% Amerikos rinkos ir iki 2009 metų apėmė jau 60%. Tuo metu prasidėjo finansų krizė ir iki 2014 metų Amerikos rinkose jų dalis sumažėjo iki 35% (Kaya, 2016). Šiuo metu didelio dažnio prekyba užima apie 55% visos prekybos Amerikos žaliavų rinkose ir apie 40% prekybos Europs žaliavų rinkose (Krauss, 2015). Didelio dažnio prekyba taip pat yra labai aktyvi ir ateities sandorių biržose. CFTC nustatė kad laikotarpiu nuo 2012 metų iki 2014 metų algoritimnė prekyba buvo atsakinga už 80% visų ateities valiutų sandorių, 67% visų ateities indeksų sandorių, 62% visų atieties žaliavų sanorių, 47% visų ateities metalų ir energijos sandorių ir 38% visų ateities agrikultūrinių sanorių (Miller and Shorter, 2016).

Prekybos strategijos, naudojančios didelio dažnio prekyba ieško sunkiai pastebimų ir trumpų rinkos pasikeitimų, kuriuos galėtų išnaudoti savo naudai, pasitlkdamos didelio našumo kompiuterius ir atlikdamos greitus sprendimus dideliais kiekiais. Šios prekybos galimybės yra labia smuklūs rikose pasikeitimai, kurie įtakoja finansinių instrument kainas, kurių paglalba gaunamas labia mažas pelnas, tačiau dėl galimybės atlikti tūkstančius prekybos singalų per sekundę, pelnas gali išaukti.

1.3 Faktoriai lemiantys mažą uždelsimą elektroninėje prekyboje

Mažas uždelsimas yra reliatyvus terminas, nes technologinės pažangos šį kriterijų daro vis mažesniu ir sistemos dirba su mažiausiais uždelsimais. Kas šiuo metu laikoma kaip mažu vėlinimu, po kelių metų gali atrodyti kaip labai didelis. Mažas vėlinimas yra esminis dalykas didelio dažnio prekyboje, todėl yra daugybė technologijų leidžiančių jį pasiekti:

• Šviesolaidžiai: šiuo metu šviesolaidžiai daugumoje vietų jau pakeitė tradicinius varinius laidus, skirtus ilgiems tinklams, jų pagalba kompanijos gali sparčiau keistis informacija.

• Duomenų perdavimo srautas: didelio dažnio prekybos firmoms yra svarbu perduoti didelius kiekius duomenų, tuo pačiu išlaikant didelį perdavimo greitį. Technologinės pažangos pagalba duomenų perdavimas nuo 1 gigabito per sekundę išaugo iki 10 gigabitų ir toliau auga.

• GPU yra plačiai naudojamas vykdyti paralelias užduotis didelio našumo sistemose ir jų architektūra keitės taip kad galėtų vykdyti tūkstančius paprastų skaičiavimų vienu metu ir paspartintų sudėtingų sistemų veikimą. https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-kim.pdf [81]

• Programuojamos loginių elementų matricos (FPGA): FPGA yra integruotos grandinės, dažniausiai naudojamos didelio našumo skaičiavimuose ir tik neseniai pradėti naudoti finansinių institucijų. Įmonės naudoja FPGA vykdyti tam tikras algoritmo užduotis, kurios yra atsikartojančios, taip paspartinant algoritmo veikimą.,

• Daugiabranduolinių procesorių naudojimas: toks procesorius yra skaičiavimo komponentas, kurį sudaro keli branduoliai ar procesoriai. Didelio dažnio prekybos firmos naudoja tokius procesorius apdoroti dideliems duomenų kiekiams, jie leidžia atlikti kelis skaičiavimus vienu metu.

1.4 Kiti įrankiai skirti lėtinimo arbitražui

Lėtinimo arbitražas pasinaudoja mažais laiko skirtumais tarp perduodamų finansinio instrumento kainų duomenų. Prekyba vykdoma remiantis informacija gauta šiek tiek anksčiau už kitus rinkos dalyvius. Norint naudotis lėtinimo arbitražu, firma turi gauti informaciją apie finansinį instrumentą anksčiau už kitus. Pagrindiniai įrankiai naudojami šiam tikslui pasiekti yra (co-located) serveriai ir raw data feeds iš prekybos platformų.

• Co-Located servers: Norint sumažinti duomenų perdavimo greitį tarp prekybos platformos ir prekybos serverio, firmos mažina fizinį atstumą tarp jų. Norint ta pasiekti didelio dažnio prekybos firmos perka nekilnojamąjį turtą kiek galima arčiau prekybos platformos ir ten stato savo prekybos serverius.

• Raw data feeds: dalis prekybos platformų siūlo individualius duomenis, susidedančius iš jų pačių geriausiai įkainuotų instrumentų ir kita su prekyba susijusia informacija, kuri gali būti panaudota prekybos firmų. Tokiu būdu didelio dažnio

prekybos firmos išvengia papildomų skaičiavimų laiko kaštų, kai reikia patiems ieškoti geriausiai įkainotų instrumentų.

Paskutinis MIT tyrimas parodė, kad šviesos greitis tampa butelio kakleliu didelio dažnio prekyboje, norint prekiauti globaliu mastu. Prekybininkai norėtų kad informacija keliautų kiek galima greičiau, tačiau tam yra fizinės ribos, nes informacija negali keliauti greičiau už šviesą. Kaip dalį šio tyrimo, mokslininkai išskyrė kelis optimalius geografinius taškus tarp skirtingų prekybos platformų iš kurių užtruktų mažiausiai laiko apsikeisti informacija. Kaip pavyzdys, optimalus taškas apsikeisti informacija tarp Niujorko ir Londono akcijų biržos būtų viduryje Atlanto vandenyno.

2 GPU didelio dažnio prekybai

Lygiagretieji skaičiavimai yra skaičiavimų forma, kuomet daugybė skaičiavimų atliekama vienu metu, vadovaujantis principu, jog sudėtingiausios problemos gali būtų išskaidytos į daugybę mažesnių uždavinių, kurie gali būti apskaičiuojami nepriklausomai vieni nuo kitų. Šis skaičiavimų tipas yra naudojamas jau daug metų, pagrinde superkompiuterių panaudojime, tačiau pastaraisiais metais šis skaičiavimų tipas yra ir vis labiau naudojamas namų kompiuteriuose. Šis išpopuliarėjimas yra siejamas su fiziniais procesorių taktinio dažnio apribojimais dėl didelių energijos sąnaudų bei naujų kompiuterių procesorių, turinčių po keletą šerdžių (angl. core) atsiradimu maždaug 2004 metais. Yra keletas skirtingų paskirstytųjų skaičiavimų architektūros klasių, skirstomų pagal tai, kokiame technikos lygmenyje yra realizuotas skaičiavimų lygiagretumas. Galima išskirti tokias grupes kaip daugiaprocesorinės sistemos, daugiabranduolinės sistemos, kompiuterių klasteriai, keleto kompiuterių sistemos ir kitos [Wik12d], [Wik12e]. Šiame darbe nagrinėsime ganėtinai naują lygiagrečiųjų skaičiavimų tipą - skaičiavimus naudojantis grafinių vaizdo kortų procesoriais (GPU).

2.1 GPU pritaikymas įprastiniams lygiagretiesiems skaičiavimams

Pagrindinė GPU paskirtis - milijonus kartų atlikti tą patį algoritmą, naudojant skirtingus duomenis, puikiai dera su lygiagrečiųjų skaičiavimų uždaviniais, ypač naudojančiais genetiniais algoritmais paremtą programavimą. Genetinių algoritmų „brangiausia“ (daugiausiai skaičiavimo laiko užimanti) dalis, individų tinkamumo

įvertinimas, yra labai panaši į grafinio vaizdo apdorojimo uždavinį. Taip pat tai pastebima ir finansiniuose, prekybos sistemų generavimo, uždaviniuose, kurie taipogi galiausiai susiveda į daugybę kartų atliekamas tas pačias vertinimo operacijas, naudojant skirtingus pradinius duomenis.

Pagrindinis fizinis skirtumas tarp CPU bei GPU lustų architektūros yra tai, jog CPU luste didelė dalis tranzistorių yra skirta atminčiai, kai GPU beveik visa luste esanti erdvė yra užpildyta vadinamaisiais Aritmetiniais loginiais vienetais (angl. ALU – Arithmetic Logic unit), kurie yra skirti aritmetinėms bei loginėms operacijoms atlikti. Kadangi vidinė lusto erdvė yra ribota ir GPU lustuose erdvė nėra išnaudota dideliam lokalios atminties kiekiui, tai savaime suprantama, jog jo skaičiavimo galimybės yra žymiai didesnės nei analogiško CPU lusto [LKC+10]. Prie viso to dar prisideda tai, kad GPU lustai yra projektuoti taip, jog galėtų palaikyti didelį skaičių aktyvių skaičiavimo gijų vienu metu. Žinoma, GPU turi ir savo apribojimus. Vienas rimtesnių buvo tai, jog programuoti GPU procesoriams yra ganėtinai sudėtingą, dauguma programavimo kalbų skirtų programuoti GPU yra žemo abstrakcijos lygio bei yra būtina išmanyti bei naudoti atitinkamas grafines aplikacijų programavimo sąsajas (API). Visa tai iš esmės keičiasi NVIDIA pristačius CUDA architektūra bei programavimo priemonės. [Fer11].

Įprasti CPU turi atlikti daug skirtingų darbų, kaip rekursinių, adaptyvių ir tarpusavyje susijusių problemų sprendimas. Šio tipo užduotys reikalauja kad didelė skaičiavimo resursų dalis būtų naudojama komunikacijai tarp duomenų ir jų kontrolės. Atvirkščiai grafiniai skaičiavimai reikalauja mažai kontrolės ir nereikia tiek daug resursų komunikacijai, lyginant pagal skaičiavimų kiekį (Kirk ir Hwu, 2010). Šios sąlygos tapo priežastimi GPU naudoti duomenų paralelizvimui. GPU yra specialiai sukurtas spręsti problemas lygiagrečiuoju būdu su dideliu aritmeniniu intensyvumu (NVIDIA, 2014).

Tipinis GPU yra sudarytas iš tarpusavyje susietų procesorių masyvų, kurie yra išdėlioti tarpusavyje susietuose multiprocesoriuose. Tokio tipo architektūra, kuri sudaryta iš daugiabranduolinių procesorių yra panaši į naudojamą CPU klasteriuose, tačiau esantis viename techninės įrangos vienete. Tipinėje grafikos plokštėje. Didžioji jos dalis yra naudojama skaičiavimams, o mažoji dalis skirta kontrolei ir atminties užduotims.

Branuolio, branduolių blokų ir jų tinklo koncepcijos yra dažnai naudojamos apibūdinti kas yra CUDA. Branduolys yra vienas iš komponentų, skirtų vykdyti jam paskirtą skaičiavimo užduotį (kernelį) su vienu duomenų vienetu. Daugybiniai branduoliai dirba lygiagrečiai atlikdami tą pačia užduotį su duomenų masyvu. Branduoliai dar yra suskirstomi į branduolių masyvus, kurie yra naudojami GPU multiprocesorių. Branduoliai esantys bloke gali dalintis duomenimis esančiaias multipocesoriaus bendroje atmintyje ir gali būti sinchronizuojami. Branduolių blokai gali būti sujungti į tinklą, kuris šiuos blokus padalina tarp multiprocesorių esančių GPU.

Branduolių blokas gali būti sugrupuotas į vienadimensinius, dudiemnsinius ar tridimensinius branduolių masyvus ir CUDA turi kintamąjį, kuris naudodamas kiekvieno branduolio indeksą esančio bloke, gali juos panaudoti. Analogiškai ir branduolių blokų tinklai gali būti visų trijų tipų masyvais. GPU galima paralelizuoti dar vienu būdu. Kiekvienas branduolių blokas, vienu metu gali naudoti tik tam tikrą kiekį branduolių. Šis skaičius branduolių vadinamas lynu. Jei branduolių kiekis bloke yra didesnis nei jis gali naudoti, GPU lauks kols atsilaisvins kiti reikalingi branduoliai, kad galėtų kuo greičiau įvykdyti užduotį.

Naudojant CUDĄ pprogramavimą, paralelizmo efektyvumas priklauso kaip duomenys bus padalinti tarp branduolių blokų. Klöckner et al. (2009) pasiūlė metodą, kuris leidžia GPU programai automatiškai nustatyti šiuos metodus.

Atminties atchitektūra naudojama GPU yra gana sudėtinga, ji turi daug tipų ir lygių. GDDR DRAM, atmintis kuri gali pasiekti 8 GB dydį, kuri dažnai vadinama globalia atmintimi ir yra pati svarbiausia atmintis (Kirk and Hwu, 2010). Globali atmintis naudojama laikyti duomenis, kurie turi būti pasiekiami visų aktyvių branduolių. Branduoliai esantys skirtinguose blokuose negali tarpusavyje komunikuoti ar būti sinchronizuojami. Tačiau branduoliai esantys tame pačiame bloke, gali naudoti jų blokui skirta bendrąja atmintimi, kuri yra daug mažesnė, tačiau ji yra spartesnė už globalę atmintį, papildomai branduoliai turi dar savo privačią atmintį ir registrus.

Be viduje GPU esančių atminčių, CUDA taip pat turi dirbti ir su tipinia CPU RAM atmintimi, kaip ir kiekviena kita programa.

Norint naudoti GPU programas, reikia mokėti teisingai panaudoti visas šias atmintis. Pirmiausia visi duomenys yra saugomi CPU RAM atmintyje. Vėliau dalis

duomenų, kurie turi būti pasiekiami GPU branduolių blokų yra perkeliami į globalė GPU atmintį. Kernelis gali būti naudojamas tik su tais duomenimis, kurie yra kaikuriose GPU atmintyse. Atlikus skaičiavimus, jų rezultatai yra išsaugomi globalioje GPU atmintyje. Kiekvienas duomenų perkėlimas tarp CPU ir GPU atminties reikalauja procesoriaus laiko, reikia turėti tai omenyje, perkialiant duomenis tarp atminčių.

http://www.scielo.br/pdf/jbsmse/v33n4/a11v33n4.pdf [82]

2.2 CUDA naudojimas

CUDA (angl. - Compute Unified Device Architecture) yra techninės bei programinės įrangos architektūra, kuri įgalina NVIDIA sukurtuose GPU vykdyti įprastinėmis programavimo kalbomis, tokiomis kaip C/C++, Fortranas, OpenCL ir kitos, parašytas programas [Nvid09].

Loginė architektūra

CUDA programa skaičiavimus su GPU inicijuoja lygiagrečiai, kviesdama kernelius (procedūros ar funkcijos analogas GPU). Vienas kernelis, pasileisdamas pasiskirsto į rinkinį lygiagrečių gijų (threads), kurias programuotojas arba kompiliatorius surūšiuoja į gijų blokus (blocks) ir gijų blokų tinklelius (grids). CUDA gijų, blokų bei tinklelių hierarchija pavaizduota 2 paveiksle.

Kiekviena gija iš gijų bloko vykdo vieną kernelio skaičiavimą bei turi unikalų gijos numerį (ID) bloko viduje, vidinius registrus, lokalią atmintį bei gražina atitinkamą rezultatą.

Gijų blokas yra rinkinys tuo pačiu metu vykdomų gijų, kurios gali bendradarbiauti tarpusavyje naudodamosis bendra atmintimi bei sinchronizacijos funkcijomis. Gijų blokas turi savo unikalų numerį (ID) tinklelyje.

Tinklelis yra gijų blokų, kurie vienu metu vykdo to paties kernelio operacijas, rinkinys (masyvas). Tinklelis gauna duomenis iš bendros įrenginio atminties, įrašo skaičiavimų rezultatus į globaliąją atmintį bei atlieka sinchronizaciją tarp priklausomų kernelių iškvietimų.

Ši loginė CUDA gijų architektūra yra tiesiogiai atkartota fiziškai GPU procesorių architektūroje: visas GPU įrenginys vykdo vienu metu vieną ar keletą kernelio tinklelių, kiekvienas GPU daugia-branduolinis procesorius vykdo gijų blokų funkcijas, o kiekvienas to procesoriaus branduolys vykdo vienos gijos skaičiavimus.

Procesorius, gijas vykdo sujungdamas jas blokais po 32. Programuotojai, norėdami pasiekti ypač gerą rezultatą skaičiavimų pagreitėjime turėtų atkreipti dėmesį į šį grupavimą bei pasirūpinti, kad visos gijos iš šios grupės vykdytų skaičiavimus tokiu pat keliu bei duomenis reikalingus skaičiavimams gautų iš netolimų atminties vietų.

CUDA programinė įranga veikia ant daugelio šiuolaikinių operacinių sistemų, tokių kaip Windows XP/Vista/7, Linux ar Mac OS bei palaiko 32 bei 64 bitų architektūrą. Oficialiai CUDA palaiko C, C++ bei Fortran programavimo kalbas, tačiau yra galimybės integruoti CUDA ir su kitoms programavimo kalbomis (Java, Python ir kitomis). Pagrindiniai įrankiai, kurių reikia norint pradėti naudoti CUDA skaičiavimus yra speciali vaizdo kortos tvarkyklės versija bei CUDA įrankių rinkinys (angl. CUDA toolkit). Visus šiuos įrankius, bei išsamią informaciją apie jų naudojimą galima rasti oficialioje interneto svetainėje [Nvid12].

Tipinis apibendrintas CUDA C++ programos veikimas, pavaizduotas 5 paveiksle, yra toks:

- Pradiniai veiksmai kompiuteryje, tokie kaip pradinių reikšmių kintamiesiems suteikimas, parametrų, reikalingų skaičiavimams, priskyrimas;
- Atminties rezervavimas GPU bei kintamųjų nukopijavimas iš kompiuterio (host) į vaizdo plokštę (device);
- Skaičiavimų ant GPU inicijavimas, kernelio paleidimas;
- Rezultatų kopijavimas iš GPU į kompiuterį;
- Tolimesni skaičiavimai naudojant gautus rezultatus iš GPU.

2.3 Linijinis modelis

Linijinis optimizavimas yra sritis, kuri tyrinėjama įvairių mokslininkų daugiau kaip 70 metų. Šis metodas tampa vis svarbesnė dalis skirtingose srityse, kaip financai ir inžinerija. Linijinis programavimas optimizuoja linijinę funkciją, kuri skirta rasti ar apskaičiuoti riboto dydžio užduotį ar funkciją. http://www.idi.ntnu.no/~elster/pubs/ipdps09-lin-prog.pdf. [83] Klasikiniai būdai skirti spręsti linijines sistemas yra naudojant iteracinius methodus, kaip Jacobi, Gauss – Seidel ir SOR, kurie yra gerai žinomi ir detaliai aprašyti. Esant labai didelėm linijinėm sistemom geriausia naudoti Krylov iteracinį metodą. Kitaip nei stacionarūs iteraciniai metodai (Jacobi, Gauss - Seidel), Krylov metodas naudoja informaciją, kuri kinta kiekvienos iteracijos metu. https://arxiv.org/ftp/arxiv/papers/1511/1511.07207.pdf

[84] Tyrime esamas algoritmas perrašoms naudojant linijinį programavimą ir naudojamos duomenų struktūros sudarytos iš masyvų ir matricų. Duomenys priklausomai nuo sprendžiamos užduoties suformuoti kaip 2D, 3D ir 4D. Papildomos dimensijos duomenyse atsiranda norint išvengti ciklų algoritme, jie pakeičiami algebrinėmis operacijomis su matricom.

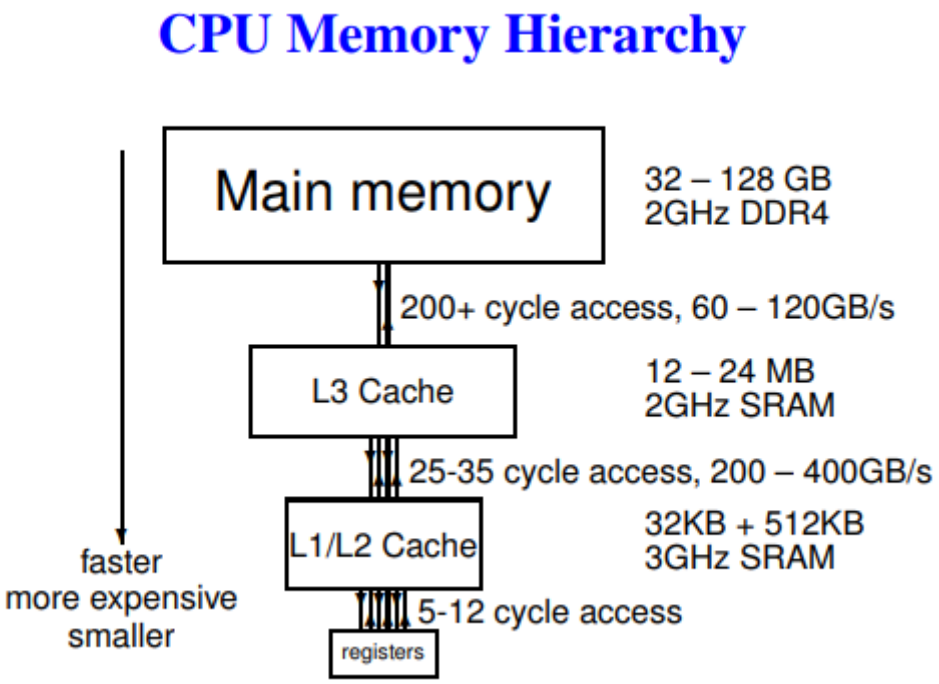
Pirmas supratęs, kad GPU galima naudoti ne tik grafikai, bet ir bendro naudojimo programoms buvo Markas Harris. Nuo tada GPU programavimo metodai evoliucionavo ir dabar yra keli būdai tam: CUDA (Compute Unified Device Architecture) pristatyta NVIDIA ir APP (Stream) pristatyta AMD. Naujas standartas OpenCL (Open Computing Language) skirtas apjungti skirtingus GPU pritaikymo metodus ir pateikti vieną bendrinę sistemą skirtą rašyti programoms heterogeninėse sistemomse naudojančiose GPU ir CPU. https://arxiv.org/ftp/arxiv/papers/1511/1511.07207.pdf [84]

Šiuo metu NVIDIA yra viena iš pirmaujančių GPU gamintojų. Jų skurta bendrinė skaičiavimo įrangos architektūra (CUDA) leidžia programuotojams išnaudoti visas jų GPU galimybes didelės apimites problemų sprendimui. CUDA sprendimai galimi su visomis NVIDIA vaizdo plokštėmės kurios yra sukurtos remiantis Tesla architektūra. NVIDIA Tesla architektūra yra sukurta naudojant keičiamo masyvus sudarytus iš keičiamo srauto multiprocesorių (SM). Vienas Tesla multiprocesorius sudarytas iš aštuonių procesoriaus branduolių (SP), dviejų specialiųjų funkcijų, procesorių instrukcijų rinkinio ir vidinės atminties. SM kuria, valdo ir vykdo funkcijas skirtinguose branduoliuose su nuliniu planavimo apkrovimu. Šiuo būdu kiekvienas branduolys dirba tik su jam skirtu duomenų paketu. GPU galima išskirti dvi skirtingas atminties rūšis: vidinė ir įreginio. Bendroji atmintis yra viena iš vidinių atminčių, kuri yra naudojama visų branduolių esančių multiprocesoriuje. Šios atminties sparta prilygsta L1 – spartinančiai atminčiaia esančiai CPU. Įrenginio atmintis yra didelio greičio DRAM atmintis su didesniu vėlinimu nei vidinė atmintis (dažniausiai iki šimot kartų lėtesnė). Įrenginio atmintis dar dalinama į rašymo – skaitymo, nano – spartinančia atmintį (globali ir lokali) ir tik skaitymui skirtą. CUDA pagalba sistemų kūrėjai gali papildyti programas su užklausomos į paralelius kernelius. Kernelio kodas tada yra vykdomas grupės branduolių susietų su GPU multiprocesorium. CUDA paralelizmas skaičiavimas

išgaunant išnaudojant branduolių hierarchiją. Banduoliai yra sugrupuoti į 1D, 2D ir 3D blokus, kurie dar yra apjungti į 1D arba 2D tinklą.

