Tartu Ülikool
Loodus- ja täppisteaduste valdkond
Matemaatika ja statistika instituut

Meelis Utt

Paralleelarvutused optsiooni hinna leidmise numbrilistes meetodites

Kindlustus- ja finantsmatemaatika eriala
Magistritöö (30 EAP)

Juhendajad: PhD Toomas Raus
PhD Eero Vainikko

Tartu 2021

Paralleelarvutused optsiooni hinna leidmise numbrilistes meetodites

Magistritöö

Meelis Utt

Lühikokkuvõte

Optsioon on finantsinstrument, mis annab optsiooni omanikule õiguse, aga mitte kohustuse osta või müüa mingit riskantset vara. Keerulisemate optsioonide hinna leidmise jaoks on vaja kasutada numbrilisi meetodeid: võremeetodid (binoom- ja trinoommeetod), Monte Carlo meetodid ning diferentsiaalmeetodid. Selleks, et arvutamisele kuluvat aega vähendada, saab kasutusele võtta paralleelarvutamise. Käesolevas töös uuritakse Euroopa, Ameerika ja Aasia optsioonide hinna paralleelselt arvutamist binoomvalemi ja Monte Carlo meetodiga. Paralleliseerime binoomvalemit Euroopa optsiooni hinna arvutamiseks, Monte Carlo meetodit: mitme alusvaraga Euroopa korvoptsiooni, Ameerika ning fikseeritud täitmishinnaga Aasia optsiooni hinna arvutamiseks. Töö raames kirjutatud koodid on kättesaadavad avalikust repositooriumist https://github.com/moledoc/parcompfin.

CERCS teaduseriala: P160 Statistika, operatsioonanalüüs, programmeerimine, finants- ja kindlustusmatemaatika.

Märksõnad. optsioonid, finantsmatemaatika, paralleelarvutamine, Monte Carlo meetod.

Parallel calculations in numerical methods of option pricing

Master's Thesis

Meelis Utt

Abstract

An option is a financial instrument that gives the holder of the option the right, but not the obligation, to buy or sell any risky assets. Numerical methods are needed to find the price of more complex options: lattice methods (binomial and trinomial methods), Monte Carlo methods and differential methods. In order to reduce the computation time, paral-

lel computing can be introduced. In this work, the calculation of the price of European, American and Asian options in parallel with the binomial formula and the Monte Carlo method are investigated. We parallelize the binomial formula for calculating the price of a European option, Monte Carlo method for calculating the price of: European basket option with multiple underlying assets, American option and Asian option with fixed strike price. The codes written in the framework of this work are available in the public repository https://github.com/moledoc/parcompfin.

CERCS research specialisation: P160 Statistics, operation research, programming, actuarial mathematics.

Keywords. options, computational finance, parallel computing, Monte Carlo method

Sisukord

Sissejuhatus **1**

1 Optsiooni hind **2**
1.1 Optsioonid . . . 2
1.1.1 Optsiooniga seotud tulu . . . 2
1.2 Black-Scholesi mudel . . . 4
1.3 Binoommeetod . . . 6
1.4 Monte Carlo meetod . . . 10
1.4.1 Euroopa optsioon . . . 12
1.4.2 Mitme alusvaraga Euroopa optsioon . . . 13
1.4.3 Ameerika optsioon . . . 14
1.4.4 Aasia optsioon . . . 16

2 Arvutuste paralleliseerimine **18**
2.1 Kasutatavad mudelid . . . 18
2.1.1 Jagatud mäluga mudel . . . 19
2.1.2 Hajusmäluga mudel . . . 21
2.1.3 Hübriidmudel . . . 23
2.2 Arvutuslikud näitajad . . . 24
2.2.1 Kasutamine numbrilistes näidetes . . . 25

3 Paralleelarvutused optsiooni hinna leidmise numbrilistes meetodites **27**
3.1 Euroopa optsiooni hinna leidmine binoomvalemiga . . . 29
3.2 Mitme alusvaraga Euroopa optsiooni hinna leidmine MC meetodiga . . . 35
3.3 Ameerika optsiooni hinna leidmine MC vähimruutude meetodiga . . . 38
3.4 Aasia optsiooni hinna leidmine MC meetodiga . . . 41

Kokkuvõte **44**

Viited **46**

Sissejuhatus

Optsioon on finantsinstrument, mis annab omanikule õiguse, aga mitte kohustuse osta või müüa mingit riskantset vara. Lihtsamatel optsioonidel, näiteks Euroopa optsioonil, on võimalik teatud tingimustel leida optsiooni hind analüütiliste valemite abil. Keerulisemate optsioonide hinna leidmise jaoks aga ei leidu analüütilisi valemeid, mistõttu tuleb selliste optsioonide hinna arvutamise jaoks kasutada numbrilisi meetodeid. Peamisteks numbrilisteks meetoditeks optsioonide hinna leidmiseks on võremeetodid (binoom- ja trinoommeetod), Monte Carlo meetod ning diferentsiaalmeetodid. Käesolevas töös vaadeldakse lähemalt binoommeetodit ning Monte Carlo meetodit. Selleks, et optsiooni hinna arvutamisele kuluvat aega vähendada, võtame kasutusele paralleelarvutamise. See on mitme arvutusliku ressursi samaaegne kasutamine arvutusliku probleemi kiiremaks lahendamiseks. Selles töös uuritakse Euroopa, Ameerika ja Aasia optsiooni hinna arvutamise paralleliseerimist.

Töö esimeses peatükis seletatakse täpsemalt, mis on optsioon ja kuidas on võimalik sellega tulu teenida. Tutvutakse töös kasutatavate numbriliste meetoditega ning kirjeldatakse kuidas nendega on võimalik erinevate optsioonide hinda arvutada.

Teises peatükis tutvustatakse töös kasutatavaid paralleelarvutamise meetodeid. Paralleelsete lähenemiste kirjeldamisel lähtutakse mälumudelite jaotusest. Tuuakse ülevaade erinevate mudelite tööpõhimõttest ning kuidas vastavat meetodit selles töös kasutatakse. Kirjeldatakse ära arvutuslikud näitajad ning seletatakse, kuidas neid kasutatakse.

Kolmandas peatükis tuuakse välja ning analüüsitakse numbriliste katsete tulemusi. Koodid katsete jaoks on kirjutatud programmeerimiskeeles C++ ning on kättesaadavad avalikust repositooriumist https://github.com/moledoc/parcompfin.

1. Optsiooni hind

1.1. Optsioonid

See peatükk põhineb allikal (Seydel, 2012 lk 1-2).

Optsioon on finantsinstrument, mis annab omanikule õiguse, aga mitte kohustuse, osta või müüa mingit riskantset vara (edaspidi alusvara) lepingus fikseeritud hinna E (edaspidi täitmishind) (ingl k *strike/exercise price*) kindlaks määratud ajahetkel või -perioodil. Optsiooni alusvaraks on tavaliselt aktsiad, aga need võivad olla ka valuuta, tarbekaubad, toorkaubad, võlakirjad, teised optsioonid jne. Kuna hind sõltub riskantse alusvara hinnast, siis optsioonid on tuletisväärtpaberite ehk derivatiivide üheks liigiks.

Optsiooni lepinguga on kindlaks määratud täitmispäev T, mis päeval või päevani saab optsiooniga õigust kasutada. Euroopa optsioon annab õiguse alusvara osta või müüa täitmispäeval T. Ameerika optsioon annab õiguse optsiooni realiseerida mistahes ajahetkel $t \leq T$. Peale täitmispäeva T möödumist on optsioon väärtusetu.

Optsioonid jagunevad ostu- (ingl k *call option*) ja müügioptsioonideks (ingl k *put option*), sõltuvalt sellest, kas optsioon annab omanikule õiguse alusvara osta või müüa.

1.1.1. Optsiooniga seotud tulu

See peatükk põhineb allikal (Seydel, 2012 lk 3-4).

Vaatleme Euroopa ostuoptsiooniga seotud tulu optsiooni täitmispäeval T optsiooni omaniku seisukohalt. Kui alusvara hind hetkel T on väiksem või võrdne täitmishind ($S_T \leq E$), siis ei ole ostuoptsiooni omanikul mõistlik kasutada oma õigust alusvara osta, kuna turult saaks selle odavamalt osta. Kui aga $S_T > E$, siis optsiooni omanikul on mõistlik kasutada oma õigust osta alusvara hinnaga E ning tal on võimalus see turul kallima hinnaga S_T maha

müüa. Kokkuvõttes saame, et ostuoptsiooniga seotud tulu avaldub kujul

$$\Psi_{ostu}(S_T, E) = \begin{cases} S_T - E, & S_T > E, \\ 0, & S_T < E \end{cases} \Rightarrow \Psi_{ostu}(S_T, E) = \max\{S_T - E, 0\}.$$

Toome sisse mugavama tähistuse

$$f^+ := \max\{f, 0\}.$$

Selle tähistuse abil saame ostuoptsiooni tulu kirjutada kujul

$$\Psi_{ostu}(S_T, E) = (S_T - E)^+.$$

Analoogse arutluskäiguga saame müügioptsiooni tulu täitmispäeval T avaldada kujul

$$\Psi_{müügi}(S_T, E) = (E - S_T)^+.$$

Ameerika optsiooni tulu täitmispäeval $t \leq T$ avaldub ostuoptsiooni korral kujul

$$\Psi_{ostu}(S_t, E) = (S_t - E)^+, \ t \leq T$$

ning müügioptsiooni korral

$$\Psi_{müügi}(S_t, E) = (E - S_t)^+, \ t \leq T.$$

Kuna optsiooni omanikul on võimalik täitmispäeval saada positiivset tulu, siis on ilmne, et optsioonilepingu sõlmimisel tuleb optsiooni omanikul tasuda mingi rahasumma optsiooni väljaandjale. Seda summat nimetame optsiooni hinnaks. Optsioonilepingut saab osta või müüa ka järelturul ning sel juhul on vajalik teada optsioonilepingu hinda. Tähistame optsiooni hinna ajahetkel t, $0 \leq t \leq T$, ning alusvara hinna S_t korral tähisega $V(S_t, t)$. On selge, et optsiooni realiseerimispäeval t on optsiooni hind võrdne optsiooni tuluga, $V(S_t, t) = \Psi(S_t, E)$.

1.2. Black-Scholesi mudel

See peatükk põhineb peamiselt allikal (Seydel, 2012, lk 10, 31-32, 38-39, 351-352).

Selleks, et leida optsiooni hind enne optsiooni täitmispäeva, on vaja teha eeldus alusvara hinna käitumise kohta. Standardseks eelduseks on, et alusvara hind käitub vastavalt stohhastilisele diferentsiaalvõrrandile

$$dS_t = \mu S_t dt + \sigma S_t dB_t, \tag{1}$$

kus B_t on standardne Browni liikumine ehk Wieneri protsess, konstant σ iseloomustab alusvara tulususe volatiilsust ning konstant μ alusvara oodatavat tulusust. Märgime, et võrrand (1) on samaväärne võrrandiga

$$d(\log(S_t)) = \mu dt + \sigma dB_t$$

ning suurus

$$d(\log(S_t)) = \log(S_{t+\Delta t}) - \log(S_t) = \log\left(\frac{S_{t+\Delta t}}{S_t}\right)$$

kirjeldab alusvara pidevat tulusust perioodil $(t, t+\Delta t)$. Seda stohhastilist diferentsiaalvõrrandit nimetatakse ka geomeetriliseks Browni liikumiseks. Wieneri protsess B_t on ajas $t \geq 0$ pidev protsess, millel on järgmised omadused:

- $B_0 = 0$;
- $B_t \sim N(0, \sqrt{t}),\ \forall t \geq 0$;
  - $E(B_t) = 0$;
  - $D(B_t) = E(B_t^2) = t$;
- iga $t \geq 0$ korral on mittekattuvate ajaintervallidega muudud $\Delta B_t := B_{t+\Delta t} - B_t$ sõltumatud;
- protsess B_t sõltub pidevalt ajast t.

Tutvume põgusalt diskreetse aja Wieneri protsessiga. Olgu $\Delta t > 0$ konstantne aja muut. Juhul kui meil on diskreetsed $t_i := i \cdot \Delta t$, kus $i = 0, 1, 2, \ldots, \frac{T}{\Delta t}$, siis väärtust B_{t_i} saab avaldada

kui

$$B_{t_i} = \sum_{k=1}^{i} \overbrace{\left(B_{k\cdot\Delta t} - B_{(k-1)\cdot\Delta t} \right)}^{:=\Delta B_k}.$$

Muudud ΔB_k on sõltumatud ning $\Delta B_k \sim N(0, \sqrt{\Delta t})$. Olgu $Z \sim N(0,1)$ ning $z \in Z$. Siis muutu ΔB_k saab arvutada kui

$$\Delta B_k = z \cdot \sqrt{\Delta t},\ \forall k = 1, 2, \ldots, i.$$

Olgu meil täidetud järgmised tingimused:

- puudub arbitraaži võimalus. Arbitraaži puudumine tähendab, et turul ei ole võimalik kasutada kauplemisstrateegiat, mille korral portfelli moodustamisel portfelli hind on null, kuid tuleviku mingil ajahetkel on portfelli väärtus iga võimaliku stsenaariumi korral mittenegatiivne ning mingi positiivse tõenäosusega on portfelli väärtus positiivne;
- riskivaba intressimäär r ja aastane volatiilsus σ on konstandid või ajas sõltuvad funktsioonid, mis on teada optsioonilepingu sõlmimise hetkel;
- ajavahemikus $0 \leq t \leq T$ ei maksta dividende;
- alusvara hind käitub vastavalt võrrandile (1);
- turg on hõõrdevaba, mis tähendab, et puuduvad tehingu tasud ja maksud; intressimäärad laenu võtmisel ja andmisel on võrdsed; kõik osapooled pääsevad varasele infole ligi; individuaalne kauplemine ei mõjuta alusvara hinda; kõik muutujad omavad reaalarvulisi väärtusi.

Nendel tingimustel saab näidata, et Euroopa tüüpi optsiooni hind $V = V(S_t, t)$, $0 \leq t \leq T$, rahuldab teist järku osatuletistega diferentsiaalvõrrandit

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0, \tag{2}$$

mida nimetatakse ka Black-Scholesi võrrandiks. Diferentsiaalvõrrand (2) ei määra optsiooni hinda üheselt. Hinna üheseks määramiseks peab funktsioon $V(S,t)$ rahuldama lõpptingimust,

et optsiooni hind täitmispäeval on võrdne optsiooni tuluga:

$$V(S_T, T) = \Psi(S_T, E).$$

Euroopa ostu- ja müügioptsiooni korral saab Black-Scholesi diferentsiaalvõrrandi lahendi leida analüütiliselt. Optsiooni hinna ajahetkel $t = 0$ saab konstantse riskivaba intressimäära ning volatiilsuse korral leida ka valemitega

$$V_{ostu}(S_0, 0) = S_0\Phi(d_1) - E \cdot e^{-rT}\Phi(d_2); \tag{3}$$

$$V_{müügi}(S_0, 0) = -S_0\Phi(-d_1) + E \cdot e^{-rT}\Phi(-d_2); \tag{4}$$

kus $\Phi(\cdot)$ tähistab standardse normaaljaotuse jaotusfunktsiooni ning

$$d_1 = \frac{\ln\left(\frac{S_0}{E}\right) + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}};$$
$$d_2 = d_1 - \sigma\sqrt{T}.$$

Keerulisemate optsioonide korral üldjuhul optsiooni hinda analüütiliselt leida ei õnnestu ning kasutada tuleb erinevaid numbrilisi meetodeid. Nendest meetoditest vaatleme alljärgnevalt binoommeetodit ning Monte-Carlo meetodit optsiooni hinna arvutamiseks.