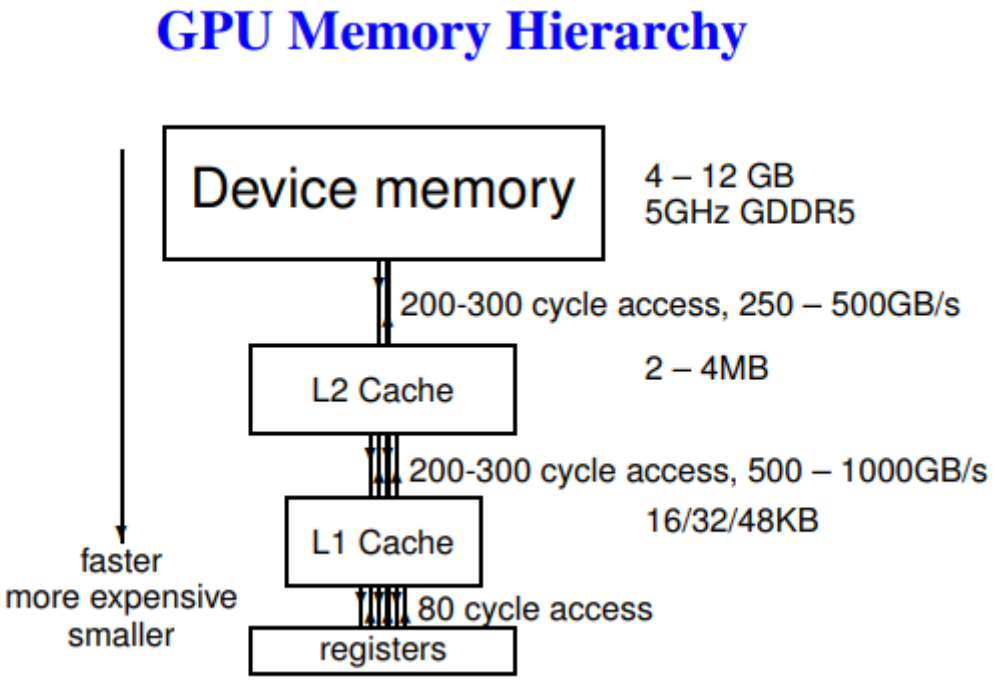
http://www.idi.ntnu.no/~elster/pubs/ipdps09-lin-prog.pdf [83]

Atminties greičių palyginimui pateikiamos CPU ir GPU atminčių hierachijos, kuriose atsipsindi atskirų atminčių greičiai, pralaidumas ir kiek užklausų per sekundę gali apdoroti.



1 pav. CPU atmintes hierarchija

Pati sparčiausia atmintis yra registruose, kur saugomos įvairios instrukcijos ir CPU registrai gali apdoroti 5 -12 užklausų vienu metu su L1/L2 spartinančiaja atmintimi. Pagal spartą šios spartinančios eina po registrų, jų talpa gali būti nuo 32 KB iki 512 KB, dirba 3GHz SRAM sparta ir vienu metu gali apdoroti 25-35 užklausas iš L3 spartinančios atminties. L3 yra žemiausio lygio, lėčiausia ir didžiausios talpos spartinančioje atmintis. Jos dydis svyruoja nuo 12 iki 24 MB, veikia 2GHz SRAM sparta, gali apdoroti iki 200 užklausų iš pagrindinės atminties ir duomenų pralaidumas tarp šių atminčių gali būti nuo 200 iki 400 GB/s. Ši atmintis yra pati talpiausia, tačiau ir pati lėčiausia. Jos dydis gali būti nuo 32 iki 128 GB, veikimo dažnis gali siekti 2GHz DDR4 ir duomenų pralaidumas 60 – 120 GB/s. Būtent atminčių pralaidumas ir atskleidžią didžiausią GPU privalumą.



2 pav. GPU atminties hierarchija

Lyginant su CPU pagrindine atminti, GPU pagrindinės atminties pralaidumas yra beveik 4 kartus didesnis ir sieka 250 – 500 GB/s, tačiau jos dydis mažesnis 4 -12 GB, 5GHz GDDR5. Pagrindinė atmintis kartu su L2 spartinančiają atminti gali atlikti 200 - 300 užklausų vienu metu. GPU L2 atminties dydis yra kelis kartus mažesnis lyginant su CPU L2 ir yra 2 – 4 MB, tačiau GPU gali turėti nuo kelių ir šimtų šių atminčių kiekvienam multiprocesoriui. Duomenų pralaidumas tarp L2 ir L1 atminčių taip pat yra daug didesnis, nei CPU ir galintis pasiekti 500 -1000 GB/s. Nors ši atmintis yra mažesnė 16/32/48 KB, nei CPU L2 tačiau kaip ir L1 jų gali būti šimtai skirtų kievienam multiprocesosiui. GPU registrų pranašumas, kad jie gali vykdyti iki 6 kartų daugiau užklausų vienu metu lyginant su CPU registrais.[83?]

2.4 Paralelūs skaičiavimai naudojant grafiniu įrenginius

Standartiniai CPU atlieka daug skirtingų užduočių, kaip rekursiniu, adaptyvių ir nepriklausomų problemų sprendimo. Šios užduotys reikalauja didelių skaičiavimo resursų, skirtų duomenų komunikacijai ir kontrolei. GPU, kitaip nei CPU reikalauja mažai resursų kontrolei ir komunikacijai, tai ir tapo motyvacija naudoti GPU paraleliems skaičiavimams. Naujausi GPU įrenginiai yra pertvarkyti spręsti negrafines užduotis. GPU jau kuris laikas naudojami spręsti įvairias didelės apmities skaičiavimo užduotis ir daugliu atveju GPU demonstravo didesnį efektyvumą vykdant šias

užduotis už CPU (Ryoo et al., 2008; Rasmusson et al., 2008; Stantchev et al., 2008; Stantchev et al., 2009; Mesquita et al., 2009).

Naudojant CUDA programavimo metodą, labai svarbu teisingai padalinti naudojamus duomenis tarp atskirų branduolių blokų ir tinklų. GPU tminties architektūra yra ganėtinai sudėtinga, nes yra daug atminties tipų ir lygių GPU kortoje. GPU esantis (GGDR) DRAM, dažniausiai įvardijama kaip globali ir pati svarbiausia atmintis. Į globalią atmintį talpinami duomenys, kurie turės būti pasiekiami visų aktyvių GPU procesorių. Skirtinų blokų procesoriai negali tarpusaviu komunikuoti ar būti sinchronizuoti, tačiau to paties bloko procesoriai gali naudotis jiems priklausančia bendrąją atmintim, kuri yra mažesnė už globalę, tačiau yra greičiau pasiekima už globalę atmintį. Papildomai dar yra konstantų ir tekstūrų tik skaitymui skirta spartinančioji atmintis skirta specifinėm GPU užduotims. Be šių viduje esančiu atminčių GPU taip pat turi komunikuoti su standartine CPU RAM atmintimi, kaip ir bet kuri kita sistema. Efektyvus GPU programų naudojimas reikalauja teisingo visų atminčių naudojimo. Pirmiausia visi duomenys yra talpinami RAM atmintyje, kuri pasiekima CPU ir GPU. Paskui dalis duomenų, kurie turi būti apdorojami GPU keliauja į GPU globaliąją atminį. Kerneliai gali būti vykdomi tik su duomenimis esančiais tam tikrose GPU atmintyse. Galutinis skaičiavimų rezultatas išsaugomas globalioje GPU atmintyje ir keliauja į CPU atmintį, kad būtų atlikti paskutinei funkcijos veiksmai. Šie duomenų perdavimai iš GPU atminties į CPU atmintį ir atvirkščiai užima dalį procesoriaus ciklų, todėl reikia atsiželgti į laiką, kurio papildomai reikia šiems veiksmams atlikti.

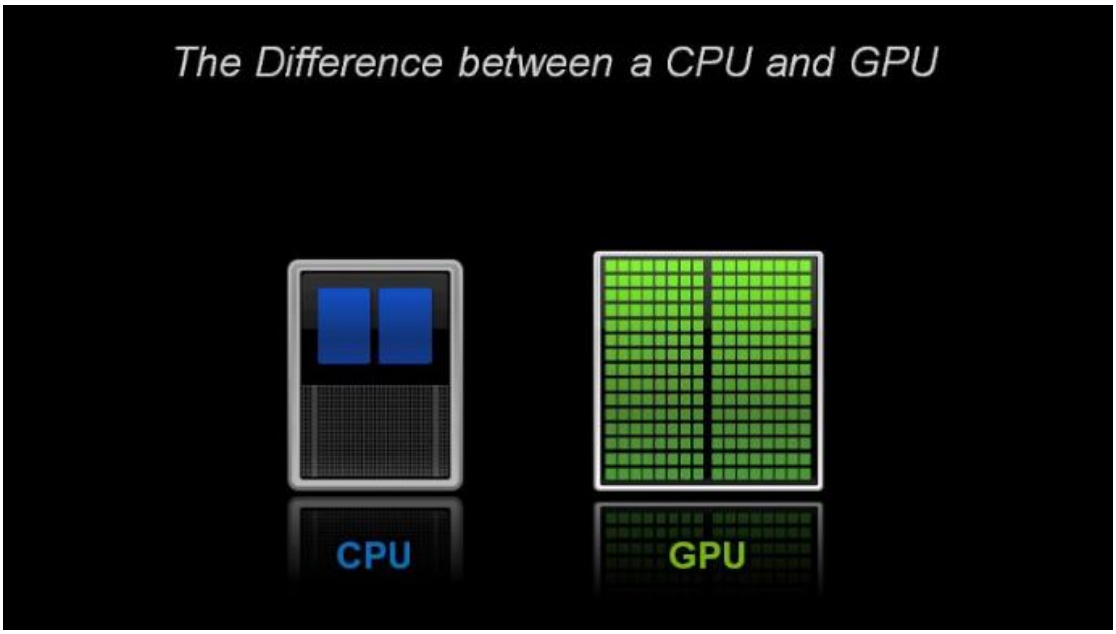
http://www.scielo.br/pdf/jbsmse/v33n4/a11v33n4.pdf [82]

2.5 CPU vs GPU

GPU ir CPU vykdo užduotis skirtingais būdais. Dažnai lyginant CPU galima vaizduoti kaip smegenis ir GPU kaip raumenis. CPU gali dirbti su daug skirtingų skaičiavimų, o GPU geriau panaudoja visą savo pajėgumą vienai užduočiai. Toks pasiskirtysmas yra todėl, nes CPU turi mažiau branduolių (danžniausiai iki 24) optimizuotų nuosekliaim procesų apdorojimui. Jo dizainas maksimizuoja procesoriaus našumą atliekant vieną užduotį darbe, tačiau užduočių apimtis gali būti plati. GPU kitaip nei CPU naudoja tūkstančius mažesnių ir efektyvesnių procesorių branduolių skirtų paraleliems skaičiavimas ir atlikti daug funkcijų vienu metu. Šiuolaikiniai GPU

suteikia didesnius skaičiavimo apdirbimo našumus, geresnį atminties pralaidumą ir efektyvumą lyginant su CPU. GPU vidutiniškai gali būti 50 – 100 kartų greistensi, vykdydami daug lygiagrečių procesų, tokių kaip mašininis komymasis arba didelių duomenų analizė.

https://medium.com/altumea/gpu-vs-cpu-computing-what-to-choose-a9788a2370c4



3 pav. CPU ir GPU palyginimas

Akivaizud skirtumas tarp CPU procesoriaus ir GPU multirpcesoriaus pateiktas višuje esančiame paveikslėlyje. Nors CPU branduoliai yra daug galingesni už GPU, tačiau dėl kartais net tūkstačiais kartų daugiau bradnuolių, GPU daug efektyviau išnaudoja lygegrečius skaičiavimus ypaę naudojant linijinį metodą.

2.6 GPU vs FPGA

Lyginant šias dvi technologijas FPGA yra efektyviau išnaudojantis energiją, o GPU labiau ekonomiškai efektyvus. FPGA yra sukurti konkuruojančius fiksuoto kalblelio operacijas su artimu techniniai įrangai programiniu sprendimu, GPU yra labiau pritaikytas lygegriatiems procesams su slankiojančiu kableliu naudojant tūkstančius mažų procesoriaus branduolių.

1 lentelė

GPU ir FPGA palyginimas

Funkcija	Analizė	Nugalėtojas
Slankiojančio kablelio	Maksimalus slaknkiojančio kablelio operacijų kiekis per sekudnė yra didesnis naudojant GPU su maksimaliu DSP palaikymu	GPU

apdorojimas		
Laiko vėlinimas	Perkelti algoritmai į FPGA demonstruoja deterministinį laiką, kuris yra su šiek tiek mažensiu vėlinimu nei lyginant su GPU.	FPGA
Apdorojimas/vatai	Matuojant GFLOPS per vatą, FPGA yra apie 3 kartus geresnis. Tačiau GPU kas met vis rodo geresnius rezultatus ir artėja prie FPGA.	FPGA
Sąsaja	GPU yra jungiamas naudojant PCIe jungtį, FPGA yra lankstesnis ir leidžia prisijungti prie kito įrenginio naudojant praktiškai bet kokią jungtį.	FPGA
Atgalinis suderinamumas	Programinė įrankos sukurtos darbui senesnėms GPU gali dirbti ir su naujomis. FPGA HDL gali būti perkeltas ir į naujesnę versiją, tačiau turi būti modifikuotas.	GPU
Lankstumas	FPGA neturi lankstumo modifikuojant sintezuojmą kodą ir perkeliant į techninę įrangą, GPU tai nesukuria papildomų trigdžių.	GPU
Dydis	Dėl maženio neregijos sunaudojimo FPGA reikalauja mažiau priemonių šilumos išskaidymui, kai yra montuojamas mažesnėje aplinkoje.	FPGA
Plėtojimas	Dauguma algoritmų yra pritaikyti darbui su GPU, o FPGA plėtojimas yra sudėtingas ir brangus.	GPU
Apdorojimas/kaina	Nors FPGA sunaudoja mažiau energijos, tačiau skaičiuojant pinigines sanaudas per vieną GFLOP, GPU yra daug pigesnis už FPGA.	GPU

http://www.bertendsp.com/pdf/whitepaper/BWP001_GPU_vs_FPGA_Performance_Comparison_v1.0.pdf [85]

Vienas iš didžiausių FPGA privalumų, tai kad FPGA mikroschemos yra techninis algoritmo variantas ir techninė įrangą yra greitesnė už programinę įrangą, taip pat FPGA naudoja mažiau energijos. GPU seniau reikėdavo daug energijos įvairiams skaičiavimas, todėl sprendimai, kurie jautrūs energijai negalėjo naudoti GPU, tačiau dabar naujos GPU plokštė sunaudoja daug mažiau energijos ir šiuo klausimu tampa dideliu konkurentu FPGA. GPU veikia programinės įrangos pagalba, todėl paleisti algoritmą užtrunka laiko. Visos instrukcijos pirmiausia turi būti paleidžiamas ir sustatomos į eilę, turi būti vykdomi matematiniai skaičiavimai ir rezultatai turi būti grąžinami į atminį. Tačiau GPU konstrukcija įgalina vykdyti daug skaičiavimų lygiagrečiai ir algoritmas vykdo skaičiavimus daug greičiau, nei tai vykto paeiliui procesoriuje. Skirtingai nuo standartinių CPU algoritmų, GPU algoritmas yra

paleidžiamas labai arti programinės įrangos taip padidinant šio sprendimo efektyvumą. GPU pranašumas pasimato naudojant slankiojančio kablelio operacijas. GPU branduoliai yra vietinės techninės įrangos plaukiojančio kablelio procesoriai. Pavyzdžiui 384 branduolių GPU gali atlikti 384 slankiojančo kablelio matematininius skaičiavimus kiekvieno ciklo metu. Papildomai GPU yra sukurti kartu su labai sparčia atmintimi ir tiesioginiu atminties pasiekiamumu (DMA), tokiu būdu GPU branduoliai gali pasiekti duomenis nendaudodami CPU ciklų. Yra daug paprasčiau atnaujinti GPU ar perašyti algoritmą skirtą GPU pakeitus techninę įrangą lyginant su FPGA, kur tai padaryti yra ganėtinai sudėtinga. GPU vartotojai turi prieigą prie nemokamų bibliotektų ir programinės įrangos kūrimo įrankių.

http://mil-embedded.com/articles/fpga-gpu-evolution-continues/ [86]

Norint pagerinti programų našumą ir energijos suvartojimo efektyvumą buvo sukurtos skirtingo akseleravimo technologijos, kaip GPU, FPGA ir ASIC. Lyginat su ASIC, GPU ir FPGA sulaukė daugiau populiarumo, nes šios tdvi technologijos yra lankstesnės ir lengviau programuojamos. Tačiau, kuri iš dviejų technologijų yra geresnė? Kad atsakyti šį klausimą buvo pasitelktas Rodinia standartas, kuris plačiai naudojamas GPU bendruomenės. Šiame darbe tyrėjai analitinį modelį su keliomis pagrindinėmis našumo reikšmėmis pipe_OPC ir e_para_factor tam kad geriau suprasti našumo skirtumus tarp FPGA ir GPU. Tyrėjai atliko analitinius tyrimus su 28nm Xilinx Virtex 7 FPGA ir NVIDIA K40c GPU. Tyrimo pabaigoje buvo nustatyta kad FPGA atlieka daugiau operacijų kiekvieno ciklo metu už GPU (pipe_OPC), tačiau turi mažesnį efektyvumą atsižvelgiant į paralelizmo faktorių (e_para_factor) lyginant su GPU. Mažesnis efektyvumas yra todėl, kad atminties pralaidumas FPGA yra lėtesnis.

https://vast.cs.ucla.edu/sites/default/files/publications/FCCM2018-paper120-final.pdf [87]

2.7 GPU atmintis

Šiuolaikiniai CPU susideda iš kelių ar keliolikos branduolių, kurie optimizuoti nuosekliam procesų vykdymui, palyginimui GPU susideda iš šimtų ar tūkstančių mažesnių ir efektyvesnių branduolių, kurie sukurti atlikti daug skirtingų užduočių vienu metu. GPU akseleruoti sprendimaidažniausiai naudoja GPU kartu su CPU paspartinti mokslinius, inžinerinius ir verslo sprendimus. Informaciją esančią GPU globalioje atmintyje užtrunka ilgiau apdoroti nei esančią GPU bendrojoje atmintyje.

Naudojant CUDA technologinį sprendimą GPU multiprocesoriaus branduolys apdoroja duomenis, kurie nėra imami tiesiai iš CPU atminties, nes tai sunaudototų daug laiko, todėl duomenys būna jau globalioje GPU atmintyje. Dėl šios priežasties labai svarbu teisingai išnaudoti atmintį esančią sistemą, kad negaišti laiko ir bereikalo neperdavinėti duomenų iš vienos atminties į kitą.

Duomenys lygegretiems skaičiavimams yra gaunami iš CPU. Atmintis skirta laikyti šiem duomenims yra išskiriama CPU ir GPU įrenginyje. Gavus visus reikalingus duomenis, jie yra perkopijuojami iš CPU į GPU atmintį. Baigus duomenų kopijavimą CPU siunčia instrukcija į GPU kad vykdytų lygegrečius skaičiavimus.

Data transfer from main memory to device memory

Consider Fig 3.1.The input data for parallel proceesing are got from the user using CPU. Memory is allocated for the data in both CPU and GPU. After getting all data, the data is copies from the main memory(CPU) to the device memory(GPU).The CPU instructs the GPU for parallel processing [2].

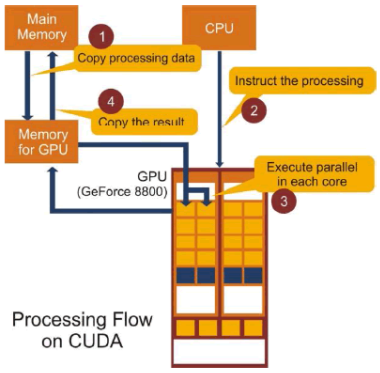


FIG 3.1

1. Copy data from main mem to GPU mem 2. CPU instructs the process to GPU 3. GPU execute parallel in each core 4. Copy the result from GPU mem to main mem

4 pav. Duomenų apdorojimas naudojant CUDA

http://www.rroij.com/open-access/gpu-acceleration-using-cuda-framework.pdf [88]

Galima išskirti šiuos esminius žingsnius:

1. Duomenų kopijavimas iš CPU pagrindinės atminties į GPU pagrindinę atmintį;
2. GPU siunčia instrukciją į GPU, kad pradėtų vykdyti procesą;
3. GPU vykdo kiekvieną skaičiavimą lygegrečiai, visuose branduoliuose;
4. Gautus rezultatus perkopijuoti iš GPU pagrindinės atminties į CPU pagrindinę atmintį.