1.3. Binoommeetod

See peatükk põhineb peamiselt allikal (Seydel, 2012 lk 16-22, 38-40).

Binoommeetod on diskreetse ajaga meetod optsiooni hetkeväärtuse $V(S_0, 0)$ arvutamiseks. See meetod on küllaltki robustne ning seetõttu leiab laialdast kasutust.

Jagame optsiooni eluea $[0, T]$ N võrdseks ajaperioodiks. Siis ühe perioodi pikkus on $\Delta t = \frac{T}{N}$. Olgu $t_i := i \cdot \Delta t,\ i = 0, 1, \ldots, N$ ning tähistame alusvara hinna ajahetkel t_i tähisega $S_i := S_{t_i}$. Binoommeetodi korral konstrueeritakse alusvara hinnapuu selliselt, et kui ajahetkel t_i, $i = 0, \ldots, N-1$, on alusvara hind S_i, siis järgmisel ajahetkel t_{i+1} saab alusvara hind omada vaid kahte väärtust: $S_{i+1} = S_i u$ tõenäosusega p või $S_{i+1} = S_i d$ tõenäosusega $1-p$, kus

u ja d on konstandid. Lihtsuse mõttes eeldame, et $0 < d < u$. Binoommudelis võib alusvara hind muutuda 2^N erineva stsenaariumi järgi, kuid alusvara hind igal ajahetkel t_i võib omada vaid $i + 1$ erinevat väärtust

$$S_0 u^j d^{i-j},\ j = 0, \ldots, i.$$

Alusvara hinna üles- ja allaliikumise kordajad u ja d ning tõenäosus p määrame kahe tingimuse põhjal. Esiteks peab arbitraaži puudumise tingimusest lähtuvalt igal ajaperioodil alusvara hind kasvama keskmiselt sama kiiresti kui riskivaba vara:

$$E(S_{i+1}|S_i) = S_i e^{r\Delta t},$$

kus r on riskivaba intressimäär ning $E(S_{i+1}|S_i)$ tähistab tinglikku keskväärtust tingimusel, et hind perioodil t_i on teada. Samas, binoommudeli korral kehtib võrdus

$$E(S_{i+1}|S_i) = pS_i u + (1-p)S_i d.$$

Viimasest kahest võrdusest järeldub, et

$$e^{r\Delta t} = pu + (1-p)d, \tag{5}$$

millest saame, et

$$p = \frac{e^{r\Delta t} - d}{u - d}.$$

Tõenäosust p nimetatakse ka riskineutraalseks tõenäosuseks.

Teiseks tingimuseks on see, et binoommudelis hinna liikumise varieeruvus peab vastama hinna varieeruvusele vastavalt eeldusele alusvara hinna käitumise kohta. Riskineutraalse tõenäosusmõõdu korral hinna tinglik dispersioon avaldub kujul

$$D(S_{i+1}|S_i) = S_i^2 e^{2r\Delta t}(e^{\sigma^2\Delta t} - 1), \tag{6}$$

kus σ on alusvara volatiilsust iseloomustav konstant. Binoommudeli korral on alusvara

dispersioon esitatav kujul

$$D(S_{i+1}|S_i) = E(S_{i+1}^2|S_i) - (E(S_{i+1}|S_i))^2 = p(S_iu)^2 + (1-p)(S_id)^2 - S_i^2(pu+(1-p)d)^2. \quad (7)$$

Asendades viimases seoses $pu + (1-p)d$ vastavalt valemile (5), saame seoste (6) ja (7) põhjal

$$e^{2r\Delta t}(e^{\sigma^2\Delta t} - 1) = pu^2 + (1-p)d^2 - e^{2r\Delta t}$$

ehk

$$e^{2r\Delta t+\sigma^2\Delta t} = pu^2 + (1-p)d^2. \quad (8)$$

Võrrandite (5) ja (8) abil saime kaks seost parameetrite u, d, p määramiseks. Selleks, et parameetrid u, d, p oleks võimalik üheselt määrata, võtame kolmandaks tingimuseks seose

$$u \cdot d = 1.$$

Asendades $d = \frac{1}{u}$ ning

$$p = \frac{e^{r\Delta t} - d}{u - d} = \frac{ue^{r\Delta t} - 1}{u^2 - 1}$$

võrrandisse (8), saame ruutvõrrandi tundmatu u suhtes kujul

$$u^2 - (e^{-r\Delta t} + e^{r\Delta t}e^{\sigma^2\Delta t})u + 1 = 0,$$

mille lahenditeks on

$$u = \beta \pm \sqrt{\beta^2 - 1},$$

kus

$$\beta = \frac{1}{2}(e^{-r\Delta t} + e^{(r+\sigma^2)\Delta t}).$$

Märgime, et parameetrid u ja d võime määrata ka teistmoodi, näiteks $u = e^{\sigma\sqrt{\Delta t}}$ ja $d = \frac{1}{u}$ (Cox-Ross-Rubensteini (CRR) valem).

Kokkuvõttes oleme saanud, et parameetrid u, d, p avalduvad kujul

$$\begin{aligned} u &= \beta + \sqrt{\beta^2 - 1}; \\ d &= \beta - \sqrt{\beta^2 - 1}; \\ p &= \frac{e^{r\Delta t} - d}{u - d}. \end{aligned}$$

Optsiooni hind hetkel $t = 0$ leitakse binoommudelis rekursiivselt ajas tagasi liikudes. Olgu

$$S_{n,i} = S_0 u^i d^{n-i},\ i = 0, 1, 2, \ldots, n.$$

Olgu $V_{n,i}$ optsiooni hind ajahetkel $n\Delta t$ ning alusvara hinna $S_{n,i}$ korral. Kõigepealt leiame optsiooni hinna ajahetkel $T = N\Delta t$. Ajahetkel T on optsiooni hind võrdne optsiooni tuluga,

$$V_{N,i} = \Psi(S_{N,i}, E),\ i = 0, \ldots, N.$$

Seejärel arvutame Euroopa optsiooni hinna rekursiivselt $n = N-1, N-2, \ldots, 0$ ajas ettepoole liikudes

$$V_{n,i} = e^{-r\Delta t}(pV_{n+1,i+1} + (1-p)V_{n+1,i}),\ i = 0, 1, \ldots, n.$$

Suurus $V_{0,0}$ ongi optsiooni hinnaks ajahetkel $t = 0$.

Ameerika optsiooni korral on võimalik optsiooni realiseerida igal ajahetkel $n\Delta t$ ning igal ajahetkel peame hindama, kas on mõtet optsiooni hoida järgmise perioodini või realiseerida optsioon. Seetõttu leitakse Ameerika optsiooni hind vastavalt valemile

$$V_{n,i} = \max\{\Psi(S_{n,i}, E), V_{n,i}^{jätk}\},$$

kus

$$V_{n,i}^{jätk} = e^{-r\Delta t}(pV_{n+1,i+1} + (1-p)V_{n+1,i}).$$

Kirjeldatud algoritmis on see probleem, et selles on väga palju üksteisest sõltuvaid arvutusi

ning seetõttu on seda algoritmi keeruline paralleliseerida. Kuid binoommudeli korral on Euroopa optsiooni hinda, mis on võrdne binoommeetodil leitud hinnaga, võimalik leida ka vastavalt valemile (Popuri & Gobbert, 2017, valem (5)):

$$V_{0,0} = e^{-rT} \sum_{i=0}^{N} C_N^i p^i (1-p)^{N-i} V_{N,i},$$

kus $V_{N,i}$ on optsiooni hind täitmisajal T ning alusvara hinna $S_{N,i}$ korral ning C_N^i tähistab kombinatsioonide arvu N-st i kaupa. Edaspidi nimetame seda valemit binoomvalemiks. Arvutuslikult on selle valemiga optsiooni hinna leidmisel see probleem, et kui N on väga suur, tekib arvutis kombinatsioonide arvutamisel ületäituvus, mistõttu ei saada õiget optsiooni hinda. Selleks, et suurendada arvutuses võimaliku perioodide arvu N, kasutame logaritmimist. Tähistame

$$y_i = \log\left(C_N^i p^i (1-p)^{(N-i)}\right).$$

Siis saame, et

$$\begin{aligned} y_i &= \log(N!) - \log(i!) - \log((N-i)!) + i\log(p) + (N-i)\log(1-p) \\ &= \sum_{j=1}^{N} \log(j) - \sum_{j=1}^{i} \log(j) - \sum_{j=1}^{N-i} \log(j) + i\log(p) + (N-i)\log(1-p) \\ &= \sum_{j=i+1}^{N} \log(j) - \sum_{j=1}^{N-i} \log(j) + i\log(p) + (N-i)\log(1-p). \end{aligned}$$

Seega saame optsiooni hinna $V_{0,0}$ leida kujul

$$V_{0,0} = e^{-rT} \sum_{i=0}^{N} e^{y_i} V_{N,i}, \tag{9}$$

kus y_i leitakse ülaltoodud valemi abil. Kuna valemis (9) on iga liidetav teineteisest sõltumatu, siis saab seda arvutust lihtsasti paralleliseerida.

1.4. Monte Carlo meetod

See peatükk põhineb allikatel (Gentle, 2003, lk 229-231; Glasserman, 2003, lk 1-3; Kroese *et al.*, 2014).

Monte Carlo meetod on juhuslikke arvude kasutamine simulatsioonides matemaatilise avaldise väärtustamiseks. Kasutatavad juhuslikud arvud võivad olla loomulikul teel tekkinud, näiteks aktsiaturgude arenemisega või tehislikult genereeritud õigesti valitud jaotusest valimit võttes. Matemaatiliseks avaldiseks võib olla määratud integraal, võrrandite süsteem või mõni keerulisem matemaatiline mudel. Lihtsaim Monte Carlo simulatsioon on määratud integraali

$$\theta = \int_D f(x)dx \tag{10}$$

ligikaudne arvutamine. Olgu meil antud juhuslik suurus X, mille tihedusfunktsioon on $f_X(x)$. Defineerime funktsiooni $g(X)$ kujul

$$g(X) = \begin{cases} \frac{f(x)}{f_X(x)}, & x \in D, \\ 0, & x \notin D. \end{cases}$$

Kuna kehtib valem

$$E[g(X)] = \int_{\mathbb{R}} g(x) f_X(x) dx,$$

siis saame, et

$$E[g(X)] = \int_D g(x) f_X(x) dx = \int_D \frac{f(x)}{f_X(x)} f_X(x) dx = \int_D f(x) dx = \theta.$$

See tähendab, et saame määratud integraali leida keskväärtuse leidmise teel. Keskväärtuse hindamiseks me kasutame Monte Carlo simulatsioone. Genereerime n juhuslikku väärtust $x_1, x_2, \ldots, x_n$ juhuslikust suurusest X. Määratud integraali (10) Monte Carlo hinnanguks saame

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} g(x_i).$$

Kasutades Suurte Arvude Seadust saame väita, et

$$\hat{\theta} \to \theta \text{ kui } n \to \infty.$$

Monte Carlo meetod on oluline finantsmatemaatikas, kuna derivatiivide väärtustamist on

tihtipeale võimalik kirja panna keskväärtuse kaudu.

Edasi vaatame kuidas me saame kasutada Monte Carlo simulatsioone Euroopa, Ameerika ja Aasia optsiooni hinna arvutamiseks.

1.4.1. Euroopa optsioon

Eeldame ühe-dimensionaalset Black-Scholesi mudeli kehtimist. Selle mudeli korral käitub alusvara hind vastavalt stohhastilisele diferentsiaalvõrrandile

$$dS_t = \mu S_t dt + \sigma S_t dB_t, \tag{11}$$

kus S_t on alusvara hind ajahetkel $t \leq T$. Täituvusajal T on selle lahendiks

$$S_T = S_0 e^{(\mu - \frac{\sigma^2}{2})T + \sigma B_T}. \tag{12}$$

Riskineutraalse tõenäosusmõõdu Q korral on parameeter μ võrdne riskivaba intressimääraga r ehk saame stohhastilise diferentsiaalvõrrandi lahendi (12) kirjutada kujul

$$S_T = S_0 e^{(r - \frac{\sigma^2}{2})T + \sigma B'_T},$$

kus B'_T on Browni liikumine riskineutraalse tõenäosusmõõdu Q korral.

Euroopa optsiooni hinna saame arvutada kui (Seydel, 2012, lk 122-124):

1. arvutame alusvara hinnad ajahetkel T simulatsioonide arvu N korral

$$(S_T)_n = S_0 e^{(r - \frac{\sigma^2}{2})T + \sigma B'_T},\ n = 1, \dots, N;$$

2. arvutame optsiooni väärtuse ajahetkel T iga simulatsiooni korral

$$(V(S_T, T))_n = \Psi((S_T)_n, E),\ n = 1, \dots, N;$$

3. arvutame optsiooni keskmise väärtuse ajahetkel T

$$\hat{E}(V(S_T,T)) = \frac{1}{N}\sum_{n=1}^{N}(V(S_T,T))_n;$$

4. arvutame optsiooni hinna ajahetkel $t = 0$, kui diskonteeritud optsiooni keskmise väärtuse

$$V(S_0,0) \approx \hat{V}(S_0,0) = e^{-rT}\hat{E}(V(S_T,T)).$$

1.4.2. Mitme alusvaraga Euroopa optsioon

See peatükk põhineb peamiselt allikal (Mehrdoust *et al.*, 2013).

Mitme alusvaraga optsiooni hind sõltub mitme alusvara väärtusest. Vaatame lähemalt mitme alusvaraga Euroopa korvoptsiooni. Selle omanikul on õigus osta või müüa mitut alusvara täituvusajal T. Euroopa korvoptsiooni tulufunktsioon on kujul

$$\left(\sum_{i=1}^{n} w_i S_{i,T} - E\right)^+ = \max\left\{\sum_{i=1}^{n} w_i S_{i,T} - E, 0\right\},$$

kus n on korvis olevate alusvarade arv, w_i on i-nda alusvara osakaal korvis, $S_{i,T}$ on i-nda alusvara hind täitumishetkel T. Kuna korvis olevad alusvarad võivad omavahel seotud olla, siis alusvara hindade arvutamisel ajahetkel T genereeritakse Browni liikumisi mitmemõõtmelisest standardsest normaaljaotusest $N(\mathbf{0},\Sigma)$, kus kovariatsioonimaatriks Σ avaldub kujul

$$\Sigma = \begin{pmatrix} 1 & \rho_{1,2} & \rho_{1,3} & \dots & \rho_{1,n} \\ \rho_{2,1} & 1 & \rho_{2,3} & \dots & \rho_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \rho_{n,1} & \rho_{n,2} & \rho_{n,3} & \dots & 1 \end{pmatrix}. \tag{13}$$

Optsiooni hinna arvutamine on analoogne alapeatükis 1.4.1 kirjeldatuga. Ainuke erinevus on tulufunktsiooni kujus ning jaotuses, millest juhuslike suurusi genereeritakse.