Dauguma dabartinių CPU (Intel, AMD ir IBM) naudoja multirpcesorių architektūrą ir kiekvienas procesoriaus branduolys turi savo pirmo lygio spartinančią atmintį (CL1). Tokio tipo muliprocesorių architektūra taip pat turi antro lygio spartinančią atmintį (CL2) ir pagrindinę atmintį RAM. CL1 dažniausiai yra

suskaidytas į instrukcijų ir duomenų spartinančią atmintį ir CL2 dažniausia yra apjungta. Keturių branduolių Intel procesoriaus Spartinančios atminties pavyzdys pateiktas žemiau:

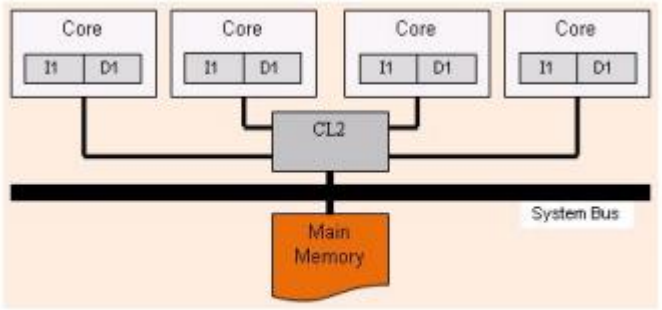


Fig. 2. Intel-like CPU cache memory organization.

5 pav. Intel tipo CPU spartinančioji atmintis

Sistemoje naudojančioje multiprocesorių CPU ir daugiaprocesorių GPU platformas, kurios palaiko CUDA, algoritmas visada pradedamas CPU. Algoritmo inicializavimas ir nuoseklios jo dalys yra vykdomos CPU. Vėliau duomenys ir lygiagrečios algoritmo dalys siunčiamos į GPU plokštę. Žemiau esantis paveiklėslis vaizduoja tipinį CPU – GPU bendradarbiavimą. Kiekvienai paraleliai algoritmo dalei aktyvuojami GPU multiprocesoriai ir pradedami vykdyti. GPU naudoja skirtinko tipo atmintis: globali atmintis yra didžiausia ir prieinama visiem skaičiavimo blokams, bei matoma visiems GPU procesoriams ir jų tinklui; bendra atmintis yra naudojama skaičiavimo bloko ir pasiekiama tik jam priklausiantiems branduoliams. Bendroji atmintis yra labai greita, tačiau daug mažesnė talpos už globalę atmintį. GPU bendroji atmintis padidina našumą, nes yra arčiau procesorių branduolių ir dedikuota tam tikram CUDA blokui.

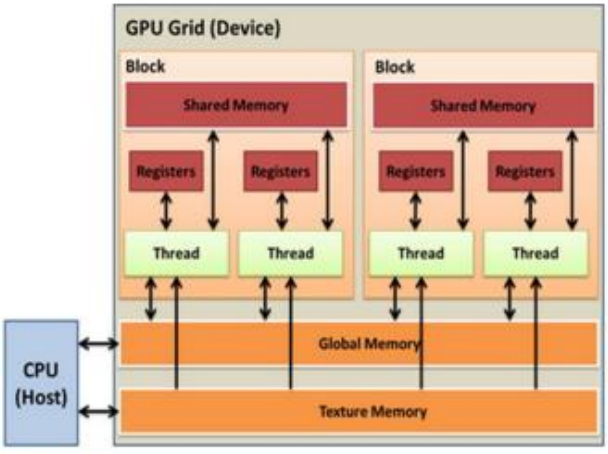


Fig. 3. GPU memory organization [2].

6 pav. CPU atminties organizavimas

Duomenų paralelizavimas yra svarbi paralelizavimo proceso technikos dalis, kuri išnaudoja lokalumo principą. Duomenų paralelizavime programa yra sukaidoma į atskiras dalis, kurios vykdo tą pačia instrukciją su skirtingais duomenimis. Masačiuceso Technologijų Instituto (MIT) tyrėjai pateikia dvi duomenų paralelizavimo strategijas: erdvinis duomenų skaidymas (SDP) ir laikinas duomenų dalinimas (TDP). Naudojant SDP strategiją, naudojant erdvinį indeksavimą duomenys yra padalinami tarp procesų (figure 4). Šio tipo strategija gali būti naudojama, kai turima didelių dimensijų erdviniai duomenys su mažai priklausomybių. Norint įgalinti komunikaciją ir sinchronicaziją tarp duomenų reikalingos papildomos instrukcijos. Duomenų padalinimas su paralelizuotu algoritmu tampa paprastas, kol algoritmas atlieka tą pačia funkciją tarp branduolių blokų su visais erdviniais indeksais.

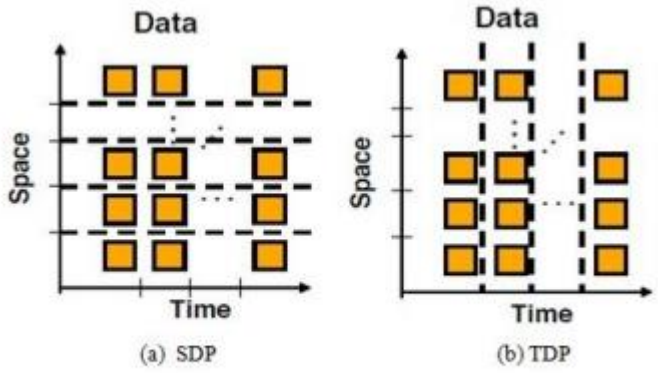


Fig. 4. Two Parallelization Strategies [7].

7 pav. Dvi galimos paralelizavimo strategijos

Naudojant TDP strategiją, pasitelkiami laikini indeksai dalinant duomenis tarp procesų. Atliekant skaičiavimus su kiekvienu procesu ir visų erdvinių indeksų susietų su jų laikinais indeksais kaip pavaizduota fig. 4. Šiuo atveju komunikacija tarp duomenų bus priklausoma nuo naudojamos paralelios aplikacijos. Tipiniu TDP atveju algoritmas duomenis naudoja nuo priskirto laikino indekso procesas vykdo visas priskirtas instrukcijas su tais duomenimis. Šio tipo strategija naudojama kai yra didelės apimties laikinos dimensijos duomenis su mažai priklausomybių. Ekeperimentai parodė kad TDP strategijų pagalba pasiekiamas geriausias pralaidumas, o su SDP strategijomis pasiekiamas mažiausias vėlinimas, tačiau su kokybės praradimu.

Tradicinio GPU skaičiavimo metodo metu duomenys (ir instrukcijos) iš CPU atminties yra perkopijuojamos į GPU atmintį kaip parodyta fig. 5. Duomenys iš

vientiso bloko/branduolio esančio CPU atmintyje perkopijuojami į GPU globalę atmintį. Šiuo atveju duomenys globalioje GPU atmintyje gali būti saugomi skirtinguose blokuose ir todėl tampa labai sunku perkelti duomenis į GPU bendrą atmintį. Naujas atminties tarp CPU ir GPU susiejimas yra reikalingas norint pagerinti bendrosios GPU atminties našumą.

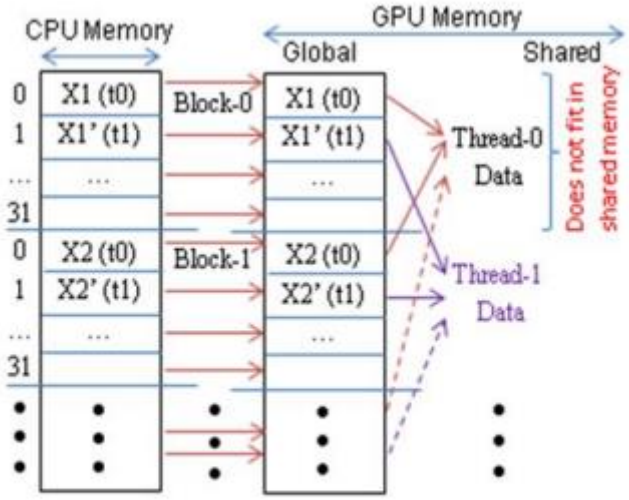


Fig. 5. Traditional CPU to GPU global memory mapping.

8 pav. Tradicinis globalios atminties komunikavimo modelis tarp CPU ir GPU

Šiame darbe autoriai pasiūlė naują metodą skirta susieti CPU – pagrindinę atmintį į GPU – globalę atmintį, tam kad padidinti sistemos našumą. Kaip matyti fig. 6 CPU duomenys turėtų būti pergrupuoti taip, kad duomenys susieti su tuo pačiu branduoliu galėtų būti išsaugomi atmintyje esančioje prie to branduolio ir duomenys eitų iš eilės. Šis duomenų pergrupavimas turėtų būti vykdomas CPU.

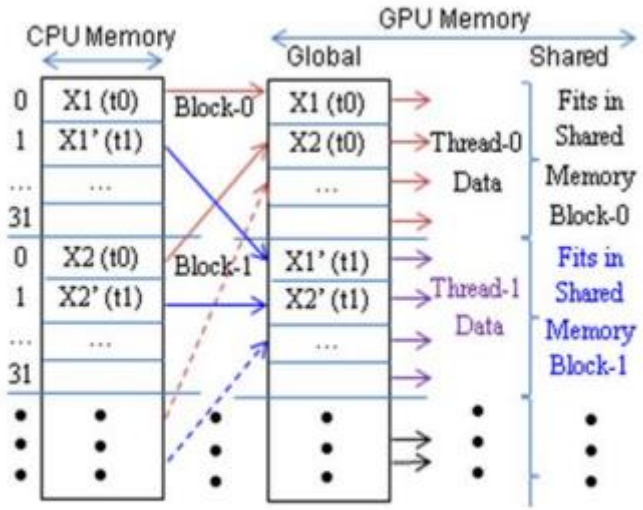


Fig. 6. Proposed CPU memory to GPU shared memory mapping.

9 pav. Siūlomas pagrindinės atmintis komunikavimas tarp CPU ir GPU

Naudojant šį duomenų susiejimo metodą duomenys X1, X2 ir kt. iš skirtingų CPU atminties vietų yra išsaugomos kartu globalioje GPU atmintyje. Kitaip nei prieš tai minėtu atveju duomenys turėtų būti saugomi kartu globalioje GPU atmintyje ir padidinti sistemos našumą. Autoriai atlikę tyrimus naudodami savo pasiūlytą metodą pastebėjo kad našumas ir vėlinimas sumažėja iki 75%.

https://www.researchgate.net/publication/286746607_A_talented_CPU-to-GPU_memory_mapping_technique [89]

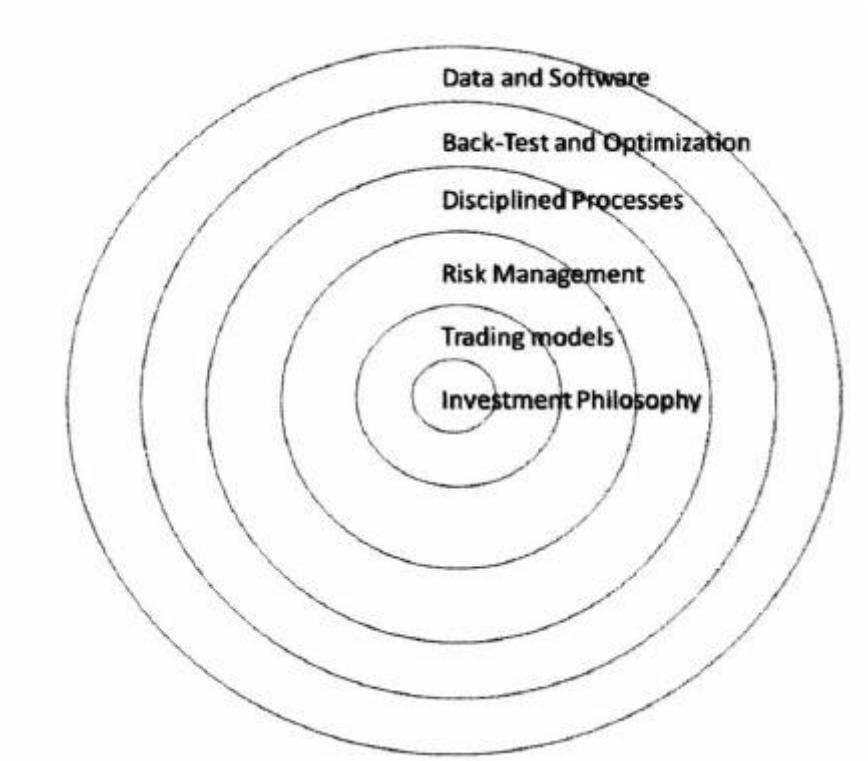
3 Didelio dažnio prekybos strategijų kūrimas ir procesų valdymas

Amerikos kariuomenės strategas J. A. Warden III, pristatė penkių žiedų sistemos modelį skirtą oro kampanijai tvirtindamas, jog beatkuri sudėtinga sistema gali būti suskirstyta į penkis koncentrinius žiedus. Kiekvienas žiedas – lyderystė, procesai, infrastruktūra, populiacija ir veiksmų vienetai – gali būti naudojamas atskirti pagrindinius sistemos elementus, kurie turėtų būti keičiami. Šis modelis buvo sėkmingai naudojamas oro pajėgų strategų pirmajame Golfo kare. Naudojant J. A. Warden III modelį galima suskirstyti prekybos sistema į sekančius lygius:

- *Investavimo filosofija*
- *Prekybos strategijos ir matematiniai modeliai*
- *Rizikos ir pinigų valdymas*

- *Procesai*
- *Testavimas ir optimizavimas*
- *Finansiniai duomenys ir prekybos programinė įranga*

Galima laikyti, jog kiekvienas iš išvardintų lygių arba žiedų yra prekybos sistemos gravitacijos centras. Tokiu atveju, šių šešių žiedų pagrindinis tikslas yra sukurti prekybos sistemą, kuri teikia norimą naudą. Norint pasiekti šį tikslą reikia išnagrinėti kiekvieną žiedą atskirai ir kartu su kitais, tada galima sukurti tvirtą ir pelningą prekybos sistemą. Sistemos kūrėjau reikia įtraukti kiek galima daugiau žiedų išryškinant jų ryšį su kitais žiedais. [73]



10 pav. Prekybos sistemos šeši žiedai

Šaltinis: [73]

Didelio dažnio prekyboje neįmanoma išsiversti be prekybos sistemų, nes prekyba vykdoma dideliais greičiais ir reikia skubiai apdoroti gauna informacija iš finansų biržų. Naudojant prekybos sistemas pasitelkiama visa kompiuterių skaičiavimo galia, kuri padeda taupyti laiką ir išvengti žmogiškųjų klaidų. Didžioji dauguma didelio dažnio prekybos firmų vadovaujas filosofija „nesistenk nuspėti rinkos, koreguok į ją“. Norint realizuoti šią filosofiją reikalinga mechaniška prekybos sistema, kuri fiksuoja trumpus laiko tarpus rinkoje, stebi juos ir analizuodama priima sprendimus. Apdorojant didelius kiekius informacijos gaunama

daug naudingos informacijos , bet ir susiduriama su iššūkiais, kurie svarbiausi yra prekybos sistemų kūrėjams. [71]

Testuojant sukuriama daug testinių sandorių naudojant didelio dažnio informacija su skirtingais modeliais. Tačiau ir čia susiduriama su didelių duomenų apdorojimo problema. Didelio dažnio informacija yra aptriukšminta, šie triukšmai gali trukdyti sėkmingai funkcionuoti prekybos sistemai, nes ji gali pasigauti triukšmą ir juo remdamasi vykdyti sprendimus, neatsižvelgiant į rinkos pokyčius. Norint dirbti su didelio dažnio informacija reikia naudoti tinkamus matematinius modelius ir technologijas, kurie padeda efektyviau apdoroti turimus duomenis testuojant ir pačioje prekybos sistemoje.[72]

3.1 Prekybos sistemos struktūra ir procesai

Finansų rinkose yra naudojami išplėstiniai tyrimai skirti kurti išmaniu matematinius prekybos modelius. Vienodai svarbu, o gal net svarbiau yra sistemos įgyvendinimo struktūra. Sistemos dizainas ir procesai yra sąjunga tarp geriausių praktikų inžinerijoje ir kiekybiniuose finansuose. Dabartiniame prekybos pasaulyje norint būti konkurencingu reikia pastoviai ieškoti, kurta ir plėtoti geresnes prekybos sistemas ." [72]. Investavimo idėjos konvertavimas į matematinį modelį ir tada į veikiančią prekybos sistemą kuo galima greičiau ir efektyviau yra svarbu norint konkuruoti rinkoje.

Yra kelios pagrindinės charakteristikos, kurios apibūdina gerą prekybos sistemą. Tushar Chande savo darbe Beyond Technical Analysis aptaria „šešis kardinolų“ taiskyklę prekybos sistemos kūrime. Kirkpatrick ir Dahlquist savo knygoje Technical Analysis [73] pateikia šias geros prekybos sistemos charakteristikas:

- *Teigiami lūkesčiai*
- *Mažas kiekis prekybos taisyklių – dešimt arba mažiau*
- *Pastovios parametrų reikšmės*
- *Galimybė prekiauti keliais kontraktais*
- *Rizikos kontrolė*
- *Pilnai automatizuota.*

Nors šios charakteristikos apibūdiną gerą prekybos sistemą, tačiau skirtingos sistemos vertinamos skirtingai. [73]

3.2 Didelio dažnio prekybos sistemos dizainas ir kūrimas

Prekybos firmos gali susidurti su daugybe kliūčių, kai jos bando valdyti prekybos sistemos valdymą ir kūrimą, tačiau šios kliūtys nėra naujos finansų sistemoms. Prekybos sistema pagrinde yra sudėtinga programinė įranga. Programinės įrangos inžinerija yra labai svarbi sritis kuriant sistemų dizainą, veikimą ir reorganizavimą. Finansų analitikai ir finansų rinkų prekiautojai gali išmokti finansų inžinerijos, tačiau norint išvengti sistemos nestabilumo ir triūsingų matematinių modelių parinkimų, inžinerija geriau patikėti specialistams.[77] [78]

Didelio dažnio prekybos sistema gali būti išskaidyta į tris pagrindines dalis: tyrimas, prototipas ir kūrimas. Kiekviena dalis turi savo vidinius procesus ir funkcijas.

Tyrimas - kiekviena prekybos sistema prasideda nuo jai skirtos investavimo filosofijos ir prekybos idėjos, kurios buvo gautos atlikus įvairius tyrimus. Atlikus tyrimą gaunama kuriamos sistemos dizaino dokumentacija.

Prototipas – šioje dalyje patikrinamos kuriamos sistemos investavimo idėjos. Tam kad patikrint reikia pasiruošti didelio dažnio istorinių duomenų. Taip pat sukuriamas sistemos modelis naudojant MATLAB ar S-PLUS. Pabaigoje yra ištestuojamas gautas modelis ir optimizuojamas pagal gautus rezultatus. Jie modelis būna patvirtinamas, tada yra paruošiama jo detali specifikacija ir perduodama kūrimo komandai.

Kūrimas – šiame etape yra realizuoja didelio dažnio prekybos sistema. Kuriant yra vadovaujamasi programinės įrangos kūrimo ir kokybės užtikrinimo procesais. Papildomai su programinės įrangos testavimu, bus paleista eksperimentinė prekybos platforma, kurios pagalba bus patikrinta visa sistema.

Prekybos sistemos kūrimas nebūtinai turi griežtai vadovautis krioklio modeliu. Iteracijų ir kilpų gali atsirasti kiekviename sistemos kūrimo etape. Nors Six Sigma ar Agile metodai gali būti naudingi, tačiau nėra standarto kuriant prekybos sistemoms.

3.2.1 Didelio dažnio prekybos sistemos kūrimas naudojant modeliais remtą sistemos inžineriją (MBSE)

Sistemų inžinerija yra tarpdisciplininis dalykas, kurio tikslas išaiškinti kaip sudėtingos sistemos turėtų būti kuriamos ir valdomos. Didelio dažnio prekyba ir yra sudėtinga Sistema, kuri reikalauja greičio, stabilumo ir patikimumo. Sistemos

inžinieriai kurdami tokias sistemas turi atsižvelgti į darbo procesus ir įrankius, skirtus dirbti su tokiomis sistemomis.

Kitos inžinerijos disciplinos pereina iš dokumentais remtų sistemų į modeliais remtus. Friedenthal, Moore ir Steiner [75] teigia, jog pagrindiniai modeliais remtų sistemos inžinerijos privalumai yra sustiprinti ryšiai tarp sistemos elementų, specifikacijų ir dizaino tikslumas, modelio integracija, galimybė pagerinti sistemos dizaino kokybę ir produktyvumą, bei sumažinti sistemos kūrimo rizikas.