1.4.3. Ameerika optsioon

Ameerika optsiooni hinna väärtuse arvutamiseks kasutades Longstaff-Schwartzi vähimruutude meetodit (Longstaff & Schwartz, 2001; Seydel, 2012, lk 138-141). Ameerika optsiooni hinna arvutamine on MC meetodiga keeruline. Optsiooni hinna arvutamiseks tuleb simuleerida alusvara hinna liikumine ning rekursiivselt tagant ettepoole arvutades saame Ameerika optsiooni hinnale hinnangu. Igal ajahetkel $t \leq T$ tuleb otsustada, kas optsiooni õigust on mõistlik kasutada või on mõistlikum optsiooni edasi hoida.

Lihtsuse mõttes kasutame M lõpliku ajasammu alusvara hinna simuleerimises. See tähendab, et MC simulatsioonis peame igal sammul $m\Delta t$, $m = 1, \ldots, M$, $\Delta t = \frac{T}{M}$ otsustama, kas jätkame optsiooni hoidmist või kasutame selle õigust müüa/osta alusvara. Kui ühe teekonna korral leida optsiooni hind kui suurim diskonteeritud tulu üle kõigi ajahetkede ning optsiooni hinnaks võtta kõigi teede keskmine, siis ülehindame Ameerika optsiooni hinda.

Selleks, et me ei ülehindaks Ameerika optsiooni hinda selle arvutamisel, kasutame MC vähimruutude meetodit. Tähistame alusvara hinda teekonna $n = 1, \ldots, N$ sammul $m = 1, \ldots, M$ tähisega $S_{n,m}$. Olgu g_n optsiooni tulu optsiooni realiseerimise hetkel teekonnal n ning τ_n ajaperioodi indeks, millal optsioon realiseeriti. Algoritm MC vähimruutude meetodil Ameerika optsiooni hinna arvutamiseks on järgnev:

- Simuleerime alusvara N teekonda M sammuga

$$S_{n,0} = S_0,\ n = 1, \ldots, N;$$
$$S_{n,m} = S_{n,m-1} e^{(r - \frac{\sigma^2}{2})\Delta t + \sigma \Delta B_m},\ n = 1, \ldots, N,\ m = 1, \ldots, M.$$

- Leiame iga teekonna tulufunktsiooni väärtuse sammul M

$$g_n = \Psi(S_{n,M}, E),\ n = 1, \ldots, N$$

  ning võtame indeksiks, millal optsiooni õigust kasutati, M:

$$\tau_n = M,\ n = 1, \ldots, N.$$

- Arvutame igas teekonna $n = 1, \ldots, N$ sammudes $m = M-1, \ldots, 1$:
  - tulufunktsiooni väärtused; teekond on rahas, kui tulufunktsiooni väärtus on suurem kui null ning vastasel korral ei ole; kui teekond on rahas, siis jätame meelde vastava alusvara hinna vektorisse $X_m = (x_1, \ldots, x_{N_m})$, kus N_m on sammul m rahas olevate teekondade arv ja
$$x_i = S_{i,m},\ i = 1, \ldots, N_m.$$
  - kui teekond on rahas, siis arvutame ja jätame meelde viimati optsiooni realiseerimise hetkel diskonteeritud optsiooni tulu vektorisse $Y_m = (y_1, \ldots, y_{N_m})$:
$$y_i = e^{-r\Delta t(\tau_n - m)}\Psi(S_{n,\tau_n}, E),\ i = 1, \ldots, N_m.$$
  - arvutame tingliku keskväärtuse $E(Y_m|X_m)$ hinnangu vastavalt valemile
$$\hat{Y} = X_{mat}\hat{\beta}_m,$$
kus $\hat{\beta}_m$ on lineaarse regressiooni mudeli $Y_m = X_{mat}\beta_m$ vähimruutude meetodil leitud kordajad, mis on leitavad valemiga
$$\hat{\beta}_m = (X_{mat}^T X_{mat})^{-1} X_{mat}^T Y_m \tag{14}$$
ning
$$X_{mat} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_{N_m} & x_{N_m}^2 \end{pmatrix}, Y_m = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N_m} \end{pmatrix}. \tag{15}$$
  - kui tulufunktsiooni väärtus rahas olnud teekonnal n on suurem kui tinglik keskväärtus $\hat{Y}$ (ehk kasutati optsiooni õigust), siis jätame meelde tulufunktsiooni väärtuse

sammul m ja võtame indeksiks, millal optsiooni realiseeriti, indeksi m:

$$g_n = \Psi(S_{n,m}, E), \text{ kui } \Psi(S_{n,m}, E) > \hat{Y};$$
$$\tau_n = m, \text{ kui } \Psi(S_{n,m}, E) > \hat{Y}.$$

- Arvutame optsiooni hinna hinnangu

$$\hat{V}(S_0, 0) = \max\left\{\Psi(S_0, E), \frac{1}{N}\sum_{n=1}^{N} e^{-r\tau_n \Delta t} g_n\right\}.$$

Paneme tähele, et saame võrrandis (14) maatriksarvutusi järgnevalt lihtsustada, kui leiame maatriksid $X_{mat}^T X_{mat}$ ja $X_{mat}^T Y_m$ vastavalt valemitele:

$$X_{mat}^T X_{mat} = \begin{pmatrix} N_m & \sum_{i=1}^{N_m} x_i & \sum_{i=1}^{N_m} x_i^2 \\ \sum_{i=1}^{N_m} x_i & \sum_{i=1}^{N_m} x_i^2 & \sum_{i=1}^{N_m} x_i^3 \\ \sum_{i=1}^{N_m} x_i^2 & \sum_{i=1}^{N_m} x_i^3 & \sum_{i=1}^{N_m} x_i^4 \end{pmatrix}; \; X_{mat}^T Y_m = \begin{pmatrix} \sum_{i=1}^{N_m} y_i \\ \sum_{i=1}^{N_m} x_i y_i \\ \sum_{i=1}^{N_m} x_i^2 y_i \end{pmatrix}. \tag{16}$$

1.4.4. Aasia optsioon

See peatükk põhineb peamiselt allikatel (Seghiouer *et al.*, 2011; Seydel, 2012 lk 2, 274, 280-281).

Aasia optsioon liigitub eksootiliste optsioonide hulka. Aasia optsiooni korral optsiooni tulu sõltub alusvara keskmisest hinnast fikseeritud ajaperioodil. Eksootiliste optsioonide hulgas jääb kehtima ostu-/müügi ja Euroopa/Ameerika optsiooni eristus. Vaatleme Euroopa tüüpi Aasia optsiooni. Siis ostu-/müügioptsiooni muutuva/fikseeritud täitmishinnaga Aasia optsiooni tulu on leitav järgmiselt:

- muutuva täitmishinnaga optsioonid (ingl k *floating strike options*)
  - ostuoptsioon: $\Psi_{ostu}(S_T, I) = (S_T - I)^+$;
  - müügioptsioon: $\Psi_{müügi}(S_T, I) = (I - S_T)^+$;

- fikseeritud täitmishinnaga optsioonid (ingl k *fixed strike options*)
  - ostuoptsioon: $\Psi_{ostu}(I,E) = (E-I)^+$;
  - müügioptsioon: $\Psi_{müügi}(I,E) = (I-E)^+$;

kus I on alusvara keskmine hind perioodi $[0,T]$ jooksul:

$$I = \int_0^T S_t dt.$$

Aasia optsiooni hindamisel tuleb lisaks hinna simulatsioonile simuleerida ka keskmist hinda I. Selleks tuleb leida esmalt integraali I lähendväärtus. Üheks võimaluseks integraali I lähendväärtuse leidmiseks on kasutada valemit

$$I_1 = \Delta t \sum_{i=0}^{M-1} S_i,$$

kuid artiklis (Lapeyre & Temam, 2000) on tõestatud, et täpsuse mõttes on parem kasutada integraali I lähendväärtuse arvutamiseks nn trapetsskeemi kujul:

$$I_{tra} = \frac{1}{M} \sum_{i=0}^{M-1} S_i \left(1 + \frac{r\Delta t}{2} + \sigma \frac{\Delta B_i}{2}\right), \tag{17}$$

kus $\Delta B_i = B_{i+1} - B_i$ on normaaljaotusega $N(0, \sqrt{\Delta t})$ juhuslikud suurused. Kui alusvara hind käitub vastavalt stohhastilisele diferentsiaalvõrrandile (1), siis valemis (17) olev alusvara hind S_i on arvutatav kui

$$S_i = S_{i-1} e^{\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma \Delta B_i}, \ i = 1, \ldots, M-1.$$

Nüüd saame fikseeritud täitmishinnaga Aasia optsiooni hinna arvutada valemiga

$$\hat{V}(S_0, 0) = \frac{e^{-rT}}{N} \sum_{i=1}^{N-1} \Psi((I_{tra})_i, E).$$

2. Arvutuste paralleliseerimine

See peatükk põhineb peamiselt allikatel (Hesham El-Rewini, 2005, lk 4-9; Thomas Rauber, 2010, lk 2-3, 10-12, 151; Roman Trobec, 2018, lk 3).

Keerulise arvutusliku probleemi kiiremaks lahendamiseks kasutatakse arvutuste paralleliseerimist, mis saavutatakse mitme arvutusliku ressurssi samaaegse kasutamisega. Paralleelarvutamine on võimalik vaid paralleelsüsteemides. Lähtudes 1966. aastal Michael J. Flynn poolt välja pakutud arvutite arhitektuuri taksonoomiast, saab arvuti arhitektuurid jaotada nelja kategooriasse:

- ühe juhisega, ühe andmevooga arhitektuur (ingl k *single-instruction single-data streams (SISD)*);
- ühe juhisega, mitme andmevooga arhitektuur (ingl k *single-instruction multiple-data streams (SIMD)*);
- mitme juhisega, ühe andmevooga arhitektuur (ingl k *multiple-instruction single-data streams (MISD)*);
- mitme juhisega, mitme andmevooga arhitektuur ((ingl k *multiple-instruction multiple-data streams (MIMD)*)).

Paralleelarvutiteks võib nimetada arvuteid, mis kuuluvad kas MIMD või SIMD klassifikatsiooni. Kuigi SIMD-tüüpi arvutitele on lihtsam programme kirjutada, siis on see arhitektuur liiga piirav. Seetõttu kuuluvad peaaegu kõik paralleelarvutid MIMD kategooriasse. Tehnoloogia arengu tõttu on paralleelarvutid jõudnud ka tarbijateni. Kõik tänapäevased arvutid kuuluvad peamiselt MIMD kategooriasse.

2.1. Kasutatavad mudelid

Flynni välja pakutud klassifikatsioon on kasulik üldpildi loomiseks, kuid täpsemaks vaatluseks on vajalik täiendav jaotamine. Üks võimalik alamjaotus, mida selles töös kasutame, on mälumudelite järgi (Thomas Rauber, 2010, lk 10-12):

- jagatud mäluga mudel (ingl k *shared memory model*);

- hajusmäluga mudel (ingl k *distributed memory model*);
- hübriidmudel.

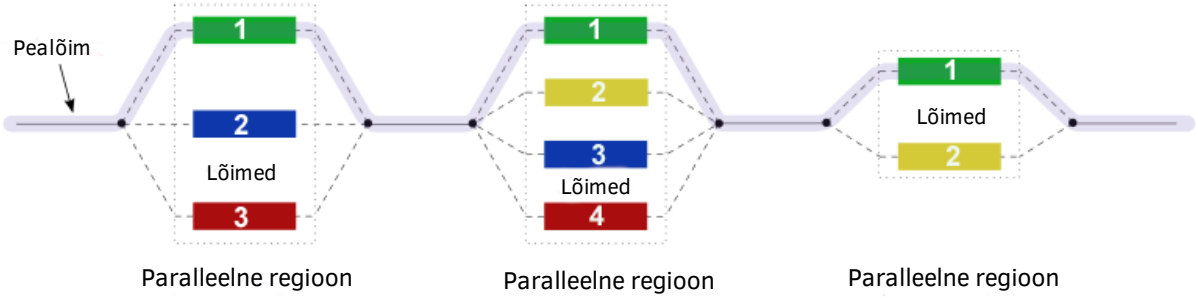
2.1.1. Jagatud mäluga mudel

See peatükk põhineb peamiselt allikatel (Roman Trobec, 2018, lk 47-50, 52, 55, 67, 73; "OpenMP juhend," 2021).

Jagatud mäluga mudeli korral kasutatakse üksteisest sõltumatuid protsessoreid, mis kõik kasutavad ühte ühist mälu. Tänapäeval koosnevad protsessorid mitmest tuumast. Igat tuuma võib vaadelda kui mitut loogilist tuuma tänu hüperhargtöötlusele (ingl k *hyperthreading*). Iga loogiline tuum on suuteline jooksutama programmi või lõime konkreetses programmis. Lõimeks (ingl k *thread*) nimetatakse instruktsioonide jada. Jagatud mäluga programmeerimisel tekib programmis paralleelsus mitme teineteisest sõltumatu lõime kasutamisel. Selle mälumudeli puuduseks on see, et mitu lõime võivad korraga kasutada sama mäluaadressiga infot, mille tulemusena tekib võidujooksu olukord (ingl k *race condition*). See võib põhjustada programmi soovimatut käitumist. Selle vältimiseks kasutatakse mälu lukustamise mehhanismi. See tähendab, et korraga pääseb ainult üks lõim jagatud mälu aadressile ligi. Selleks, et lõim saaks infot konkreetsel mäluaadressil muuta, peab lõim kõigepealt aadressi lukustama ning pärast muutuse tegemist aadressi jälle lukust vabastama. Kui aadress on juba lukus, peab lõim ootama seni kuni aadress on vabastatud.

Käesolevas töös kasutatakse jagatud mäluga programmeerimiseks OpenMP ("OpenMP koduleht," 2021) (edaspidi OMP) täiendust C++ programmeerimiskeelele. See täiendus sisaldab endas kompilaatori direktiive, abifunktsioone ning käsurea muutujaid. Direktiivid annavad kompilaatorile juhised kuidas vastavast lähtekoodist paralleelset masinkoodi kompileerida. Abifunktsioonid aitavad programmeerijal kontrollida programmi paralleelsust programmi tööajal. Käsurea muutujate abil on võimalik programmi paralleelsust kontrollida enne programmi käivitamist. See tähendab, et programmi paralleelsust on võimalik mitut moodi kontrollida ning see annab programmeerijale vabaduse kasutada seda meetodit, mis talle konkreetse probleemi lahendamiseks sobib.

OMP programm käivitatakse ühel pealõimel. Kui tööajal jõutakse paralleelse regioonini, loob pealõim eelnevalt defineeritud arvu paralleelseid lõimesid. Paralleelse regiooni töö jaotatakse lõimede vahel ära. Kui vastava regiooni töö on tehtud, koondatakse tulemused pealõime ning kõik lõimed peale pealõime lõpetavad. Edasi toimub töö järjestikusel viisil pealõimes kuni järgmise paralleelse regioonini. Seda käitumist illustreerib joonis 1.



Joonis 1. OpenMP tööd illustreeriv joonis, 2021-05-14, https://hpc.llnl.gov/openmp-tutorial

Selles töös kasutati abifunktsioone, et kontrollida programmi paralleelsust. Põhiliselt kasutati abifunktsiooni

```
void omp_set_num_threads(int lõimed),
```

et seada arvutusteks kasutatavate lõimede arv. Algoritmides ja meetodites, mida selles töös käsitletakse, ei olnud palju teineteisest sõltumatuid arvutusplokke ning arvutused olid koondunud tsüklitesse. Seetõttu kasutati peamiselt OMP *for* direktiivi programmide paralleeliseerimisel. Direktiividele on võimalik anda täiendavaid juhiseid, et täpsustada paralleliseerimist vastava direktiivi ulatuses. Näide kasutatud OMP *for* direktiivist on järgmisel kujul.

```
#pragma omp for schedule(dynamic,1000) nowait reduction(+:result)
```

Näites toodud direktiivi tingimused täpsustavad järgmist:

- *schdule(tüüp [,tükk])* - kirjeldab kuidas tsükli iteratsioonid lõimede vahel jaotuvad; tingimus *schdule(dynamic,1000)* jaotab programmi tööajal iteratsioonide arvu tükkideks, milles igas on kuni 1000 iteratsiooni ning dünaamiliselt jaotab need lõimede vahel: kui üks lõim lõpetab, antakse sellele järgmine tükk tegemiseks;

- *nowait* - annab direktiivile märku, et paralleelse tsükli lõpus ei ole lõimedel vaja sünkroniseerida;
- *reduction(operaator: muutujate järjend)* - iga muutuja jaoks tehakse igas lõimes koopia, kuhu salvestatakse lõime vahetulemus ning paralleelse regiooni lõpus koondatakse muutujad operaatori abil üheks muutujaks; tingimuse *reduction(+:result)* korral arvutatakse igas lõimes *result* väärtus ning paralleelse tsükli lõpus summeeritakse kõikide lõimede *result* väärtused samanimelisse muutujasse kokku.

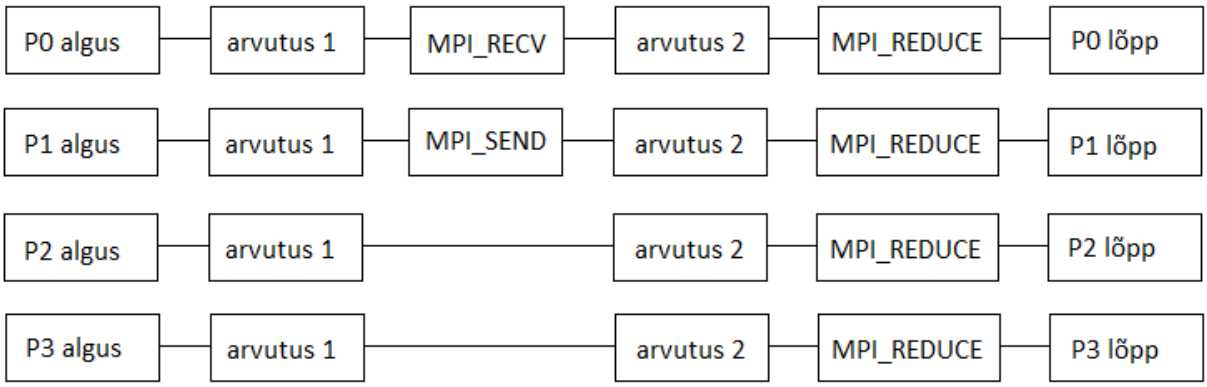
2.1.2. Hajusmäluga mudel

See peatükk põhineb peamiselt allikatel (Thomas Rauber, 2010 lk 214-227; Roman Trobec, 2018, lk 87-89).

Hajusmäluga mudeli korral kasutatakse protsesse, mis asuvad erinevatel protsessoritel. Protsesside vahel toimub sõnumite teel suhtlus, et tööd koordineerida ning infot vahetada. Protsesside vaheline suhtlus on aeglasem kui jagatud mälus olevate lõimede vaheline suhtlus. See-eest on hajusmäluga mudeli korral võimalik efektiivselt kasutada rohkemaid protsessoreid. Selleks, et protsessid protsessorites saaks omavahel suhelda kasutatakse platvormi neutraalset sõnumite edastamise liidest (ingl k *message passage interface (MPI)*). Hajusmäluga mudeli korral on suurimaks pudelikaelaks protsesside omavaheline suhtlus. Seetõttu on paralleelses programmeerimises oluline jälgida protsesside vahelist suhtlust, selle sagedust ning jagatava informatsiooni suurust. MPI-s on võimalik kasutada nii kahe protsessi vahelist suhtlust (*point-to-point communication*) kui ka kollektiivset suhtlust (ingl k *collective communication*). Joonis 2 illustreerib võimaliku MPI lähenemisega tehtud programmi tööd, kus kasutatakse nii protsesside vahelist kui ka kollektiivset suhtlust.

Selleks, et tulemused oleksid võrreldavad OpenMP lahendustega, kasutatakse C++ programmeerimiskeelt MPI lahenduste kirjutamisel. Kasutatakse *openmpi* MPI implementatsiooni ("OpenMPI koduleht," 2021).

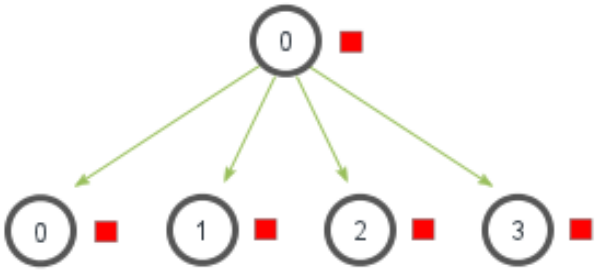
Selles töös kasutatakse kollektiivset suhtlust, kuna ei olnud tarvidust kasutada kahe protsessi vahelist suhtlust. Kasutati järgmisi kollektiivseid suhtlusviise:



Joonis 2. MPI tööd illustreeriv joonis

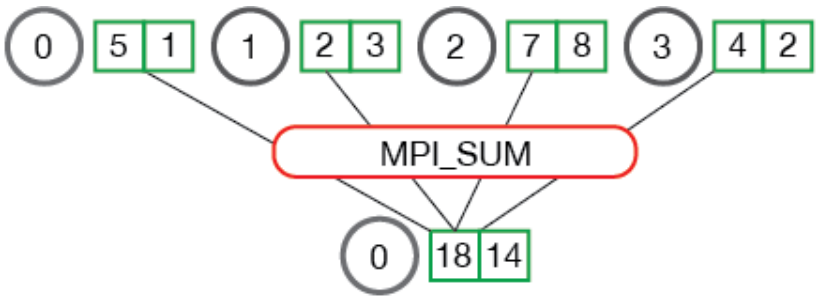
```
int MPI_Bcast(void *sõnum, int loend,
    MPI_Datatype tüüp, int juur, MPI_comm suhtleja);
int MPI_Reduce(void *saadetav_puhver, void *saadav_puhver, int loend,
    MPI_Datatype tüüp, MPI_Op operatsioon, int juur, MPI_comm suhtleja);
int MPI_Allreduce(void *saadetav_puhver, void *saadav_puhver, int loend,
    MPI_Datatype tüüp, MPI_Op operatsioon, MPI_comm suhtleja).
```

Funktsioon *MPI_Bcast* imiteerib ülekannet (ingl k *broadcast*) ehk protsessist *juur* saadetakse kõikidele teistele protsessidele *sõnum*, mille sisuks on *loend* elemendiga andmehulk ning need on andmetüübiga *tüüp*, näiteks *MPI_DOUBLE, MPI_INT*. Selle tööd illustreerib joonis 3.



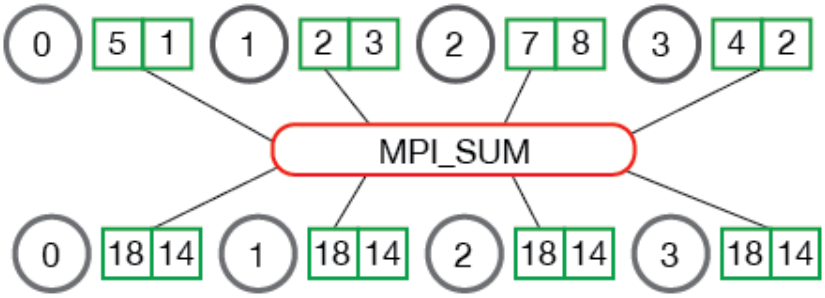
Joonis 3. *MPI_Bcast* tööd illustreeriv joonis, Wes Kendall, 2021-05-14, https://mpitutorial.com/tutorials/mpi-scatter-gather-and-allgather/

Funktsioon *MPI_Reduce* koondab protsessi *juur* muutuja *saadetav_puhver* väärtused muutujasse *saadav_puhver*, kasutades funktsiooni *operatsioon*. Siin näitab *loend* mitu elementi on *saadetav_puhver*-s ning *tüüp* näitab, analoogselt *MPI_Bcast*-le, puhvris olevate elementide andmetüüpi. Selle tööd illustreerib joonis 4.



Joonis 4. *MPI_Reduce* tööd illustreeriv joonis, Wes Kendall, 2021-05-14, https://mpitutorial.com/tutorials/mpi-reduce-and-allreduce/

Funktsioon *MPI_Allreduce* erineb funktsioonist *MPI_Reduce* selle poolest, et *saadav_puhver* koondatakse kõikidesse protsessidesse. Selle tööd illustreerib joonis 5.



Joonis 5. *MPI_Allreduce* tööd illustreeriv joonis, Wes Kendall, 2021-05-14, https://mpitutorial.com/tutorials/mpi-reduce-and-allreduce/

Käesolevas töös kasutataksegi kollektiivsetes suhtlusviisides koondavaks operaatoriks *MPI_SUM*-i, mis summeerib *saadetav_puhver* väärtused muutujasse *saadav_puhver*.

2.1.3. Hübriidmudel

Selles peatükis kasutatakse allikaid (Rabenseifner, 2003; "OpenMP juhend," 2021).

Hübriidmudel on selline mudel, kus kasutatakse nii jagatud kui ka hajusmäluga mudeli lähenemisi. Üks võimalus oleks teostada sõlmede vaheline suhtlus kasutades MPI-d ning arvutuslikult intensiivne töö kasutades OMP-d. Hübriidmäluga mudelil on aga palju erinevaid võimalusi kuidas MPI-d ja OMP-d kombineerida, kusjuures igal võimalusel on omad positiivsed ja negatiivsed küljed. Kuna selles töös põhirõhk ei ole hübriidmudelil, siis numbrilistes katsetes ei keskenduta hübriidlähenemisele nii põhjalikult kui OMP ja MPI lähenemistele. Näiteülesanded lahendatakse lihtsa hübriidlahendusega võrdluse eesmärgil, kus kasutatakse

võrdne arv protsesse ja lõimesid. See tähendab, et kui joonisel on hübriidlahenduse korral märgitud paralleelsuse astmeks 64, siis kasutatakse 8 protsessi, kus igal protsessil kasutatakse 8 lõime, kusjuures protsess ise on pealõimeks.

2.2. Arvutuslikud näitajad

See peatükk põhineb peamiselt allikal (Thomas Rauber, 2010 lk 3-5, 151, 162-164; Roman Trobec, 2018, lk 10).

Arvutusliku probleemi paralleelseks lahendamiseks on tarvis paralleelset programmi. Seda on tihtipeale keerulisem implementeerida kui järjestikust programmi, kuid võib anda oluliselt väiksema tööaja. Lisaks paralleelse programmi tööajale seonduvad paralleelarvutamisega ka mitmed arvutuslikud näitajad. Nende näitajate abil on võimalik hinnata paralleelse programmi kiirendust järjestikuse programmi suhtes, efektiivsust, skaleeruvust jm. Käesolevas töös keskendume põhiliselt kiirendusele ning käsitleme ka efektiivsust.

Olgu meil arvutuslik probleem ning seda probleemi lahendav järjestikune algoritm A_{ser} ja paralleelne algoritm A_{par}. Olgu algoritmi A_{ser} tööaeg T_{ser} ja algoritmi A_{par} tööaeg T_{par}. Paralleelse algoritmi kiirenduseks nimetatakse suhet

$$S = \frac{T_{ser}}{T_{par}}.$$

Kui järjestikune algoritm on valitud selliselt, et see ei ole kiireim teadaolev lahendus selle probleemile või on rakendatud paralleelset algoritmi ühe lõimega/protsessiga, siis nimetatakse kiirendust S suhteliseks kiirenduseks.

Paralleelse algoritmi A_{par} efektiivsuseks nimetatakse suhet

$$E = \frac{S}{p},$$

kus S on (suhteline) kiirendus ja p on paralleelsete protsesside/lõimede arv, mida paralleelses töös kasutati. Efektiivsus näitab keskmiselt kui suure osa ajast lõim/protsess tööd tegi. Teisiti, väärtus $1 - E$ näitab keskmiselt kui suure osa ajast lõim/protsess oli tegevuseta. Paneme

tähele, et

$$T_{par} \leq T_{ser} \leq p \cdot T_{par} \Rightarrow \frac{T_{par}}{T_{par}} \leq \frac{T_{ser}}{T_{par}} \leq p \cdot \frac{T_{par}}{T_{par}} \Rightarrow 1 \leq S \leq p \Rightarrow \frac{1}{p} \leq \frac{S}{p} \leq \frac{p}{p} \Rightarrow \frac{1}{p} \leq E \leq 1.$$

Sellest järeldub, et paralleelse algoritmi A_{par} jooksutamine paralleelselt p protsessil saab olla mitte rohkem kui p korda kiirem kui algoritmi A_{par} jooksutamine ühel protsessil. Paralleelne algoritm saavutab suurima efektiivsuse, kui iga protsess teeb $\frac{1}{p}$ osa tööst. See tähendab, et teoreetiliselt on parim võimalik kiirendus lineaarne, $S = p$. Harva on aga saavutatav superlineaarne kiirendus, $S > p$. See on saavutatav kui teostatav paralleelarvutus mahub ära protsessorite vahemällu, samal ajal kui järjestikarvutus vahemällu ei mahtunud. Tavaliselt ei küündi paralleelse programmi kiirendus isegi lineaarse kiirenduseni, kuna parallelismi jaoks on tarvis hoida mälus teatud lisaruumi. Seda kasutatakse protsesside vaheliseks suhtlemiseks ja sünkroniseerimiseks või see on tekkinud ooteaegadest programmis, kuna töömaht on protsesside vahel erinevalt jaotunud.