Yra daug galimų MBSE (model based system engeneering) kūrimo kalbų. Šiame darbe aptariamos SysML ir objektų – procesų metodologija. Šios kalbos bus naudojamas kuriant didelio dažnio prekybos sistemų architektūros. Abi šios kalbos pilnai tinka modeliuoti didelio dažnio sistemas ir turi skirtingus loginius procesus, tačiau padeda siekti to paties tikslo: kalibruoti, apjungti ir patikrinti sukurtą modelį. [75]

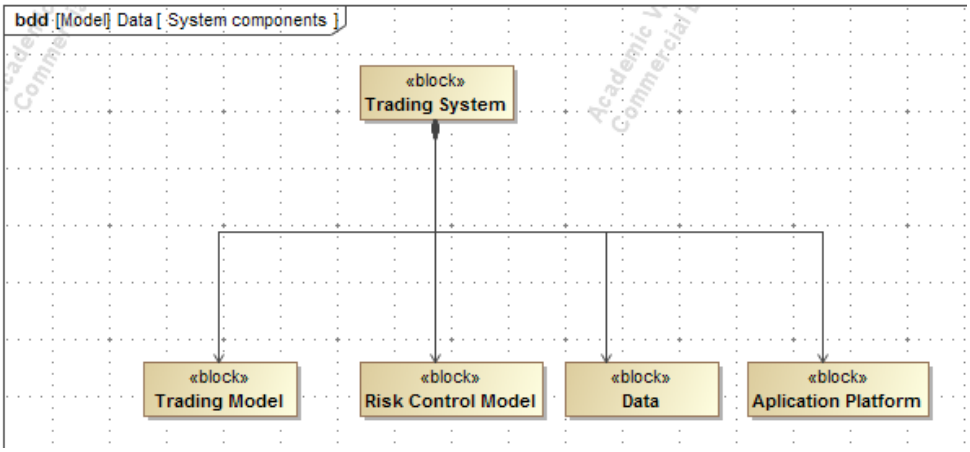
SysML yra grafinė modeliavimo kalba kuri palaiko sudėtingų sistemų analizavimą, specifikacijas, dizainą ir tikrinimą. Ši modeliavimo kalba suteikia įrankius užfiksuoti modeliuojamos sistemos informaciją. Pasirinktas metodas nurodo, kurie veiksmai yra vykdomi, jų tvarkymas ir kokie modelio artefaktai yra sukurti atspindėti kuriamai sistemai.[75]

Objektų – procesų metodologija (OPM) yra modeliavimo kalba skirta modeliuoti sudėtingas sistemas kurios susideda iš žmonių, fizinių objektų ir informacijos. Ši modeliavimo kalba yra formali paradigma skirta sistemų kūrimui, gyvavimo ciklo palaikymui ir evoliucijai. OPM apjungia grafinius modelius su natūralios kalbos sakiniais, skirtis išreikšti funkcijas, struktūrą ir integruotas sistemos į modelį veikimą. [76]

SysML yra naudinga žmonėms, kurie jau turi programinės įrangos kūrimo patirties. Naudojant šią kalbą yra paprasčiau aprašyti sistemos procesus ir funkcijų kilpas esančias, kuriamoje sistemoje. Ši modeliavimo kalba yra paprastesnė už OPM, kai tenka aiškinti sistemos procesus finansų inžinieriams, prekiautojams ir kitiems asmenims, kurie neturi žinių apie sistemos architektūrą. Objektų – procesų metodologija yra puikus įrankis, kuris identifikuoja visus sistemos objektus ir vidinius procesų ryšius tarp tų objektų.

Abi išvardintos kalbos yra tinkamos modeliuoti didelio dažnio prekybos sistemas. Kuris modeliavimo kalba geresnė, priklauso nuo inžinieriaus keliamų reikalavimų, sistemos sudėtingumo ir nuo kuriamos sistemos pobūdžio.

Kai buvo kuriam didelio dažnio prekybos Sistema naudojant OPM, modelis buvo padalintas į kelias skirtingas diagramas, kurios atvaizdavo skirtingus procesus skirtinguose OPM lygmenyse. Aukštesnio lygio diagrama atvaizduoja ryšį tarp jo ir žemesnio lygio diagramų. Kiekviena sukurta diagrama yra išplėtimas prieš tai buvusios diagramos, kuri atvaizduoja pasirinkta sistemos procesą.



11 pav. Prekybos sistemos komponentai

Kaip buvo aptarta prieš tai didelio dažnio prekybos sistema yra išskaidyta į šešis lygius:

- *Investavimo filosofija*
- *Prekybos strategijos ir matematiniai modeliai*
- *Rizikos ir pinigų valdymas*
- *Procesai*
- *Testavimas ir optimizavimas*
- *Finansiniai duomenys ir prekybos programinė įranga*

Šie šeši prekybos sistemos lygiai gali būti įtraukti kuriant sistemą - rizikos valdymas, prekybos strategijos, finansiniai duomenys ir prekybos programinė įranga didelio dažnio sistemos komponentai/objektai , o investavimo filosofija, testavimas ir optimizavimas yra procesai. Vadovaujantis MBSE būdu viršuje esantis paveikslėlis atvaizduoja pirmo lygio išskaidymą:

- *Prekybos strategijos/modeliai*
- *Duomenys*
- *Rizikos valdymo modelis*

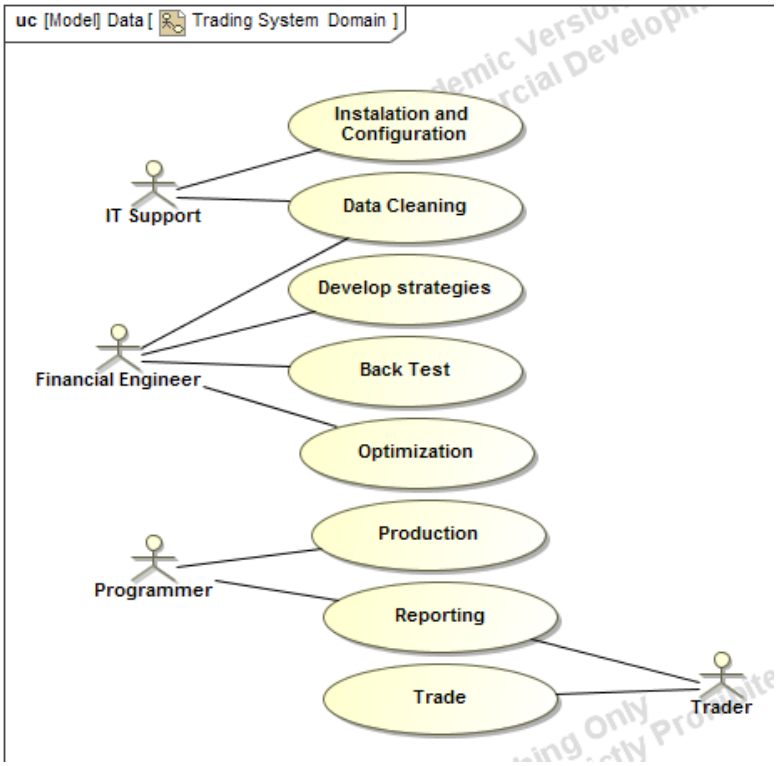
- *Aplikacijos platforma*

Yra daug vartotojų, kurie tiesiogiai ar netiesiogiai yra susyja su didelio dažnio prekybos sistema. Keturi tiesioginiai vartotojai, kurie yra susyja su sistemos kūrimu – biržų prekiautojai, finansų analitikai, programuotojai ir IT prižiūrėtojai. Biržos prekiautojai yra galutiniai sistemos vartotojai. Automatinės prekybos sistemos reikalauja nedidelio žmogaus įsikišimo ir prekiautojai tik stebi kaip vyksta prekyba ir gali vykdyti mažus pakeitimus.

Finansų analitikai/inžinieriai daugiausiai laiko praleidžia tyrinėdami ir kurdami naujas strategijas, bei jas testuodami ir optimizuodami. Jų pagrindinis dėmesys yra skiriamas kurti algoritmą ir modeliui sistemos prototipą naudojant SAS, MATLAB arba S-PLUS. Jie gali palikti grynai techninius klausimus, problemų sprendimus, duomenų bazių kūrimą programuotojams.

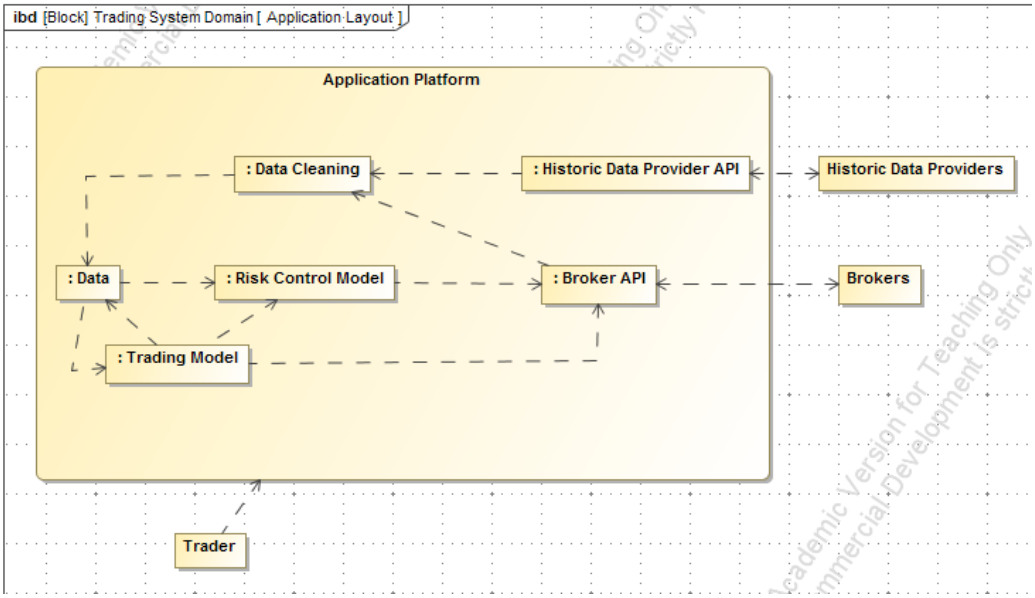
Programuotojai yra atsakingi už sistemos programavimą ir jie turi turėti gerus C++, C# ir JAVA įgūdžius. Jie paima patvirtintus sistemų prototipus ir paverčia juos į realias sistemas. Programuotojams nereikia suprasti finansų rinkų.

IT prižiūrėtojai suteikia palaikymo funkciją, diegimą, konfigūravimą ir mokymą.



12 pav. Panaudojimo atvejų diagrama

Viršuje esanti paveikslėlis vaizduoja panaudojimų atvejų diagramą, skirtą prekybos sistemos kūrimui. Skirtingos firmos organizuojančios prekybų sistemų kūrimą, gali keisti panaudojimo atvejų modelį, tačiau pateiktas šiame darbe atvaizduoja esmines dalis.



13 pav. Realaus prekybos proceso diagrama

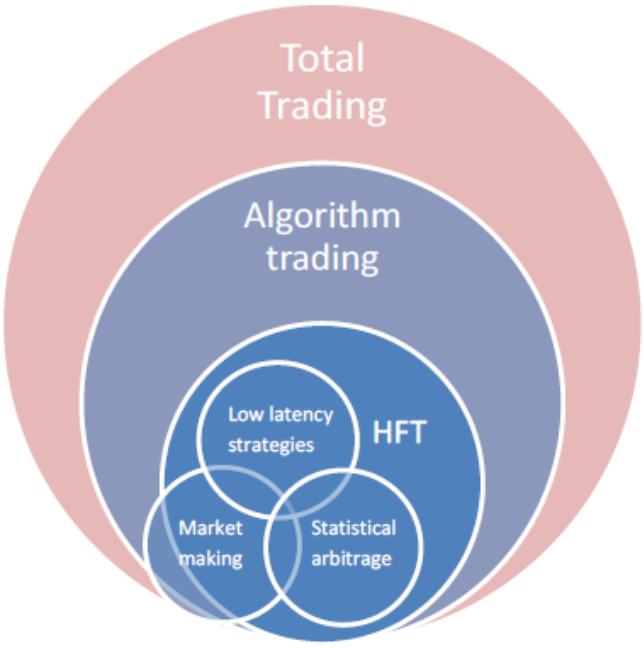
Ketvirtame paveikslėlis atvaizduoja prekybos procesą: istorinė informacija pateikta duomenų tiekėjų keliauja per duomenų valymą, kuris taip pat gali būti naudojamas apskaičiuoti rizikas ir kitus rinkos veiksnius. Matematiniai modeliai skirti skaičiavimams yra gerai ištestuojami ir tada diegiami į prekybos sistemą. Prekybos modelis turi nuskaitinėti informaciją realiu laiku gaunamą iš brokerių. Visi prekybos sprendimai yra įvertinami rizikos vertinimo modelio ir kai prekybos signalai yra patvirtinami jie siunčiami atgal brokeriui.

Apibendrinus didelio dažnio prekybos sistema yra pilnai automatizuotas procesas, skirtas apdoroti gaunamus duomenis iš elektroninių biržų, perleisti informaciją per sprendimų logikos modulį, kurti užsakymus ir gauti informaciją apie patvirtinus užsakymus. Prekiautojas nesikiša į sistemos veikimą, o tik stebi kaip viskas veikia.

Papildomai 4 paveikslėlis parodo aplikacijos platformos komponentus, kurie gali būti išskaidomi į objektus: API modulis, duomenų valymo modulis, atvaizdavimo modulis ir vykdymo modulis. API modulis egzistuoja tik tam, kad sujungtų duomenų tiekėjus su prekybos sistema. Duomenų tiekėjai gali būti skaidomi į du kitus objektus: istorinių duomenų tiekėjai ir realaus laiko duomenų tiekėjai.

3.3 Didelio dažnio prekybos strategijos

Tyrimai atskleidžia, kad daugiau nei pusę visų prekybų sudaro algoritminė prekybą, kuri atliekama kompiuterių ir robotų. Didžiąją algoritminės prekybos dalį sudaro didelio dažnio prekyba.[44]



Šaltinis: [44]

14 pav. Skirtinų prekybos metodų užimama dalis rinkose.

Dauguma strategijų, kurios naudoja didelio dažnio duomenis, nėra naujos ir buvo naudotos ankščiau. Tačiau, techniniai sprendimai ir patobulinimai, atsiradę didelio dažnio prekybos dėka, leido išnaudot šias strategijas visu pajėgumu. Visos strategijos, su kuriomis susiduriama prekiaujant dideliu dažniu galima suskirstyti į tris pagrindines grupes t.y. rinkos kūrimo (market making), statistinis arbitražas (statistical arbitrage) ir mažo vėlavimo (low latency).[44] Detalesnis jų veikimas yra toks:

- Rinkos kūrimas - rinkos kūrimo strategijos, rinkose kuria likvidumą, taip padedančios suaktyvinti ir išjudinti lėtai veikiančias rinkas. Šio tipo strategijos naudoja algoritmus, kurie apskaičiuoja finansinio instrumento kainas (pirkimo arba pardavimo), kurios gali būti pasiūlytos naudojant didelio dažnio duomenų perdavimą. Tipiškai apskaičiuojama finansinio instrumento kaina, kuri yra vienoje rinkoje lyginant, tą patį finansinį instrumentą, kitoje rinkoje. Tokiu būdu skirtumas, tarp kainų yra išnaudojamas algoritmo ir taip vykdoma prekyba.[10]

- Mažas vėlavimas - pagrindinis faktorius, lemiantis sėkmingą prekybą, naudojant mažo vėlavimo strategijas, būti greitesniam už rinką. Mažo vėlavimo prekybą sudaro daug įvairių skirtingų strategijų. Jų sėkmė priklauso nuo sistemos greitumo arba greitesnės ir tikslesnės informacijos gavimo iš prekybos platformos.

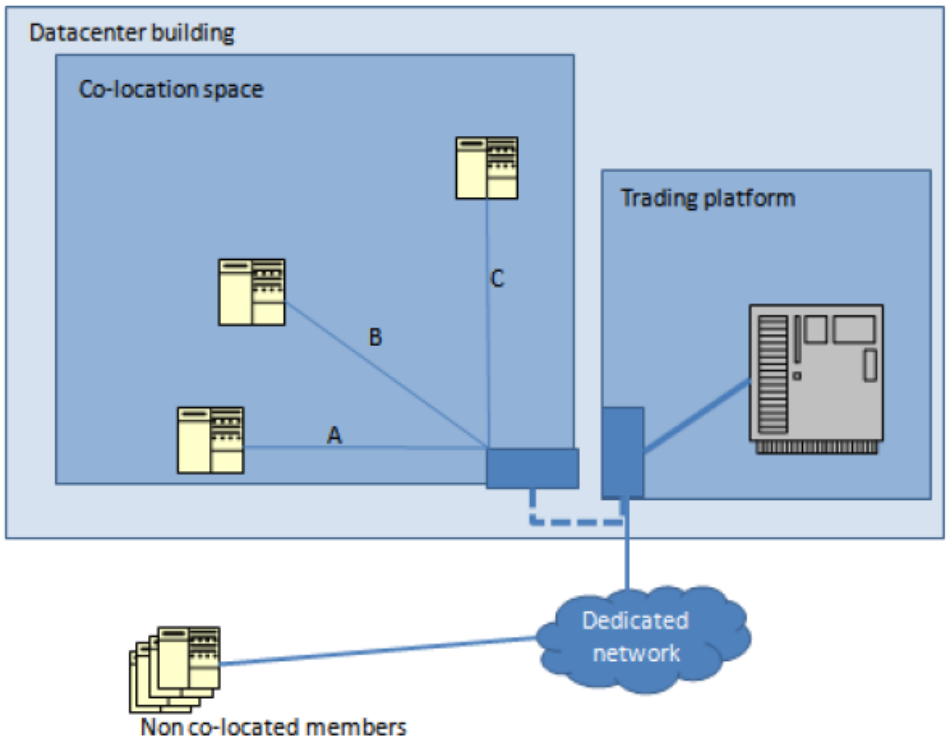
1. Paskutinių užsakymų paieška atidarant staigias pozicijas arba jų atšaukimas. Tokiu būdu investuotojas visada sumoka maksimalią kainą už užsakymą, o skirtumas atitenka didelio dažnio prekybos naudotojui.

2. Analizuojama kiti rinkoje esantys algoritmai, bei stebimas jų veikimas, atsižvelgiant į pateikiamus užsakymus. Kai tik sužinoma, kaip nors vienas algoritmas veikia, šios žinios gali būti panaudojamos prekiaujant ir aplenkiant, kitus rinkos dalyvius.

3. Eiti per rinką mažais ir pastoviais užsakymais, tam kad išnaudoti rinkos nepastovumus ir gauti pelną iš prieš tai atidarytų pozicijų.

4. Patiems sukurti geriausią nacionalinį pirkt - parduoti užsakymą (NBBO), kuris pasirodo NBBO viešose paskyrose tik praėjus keliomis milisekundėm ir išnaudoti tą laiką.[4]

- Statistinis arbitražas - tikslas yra stebėti finansinius instrumentus. Stebėjimo metu laukiama nukrypimo, nuo planuojamo, šio instrumento, judėjimo ir tai traktuojamas kaip prekybos signalas, kurio pagalba galima nuspėti kada kaina grįš prie normalaus judėjimo. Statistinis arbitražas dažnai yra vadinamas porų prekyba, kurios tikslas stebėti porą finansinių instrumentų ir laukti kada jų kainos nukryps nuo normalaus judėjimo.[44] Vykdant prekyba naudojant didelio dažnio duomenis, labai svarbu kuo greičiau ir su kiek galima mažesniu vėlinimu, gauti ir apdoroti duomenis. Vienas iš būdų tai padaryti - pastatyti serverį, kuriame yra prekybos algoritmas, kiek galima arčiau prekybos platformos. Dėl šios priežasties, labai svarbu parinkti tinkamą vietą, kur bus statomas serveris, su prekybos algoritmu.



Šaltinis: [44]

15 pav. Prekybos platformų vietos parinkimo palyginimas.

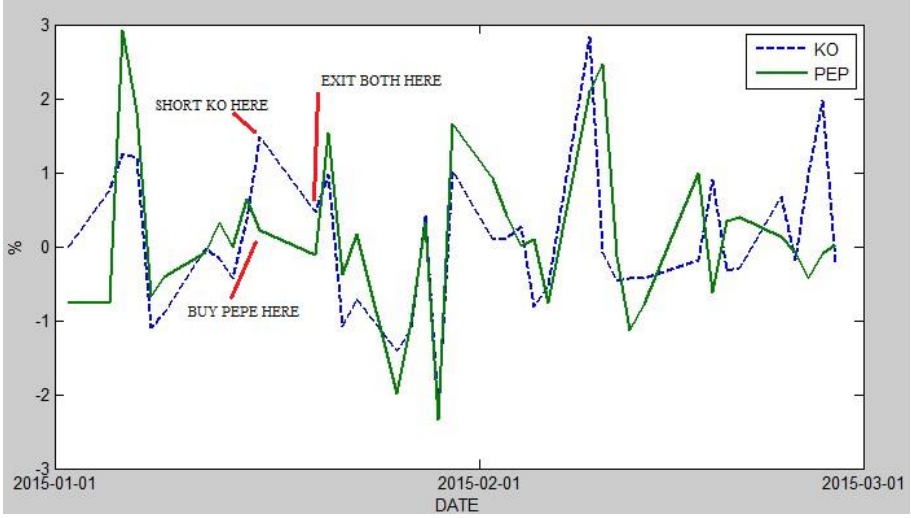
Dauguma prekybos platformų net siūlo išsipirkti vietas serveriams, tame pačiame pastate, kur yra jų siūloma prekybos platforma, taip sumažinant atstumą tarp prekybos serverio ir prekybos platformos. Tokiu būdu informacija iš prekybos platformos nukeliauja trumpiausią galimą kelią iki serverio, taip sumažindama informacijos vėlinimą iki minimumo.

Porų prekyba yra neutrali rinkos statistinio arbitražo strategija, kuri remiasi finansinių instrumentų kainos konvergencija. Pirmiausia šio tipo strategijos turi rasti du finansinius instumentus, kurių kainos juda kartu t. y. koreliuoja tarpusavyje. Atvejais, kai šie du finansiniai instrumentai nukrypsta vienas nuo kito yra išnaudojami porų prekybos strategijos. Pigesnis finansinis isntrumentas yra perkamas ir brangesnis yra parduodamas ir kai kainos grįžta į savo istorinį vidurkį pozicijos uždaromos. Šios strategijos pagrindinis tikslas yra išnaudoti laikinus finansinių instumentų klaidingus kainų nustatymus.[28]

Visos finansų rinkos yra paremtus tuo pačiu principu: pirkti maža kaina ir parduoti kai ji pakyla. Dėl šios priežasties yra svarbu sukurti prekybos strategiją su mažiausia rizika. Reikia paminėti kad tikrasis arbitražas yra prekybos strategija su

nuline rizika. Kaip pavyzdy galima pateikti prekyba tuo pačiu finansiniu instumentu skirtingose rinkose. Kito tipo prekyba yra statisninis arbitražas, kuris jau nėra be rizikos. Šio tipo strategijos tiki gauti didesnį pelną nei yra rizika. Prieš vykdant prekybatokio tipo strategijai reikia nustatyti kurio finansinio instrumento kaina yra perdidelė ir kurio permaža. [24]

Pateiktame pavyzdyje yra dvi koreliuojančios akcijos KO (Coca-Cola) ir PEP (Pepsi). Istoriniu laikotarpiu abiejų akcijos juda panašiai, labai nenutoldamaos viena nuo kitos.



16 pav. Statistinis arbitražas.

Atvejais kai skirtumas tarp abiejų akcijų staigiai padidėja ar sumažėja yra išnaudojamas porų prekybos strategijos ir laukiama, kol jų skirtumas grįš prie istorinio vidurkio.

3.3.1 Statistinis arbitražo naudojant koreliaciją

Ryšys tarp dviejų finansinių instrumentų, kai jie sekavienas kitą yra vadinamas koreliacija. Kai finansinių instrumentų pora yra labai koreliuota, porų prekybos strategija turi ieškoti smulkių pasikeitimų jų korelieacijoje ir išnaudot atvejus kai šie finansiniai instrumentai vienas nuo kito nutolsta. Tam kad rasti koreliuojančia porą yra skaičiuojamas koreliacijos koeficiantas arba naudojamas minimalaus atstumo kriterijus kaip kvadratinis skirtumas tarp dviejų finansinių instrumentų normalizuotų kainų. Finansiniam instrumentui parenkama ta pora, kuri turi mažiausia dvadratinį skirtumą (Lau et al., 2016; Miao and Clements, 2002).

3.3.2 Statistinis arbitražo naudojant kointegraciją

Kointegracijos metodas naudoja matematinį modelį, kurie buvo sukurti Engelio ir Grangerio (Engle and Granger, 1987). Šis modelis sulaukė didelio ekonomistų susidomėjimo per pastaruosius 20 metų. Kointeracija teikia, kad yra situacijų, net kai yra dvi nestacionarios laiko eilutės, tam tikra linijinė šių laiko eilučių kombinacija yra stacionari. Šios dvi laiko eilutės juda kartu fiksuotais žingsniais. Jų kointegracija gali būti apibūdinama taip: x_t ir y_t yra dvi laiko eilutės, kurios yra nestacionarios. Jei yra toks parametras γ, kuriam tinka tokia lygtis:

$$z_t = y_t - \gamma x_t \qquad (1)$$

Jei ši lygtis yra stacionarus procesas, tada x_t ir y_t yra kointerguoti. Šis metodas tapo labia svarbus ieįkant ryšių tarp finansinių instrument (Miao, 2014).

3.4 Didelio dažnio prekybos nauda

Tokio tipo prekyba sumažina skirtumas tarp pirk – parduok signalų kainų, padidina prekybos greičius, didina rinkų likvidumą, sumažina rinkų nepastovumus, kitimus ir prekybos kaštus, bendrai didelio dažnio prekyba didina rinkos efektyvumą.

Dėl greičio, kurį sukuria didelio dažnio prekyba, rinkos kūrėjai gali pastoviai reguliuoti pirkti ir parduoti kainas, kurias jie siūlo rinkoms esant skirtingoms situacijoms. Tai reiškia, jog jie gali išlikti arčiau norimos kainos, nedidinant savo prekybos rizikos. Padidėjęs prekybos greitis, reiškia jog yra mažesnis skirtumas tarp pirk – parduok kainų, tai lemia žemesnius prekybos kaštus, nes prekybos dalyviai gali prekiauti patrauklesnėmis jiems sąlygomis.

Kita didelio dažnio prekybos nauda yra likvidumo kūrimas įvairiose rinkose. Rinkos, kurios iki to nebuvo likvidžios ir prekyba vykdavo labai lėtai, integravus didelio dažnio prekybos sistemas, padėjo suaktyvinti prekybą. Pirk – parduok kainų skirtumas dažnai naudojamas nustatyti rinkos likvidumui. Kuo mažesnis skirtumas tarp kainų, tuo likvidesnė yra rinka. Kitas faktorius, kuris parodo likvidumą yra rinkos gylis t.y. norint pakeisti akcijų kainą, reikalinga pateikti labai didelį užsakymą, tokiu būdu parduodama daugiau akcijų ar kitų finansinių instrumentų.[4][44]

Greitis, kuris parodo, kaip greitai investuotojų užsakymai yra vykdomi ,padidėja, dėl didelio dažnio prekybos naudojimo. Tai reškia, jog yra mažiau laiko netiksliems kainų pasikeitimams, tarp užsakymo pateikimo ir jo vykdymo. Tarp laiko,

kada užsakymas yra pateikiamas („ankstyvo“ investuotojo) ir kada jis pradedamas vykdyti („vėlyvo“ investuotojo), gali atsirasti svarbios informacijos, kuria nepasinaudoja „ankstyvasis“ investuotojas. Dėl šios priežasties gali atsirasti klaidingo pasirinkimo problema: „vėlyvas“ investuotojas gali pradėti prekiauti prieš „ankstyvą“ investuotoją išlaukdamas naujos informacijos, taip įgaudamas kainos pranašumą. Didelio dažnio duomenų naudojimas sumažina tokios situacijos galimybę.[7][44]

Didelio dažnio prekybos strategijos naudojamas ir esant labai nestabiliai rinkai, taip mažinamas jos nestabilumas, didinamas rinkos likvidumas ir pateikiama stabilesnė kainų informacija. Skirtumas tarp pirk –parduok kainų, esant rinkų nestabilumui padidėja, tačiau kaip jau buvo minėta, didelio dažnio prekybos strategijos šį skirtumą mažina, taip stabilizuojant rinką.[7][44]

3.5 Didelio dažnio prekybos rizika

Visos rizikos, kurios yra susijusios su didelio dažnio duomenų naudojimu, galima suskirstyti į prekybos ir IT rizikas.

Viena iš prekybos rizikų yra, kad maži ir dideli rinkos dalyviai naudoja tas pačias strategijas. Visos didelio dažnio strategijos yra vykdomos algoritmų, todėl yra galima grandininė reakcija, jog strategijos pradės viena kitą remti iki maksimumo. Dažnai, kai atsiranda rinkos nestabilumas, kurio neina paaiškinti, yra kaltinami prekybos algoritmai.

Didelio dažnio strategijų naudotojai dažnai didžiuojasi tuo, jog didina rinkų likvidumą net esant nestabilumams rinkose. Tačiau nėra garantijos, jog šių strategijų naudotojai ir toliau taip elgsis atsiradus naujoms sąlygoms rinkose.

Didesni rinkos dalyviai teigia, jog didžiausia rizika, tokio tipo strategijų yra, jog nėra jokio šių strategijų naudotojų įsipareigojimų, kur jie vykdo prekybą. Tai reiškia, jog šie rinkos dalyviai neįsipareigoja naudoti būtent tas strategijas, kurios teikia didžiausia naudą rinkoms.

Taip pat yra manančių, jog didelio dažnio prekyba padidino prekybos kaštus. Pavyzdžiui yra tyčia sumažinamas užsakymo kiekis, parodant mažesnį finansinio instrumento kainos pasikeitimą. Tokiu būdu didelio dažnio prekybos naudotojai kartais net gauna premijas už fiktyvaus likvidumo kūrimą, o kiti rinkos dalyviai, gavę klaidingą informaciją, prisiima atsakomybę už likvidumo mažinimą ir turi susimokėti.[44]

Dažniausiai rizikos, kurios susijusios su IT yra siejamos su stebėjimo ir informacijos apdorojimu, nes būtent atliekant šiuos veiksmus yra papildomai sugaištama laiko, kas didelio dažnio prekyboje yra labai svarbu.

Siekdamos sumažinti informacijos perdavimo vėlinimą, prekybos platformos naudoja skirtingas priemones tam pasiekti. Dėl šios priežasties atsiranda kita rizika, ar naudojamas metodas yra efektyvus ir ar jis nepanaikina dalies informacijos.

Technologiniai pranašumai, kuriuos turi ir naudoja didelio dažnio prekybos investuotojai, kelia klausimą, ar nėra pasitelkiamos šios technologijos naudojant manipuliuojančias strategijas. Tačiau nėra aiškaus atsakymo į šį klausimą. Dalis naudojamų strategijų gali būti manipuliuojančiomis, nors nebuvo tam kurtos, jos tiesiog naudoja tam tikrus metodus, kurie suteikia privalumus prieš kitus rinkos žaidėjus.[8]

• Imitavimas: pateikiant prekybos platformom užsakymą, kurio neketinama vykdyti, taip dirbtinai reguliuojant finansinio instrumento kainą.

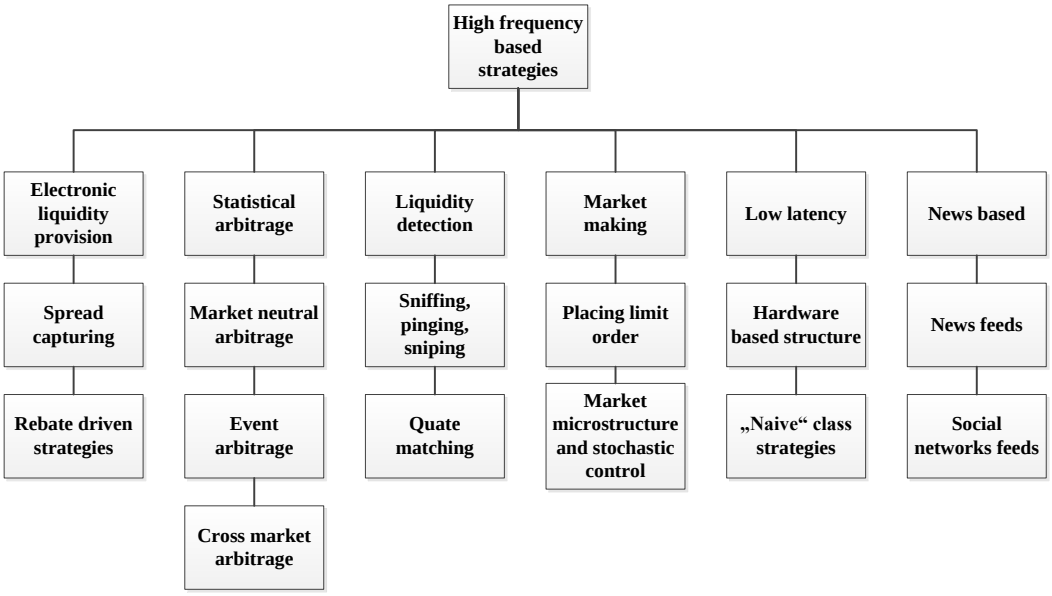
• Sluoksniavimas: investuotojas, kuris kuria tik pirkti arba parduoti užklausas, pateikia didelį kiekį užsakymų su skirtingomis kainomis. Taip dirbtinai sukuriamas spaudimas pirkti arba parduoti pusėje.

• Užsakymų blokavimas: susikuriama daug klaidingų užsakymų su pirkti arba parduoti kainomis, tam kad gauti geresnes kainas priešingu atveju.

• Užsakymo numatymo strategijos: investuotojas ieško didelių pirkėjų, tam kad nupirktų prieš juos, taip jis išnaudoja didelių pirkėjų įtaką.

• Momentinio įėjimo strategijos: sukuriama daug užsakymų, tam kad tuo pačiu būtų kuriami pokyčiai kainose.[44]

Į klausimą ar naudojama didelio dažnio prekybos strategija yra kenksminga, priklauso tik nuo investuotojo tikslų ir kaip jis tas strategijas naudos.



Šaltinis: sudaryta autoriaus

17 pav. Didelio dažnio prekybos strategijos.

Didelio dažnio prekybos strategijos galima suskirstyti į šešias grupes: elektroninio likvidumo suteikimas, statistinis arbitražas, likvidumo aptikimas, rinkos kūrimas, mažas vėlavimas ir naujienomis besiremiančios strategijos. Visų šių strategijų principas nesiskiria nuo lėto dažnio naudojamų strategijų, tačiau didelio dažnio duomenų naudojimas leidžia jas išnaudoti skirtingai ir dažnu atveju gaunami kitokie rezultatai, nei naudojant lėto dažnio duomenis. Visų strategijų tyrimas užimtų daug laiko, todėl šiame darbe koncertuotasi ties statistinio arbitražo (porų prekybos) strategija.

4 Tyrimui naudotos strategijos

Kai yra surastos prekybos poros, duotam prekybos langui, pradedama ieškoti prekybos signal, šių signal parinkimas skiriasi nuo naudojamų strategijų. Darbe kaip pagrbindas naudojamos trys porų prekybos strategijos, kurias keičiant ir papildant viena kitą sukuria daugiau prekybos strategijų. Pirmoji porų prekybos strategija pirmą kartą buvo pristatya J. Caldeira ir G. V. Moura darbe (Caldeira and Moura, 2013). Jų pasliūlytoje strategijoje reikia pirma rasti kriterijų ε_t, kuris yra skirtumas tarp rastos finansinių instrument normalizuotų kainų poros:

$$\varepsilon_t = P(i,t) - p(i,t) \qquad (1)$$

Šioje lygtyje ε_t yra skirtumas tarp vieno finansinio instrument normalizuotos kainos *P(i,t)* ir jo poros normalizuotos kainos *p(i,t)* tuo laiku *t*. Radus šį skirtumą skaičiuojamas kriterijus z_t:

$$z_t = \frac{\varepsilon_t - \mu_t}{\sigma_t} \qquad (2)$$

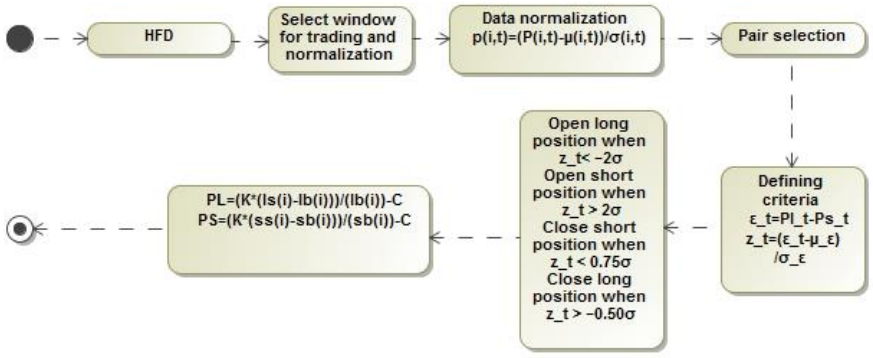
Čia μ_ε yra vidurkis σ_ε yra standartinis nuokrypis poros rasto skirtumo ε_t praėjusio prekybos lango metu. Kai kriterijus z_t yra randamas, strategija gali pradėti ieškoti prekybos signalų (Caldeira and Moura, 2013):

Atidaroma pirkimo pozicija, kai $z_t < -2\sigma$;

Atidaroma pardavimo pozicija, kai $z_t > 2\sigma$;

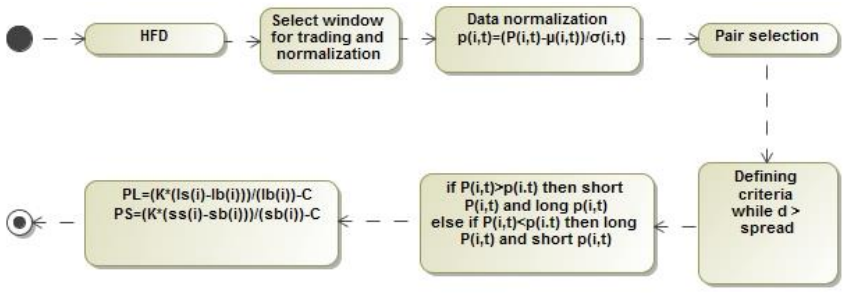
Uždaroma pardavimo pozicija, kai $z_t < 0.75\sigma$;

Uždaroma pardavimo pozicija, kai $z_t > -0.50\sigma$ (Caldeira and Moura, 2013).



18 pav. J. Caldeira and G. V. Moura's pairs trading strategy

Tam kad pozicijos nebūtų laikomos atidarytos labia ilgą laikotarpį yra nustatomas laikotarpis, kiek maksimaliai galima laikyti atidarytą poziciją. Lyginant prekybos strategijas pradiniai parametria yra nustatomi vienodi (prekybos langas, maksimalus periodas laikyti pozicijas, komisinis mokestis). Antroji strategija, kuria remtasi šiame darbe, buvo pristatya M. S. Perlini tyrime (Perlin, 2009). Kaip ir prieš tai buvusioje statgijoje pirma reikia rasti skirtumą tarp normalizuotų poros kainos ε_t ir atsižvelgiant į jį yra ieškoma prekybos signalų lyginat šį skirtumą su pradžioje prekybos nustatytu parametru *d*, kurį nustato pats rinkos dalyvis. Upon calculating the difference, long and short positions are established.



19 pav. M. S. Perlini's pairs trading strategy

Radus prekybos pozicijas, jos atidarytos laikomos tol kol nesumažėja skirtumas tarp normalizuotų poros kainų arba pasiekiamas laikotarpis laikyti atidarytas pozicijas. Detalesnis prekybos signal radimas pateikiamas žemiau:

Kol $d > \varepsilon_t$;

Jei *A(i,t)>B(i.t)*, tada parduodam *A(i,t)* ir perkam *B(i,t)*;

Kitu atveju, perkam *A(i,t)* ir parduodam *B(i,t)* (Perlin, 2009).

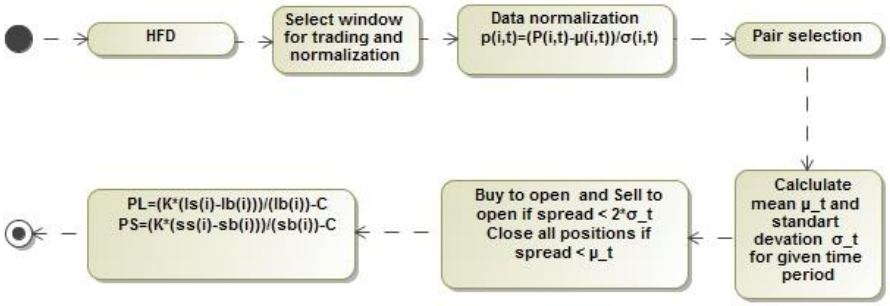
Pagrindinė M. S. Perlini prekybų strategijos logika stebėti skirtumą tarp normalizuotų poros kainų ir laukti kol kriterijus *d* yra pasiekiamas ir tada pradėti prekybą. Tikimasi kad kainos grįš į istrinį vidurkš ir skirtumas tarp kainų grįš į buvusę reikšmę, šis skirtumo judėjimas išnaudojamas prekybos strategijos. (Perlin, 2009).

Paskutinė strategija remiasi D. Herlemont tyrimu, kur jis naudojo porų prekybos strategiją su dienų uždarymo kainomis. (Herlemont, 2013).

Naudojant šią strategija pirmiausia randamas vidurkis μ_t ir standartinis nuokrypis σ_t rasto skirtumo tarp normalizuotų poros kainų duotu prekybos laikotarpiu. Kaip abu šie kriterijai yra rasti, strategija gali pradėti ieškoti prekybos signalų. Kai skirtumas tarp rastos finansinių instrumentų normalizuotų kainų poros *A* ir *B* yra $<2*\sigma_t$:

Kai $A_t > B_t$, tada parduodamas A_t ir perkamas B_t;

Kai $A_t < B_t$, tada perkamas A_t ir parduodamas *Bt* (Herlemont, 2013).



20 pav. D. Herlemon's pairs trading strategy

Pozicijos laikomos atidarytios tol, kol skirtumas tarp normalizuotų poros kainų A_t ir B_t yra $< \mu_t$, arba periodas laikyti atidarytas pozicijas yra pasiektas (Herlemont, 2013).

Pasibaigus prekybos laikotarpiui svarbu nustatyti kiekvienos strategijos efektyvumą, kuris nustatomas pagal pelno/nuostolio dydį laikotarpio pabaigoje. Tam kad rasti pelną/nuostolį strategijų randmas kiekvieno prekybos signal kainų skirtumas t.y. uždirbta ar ne uždarius kiekvieną poziciją, lygtis esanti apačioje detalizuoja šį skaičiaimą (Perlin, 2009; Vaitonis and Masteika, 2016):

$$PL = \sum_{i=1}^{K} (ls(i) - lb(i) \quad (3)$$

$$PS = \sum_{i=1}^{K} (ss(i) - sb(i) \quad (4)$$

Kintamasis *PL* rodo pelną iš pirkimo pozicijų ir kintamasis *PS* – iš pardavimo pozicijų, kintamasi *i* atstoja atskirus prekybos signalus ir kada pozicijos atidaromos ir uždaromos. Pirkimo pozicijų pelnas randmas skaičiuojant skirtumą tarp finansinio instrument kainos kai jis buvo nupirktas – *ls* ir parduotas – *lb,* šis skirtumas padauginamas iš komisinio mokesčio *K,* jeis jis yra taikomas. Pardavimo pozicijų pelnas randmas skaičiuojant skirtumą tarp finansinio instrument kainos kada jis buvo parduotas – *ss* ir nupirktas – *sb,* padauginus iš komisinio mokesčio *K*. Pabaigoje suskaičiuojamas bendras pelnas/nuostolis (Perlin, 2009; Vaitonis and Masteika, 2016):

$$TP = PL + PS \quad (5)$$

Baigus visus skaičiavimus pateikiamas kiekvienos strategijos rezultatas.

4.1 Porų atrinkimas naudojant kointegraciją

Viena iš pagrindini poros prekybos dalių yra koreliuotų finansinių instrumentų radimas, kurs dažniausia remiasi coinegracijos testais. Prieš tai buvo pateiktas mažiausių kvadratų metodas, kuris detaliau yra mažiausio atstumo metodas, o čia bus aptariamas kointegracijos metodas porų ieškojimui. (Driaunys et al., 2014; Masteika and Vaitonis, 2015; Vaitonis and Masteika, 2016).