2.2.1. Kasutamine numbrilistes näidetes

Nimetame **arvutuse tööajaks** aega, mis kulub vaid optsiooni väärtuse arvutamiseks. Nimetame **programmi tööajaks** aega, mis kulub kogu programmi tööks, v.a tulemuste kuvamiseks. Siia kuulub käsurea muutujate väärtustamine, lõimede ja protsesside loomine ning lõpetamine. Käesolevas töös vaatame peamiselt arvutuse tööaega. Seda seetõttu, et paralleliseerime vaid optsiooni väärtuse arvutust ning selle väliselt programmi ei paralleliseeri.

Kuigi leidub olukordi, kus paralleelselt arvutamine on kiirem arvutuse seisukohast, siis võib programmi järjestikune käivitamine olla kokkuvõttes kiirem ning seetõttu mingitel tingimustel mõistlikum kasutada. Selline olukord eksisteerib näiteks juhul kui paralleelsete lõimede/protsesside loomine arvutusega võrreldes töömahukas.

Kõiki kiirendusi käsitleme suhteliste kiirendustena, kuna võrreldakse paralleelse programmi järjestikust ja paralleelset käivitamist. Kiirenduste leidmisel kasutame arvutuse tööaegu. Seetõttu peamine fookus on arvutuste suhtelistel kiirendustel, kuid välja tuuakse ka märkimisväärne info programmi kiirenduse kohta, kui see leidub.

Keskendume vaid arvutuse efektiivsusele, kuna paralleliseeritud osad asuvad just optsiooni väärtuse arvutamise osas. Seetõttu programmi efektiivsuse uurimine ei anna lisainfot paralleelsete lähenemiste kohta.

3. Paralleelarvutused optsiooni hinna leidmise numbrilistes meetodites

Uurime, kas peatükis 1 kirjeldatud optsioonide hinna arvutamisele kuluvat aega on võimalik vähendada kasutades peatükis 2 kirjeldatud paralleelseid meetodeid. Selleks teostame programmeerimiskeeles C++ peatükis 1 kirjeldatud optsioonide hinna arvutamiseks paralleelsed programmid. Ajaliste tulemuste saamiseks käivitame paralleelsed koodid nii järjestikusel kui ka paralleelsel viisil. Loeme optsiooni hinna arvutamise edukalt paralleliseerituks, kui saavutame ajalise võidu ning arvutatud optsiooni hind on piisavalt täpne. Kontrollime arvutatud optsioonide hindade korrektsust järgmiselt:

- analüütiliste valemite (3) ja (4) abil;
- tavalise binoommeetodi abil;
- netikalkulaatorit kasutades;
- artiklist (Mehrdoust *et al.*, 2013), kus kontrollväärtusena kasutatakse arvutustulemusi, mis on saadud iteratsioonide arvuga $N = 10000$, kasutades keskmiste Monte Carlo meetodit (ingl k *Mean Monte Carlo method (MMC)*).

Analüütiliste valemite tulemustega kontrollime binoomvalemiga arvutatud optsiooni hinna väärtust. Tavalise binoommeetodiga kontrollime MC meetodiga arvutatud Ameerika optsiooni hinda. Netikalkulaatorist saadud väärtusi kasutame MC meetodiga arvutatud Aasia optsiooni hinna kontrolliks. MC meetodiga arvutatud mitme alusvaraga Euroopa optsiooni hinda võrdleme artiklis leitud hindadega. Käesolevas töös on arvutatud optsiooni hind piisavalt täpne, kui selle absoluutne erinevus kontrollväärtusest ei ületa $0,05$.

Kuna teostatud programmide tulemuste (vead, kiirendused, efektiivsused) esmastel uurimistel selgus, et ostu- ja müügioptsioon käitusid teineteisega analoogselt, siis töös tuuakse välja vaid tulemused ostuoptsiooni korral. Samuti tehti arvutused, kus võimalik, erinevate ajasammude ja alusvarade arvuga. Ka nende eristuste tulemused olid analoogsed, mistõttu käsitleme vaid ühte ajasammude/alusvarade arvu väärtust.

Lihtsuse mõttes kasutatakse samu väärtusi kõikide optsioonide, v.a mitme alusvaraga Euroopa

optsiooni, korral:

$$S_0 = 100;\ E = 110;\ r = 0,02;\ \sigma = 0,75;\ T = 1.$$

Mitme alusvaraga Euroopa optsiooni hinda on ülaltoodud parameetrite korral keeruline kontrollida, mistõttu kasutatakse optsiooni hinna arvutamisel artiklis (Mehrdoust *et al.*, 2013) kasutatud väärtusi:

$$S_{0,i} = 100;\ E = 100;\ r = 0,1;\ \sigma_i = 0,2^1;\ T = 1;\ \rho_{i,j} = 0,5\,(i \neq j);\ w_i = \frac{1}{n};\ i,j = 1,\dots,n,$$

kus n on alusvarade arv, $S_{0,i}$ on i-nda alusvara hind ajahetkel $t = 0$, w_i on i-nda alusvara kaal korvis ning $\rho_{i,j}$ on Browni liikumiste vahelised korrelatsioonikordajad.

Väärtused millega arvutatud optsioonide hindu võrdleme on toodud tabelis 1. Veerus M olev väärtus tähistab Ameerika optsiooni korral simuleeritava alusvara hinna teekonna pikkust ning Aasia optsiooni korral ajasammude arvu, mida kasutatakse alusvara keskmise hinna simuleerimiseks. Veerus V on toodud optsioonide hinna väärtused, mida me kasutame arvutatud optsiooni hinna väärtuste võrdlemiseks.

Tabel 1. Kontrollväärtused ostuoptsioonide korral

Optsioon	**E**	**Alusvarasid**	**M**	**V**
Euroopa	110	1	-	26,61
Euroopa	100	4	-	11,90
Ameerika	110	1	200	26,61
Aasia	110	1	200	13,71

Näiteid lahendatakse järgmiste näitajatega virtuaalmasinas:

- operatsioonisüsteem (OS): GNU/Linux Ubuntu 20.04.2 LTS;
- Linux kerneli versioon: 5.4.0-72-*generic*;
- kompilaatori gcc versioon: 9.3.0;
- OpenMPI versioon: 4.0.3;

[1] Artiklis märgitakse, et kasutatakse $\sigma^2 = 0,2$, kuid arvutuste tegemine näitas, et artiklis on tegemist trükiveaga ning tegelikult kasutatakse $\sigma = 0,2$.

- OpenMP versioon: 4.5;
- muutmälu (ingl k *random acces memory (RAM)*): 64Gb;
- tuumade arv: 32.

Lähtudes Amdahli seadusest (Hesham El-Rewini, 2005 lk 54) saame väita, et mainitud arvutis on teoreetiliselt parim 32-kordne kiirendus.

Kõik kasutatud koodid ja arvutustulemused on kättesaadavad avalikust repositooriumist https://github.com/moledoc/parcompfin.

3.1. Euroopa optsiooni hinna leidmine binoomvalemiga

Nagu punktis 1.3 mainitud, siis tavalist binoommeetodit on keeruline paralleliseerida. Seetõttu uurime vaid binoomvalemi paralleliseerimist. Esmalt kontrollime binoomvalemiga leitud optsiooni hinna täpsust. Näeme jooniselt 6, et kõikidel teostatud koodidel, olenemata paralleelsusastmest, on väga väikesed arvutusvead: absoluutne arvutusviga jääb alla seatud $0,05$ piiri. Saame lugeda täpsuse kriteeriumi täidetuks.

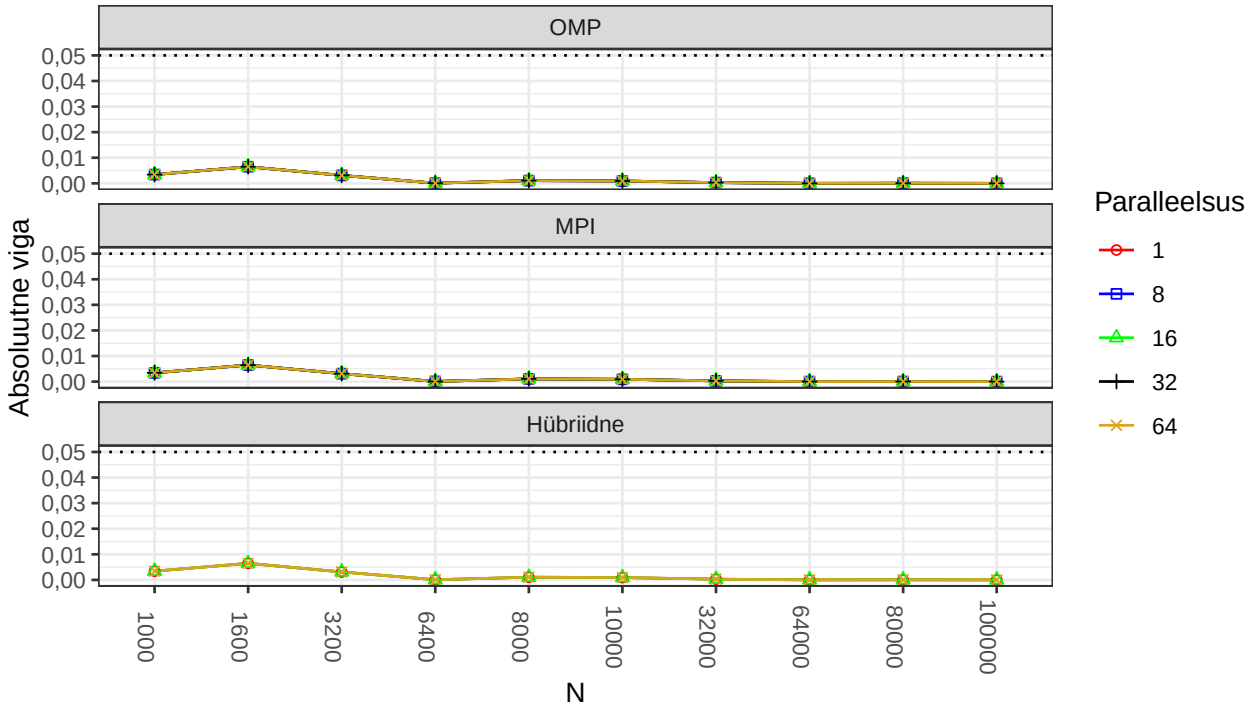
Edasi uurime kas paralleelsel käivitamisel saavutatakse ajaline võit. Alustame arvutuse tööaegade uurimisega. Näeme jooniselt 7, et kõik paralleelsed meetodid saavutavad ajalise võidu, kui iteratsioonide arv on piisavalt suur. Kui järjestiku käivitatud koodi tööaeg iteratsioonide arvuga $N = 100000$ on veidi üle 150 sekundi, siis näiteks OMP lahendusega arvutatakse 8 lõimega sama iteratsioonide arvu korral optsiooni hind ligikaudu 25 sekundiga. See tähendab, et saavutatakse ligikaudu 6-kordne suhteline kiirendus. Tuletame meelde, et suhteline kiirendus on suhe

$$S = \frac{T_{ser}}{T_{par}},$$

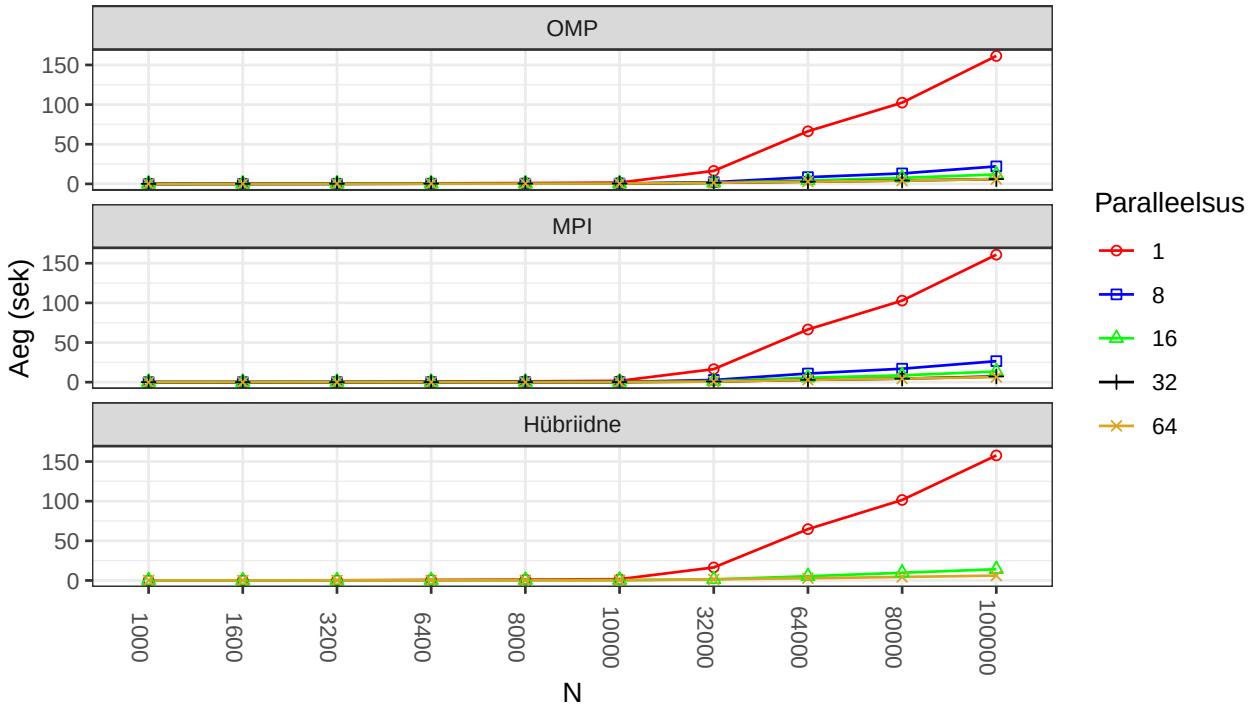
kus T_{ser} on järjestiku ning T_{par} on paralleelselt käivitatud koodi tööaeg.

Kuigi jooniselt 7 saame aimduse kui suure ajalise võidu saavutame, siis täpsema ülevaate saamiseks uurime joonist 8, millel on kujutatud arvutuse suhtelised kiirendused.