Kointegracijos metodas susideda iš sekančių žingsnių:

1. Rasti tuos finansinius instrumentus kurie potencialiai gali būti koinegruoti;

2. Kai randama potenciali pora, tikrinama hipotezė jog ši pora yra tikrai kointegruota tarpusavyje remiantis istoriniais duomenimis;

3. Tikrinamos rastos poros, ar su jomis galima vykdyti prekybą (Vidyamurthy, 2004).

Šios dallies tikslas yra rasti poras, kurios turi linijinę kombinaciją, kuri turi nuspėjamą elementą. Šis elemtentas yra nekoreliuotas lyginat su rinkos judėjimu. Turint omenyje šį tikslą, pirmiausia randami skirtumai visų galimų finansinių isntrumentų normalizuotų kainų porų ir tikrinama ar šie skirtumai yra stacionarūs. Šaiame darbe tai atliekama tikrinant ar turimos duomenų eilutės yra inegruotos ta pačia tvarka ir tam naudojamas Padidintas Dickey Fuller testas (ADF), kuris yra praplėstas Dickey Fuller testas (Caldeira and Moura, 2013). Šiame žingsnyje tikrinama nulinė hipotezė ar tikrinamas procesas nėra stacionarus. Jei tikrinama finansinių instrument pora yra koinegruota, tai jų normalizuotų kainų skirtumas turėtų būti syacionarus (Bogoev and Karam, 2016; Mushtaq, 2011).

Jei ADF testas yra išlaikoms, tuomet atliekamas kointegracijos testas su visomis galimomis porų kombinacijomis. Kointegracijos tikrinimui naudojamas dviejų žingsnių Engle ir Granger metodas kartu su Johansen testu (Caldeira and Moura, 2013).

Engle ir Granger teorema teigia, kad jeigu yra dvi ar daugiau duomenų eilučių esančių y_t yra kointegruotos, tada yra klaidos taisymo būdas, kuris gali būti atvaisduotas tokia lygtimi:

$$\Delta y_t = \mu + \gamma z_{t-1} + \varepsilon_t \qquad (6)$$

Šioje lygtyje γ yra koeficientų matrica $n \times r$, μ yra konstantų vektoriaus ($n \times 1$), z_{t-1} yra vektorius $r \times 1$, kuris atitninką sąlygą $r \leq n - 1$, $z_t = \alpha'$, vektorius y_t yra kointegruotas jei yra tokio dydžio $n \times r$ matrica α, kad $z_t = \alpha'$ ir y_t su ε_t yra stacionarus (LeSage, 1999).

Johansen testo pagalba nustatomas koinegruotų ryšių skaičius ir pasitelkdamas daug kintamųjų pralpečia dviejų žingsniu Engle ir Granger metodą (Caldeira and Moura, 2013). Visi šie skaičiavimai yra perkelti į algoritmą, kuris vykdo visus skaičiavimus MATLAB aplinkoje.

4.2 Duomenų normalizavimas

Norint turimus duomenis taikyti mūsų paruoštuose algoritmuose, reikia juos normalizuota tam kad būtų galima tarpusavyje lyginti. Pirmas žingsnis yra suvienodinti visų duomenų laiko eilutes. Pavyzdžiui, jei turime duomenis su šia laiko

eilute vieno finansinio instrumento 17:00:00.869053009 ir kitą finansinį instrumentą su šia laiko eilute 17:00:00.825207610, šios laiko eilutės turi atsispindėti abiejuose finansiniuose instrumentuose. Galimi duomenų variantai su laiko eilutėmės pateikti lentelėje 1 ir lentelėje 2.

2 lentelė

Nanosekundinių duomenų pavyzdys

Receiving Date	Receiving Time	Symbol	Asset	Entry Type	Entry Price
20150809	17:00:00.869053009	NGF6	NG	A	3227
20150809	17:00:00.869053009	NGF6	NG	B	3221
20150809	17:00:00.930168164	NGF6	NG	A	3226
20150809	17:00:00.930168164	NGF6	NG	B	3221
20150809	17:00:01.017456320	NGF6	NG	A	3226
20150809	17:00:01.017456320	NGF6	NG	B	3219
20150809	17:00:01.059840559	NGF6	NG	A	3227
20150809	17:00:01.059840559	NGF6	NG	B	3219
20150809	17:00:01.156791713	NGF6	NG	A	3238
20150809	17:00:01.156791713	NGF6	NG	B	3216
20150809	17:00:01.204683812	NGF6	NG	A	3238
20150809	17:00:01.204683812	NGF6	NG	B	3216
20150809	17:00:01.205605232	NGF6	NG	A	3238
20150809	17:00:01.205605232	NGF6	NG	B	3215
20150809	17:00:01.206755867	NGF6	NG	A	3238
20150809	17:00:01.206755867	NGF6	NG	B	3215
20150809	17:00:01.207350519	NGF6	NG	A	3231
20150809	17:00:01.207350519	NGF6	NG	B	3215
20150809	17:00:01.208805474	NGF6	NG	A	3231
20150809	17:00:01.208805474	NGF6	NG	B	3217
20150809	17:00:01.224604710	NGF6	NG	A	3233
20150809	17:00:01.224604710	NGF6	NG	B	3217

3 lentelė

Nanosekundinių duomenų pavyzdys

Receiving Date	Receiving Time	Symbol	Asset	Entry Type	Entry Price
20150809	17:00:00.825207610	HOF6	HO	A	16040
20150809	17:00:00.825207610	HOF6	HO	B	15950
20150809	17:00:00.826021615	HOF6	HO	A	16035
20150809	17:00:00.826021615	HOF6	HO	B	15950
20150809	17:00:00.838609766	HOF6	HO	A	16040
20150809	17:00:00.838609766	HOF6	HO	B	15950
20150809	17:00:00.865890817	HOF6	HO	A	16040
20150809	17:00:00.865890817	HOF6	HO	B	15945

20150809	17:00:00.866430043	HOF6	HO	A	16040
20150809	17:00:00.866430043	HOF6	HO	B	15944
20150809	17:00:00.867756129	HOF6	HO	A	16040
20150809	17:00:00.867756129	HOF6	HO	B	15943
20150809	17:00:00.869125205	HOF6	HO	A	16040
20150809	17:00:00.869125205	HOF6	HO	B	15938
20150809	17:00:00.875541527	HOF6	HO	A	16040
20150809	17:00:00.875541527	HOF6	HO	B	15934
20150809	17:00:00.884336757	HOF6	HO	A	16040
20150809	17:00:00.884336757	HOF6	HO	B	15928
20150809	17:00:01.025686712	HOF6	HO	A	16040
20150809	17:00:01.025686712	HOF6	HO	B	15950
20150809	17:00:01.029573686	HOF6	HO	A	16019
20150809	17:00:01.029573686	HOF6	HO	B	15950

Jie turimame finansinio instrument duomenyse atsiranda nauja laiko eilutė, ji užpildoma su prieš tai buvusiomis pirkimo pardavimo kainomis. Tokiu būdu užpildomos visos naujai atsiradusios laiko eilutės visuose finasinių instrument duomenyse (Masteika and Vaitonis, 2015; Vaitonis and Masteika, 2016).

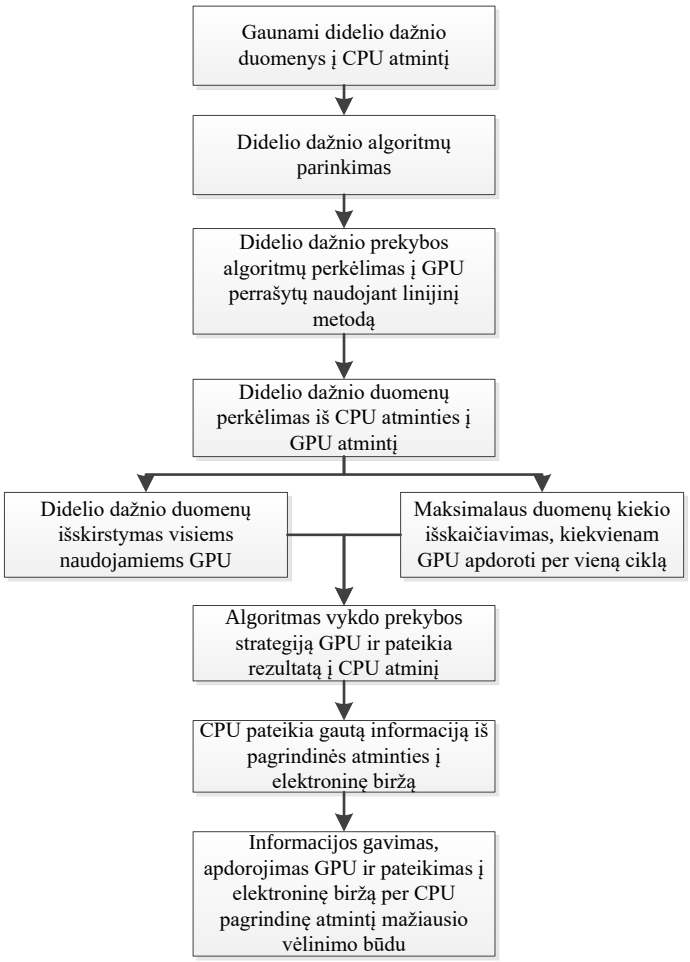
Supildžius šias naujai atsiradusias eilutes, sekantis žvilgnis yra suvienodinti visas finansinių instrument kainas. Atskiri finansiniai instrumentai turi skirtingas kainas ir norint juos lyginti, ieškoti tarpusavio koreliacijos, prekybos signalų, reikia kainas suvienodinti, kad būtų galima atlikti šiuos veiksmus. Kiekvienam turimam finansiniam instrumentui *P(i,t)* yra suskaičiuojamas vidurkis $\mu(i,t)$ ir standartinis nuokrypis $\sigma(i,t)$ pasirinktam prekybos langui, pritaikoma apačioje esanti lygtis (Perlin, 2009):

$$p(i,t) = \frac{P(i,t) - \mu(i,t)}{\sigma(i,t)} \qquad (7)$$

Gauta reikšmė *p(i,t)* yra normalizuota finansinio instrument kaina *i* laikotarpiu *t* (Perlin, 2009). Šis normalizavimo metodas dar kitaip vadinamas z-score. Yra ir kitų duomenų normalizavimo metodų: mini – max, dežimtainis dalinimas, slenkantis langas ir kt. Tačiau šis tyrimas remiasi duomenų svarba, todėl normalziavimo metodas, kuris yra dažniausiai naudojamas statistiniame arbitraže buvo pasirinktas (Bogoev and Karam, 2016).

4.3 Tyrime naudojamas metodas

Šiame darbe norint pasiekti užsibrėžtus tikslus naudojamas metodas, kuris detaliau pateikiamas žemiau esančiame paveikslėlyje. Naudojamą metodą galima išskaidyti į aštuonis pagrindinius žingsnius.



21 pav. Tyrime naudojamas modelis

Pirmo žingsnio metu mikrosekundiniai ar nanosekndiniai duomenys yra gaunami iš elektroninės biržos ir keliauja į CPU pagrindinę atmintį. Gavus didelio dažnio duomenis yra parenkamos strategijos, kurioms toliau bus vykdoma didelio dažnio prekyba. Parinktus prekybos algoritmus reikia perkelti į GPU. Kaip jau minėta GPU efektyviausiai dirba, kai nėra naudojami cikla, todėl visos perkeltos stratetgijos yra perašytos linijiniu būdų, minimalizuojant arba visai atsisakant skaičiavimo ciklų. Ketvirtas žingsnis yra duomenų iš CPU pagrindinės atminties prekėlimas į GPU atmintį. Perkeliant duomenis į reikia įvertinti visus sistemoje esančius GPU ir apskaičiuoti maksimalų duomenų kiekį, kokį galimą pergelti, kad užtektų turimos GPU atminties ir tokiu būdų išskirtstyti duomenis tarp visų GPU. Atlikus šiuos

veiksmus, CPU siunčia instrukciją į GPU, kad inicijuotų parinktus didelio dažnio prekybos algoritmus. Jiems bagus vykdyti skaičiavimus rezultatai keliauja iš GPU atgal į CPU atmintį. Iškart gavęs informaciją CPU pateikia prekybos informaciją elektroniniai biržai. Informacijos gavimas, apdorojimas GPU, jos grąžinimas į CPU ir informacijos pateikimas elektroninei biržai atliekamas mažiausio vėlinimo būdu. Pabaigoje skaičiuojami rezultatai ir apskaičiuojamas laikas, kiek užtrunka nuo informacijos gavimo iki prekybos signal pateikimo elektroniniai biržai.

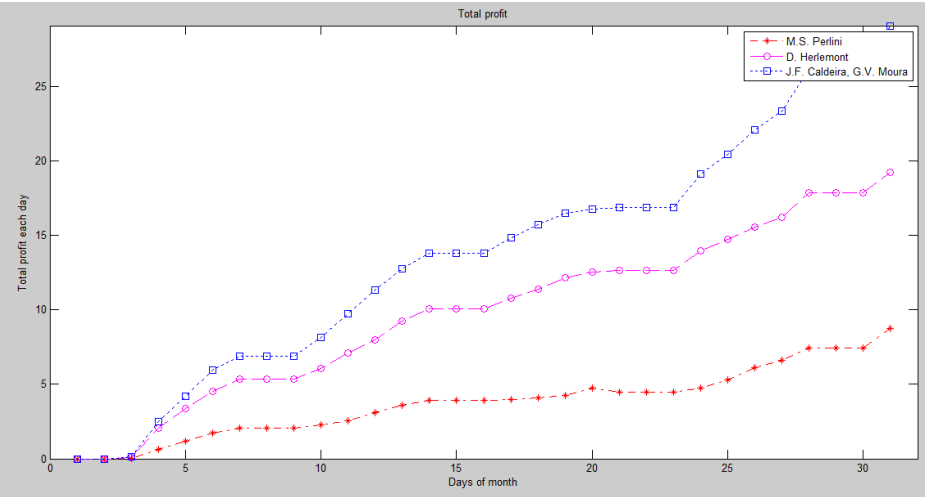
4.4 Porų prekyba naudojant nanosekundinius duomenis

Visos trys didelio dažnio por prekybos strategijos remiasi neutralios rinkos strategijos pozicija. Jos atidaro pirkimo ir pardavimo pozicijas tuo pačiu metu, kai tik atsiranda prekybos signalas. Visi pirkimai – pardavimai buvo atlktinaudojant rinkos kainas, tai reiškia kad pirkimai buvo atlikti su pasiūlos ir pardavimai su pirkimo kainomis.

Vykdant porų prekybą vienas finansinis instrumentas negali priklausyti daugiau nei vienai porai ir jis gali būti tik perkamas arba parduodamas, kol nėra uždaroma pozicija iki sekančio prekybos signal atsiradimo. Šiame tyrime naudoti ateities sandorių nanosekundiniai duomenys ir kadangi stebimas tik pačių strategijų efektyvumas, buvo netaikomi komisiniai mokesčiai.

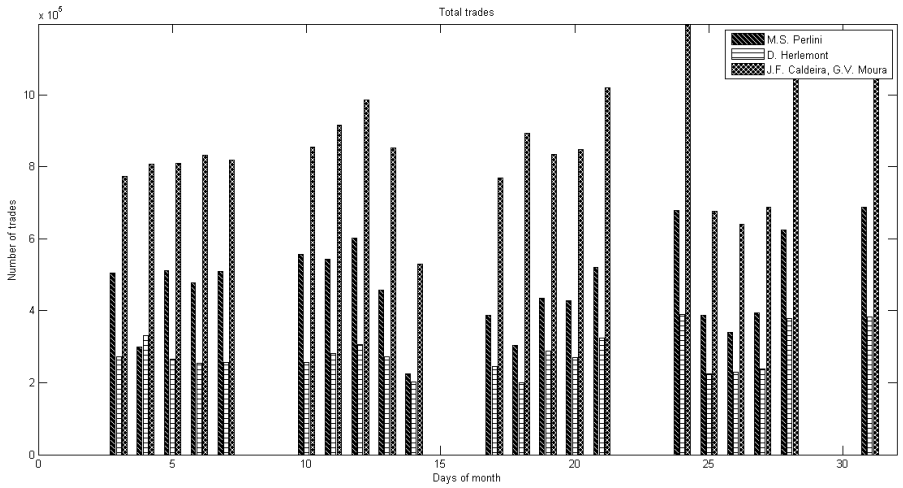
Tyrimo pabaigoje buvo pastebėta, kad visos trys didelio dažnio prekybos strategijos buvo pelningos neatsižvelgiant į komisinius mokesčius.

Grafikas esantis apačioje iliustruoja visa nnosekindinį prekybos laikotarpį nuo 2015-08-1 iki 2015-08-31 ir kokius pelnos generavo kiekviena didelio dažnio porų prekybos strategija.



22 pav. Gautas pelnas naudojant nanosekundinius duomenis prekybos laikotarpio pabaigoje naudojant tris strategijas

Įdomu matyti ir kiek sandorių atliko kiekviena didelio dažnio porų prekybs strategija kievieną prekybos dieną atskirai, naudojant nanosekundinius duomenis.



23 pav. Atliktų sandorių kiekis kiekvieną prekybos diena naudojant visas tris strategijas

Sandorių kiekies kievienai dienai priklauso nuo prekybos strategijos taisyklių skirtų rasti prekybos signalus: kuo porų prekybos strategijos taisyklės jautresnės t. y. pastebi ir smulkius rinkos pasikeitimus, tuo daugiau prekybos signal bus rasta ir atlikta daugiau sandorių. Tačiau nevisada didelis sandorių kiekies reiškia, kad prekybos strategija bus efektyvi ir pelninga.

4 lentelė

Tyrimo rezultatai naudojant mikrosekundinius ir nanosekundinius duomenis

	M. S. Perlini	D. Herlemont	J. Caldeira ir G. V. Moura
Profit with nanosecond data	8.74%	19.27%	29.11%
The number of total trades	9878628	5869860	18051372
Profit with microsecond data	4.01%	3.39%	4.75%
Number of the total trades	1491576	2135360	2538979

Aukščiau esančioje lentelėja galima matyti procentinius pelningumus ir kiek sandorių atliko kiekviena didelio dažnio prelkybos strategija. M. S. Perlini laikotarpio pabaigoje turėjo 8,74% pelną ir atliko 9878628 sandorius, D. Herlemont strategija atliko mažiausiai sandorių 5869860 ir turėjo 19,27% pelną ir trečioji J. Caldeira ir G.V. Moura porų prekybos strategija atliko didžiausią kiekį sandorių – 18051372 ir turėjo didžiausią pelną – 29,11%. Žemiau esančioje lentelėje galima matyti vidutį kiekvieno sandorio pelną naudojan nanosekundinius ir mikrosekundinius duomenis.

5 lentelė

Vidutinis pelningumas naudojant skirtingo dažnio duomenis

	M. S. Perlini	D. Herlemont	J. Caldeira ir G. V. Moura
The average return with microsecond data	0.00000088%	0.00000328%	0.00000161%
The average return with nanosecond data	0.00000269%	0.00000159%	0.00000187%

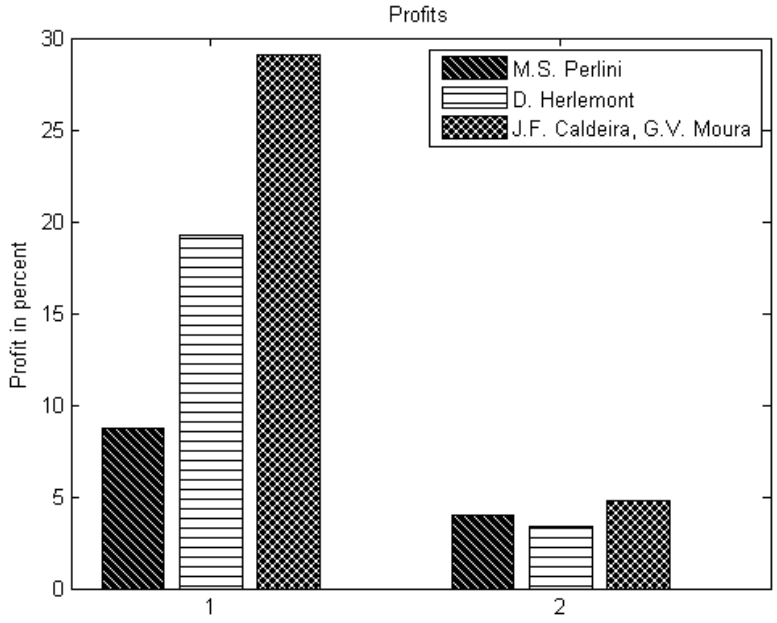
Tačiau vidutinis pelnas už sandorį nevisada parodo efektyviausią prekybos strategiją. Tam kad rasti efektyviausią strategiją buvo skaičiuojamas Šarpo rodiklis visoms didelio dažnio porų prekybos strategijoms su nanosekundėmis ir mikrosekundėmis.