Näeme, et OMP ja hübriidlahendusega ei saavutata häid kiirendusi, kui iteratsioonide arv on väiksem. MPI lähenemise korral näeme arvutuses kiirendusi ka väiksemate iteratsioonide



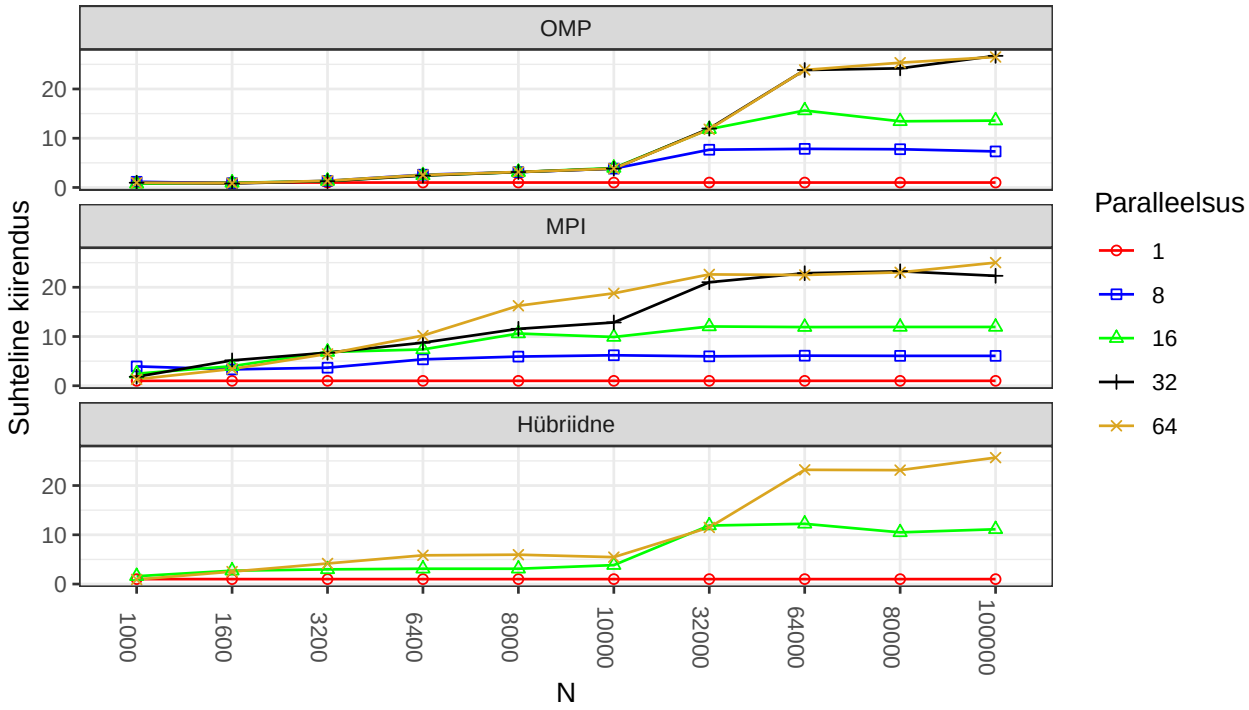
Joonis 6. Binoomvalemi arvutusvead Euroopa ostuoptsiooni hinna arvutamisel; punktiiriga tähistatud absoluutne viga $0,05$



Joonis 7. Binoomvalemi arvutuse tööajad Euroopa ostuoptsiooni hinna arvutamisel

arvu korral. Kui iteratsioonide arv on piisavalt suur, saavutavad kõik paralleelsed meetodid arvutuses sarnaseid kiirendusi. Samuti paneme tähele, et toimub kiirenduste koondumine ehk mingist iteratsioonide arvust alates ei saavutata paralleelsete meetoditega enam paremaid kiirendusi.

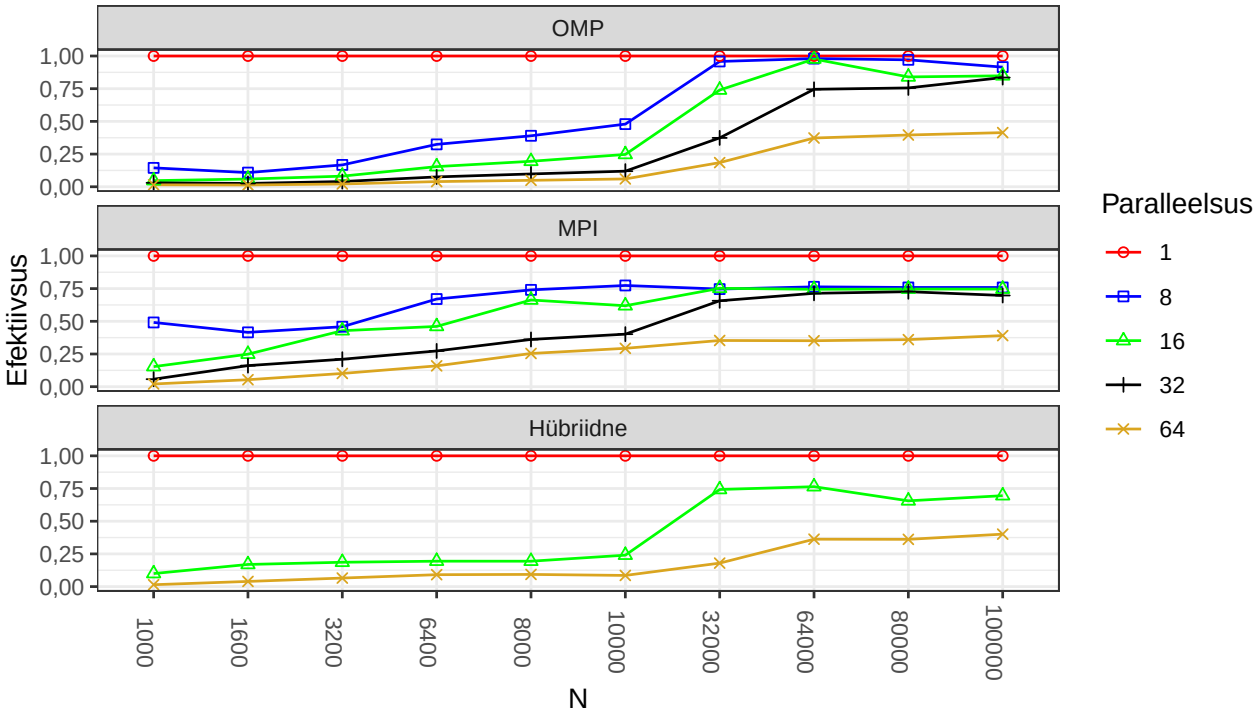
Jooniselt 8 paneme veel tähele seda, et 32 protsessi/lõime kasutamine annab meile märgatavalt parema kiirenduse võrreldes 16 protsessi/lõime kasutamine. See-eest 64 protsessi/lõime kasutamine 32 asemel märgatavat võitu ei anna. Seda nii OMP kui ka MPI lahenduse korral. Tulemus on loogiline, kuna arvutused tehti arvutis, millel on 32 tuuma. Kuna aga tuumadel on hüperhargtöötluse võimekus, oli lootus, et täiendavate protsesside/lõimede kasutamine annab täiendavat võitu, mida siin ei täheldata.



Joonis 8. Binoomvalemi arvutuse suhtelised kiirendused Euroopa ostuoptsiooni hinna arvutamisel

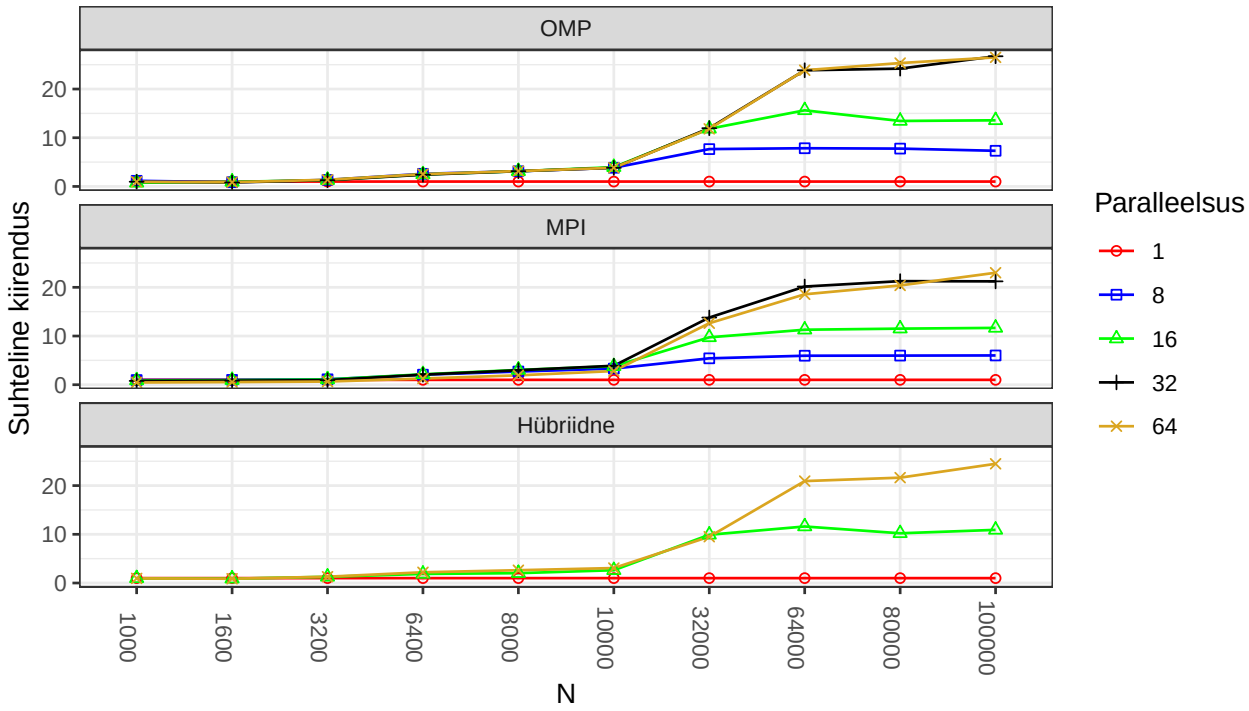
Juhime tähelepanu sellele, et kui OMP programmi käivitada 8 või 16 lõimega, siis saavutatud kiirendused lähenevad kasutatud lõimede arvule. Seetõttu on ootuspärane, et nendel juhtudel saavutatakse ligikaudu lineaarne kiirendus. Seda saame kontrollida arvutuse efektiivsuse jooniselt.

Jooniselt 9 näeme, et tõepoolest OMP lahendus saavutab ligikaudu lineaarse kiirenduse kui kasutatakse 8 või 16 lõime. Samuti paneme tähele, et kui kasutatakse 32 lõime, saavutab OMP programm suuremate iteratsioonide arvu korral hea efektiivsuse (nimetame heaks efektiivsuseks $E \geq 0,75$). Seevastu, kui kasutatakse 64 lõime, siis on paralleelse programmi töö vähem efektiivne. See tähendab, et leidus rohkem ajahetki, kus mõni lõim oli tegevuseta. See võib tuleneda sellest, et kuna arvutil on 32 tuuma, siis mõni lõim pidi ootama järjekorras, et saaks tuumas tööd teha. Paralleelsusastme 64 korral näeme MPI ja hübriidlahenduse korral analoogset käitumist OMP lahendusele. MPI lahenduse korral näeme, et kui kasutati 32 protsessi või vähem, siis saavutatakse enam-vähem hea efektiivsus. MPI lahendusega lineaarse kiirenduseni ei jõuta.



Joonis 9. Binoomvalemi arvutuse efektiivsused Euroopa ostuoptsiooni hinna arvutamisel

Lõpetuseks uurime, kui palju annavad paralleelsed lähenemised ajalist võitu programmi tööaja mõttes. Näeme jooniselt 10, et MPI ja hübriidsel lahendusel, võrreldes joonisega 8, vähenesid suhtelised kiirendused väikesemate iteratsioonide arvu korral, kuid OMP lahendusel jäid kiirendused sisuliselt samaks. Suuremate iteratsioonide arvu korral jäid OMP ja hübriidse lahenduse kiirendused suhteliselt samaks, kuid MPI kiirendused vähenesid veidi.



Joonis 10. Binoomvalemi programmi suhtelised kiirendused Euroopa ostuoptsiooni hinna arvutamisel

Selleks, et aru saada, miks see nii on, uurime kui palju aega kulub paralleelsete protsesside/lõimede loomisele ja lõpetamisele. Tabelis 2 on välja toodud erinevate protsesside arvu loomisele ja lõpetamisele kulunud keskmised ajad ning palju igale üksikule protsessile keskmiselt aega kulus.

Tabel 2. MPI protsesside loomisele ja lõpetamisele kulunud keskmised ajad sekundites

Protsesse	**Keskmiselt kokku**	**Keskmiselt igale**
1	0,2680	0,2680
8	0,3158	0,0395
16	0,3405	0,0213
32	0,3894	0,0122
64	0,6100	0,0095

Näeme, et protsesside loomine on küllaltki ajakulukas tegevus. Paneme tähele, et mida rohkem protsesse on, seda rohkem aega kulub kõikide protsesside loomiseks, kuid iga üksiku protsessi loomisele kuluv keskmine aeg väheneb. Selle põhjal saame öelda, et saavutame programmi töös kiirenduse MPI ja hübriidses lähenemises, kui protsessidele kuluv aeg moodustub kogu arvutuse ajast väikese osa. Kuna väiksemate iteratsioonide arvu korral oli protsesside loomisele

ja lõpetamisele kuluv aeg suur võrreldes arvutusajaga, siis MPI ja hübriidlahenduse programmi suhtelised kiirendused ei ole nii head kui arvutuse suhtelised kiirendused.

Nüüd vaatame lõimedele loomisele ja lõpetamisele kuluvat aega. Selleks vaatame tabelit 3. Sellelt näeme, et lõimede loomine ja lõpetamine on võrreldes protsessidega väga kiire. Sarnaselt protsesside loomisele näeme, et iga üksiku lõime loomisele kulub keskmiselt vähem aega, kui lõimede arv suureneb. Erinevus on aga selles, et 1 lõime ja 64 lõime loomisele kuluv aeg ei erine teineteisest väga palju. Saame väita, et lõimede loomine on konstantsema ajaga kui protsesside loomine. Kuna lõimede loomisele ja lõpetamisele kuluv aeg on väga väike, siis see ei mõjuta programmi suhtelisi kiirendusi, mistõttu OMP lahenduse korral programmi ja arvutuse suhtelised kiirendused langevad kokku.

Tabel 3. OMP lõimede loomisele ja lõpetamisele kulunud keskmised ajad sekundites

Protsesse	**Keskmiselt kokku**	**Keskmiselt igale**
1	0,0000291	0,0000291
8	0,0000367	0,0000046
16	0,0000301	0,0000019
32	0,0000337	0,0000011
64	0,0000361	0,0000006

Kuna protsesside/lõimede loomisele/lõpetamisele kuluv aeg sõltub arvutist ja mitte arvutusmeetodist, siis võime eeldada, et kuluv aeg on sõltumata optsioonide hinna arvutusmeetodist analoogne tabelites 2 ja 3 väljatooduga. Seetõttu järgmiste tulemuste analüüsis me protsessi-de/lõimede loomisele ja lõpetamisele kuluvat aega enam eraldi ei käsitle.

Nägime, et nii OMP kui ka MPI lahendustega saavutatakse korrektsed optsiooni hinna väärtused. Veel nägime, et kasutades paralleelselt arvutamist, on võimalik saavutada suur ajaline võit võrreldes järjestiku käivitatud programmiga. Seetõttu väidame, et binoomvalemit on võimalik edukalt paralleeliseerida. Kui nüüd arvesse võtta programmi suhtelisi kiirendusi, siis OMP lähenemine on paralleliseerimiseks eelistatud. Seda seetõttu, et OMP lähenemisega saavutati paremad programmi suhtelised kiirendused kui MPI lähenemisega. Lisaks on binoomvalemit kergem OMP-ga paralleeliseerida.