6 lentelė

Procentinis pelnas kiekvieną prekybos dieną

Date	M. S. Perlini in ns.	M. S. Perlini in ms.	D. Herlemont in ns.	D. Herlemont in ms.	J. Caldeira and G. V. Moura in ns.	J. Caldeira and G. V. Moura in ms.
2015-08-3	0.2991	0.2861	0.9131	0.1344	1.246	0.0482
2015-08-4	0.329	0.0951	1.1628	0.2393	1.257	0.0274
2015-08-5	0.5558	0.2684	1.3029	0.3997	1.6732	0.0153
2015-08-6	0.5296	0.2605	1.1581	0.3367	1.81	0.2466
2015-08-7	0.3334	0.1725	0.8016	0.254	0.911	0.1888
2015-08-10	0.2338	0.2397	0.7086	0.2693	1.251	0.1669
2015-08-11	0.2551	0.291	1.0476	0.3316	1.616	0.2252
2015-08-12	0.5766	0.3343	0.8812	0.107	1.596	0.3285
2015-08-13	0.4855	0.2346	1.272	0.1013	1.403	0.3322
2015-08-14	0.3569	0.1116	0.8347	0.0929	1.06	0.2173
2015-08-17	0.0275	0.0753	0.7157	0.0425	1.026	0.1978
2015-08-18	0.1219	0.0597	0.6141	0.037	0.866	0.1121
2015-08-19	0.1439	0.1856	0.7636	0.0425	0.812	0.2317
2015-08-20	0.4791	0.1352	0.3591	0.0443	0.239	0.1909
2015-08-21	-0.25	0.0917	0.118	0.0754	0.11	0.1201
2015-08-24	0.2562	0.2031	1.3034	0.1552	2.273	0.2941
2015-08-25	0.5519	0.086	0.7722	0.0504	1.335	0.1795
2015-08-26	0.8523	0.1502	0.8247	0.0539	1.613	0.2987
2015-08-27	0.5036	0.1992	0.6949	0.1271	1.285	0.3636
2015-08-28	0.7844	0.2974	1.6071	0.319	2.786	0.5674
2015-08-31	1.3203	02296	1.4236	0.1759	2.944	0.3922

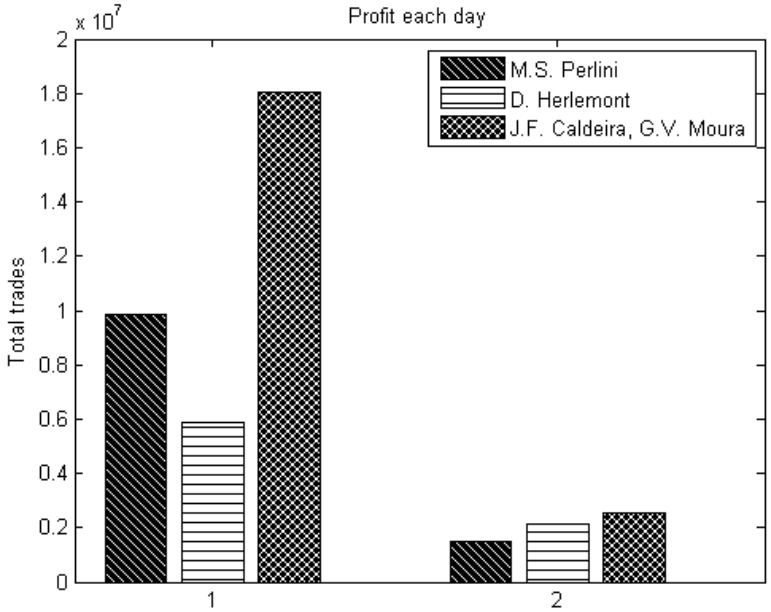
Prieš skaičiuojant Šarpo rodiklį reikia rasti kaip kiekvieną didelio dažnio prekybos strategija atrodė kiekvieną dieną atskirai su abiejų tipo duomenimis. Lentelė esanti viršuje atvaizduoja visų prekybos strategijų pelningumą procentais kiekvienai diena su mikrosekundiniai ir nanosekundiniais duomenimis. Galima pastebėi, jog nė viena diena nebuvo nuostolinga. Grafikas apačioje atskirai parodo strategiojų galutinio prekybos laikotarpio pelningumą su mikrosekundiniais (2) ir nanosekundiniais (1) duomenimis.



24 pav. Procentinis pelnas naudojant visas tris strategijas su mikrosekundiniais ir nanosekundiniais duomenimis

Grafikas atskldeidžia akivaizdų skirtumą tarp duomenų naudojimo skirtumą, didesnio dažnio duomenys duoda didesnį pelną. Tačiau prieš priimant galutines išvadas reiktų apskaičiuoti Šarpo rodiklį, kad būtų galima tiklsiau nustatyi kurio tipo duomenys efektyvesni ir kuri strategija buvo geriausia.

Taip pat įdomu palyginti sandorių skirtumą naudojant skirting dažnio duomenis, jų palyginimą galima matyti grafike esančiame žemiau. Nieko nuostabaus kad nanosekundiniai duomenys turėjo daugiau sandorių už mikrosekundinius duomenis.



25 pav. Bendras sandorių kiekis naudojant ris strategijas išskiriant mikrosekundinius ir nanosekundinius duomenis

Surinkus vis1 reikaling1 informaciją apie visas tris didelio dažnio porų prekybos strategijas su mikrosekundiniais ir nanosekundiniais duomenimis galima skaičiuoti Šarpo rodiklį. Jis skaičiuojamas naudojant šią formulę $S=\frac{\mu_{PnL}}{\sigma_{PnL}}$, kur μ_{PnL} yra pelno ar nuostolio vidurkis ir σ_{PnL} yra pelno ar nuostolio *PnL* standartinis nuokrypis.

7 lentelė

Visų strategijų Šarpo rodiklis naudojant mikrosekundinius ir nanosekundinius duomenis

	Sharpe ratio of M. S. Perlini strategy	Sharpe ratio of D. Herlemont strategy	Sharpe ratio of J. Caldeira and G. V. Moura strategy
Microsecond data	1.3380	1.4240	1.7651
Nanosecond data	1.3144	2.6388	2.0442

Kuo didesni Šarpo rodiklis tuo geresnė porų prekybos strategija. Iš lentelės viršuje matyti kad J. Caldeira ir G. V. Moura's didelio dažnio porų prekybos strategija naudodama mikrosekundinius duomenis byvo geriausia. Tačiau, kai reikėjo naudoti nanosekundinius duomenis šią strategija aplenkė D. Herlemont didelio dažnio porų prekybos strategija. Rezultatai atskleidžia, kad vien didelis sandorių kiekies nereiškia kad prekybos strategija yra efektyvi. Taip pat matyti kad būtent naudojant nanosekundinius duomenis Šarpo rodikliai gauti didesni dviejų strategijų iš trijų ir tai parodo didelio dažnio duomenų naudingumą. Didelio dažnio duomenys leidžia rinkos dalyviui dirbti su naujausia informacija ir galima parinkti nedidelius prekybos langus dėl gaunamo didelio duomenų kiekio. Tokiu būdu lengviau pastebimi net smulkiausi rinkos nukrypimai. Kitaą svarbi didelio dažnio duomenų svarba, kad rinkos dalyvis gali aplenkti kitus, kurie dirba su lėtesnio dažnio duomenimis ir gali priimti sprendimus daug greičiau už kitus.

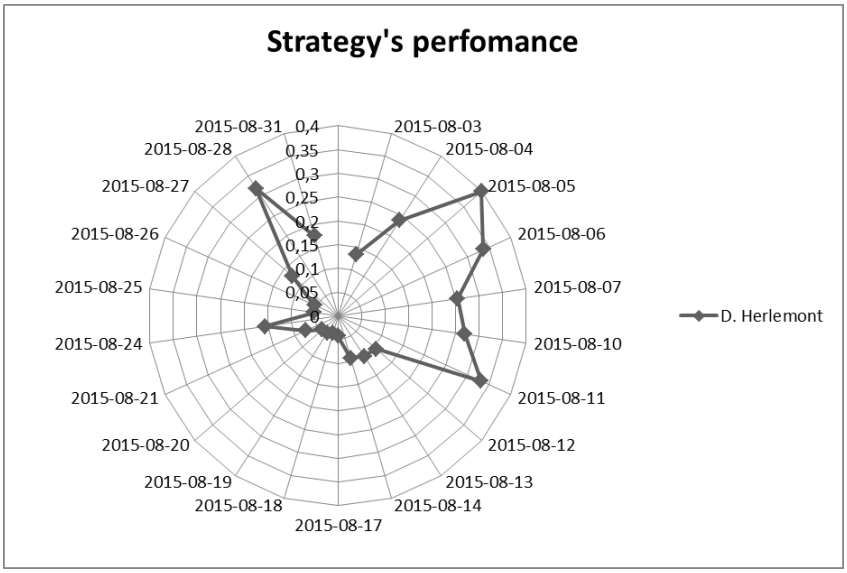
4.5 Porų prekyba su nanosekundiniais duomenimis naudojant CPU ir GPU

Du pagrindiniai algoritminės prekybos kriterija yra greitis kuriuo tie patys skaičiavimai gali būti atliekami su dideliu kiekiu skirting duoemnų ir galimybė tai suprogramuoti. Dėl šios priežasties tradicinė techninė įranga, kaip centrinis procesorius, yra netinkama. Centrinis kompiuterio procesorius yra sukurtas taip, kad altiktų veiksmus vieną paskui kitą, tačiau norint sparintiti skaičiavimus juos reikia paralelizuoti ir atlikti daug vienodų skaičiavimu tuo pačiu metu.

Šio tyrimo metu buvo naudojamas 2 branduolių CPU Intel i5 - 3230M 2,6 GHz ir GPU su 96 CUDA branduoliais GeForce 710M. Pirmiausia prekybos strategijai buvo naudojamas tik centrinis procesorius. Naudoojant “parfor” fikncją MATLAB aplinkoje, leidžia paralelizuoti skaičiavimus ir pačiam CPU padalinant juos tarp branduolių. Šioje dalyje buvo rasti skaičiavimai, kuriuos galima paralelizuoti CPU ir taip buvo optimizuojama porų prekybos strategija.

Norint perkelti skaičiavimus į GPU buvo pasitelktos būtent šiai įrangai MATLAB pritaikytos funkcijos kaip “gpuArray” ir “arrayfun”. GpuArray pagalba sukuriamas duomenų masyvas GPU, o arrayfun paralelizuoja skaičiavimus ir atlieka skaičiavimus su kievienu GPU masyvo element. Šių funkcijų pagalba visi skaičiavimai atliekami GPU, o ne CPU. Visi duomenys esantys CPU amintyje atituria GPU atmintyje.Visos manipuliacijos atliktos MATLAB aplinkoje ir galutinis skaičiavimo rezultatas atsiduria GPU atmintje. Baigus visus skaičiavimus esamus duomenis reikia grąžinti į CPU atmintį [10][11].

Šio eksperimento metu buvo paralelizuoti porų aptikimo, prekybos signalų nustatymo ir pelno skaičiavimo funkcijos. Šias funkcijas buvo galima paralelizuoti, nes jos atlikdavo tuos pačius veiksmus tik vis su kitais duomenimis. Pabaigoje ekspremineto buvo suskaičiuotas strategijos pelnas, kuris pateiktas grafike esančiame žemiau.



26 pav. Gautas strategijos pelnas kiekvieną prekybos dieną

Mūsų eksperimento esmė nebuvo nustatyti strategijos pelingumą, bet radus galimybes paralelizuoti skaičiavimus CPU ir vėliau juos perkėlus į GPU pagreitinti prekybos algorimto veikimą. Žemiau esanti lentelė vaizduoja kiek duomenų buvo

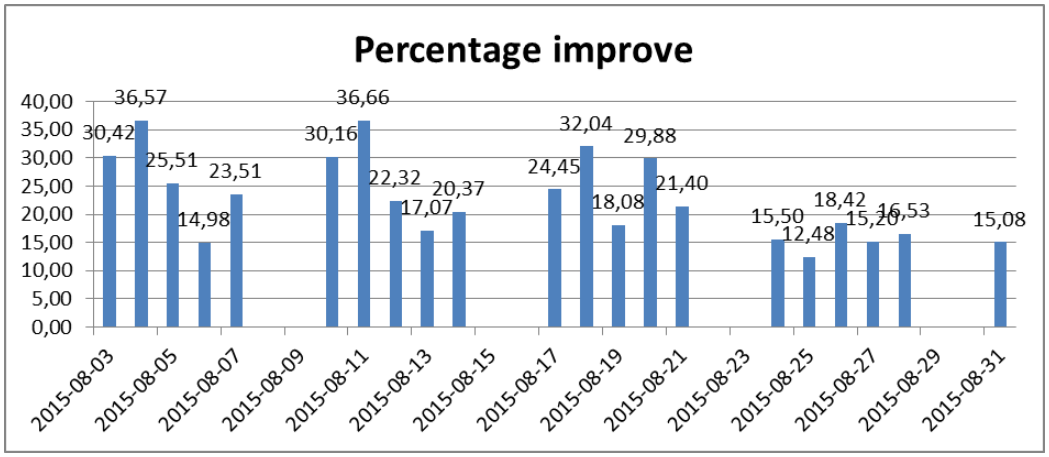
apdorota kiekvieną dieną ir kiek užtreuko didelio dažnio porų prekybos strategija juos apdorodoama, kai naudojo CPU ir GPU.

8 lentelė

CPU ir GPU palyginimas

Date	**Intel i5 - 3230M 2,6 GHz,2 cores (in seconds)**	**GeForce 710m, 96 CUDA Cores (in seconds)**	**Number of records processed**
2015-08-03	2991,80	2081,60	6096505
2015-08-04	2208,10	1400,50	4579465
2015-08-05	2393,70	1783,10	5793525
2015-08-06	3040,90	2585,3	5595770
2015-08-07	2650,10	2027,1	5586360
2015-08-10	4410,80	3080,70	5732355
2015-08-11	4980,30	3154,50	6249980
2015-08-12	2769,20	2151,20	6758875
2015-08-13	4122,60	3419,00	5666900
2015-08-14	1325,90	1055,80	4227335
2015-08-17	1550,00	1171,10	4879990
2015-08-18	1912,10	1299,50	4364540
2015-08-19	4002,30	3278,70	5666700
2015-08-20	4449,00	3119,43	5411145
2015-08-21	4311,70	3389,10	5946205
2015-08-24	4809,40	4064,00	7710745
2015-08-25	3960,20	3466,10	5105175
2015-08-26	3187,60	2600,40	5119660
2015-08-27	5004,90	4244,20	7963320
2015-08-28	5287,10	4413,10	7721975
2015-08-31	5409,70	4594,10	8613445

Lentelėje vaizduojama kiek sekundžių užtruko algoritmas naudodamas skirting tipo techninę įrangą, Procentinis skirtuams, kiek paspartėjo skaičiavimai naudojant GPU pateikti grafike esančiame žemiau.



27 pav. Algoritmo vėlinimo sumažinimas jį pritaikius GPU

Kaip matyti iš pateikto grafiko algoritmą pritaikius dirbit su GPU, kuris turėjo 96 CUDA branduolius, skaičiavimai pagreitėjo nuo 12% iki 36% procentų lyginat su CPU. Skaičaivimų pagreitėjimo skirtumas atsiranda dėl to, kad skirtingomis dienomis randamas skirtingas kiekies porų ir prekybos signal. Kuo didesnį kiekį duomenų skaičiavimai galima paralelizuoti, tuo greičiau vyksta skaičiavimai. Ekperimentas parodo ne tik didelio dažnio duomen svarba, bet ir kaip technologiniai pakeitimai gali įtakoti algoritmo veikimą.

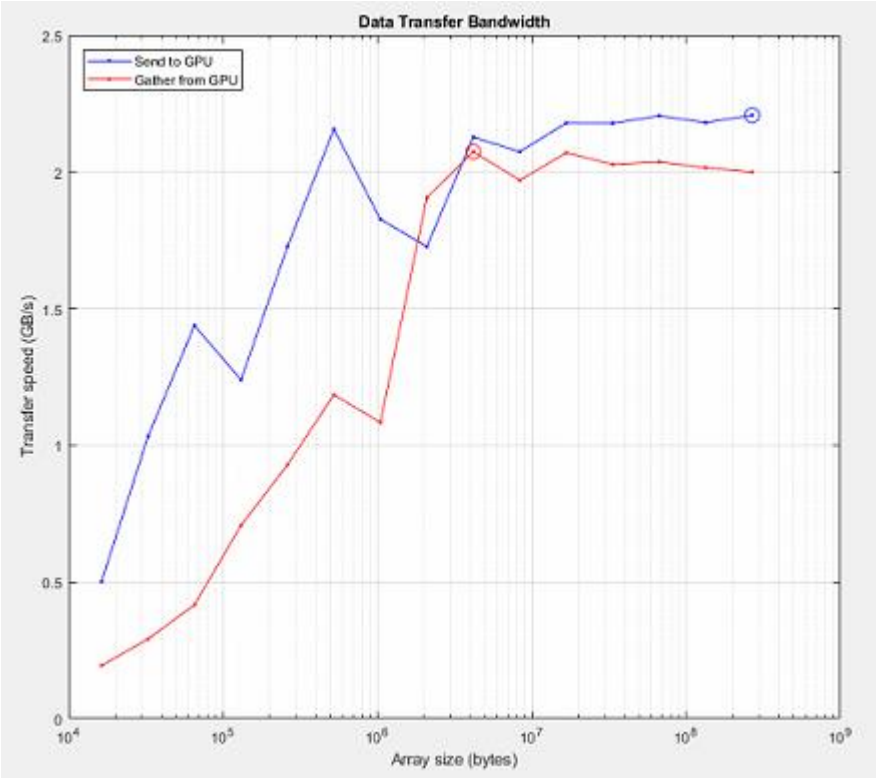
9 lentelė

CPU ir GPU palyginimas naudojan tiesinių lygčių metodą

M. S. Perlini			D. Herlemont			J. Caldeira and G. V. Moura		
Nonlinear CPU	Nonlinear GPU	Linear GPU	Nonlinear CPU	Nonlinear GPU	Linear GPU	Nonlinear CPU	Nonlinear GPU	Linear GPU
3082,01	2212,58	7,56	2991,80	2234,69	7,06	3188,30	2175,46	7,13
2312,21	1866,77	5,92	2208,10	1625,87	5,52	2300,20	1836,70	5,55
2548,00	2275,70	7,41	2393,70	2033,04	6,80	2577,20	2238,40	6,87
3157,23	1980,72	7,02	3040,90	1796,88	6,51	3025,80	1741,62	6,56
2757,20	2020,74	7,01	2650,10	1617,00	6,67	2736,30	2010,04	6,75
4930,11	1723,71	7,12	4410,80	2397,89	6,72	4850,30	2409,41	6,79
5182,32	2513,59	7,58	4980,30	2833,62	7,22	5144,90	2215,78	7,22
2900,12	2174,64	8,21	2769,20	2007,54	7,68	2837,10	2075,58	7,73
4211,10	2082,78	7,08	4122,60	2287,30	6,55	4222,80	2591,05	6,65
1398,00	1208,22	5,52	1325,90	1204,28	5,24	1483,00	1284,10	5,21
1602,56	1491,32	6,21	1550,00	1342,17	5,87	1597,30	1333,63	5,86
2102,70	1455,73	5,66	1912,10	1695,52	5,33	2074,60	1570,07	5,40
4179,40	1633,95	6,92	4002,30	1620,07	6,61	4154,70	1495,48	6,63
4982,20	2130,43	6,65	4449,00	1948,76	6,41	4591,40	2098,70	6,42
4591,80	1883,19	7,26	4311,70	2111,37	7,22	4421,70	1941,67	6,89
5080,60	3185,21	9,07	4809,40	3193,18	8,61	4982,50	3306,48	8,74
4229,10	1810,58	6,44	3960,20	2200,31	5,94	4074,30	2154,72	6,06
3377,20	2266,08	6,38	3187,60	2564,24	6,03	3269,70	2261,87	6,05
5305,40	3110,35	9,42	5004,90	3494,92	8,87	5238,00	3089,66	9,09
5698,10	3186,38	9,07	5287,10	3642,83	8,59	5534,70	3769,10	8,71
5671,80	3835,31	10,08	5409,70	3033,25	9,51	5583,40	3617,43	9,64

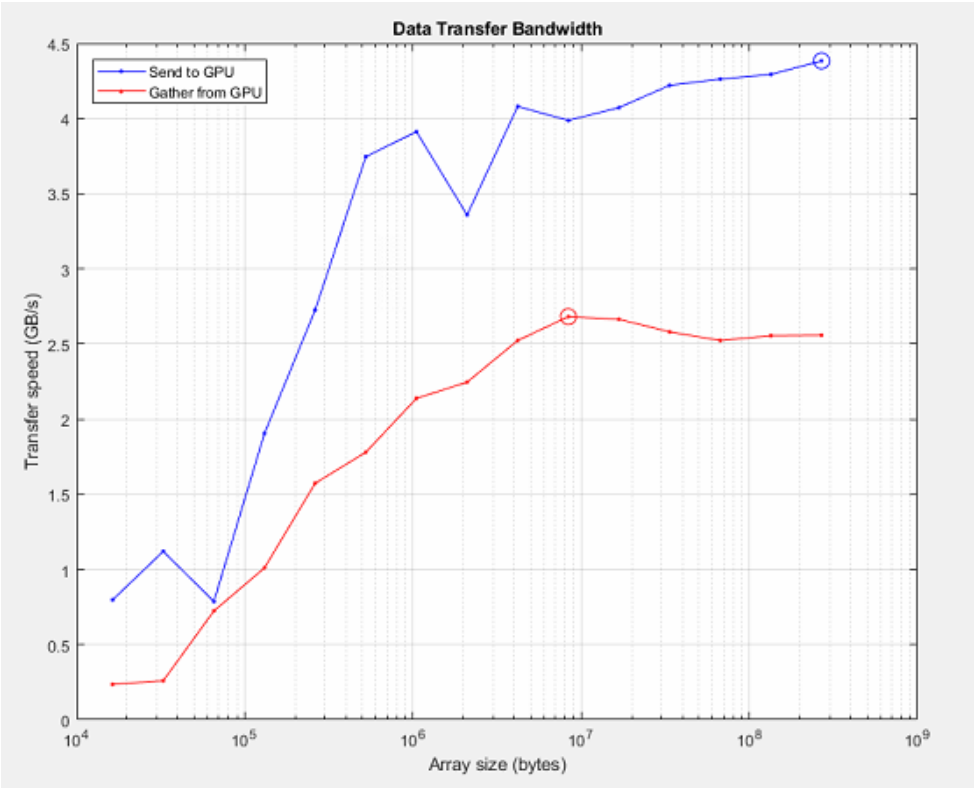
Iš pateiktos lentelės matyti, kad procentinis pagreitiėjimas nelinijinio metodo, kuris naudoja ciklus ir dažna duomenų perkėlinėjamą tarp CPU ir GPU atminčių, perkėlųs iš grynai CPU į GPU nanosekundinių duomenų apdorojimas pagreitėjo nuo 10% iki 60%. Norint įvertinti visą galimą GPU efektyvumą algoritmą reikėjo perašyti, kad jis galėtų pilnai išnaudoti visus GPU privalumos ir tam reikėjo pasitelkti linijinį metodą. Šio metodo dėka, pilnai ciklų išvengti nepavyko, tačiau jų sumažėja iki vieno, kuris skirtas perdavinėti tam tikro dydžio duomenų paketus. Procentinis prageitėjimas iš nelinijinio metod į linijinį GPU metodą siekia apie 99%. Tai parodo, kiek kartų GPU yra efektyvesnis apdorojant didelio kiekio duomenis, kur reikalingas mažas vėlinimas. Šioje tyrimo dalyja naudotas CPU Xeon E5-2650 su 8 branduoliais ir 20 MB spartinančiaja atmintimi, bei GPU Nvidia GTX – 1060 6GB su 1280 CUDA branduoliais.