3.2. Mitme alusvaraga Euroopa optsiooni hinna leidmine MC meetodiga

Jätkame MC meetoditega arvutatavate optsiooni hindade näidetega. Esiteks uurime, kas ja kui hästi on mitme alusvaraga Euroopa optsiooni hinna arvutamine paralleliseeritav.

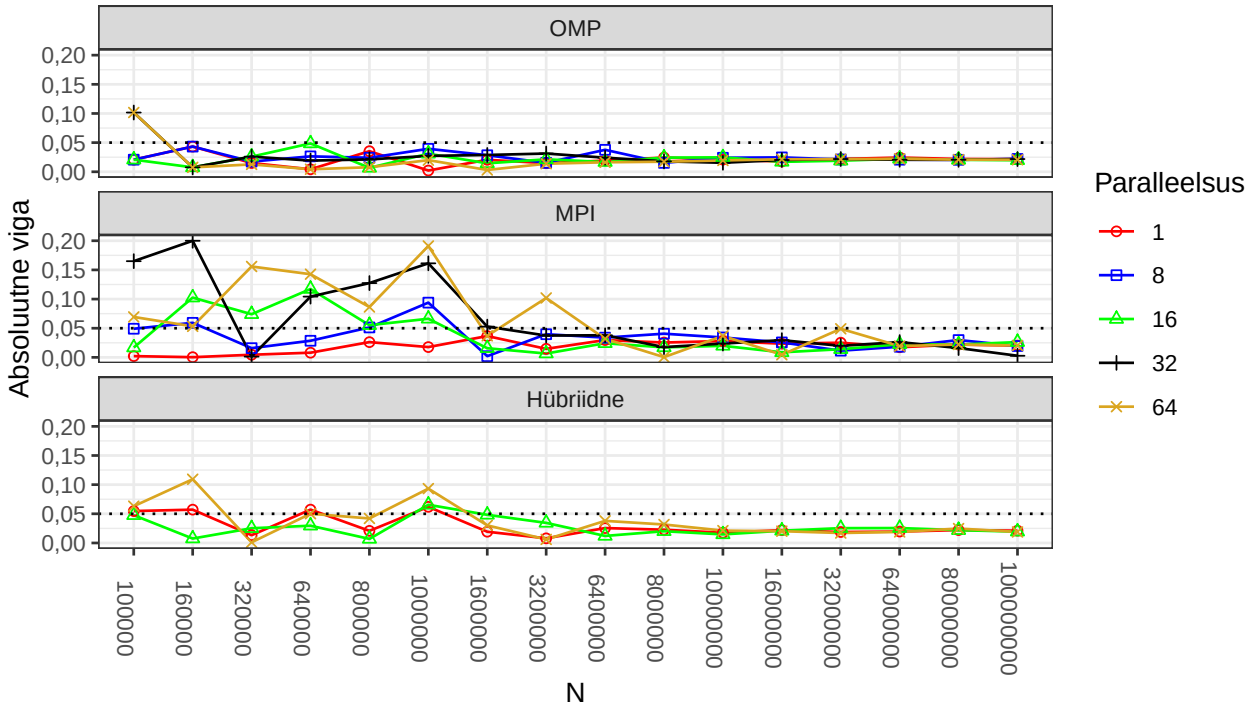
Nagu peatüki 3 alguses mainitud, siis kasutame teostatud programmide õigsuse kontrolliks artiklis (Mehrdoust *et al.*, 2013) väljatoodud tulemusi. Kuna artiklis tehti arvutused läbi 4 ja 10 alusvaraga, siis kasutame samu alusvarade arve oma arvutustes. Esmane tulemus näitas, et paralleelarvutuse tulemused ei sõltu oluliselt alusvarade arvust, mistõttu toome välja vaid nelja alusvaraga arvutatud tulemused.

Kuna C++ ei eksisteeri mugavat funktsiooni juhuarvude ükshaaval genereerimiseks mitmemõõtmelisest standardsest normaaljaotusest $N(\mathbf{0}, \Sigma)$, siis enne optsiooni hinna arvutamist genereerime valimi mainitud jaotusest ning kasutame selle väärtusi arvutuste tegemisel. Lihtsuse mõttes kasutame konstantseid Browni liikumise korrelatsioone kovariatsioonimaatriksis Σ. Samuti kasutame korvis olevate alusvarade jaoks lihtsuse mõttes võrdseid kaale. Programmi on lihtne muuta selliseks, et alusvarad on korvis erinevate kaaludega ning Browni liikumised on erinevalt korreleeritud.

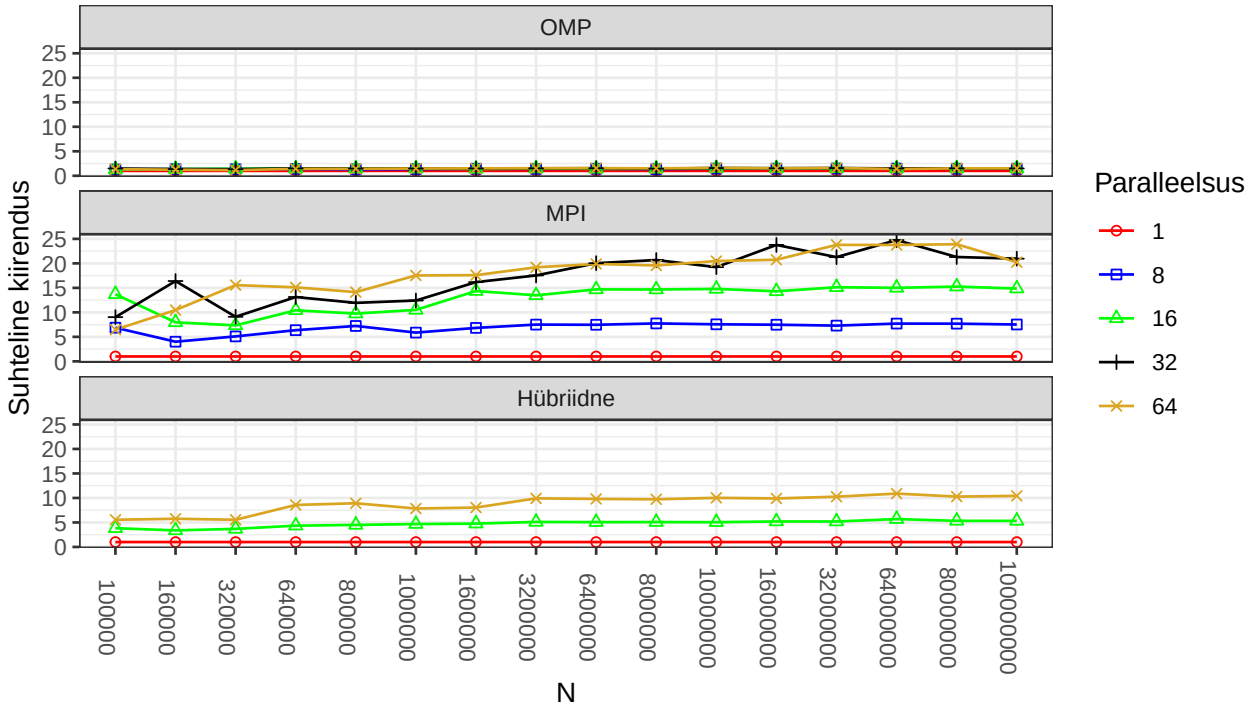
Enne ajaliste tulemuste vaatamist veendume, et teostatud koodidega arvutatud optsiooni hinnad on korrektsed. Jooniselt 11 näeme, et väiksemate iteratsioonide arvu korral on MPI lahenduse arvutusvead suuremad, kui töös lubatud. Iteratsioonide arvu suurenedes koonduvad arvutusvead töös seatud piiridesse. OMP ja hübriidse lahenduste korral näeme, et arvutusvead jäävad üldiselt lubatud vea piiridesse.

Binoomvalemi analüüsist teame, et tööaegade jooniselt on keeruline täpselt näha saavutatud ajalisi võite. Seetõttu uurime kohe arvutuse suhtelisi kiirendusi (joonis 12).

Näeme, et OMP lähenemine annab vaid konstantse paarikordse kiirenduse. See on tingitud sellest, et optsiooni hinna arvutamises oli ajamahukaim mitmemõõtmelisest normaaljaotusest juhuslike arvude genereerimine. Kuna seda ei suudetud OMP lähenemisega edukalt paralleliseerida, siis ei saavutatud ka suurt ajalist võitu. Saavutatud kiirendus tuleneb arvutustsükli



Joonis 11. MC meetodi arvutusvead nelja alusvaraga Euroopa ostuoptsiooni hinna arvutamisel; punktiiriga tähistatud absoluutne viga $0,05$



Joonis 12. MC meetodi arvutuse suhtelised kiirendused nelja alusvaraga Euroopa ostuoptsioonihinna arvutamisel

paralleliseerimisest.

MPI lähenemise korral arvutatakse igas protsessis mitmemõõtmelise normaaljaotusest valim ning kuna töömaht on protsesside vahel jaotatud, peab iga protsess palju väiksema valimi arvutama kui järjestiku käivitatud programm. Seetõttu saavutame MPI lahenduse korral ajalise võidu. Sellise lähenemise korral kasutatakse igas protsessis valimi genereerimise jaoks erinevat seemet, mis võib mõjutada väiksemate iteratsioonide arvu korral arvutustäpsust.

Suhteliste kiirenduste suuruse põhjal saame väita, et hübriidlahenduse ajaline võit on suuresti tingitud MPI lähenemise poolt. Kuna aga hübriidses lahenduses kasutatakse vähem paralleelseid protsesse kui MPI lahenduses, siis ei saavutata nii häid kiirendusi kui MPI lahendusega.

Kui MPI lahendusega kasutatakse arvutuses 8 või 16 protsessi, siis nende kiirendused koonduvad vastava protsesside arvu kordseks kiirenduseks. Binoomvalemi analüüsis nägime, et sellisel juhul saavutatakse ligikaudu lineaarne kiirendus. Saame järeldada, et MPI lahendusega saavutatakse ligikaudu lineaarne kiirendus, kui kasutatakse 8 või 16 protsessi. Samuti täheldame, et 32 protsessi kasutamine annab märgatavat võitu vähemate protsessi kasutamise ees, kuid rohkem kui 32 protsessi kasutamine täiendavat võitu kiirendustes ei anna.

Arvutuse suhteliste kiirenduste joonise põhjal saame öelda, et käesoleva töö raames ei suudetud OMP ja hübriidse lähenemisega mitme alusvaraga Euroopa optsiooni hinna arvutamist MC meetodil edukalt paralleliseerida. Kui OMP lähenemises suudetaks edukalt ära paralleliseerida mitmemõõtmelisest normaaljaotusest valimi genereerimine, siis leidub võimalus, et ka OMP lähenemisega saavutatakse häid suhtelisi kiirendusi. Kui lähtuda lisaks arvutuse suhtelistest kiirendustest ka arvutusvigadest (joonis 11), siis saame väita, et mitme alusvaraga Euroopa optsiooni hinna arvutamist käesolevas töös suudeti edukalt paralleeliseerida MPI lähenemisega, kui iteratsioonide arv N on suur.

3.3. Ameerika optsiooni hinna leidmine MC vähimruutude meetodiga

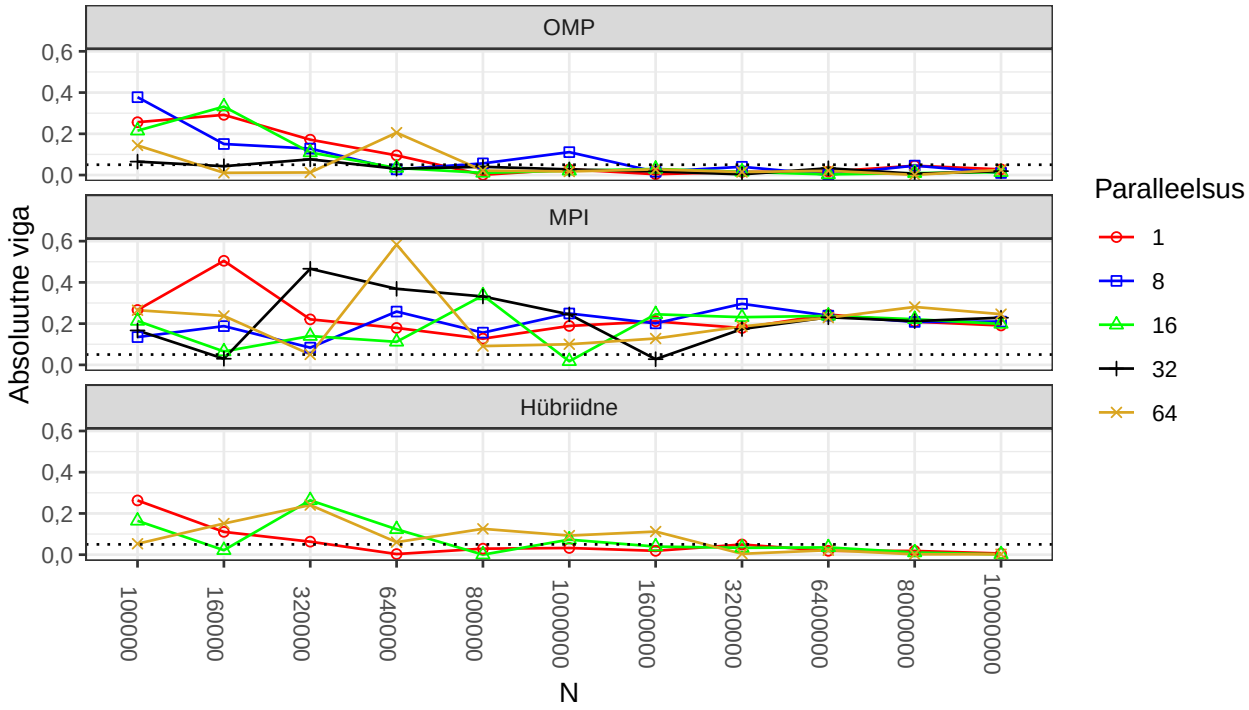
Edasi uurime, kas MC vähimruutude meetodit Ameerika optsiooni hinna leidmiseks on võimalik edukalt paralleliseerida. Näiteülesanne lahendati juhul, kui alusvara hinna teekonnas oli $M = 200$ ja $M = 1000$ sammu. Ainuke erinevus $M = 200$ ja $M = 1000$ kasutamise vahel oli see, et kui kasutati teekonna pikkuseks $M = 1000$ sammu, siis kõik arvutused kestsid ligikaudu 5 korda kauem. Seetõttu, kuna kiirendustega seotud tulemused olid $M = 200$ ja $M = 1000$ korral analoogilised, siis tuuakse välja vaid $M = 200$ tulemused.

Selleks, et vähendada arvutatud optsiooni hinna hajuvust ning alusvara teekonna arvutamiseks vajaminevate juhuslike väärtuste arvu, kasutame antiteetilisi muutujaid. Täpsemalt kasutame simulatsioonis $\left(\frac{N}{2} + \frac{N}{2}\right) \cdot M$ antiteetilist normaaljaotusega $N(0, \sqrt{\Delta t})$ juhusliku suurust. Antiteetilised muutujad on juhuslikud suurused, mis on negatiivselt korreleeritud ning nende kasutamise peamine eesmärk ongi tavaliselt MC meetodi hinnangu varieeruvuse vähendamine (Gentle, 2003, lk 246).