Tolimesnėje tyrimo dalyje bus apjungiamos dvi vaizdo plokštės: jau minėta GeForce GTX 1060 6GB ir GeForce GTX 1070 Ti turinti 2432 CUDA branuolius. Prieš pradedant apjuginėti šias dvi plokštes yra svarbu nustatyti kaip greitai galima perduotis duomenis iš CPU atminties į GPU atmintį esančią abiejose plokštėse. Todėl, svarbu nuststyi duomenų pralaidumą įrašymui į GPU atmintį ir nuskaitymui. Kadangi abi plokštės yra prijungtus naudojant PCI jungtį, dalis pralaidumo priklauso nuo šios jungties greičio, bei nuo to kas dar yra prijungta prie pagrindinęs ir grafinės plokštės. Duomenų pralaidumui atliekamas testas MATLAB aplinkoje funkcijos *gpuArray* pagalba yra perkeliami duomenys į GPU atmintį iš CPU atminties ir *gather* funcijos pagalba yra grąžinami duomenys į CPU atmintį. Sistomeje naudojama PCI express v.3, kurios teorinis pralaidumas yra 0,99 GB/s vienai linijai. Jei naudojama PCI express x16, tokiu atveju teorinis duomenų pralaidumas 15.75GB/s. Ekeperimento metu GTX 1060 6GB naudoja PCI express x16, o GTX 1070 Ti naudoja PCI express x4. Testo metu paskaičiuojamas abiejų plokščių duomenų įrašymo ir nuskaitymo pralaidumas.



28 pav. GTX 1070 Ti duomenų pralaidumas

GTX 1070 Ti atveju didžiausias duomenų įrašymo pralaiduma buvo pasiektas 2.20726 GB/s, o didžiausias duomenų nuskaitymo pralaidumas buvo 2.0736 GB/s.



29 pav. GTX 1060 6GB duomenų pralaidumas

Naudojant GTX 1060 6GB didžiausias duomenų įrašymas buvo 4.38328 GB/s, o didžiausias duomenų nuskaitymas buvo pasiektas 2.68154 GB/s. GTX 1070 Ti mažesnis duomenų pralaidumas gali būti dėl to, nes prie šios plokštės yra prijungtas kompiuterio vaizduoklis ir ji naudojama kaip sistemos grafinė plokštė. Dėl šių priežasčių, dalis šios grafinės plokštės resursų tenka apdoroti grafinius elementus.

5 Literatūra

1. Gatev E., Goetzman W. N., Rouwenhorst K. G. (1999) “Pairs Trading: Performance of a Relative Value Arbitrage Rule", Yale school of management.
2. Nath, P. (2003) "High Frequency Pairs Trading with US Treasury Securities: Risks and Rewards for Hedge Funds", Londono business school.
3. Perlini M. S. (2007) “Evaluation of Pairs Trading Strategy at Brazilian Financial Market", ICMA University.
4. Jonathan Chiu, Daniel Wijaya Lukman, Kourosk Modarresi, Avinayan Senthi Velayutham. (2011) “High – frequency Trading” Stanford University.
5. Simon Bernardi, Alessandro Gnoatto. (2010) “A Pairs Trading Strategies applied to the European Banking Sector” Zurich University.
6. The Holy Grail Of Trading Has Been Found: HFT Firm Reveals 1 Losing Trading Day In 1238 Days Of Trading. Tyler Durden. zerohedge.com [žiūrėta 2016 m. sausio 10 d.] Prieiga per internetą: <http://www.zerohedge.com/news/2014-03-10/holy-grail-trading-has-been-found-hft-firm-reveals-1-losing-trading-day-1238-days-tr>

7. Allen Carion. (2013) “Very fast money: High – frequency trading on the NASDAQ”, Journal of Financial Markets, vol. 16, no. 4, p. 680 – 711.
8. Joel Hasbrouck, Gideon Sarr. (2013) “Low – latancy trading”, Journal of Financial Markets, vol. 16, no. 4, p. 646 – 679.
9. Nils Fagerburg. (2012) “Pairs Trading System”, Brussels Liberty University.
10. Murat Ahmed, Anwi Chai, Xiaowei Ding, Yunjiang Jiang, Yunting Sun.(2009) “Statistical Arbitrage in High Frequency Trading Based on Limit Order Book Dynamics”, nepublikuotas darbas.
11. Thomas A. Hanson, Joshua R. Hall. (2013) “Statistical Arbitrage Trading Strategies and High Frequency Trading”, Kent State Univeristy.
12. Jay Desai, Arti Trivedi, Nisarg A. Joshi.(2012) “The case of Gold and Silver”: A New Algorithm for Pairs Trading”, Shri Chimanbhai Patel business and reserch institute.
13. Miika Sipila. (2013) “Algorithmic Pairs Trading: Empirical Investigation of Exchange Trade Funds”, Master Thesis, Faculty of finance Aalto University.
14. Kun Zhu. (2005) “A Statistical Arbitrage Strategy”, magistro darbas, Department of Numerical Analysis Royal Institute of Technology.
15. Joao Frois Caldeira, Guilherme Valle Moura. (2013)“Selection of a Portfolio of Pairs Based on Cointegration: A Statistical Arbitrage Strategy”, Rev. Bras. Financas, vol. 11, no. 1, p. 49 – 80.
16. D. Herlemont. Pairs Trading, Convergence Trading, Cointegration. Quantitative Finance. 2013, Volume 12, No. 9
17. Robert J. Elliot, John Van Der Hoek, William P. Malcolm.(2005) “Pairs Trading”, Quantitative Finance, vol 5, no 3, p. 271 – 276.
18. Jozef Rudy. (2011) “Four Essays in Statistical Arbitrage in Equity Markets”, daktaro laipsnio tezė, John Moore University.
19. Christoffer Haakon Hoel. (2010) “Statistical Arbitrage Pairs: Can Cointegration Capture Market Neutral Profits?”, magistro darbas, Norvegijos ekonomikos mokykla.
20. Joel Hasbrouck, Gideon Saar. (2013) “Low – latency trading”, Journal of Financial Markets, vol. 16, no. 4, p. 646 – 679.

21. Steve Hogan, Robert Jarrow, Melvyn Teo, Mitch Warachka. (2004) “Testing market efficiency using statistical arbitrage with applications to momentum and value strategies”, Journal of Financial Economics, vol. 73, p. 525 – 565.
22. GATEV E., GOETZMAN W. N., ROUWENHORST K. G.(2006) Pairs Trading: Performance of a Relative Value Arbitrage Rule. The Review of Financial Studies, Volume 19, Issue 3, p. 797 - 827. ISSN 1465-7368.
23. HASBROUCK Joel, SARR Gideon.(2013) Low – latancy trading. Journal of Financial Markets, Volume 16, Issue 4, p. 646 – 679. ISSN: 1386-4181.
24. ALDRIDGE I. (2013), High-Frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems. John Wiley & Sons, 368 p. ISBN: 978-0-470-57977-0.
25. CIFU, D.A. (2014). Virtu Financial inkorporacija, forma S-1, Registracijos pareiškimas pagal vertybinių popierių aktą 1933. Sec.gov, [žiūrėta 2016 m. vasario 5 d.]. Prieiga per internetą: https://www.sec.gov/Archives/edgar/data/1592386/000104746914002070/a2218589zs-1.htm
26. ZUBULAKE P., LEE S. (2011). The High frequency game changer: how automated trading strategies have revolutionized the markets. Aite group. Wiley trading, The high – frequency,
162 p. ISBN: 978-0-470-77038-2.
27. CFTC, SEC (2011). Rekomendacijos kaip atsakas į reglamentavimus prasidėjusius 2010 m. gegužės 6d: Jungtinės CFTC-SEC patariamojo komiteto ataskaitos santrauka susidariusiems reguliavimo klausimamas. Sec.gov, [žiūrėta 2016 vasario 22 d.]. Prieiga per internetą: https://www.sec.gov/spotlight/sec-cftcjointcommittee/021811-report.pdf
28. BROGAARD J., HENDERSHOTT T., RIORDAN R. (2013) High frequency trading and price discovery. ECB Lamfalussy fellowship programme, Working paper series, , European central bank Press, Nr. 1602. ISSN 1725-2806.
29. CARRION, A. (2013) Very fast money: High – frequency trading on the NASDAQ. Journal of Financial Markets, vol. 16, no. 4, p. 680 – 711.
30. MIAO, J.G. (2014) High Frequency and Dynamic Pairs Trading Based on Statistical Arbitrage Using a Two-Stage Correlation and Cointegration Approach.

International Journal of Economics and Finance, vol.6, no. 3, p. 96-110. ISSN 1916-9728.

31. CARTEA, A., PENALVA, J. (2012) Where is the value in high frequency trading? Quarterly Journal of Finance, vol. 2, no. 3, p. 1–46. ISSN: 1059-0560.

32. MENKVELD, A.J. (2013) High frequency trading and the new market makters. Journal of Financial Markets, vol. 16, no.4, p. 712-740. ISSN: 1386-4181.

33. ACWORTH W. (2011) Record Volume 2010 (Annual Volume Survey). Futures Industry, p. 12-29.

34. FERGUSON, R., LASTER, D. (2007) "Hedge Funds and systemic risk" in Hedge funds, credit risk transfer and financial stability by Cole, R.T., Feldberg, G., Lynch D. Financial Stability Review- Special issue on hedge funds, Banque de France, no. 10, p. 45-54.

35. MASTEIKA S., RUTKAUSKAS A.V. (2012) Research on futures trend trading strategy based on short term chart pattern, Journal of Business Economics and Management, Vol. 13, No.5, p. 915-930. ISSN 1611-1699.

36. DUNIS C., GIORGIONI G., LAW, J., RUDY J. (2010) Statistical arbitrage and high-frequency data with an application to Eurostoxx 50 equities. Alternative investment analyst review: Trading strategies, p.35-57.

37. VIDYAMURTHY G. (2004) Pairs Trading – Quantitative Methods and Analysis. John Whiley & Sons, 224 p. ISBN-10: 0471460672.

38. TA Guide to the HFT Arms Race Holy Grail. Brian Durwood ir Nick Granny. Wallstreetandtech.com [žiūrėta 2016 m. kovo 5 d.] Prieiga per internetą: http://www.wallstreetandtech.com/trading-technology/a-guide-to-the-hft-arms-race/d/d-id/12686

39. What Is The Technology Stack Like Behind A High-Frequency Trading Platform. Vlad Tanev. Forbs.com [žiūrėta 2016 m. kovo 11 d.] Prieiga per internetą: < http://www.forbes.com/sites/quora/2014/01/07/what-is-the-technology-stack-like-behind-a-high-frequency-trading-platform >

40. Preparing for hardware-accelerated High Frequency Trading. David Troman. Paconsulting.com [žiūrėta 2016 m. kovo 11 d.] Prieiga per internetą: < http://www.paconsulting.com/our-thinking/preparing-for-hardware-accelerated-high-frequency-trading >

41. MPC-C Series. Maxeler.com [žiūrėta 2016 m. kovo 11 d.] Prieiga per internetą: < http://www.maxeler.com/products/mpc-cseries >

42. Credit Writedowns. The daily links posts. Edward Harrison. Creditwritedowns.com [žiūrėta 2016 m. liepos 10 d.] Prieiga per internetą: https://www.creditwritedowns.com/2010/12/links-2010-12-08.html

43. Antoine B., Cyrille G., Carlos A. R., Christian W., Steffen N. High-frequency trading activity in EU equity markets. Economic Report. 2014, Volume 1.

44. The authority of the financial markets. (2010). Report: High frequency trading: The application of advanced trading technology in the European Marketplace.

45. Bondarenko, O. (2003). Statistical arbitrage and security prices. Review of Financial Studies, 16, 875 – 919.

46. Whistler, M. (2004). Trading Pairs. Hoboken, New Jersey: John Wiley & Sons, Inc.

47. Why pairs trading? Pairstradefinder.com [žiūrėta 2016 m. liepos 16 d.] Priega per internetą: < http://www.pairtradefinder.com/>

48. Chen, H., Chen, S., & Li, F. (2012). Empirical investigation of an equity pairs trading strategy. Working paper.

49. Bowen, D., Hutchinson, M., & O'Sullivan, N. (2010). High frequency equity pairs trading: Transaction costs, speed of execution, and patterns in returns. Working paper.

50. Siy-Yap, D. (2009). Evaluation of the pairs trading strategy in the Canadian market. Working paper.

51. Gesina Goter. (2006). Pairs trading. Working paper.

52. Jeff Blokker, Emile Chamoun, Ibrahim Jreige,Paris Georgoudis, Sameh Galal. (2010). Statistical Arbitrage. Working paper.

53. Robert J. Bianchi, Michael E. Drew, Roger Zhu. (2009). Pairs Trading Profits in Commodity Futures Markets. Department of Accounting, Finance and Economics. Working paper.

54. Driaunys K., Masteika S., Sakalauskas V., Vaitonis M. An algorithm-based statistical arbitrage high frequency trading system to forcast prices of natural gas futures.. Transformations in bussines and economics. 2014, Volume 13, No. 3, psl. 96–109.

55. Antoine B., Cyrille G., Carlos A. R., Christian W., Steffen N. High-frequency trading activity in EU equity markets. Economic Report. 2014, Volume 1

56. Gomber p., Arndt B., Lutat M., Uhle T. (2011). High – frequency trading. [žiūrėta 2016 m. rugpjūčio 6 d.] Prieiga per internetą:< http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1858626>

57. Agarwal A. (2012). High frequency trading: evolution and the future. [žiūrėta]. Prieiga per internetą:< https://www.capgemini.com/resource-file-access/resource/pdf/High_Frequency_Trading__Evolution_and_the_Future.pdf>

58. Miller R. S., Shorter G. (2016). High frequency trading: Overview of recent developments. [žiūrėta 2016 m. rugpjūčio 8 d.]. Prieiga per internetą: < https://www.fas.org/sgp/crs/misc/R44443.pdf>

59. Goldstein M. A., Kumar P. K., Graves F. C. (2014). Computerized and High-Frequency Trading. Forthcoming, The Financial Review, May 2014, Volume 49, No. 2, psl. 177 – 202.

60. Vinit J., Falke B. (2011). A Low-Latency Solution for HighFrequency Trading from IBM and Mellanox. [žiūrėta 2016 m. rugpjūčio 8 d.]. Prieiga per internetą:< http://www.mellanox.com/pdf/whitepapers/Low-Latency-Solution-for-High-Frequency-Trading-from-IBM-and-Mellanox.pdf>

61. Byrne D. (2015). Algo Speed High-Frequency Trading Solution. [žiūrėta 2016 m. rugpjūčio 8 d.]. Prieiga per internetą: <http://www.cisco.com/c/dam/en_us/solutions/industries/docs/finance/c22-658397-01_sOview.pdf>

62. Burton G. M. The Efficient Market Hypothesis and Its Critics. Journal of Economic Perspectives. 2003, Volume 17, No. 1, psl. 59 – 82.

63. Carrion, A. Very fast money: High – frequency trading on the NASDAQ. Journal of Financial Markets. 2013, Vokume 16, No. 4, psl. 680 – 711.

64. Driaunys K., Masteika S., Sakalauskas V., Vaitonis M. An algorithm-based statistical arbitrage high frequency trading system to forcast prices of natural gas futures.. Transformations in bussines and economics. 2014, Volume 13, No. 3, psl. 96–109.

65. Miao, J.G. High Frequency and Dynamic Pairs Trading Based on Statistical Arbitrage Using a Two-Stage Correlation and Cointegration Approach. International Journal of Economics and Finance. 2014, Volume 6, No. 3, pages 96-110.

66. Zubulake P., Lee S. The High frequency game changer: how automated trading strategies have revolutionized the markets. Aite group. Wiley trading, The high – frequency. 2011.
67. Cifu, D.A. FORM S-1, Registration Statement Unde The Securities Act Of 1933. Virtu Financial, Inc.. 2014 [žiūrėta 2016.12.05]. Prieiga per internetą: https://www.sec.gov/Archives/edgar/data/1592386/000104746914002070/a2218589zs-1.htm
68. Antoine B., Cyrille G., Carlos A. R., Christian W., Steffen N. High-frequency trading activity in EU equity markets. Economic Report. 2014, Volume 1.
69. Gatev E., Goetzmann W. N., Rouwenhorst K. G. Pairs Trading: Performance of a Relative-Value Arbitrage Rule. The Review of Financial Studies. 2006, Volume 19, No. 3, psl. 797-827.
70. Perlin, M. Evaluation of Pairs Trading Strategy at Brazilian Financial Marke. Journal of Derivatives & Hedge Funds. 2009, Volume 15, psl. 122-136.
71. M. M. Dacorogna, R. Gengay, U. Miller, R. B. Olsen, O. V. Pictet. An introduction to High-Frequency Finance. Academic Pres. 2001.
72. Kumiega, B. V. Vliet. Quality Money Management: Process Engineering and Best Practices for Systematic Trading and Investment. Academic Press. 2008
73. D. Kirkpatrick, J. R. Dahlquist. Technical Analysis: The Complete Resource for Financial Market Technicians. FT Press. 2008.
74. L. Bernstein, C. M. Yuhas. Trustworthy Systems Through Quantitative Software Engineering. Wiley-IEEE Computer Society Press. 2005
75. S.Friedenthal, A. Moore, R. Steiner. A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann . 2008
76. D. Dori , Object-Process Methodology. Springer. 2002
77. Turing Finance. Algorithmic Trading System Requirements. 2014 [žiūrėta 2016.12.05]. Prieiga per internetą: http://www.turingfinance.com/algorithmic-trading-system-requirements-post/
78. Ch. Li, G. Hains, Y. Khmelevsky, B. Potter, J. Gaston, et al.. Generating a Real-Time Algorithmic Trading System Prototype from Customized UML Models (a casestudy). [Technical Report] TR-LACL-2012-09, 2012, pp.14.

79. S. Seth. Build a Profitable Trading Model In 7 Easy Steps. 2015 [žiūrėta 2017.01.05]. Prieiga per internetą: http://www.investopedia.com/articles/active-trading/010614/build-profitable-trading-model-7-easy-steps.asp
80. Ch. Lu. High Frequency Trading: Price Dynamics Models and Market Making Strategies. [Technical Report] UCB/EECS-2012-144, 2012.
81. S. Kim, S. Huh, Y. Hu, A. Wated, M. Silberstein, X. Zhang ir E. Witchel, 2014. GPUnet: *Networking Abstractions for GPU Programs*. ACM Transactions on Computer Systems, Vol. 34 No. 3.
82. J. Labaki, L. O. S. Ferreira ir E. Mesquita, 2011. *Constant Boundary Elements on graphics hardware: a GPU-CPU complementary implementation*, Journal of the Brazilian Society of Mechanical Sciences and Engineering, Vol.33 No.4.
83. D. G. Spampinato ir A. C. Elster, 2009. *Linear optimization on modern GPUs*, in: 2009 IEEE International Parallel and Distributed Processing Symposium (IPDPS), ISSN 1530-2075, pp. 1–8.
84. B. Oancea, T. Andrei ir R. M. Dragoescu, 2012. *Improving the Performance of the Linear Systems Solvers using CUDA*, Proceedings of Challenges of the Knowledge Society (CKS), pp. 2036–2046.
85. Berten, *GPU vs FPGA Performance Comparison*. 2015 [žiūrėta 2017.04.15]. Prieiga per internetą: http://www.bertendsp.com/pdf/whitepaper/BWP001_GPU_vs_FPGA_Performance_Comparison_v1.0.pdf.
86. C. Adams, 2014. FPGA or GPU? - The evolution continues. [žiūrėta 2017.04.15]. Prieiga per internetą: http://mil-embedded.com/articles/fpga-gpu-evolution-continues/.
87. J. Cong, Z. Fang, M. Lo ir S. Zhang, 2018. *Understanding Performance Differences of FPGAs and GPUs*. The 26th IEEE International Symposium on Field-Programmable Custom Computing Machines, April 29 - May 1, Boulder, CO, USA.
88. P. P. Nambiar, V. Saveetha, S. Sophia ir V. A. Sowbarnika, 2014. *GPU Acceleration Using CUDA Framework*. International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2 No 3, pp. 200 – 205.
89. A. Asaduzzaman, D. Gummadi ir C. M. Yip, 2014. *A talented CPU-to-GPU memory mapping technique*. IEEE SOUTHEASTCON 2014, Lexington, KY.

90. Masteika S., Vaitonis M. 2015, *Quantitative Research in High Frequency Trading for Natural Gas Futures Market*, Business Information Systems Workshops, Springer International Publishing, Vol. 228,p. 29-35.

91. Vaitonis M., Masteika S., 2016. *High frequency statistical arbitrage strategy engineering and algorithm for pairs trading selection*. 7th International Workshop on Data Analysis Methods for Software Systems [abstracts book], Druskininkai, Lithuania, December 3-5, 2015. ISBN 978-9986-680-58-1. p. 51.

92. Vaitonis M., Masteika S., 2016. *Research in high frequency trading and pairs selection algorithm with Baltic region stocks*. Information and Software Technologies. 22nd International Conference, ICIST 2016, Druskininkai, Lithuania, October 13-15, 2016, Proceedings. ISBN 978-3-319-46254-7, p.p. 208 – 217.

93. Vaitonis M., 2017. *Pairs Trading Using HFT in OMX Baltic Market*. Baltic J. Modern Computing, Vol. 5(2017), No. 1, 37-49.

94. Vaitonis M., Masteika S., 2017. *Statistical Arbitrage Trading Strategy in Commodity Futures Market with the Use of Nanoseconds Historical Data*, Information and Software Technologies: 23rd International Conference, ICIST 2017, Druskininkai, Lithuania, October 12–14, 2017, Proceedings. R. Damaševičius and V. Mikašytė (Eds.): ICIST 2017, CCIS 756, pp. 303–313, ISBN 978-3-319-67642-5.

95. Vaitonis M., Masteika S., 2018. *Experimental Comparison of HFT Pair Trading Strategies using Microsecond and Nanosecond Future Commodity Contracts Data*. Baltic J. Modern Computing, Vol. 6(2018), No. 2, 195-216, ISSN 2255-8950