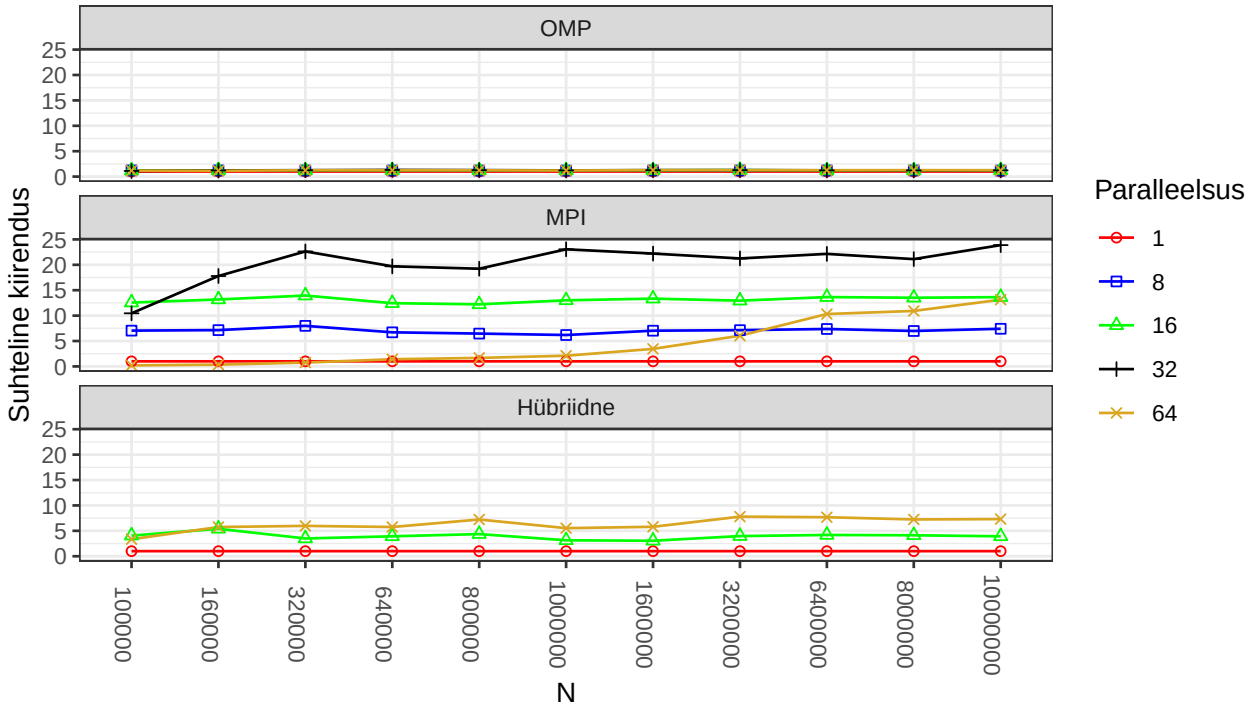
Vähimruutude meetod teostati C++'s mitmel erineval moel. Katsetati nii standardset vektorite kogumit (*std::vector<>*) kui ka täienduse *Eigen* vektoreid ja maatrikseid. Kuna selle töö raames saavutatakse paremad ajalised võidud ja arvutustäpsused standardse vektorite kogumiga, siis on ka selle lähenemisega saadud tulemused välja toodud.

Esmalt vaatame, kas teostatud koodid arvutavad optsiooni hinda korrektselt. Jooniselt 13 näeme, et OMP ja hübriidlahendusega arvutatud optsiooni hinna arvutusvead koonduvad iteratsioonide arvu kasvades töös lubatud piiridesse. Huvitaval kombel koonduvad MPI lahenduse arvutusvead ligikaudu $0,2$-ks. Selle põhjal ei täida MPI lahendus töös seatud arvutustäpsuse kriteeriumit.

Nagu ka mitme alusvaraga Euroopa optsiooni korral, alustame kohe arvutuse suhteliste kiirenduste uurimist. Sarnaselt mitme alusvaraga Euroopa optsiooni hinna arvutamisele, ei saavutata ka Ameerika optsiooni hinna arvutamises OMP ja hübriidlahendusega häid kiirendusi. Põhjus, miks me OMP lahendusega kiirendust ei näe, on sarnane mitme alusvaraga



Joonis 13. MC vähimruutude meetodi arvutusvead Ameerika ostuoptsiooni hinna arvutamisel, kasutades $M = 200$ sammu alusvara hinna simulatsioonis; punktiiriga tähistatud absoluutne viga $0,05$



Joonis 14. MC vähimruutude meetodi arvutuse suhtelised kiirendused Ameerika ostuoptsiooni hinna arvutamisel, kasutades $M = 200$ sammu alusvara hinna simulatsioonis

Euroopa optsiooni hinna arvutamise juhule. Teostatud järjestikuses programmis kulus veidi rohkem kui pool arvutusaega alusvara hinna teekondade simuleerimisele. Kuna selle töö raames OMP-ga teekondade simuleerimist edukalt paralleeliseerida ei suudetud, siis ei ole ka tulemustes head kiirendust näha. Lisaks, kuna optsiooni hinna arvutamises oli palju üksteisest sõltuvaid arvutusi, siis OMP lahendusega ei suudetud väga efektiivselt ka optsiooni hinna arvutust paralleliseerida. Kui OMP lähenemisega suudetaks ära paralleliseerida alusvara hindade simuleerimine, siis oleks lootust saavutada ajaline võit.

Seevastu oli MPI lähenemisega alusvara hindade simuleerimise paralleliseerimine palju lihtsam, mistõttu näeme ka paremaid kiirendusi. Alusvara hindade simuleerimise paralleliseerimise lähenemine on analoogne mitmemõõtmelisest normaaljaotusest valimi arvutamisega, mida nägime mitme alusvaraga Euroopa optsiooni hinna arvutamisel.

Paneme tähele, et MPI lahendusega saavutatakse ligikaudu lineaarne kiirendus, kui kasutatakse protsesside arvu 8 või 16 programmi käivitamisel. Huvitav on tähelepanek, et 64 protsessi kasutamine on väiksemate iteratsioonide korral isegi aeglasem kui programmi järjestiku käivitamine. See on tingitud sellest, et 64 protsessi loomisele ja lõpetamisele kulub palju aega võrreldes arvutuse enda ajaga. Lisaks veel see, et kuna Ameerika optsiooni hinna arvutamisel on tarvis rohkem protsesside vahelist suhtlust kui näiteks mitme alusvaraga Euroopa optsiooni hinna arvutamise korral, siis kulub ka suhtlemisele rohkem aega.

Kui arvesse võtta nii suhtelisi kiirendusi kui ka arvutusvigu, siis töös seatud kriteeriumite põhjal Ameerika optsiooni hinna arvutamist MC vähimruutude meetodil edukalt paralleliseerida ei suudetud. Tingimusel, et suudame MPI lahendusega saavutada töös lubatud arvutusviga, saaksime väita, et Ameerika optsiooni hinna arvutamine MC vähimruutude meetodil on edukalt paralleliseeritav MPI lähenemisega.

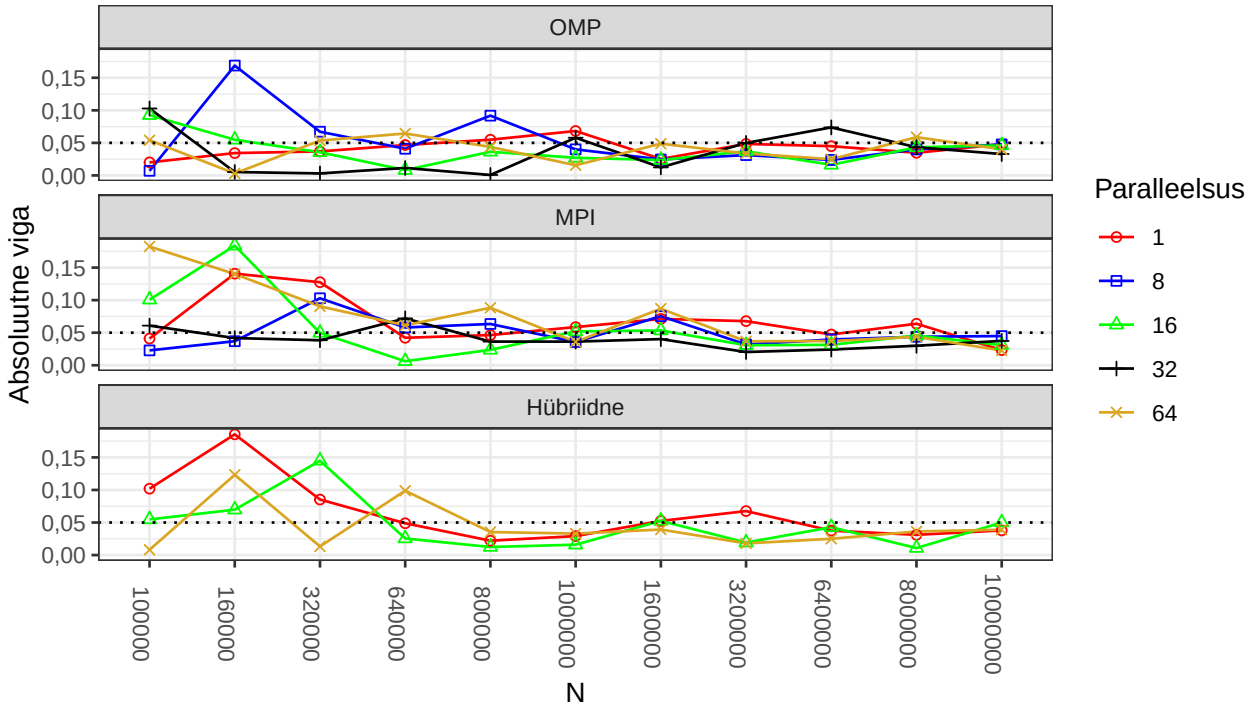
Peatüki alguses sai mainitud, et Ameerika optsiooni hinna arvutamist üritati lahendada kasutades mitut erinevat C++ lähenemist. Töös väljatoodud tulemused on saadud standardset vektorite kogumit kasutades. Selle lähenemise korral oli võimalik edukamalt MPI lähenemist paralleliseerimiseks kasutada. Seda seetõttu, et selle lähenemisega oli võimalik mõistlikult ära kasutada valemeid (16), mis lihtsustas protsesside vahelist suhtlust. Kui katsetati täiendust *Eigen*, siis vektorite ja maatriksitega töötamine muutus lihtsamaks, kuid protsesside vaheline

suhtlus muutus keeruliseks ning ajaliselt kulukaks. Autor näeb potentsiaali OMP lähenemisel *Eigen* täiendust kasutades. Nagu eelnevalt mainitud, siis OMP lahendus saab head ajalist võitu saavutada vaid siis, kui alusvara hinna simuleerimine on edukalt paralleliseeritud.

3.4. Aasia optsiooni hinna leidmine MC meetodiga

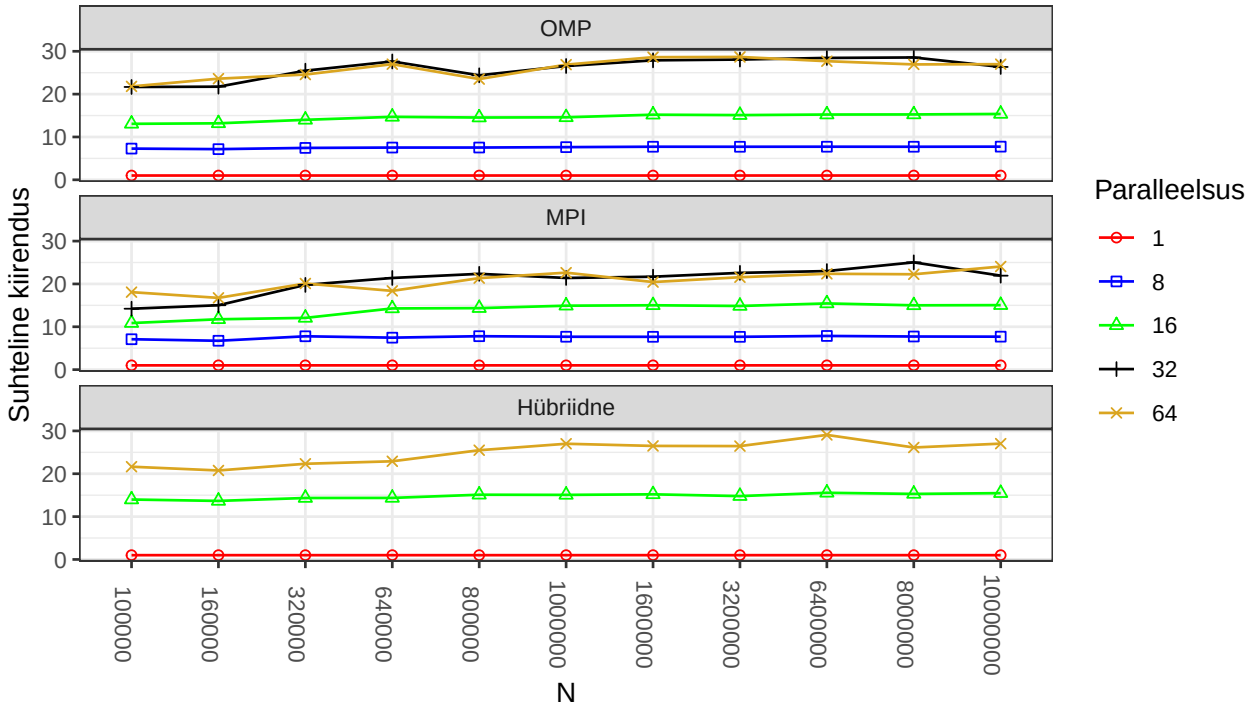
Viimaseks vaatame Aasia optsiooni hinna arvutamist, milles lähtume punktist 1.4.4. Arvutused tehti läbi juhtudel, kus kasutati $M = 200$ ja $M = 1000$ ajasammu keskmise alusvara hinna arvutamisel. Kuna kiirenduse tulemused olid mõlemal juhul analoogilised, tuuakse välja vaid tulemused $M = 200$ korral. Kui kasutati $M = 1000$ ajasammu Aasia optsiooni hinna arvutamisel, siis kõik tööajad olid ligikaudu 5 korda suuremad. Sama käitumist nägime ka Ameerika optsiooni hinna arvutamisel.

Kõigepealt uurime kas teostatud koodidega saadakse korrektsed optsiooni hinnad. Jooniselt 15 näeme, et kõikide paralleelsete meetodite korral koonduvad arvutusvead iteratsioonide arvu suurenedes töös lubatud piiridesse.



Joonis 15. MC meetodi arvutusvead Aasia ostuoptsiooni hinna arvutamisel, kasutades $M = 200$ ajasammu väärtust keskmise alusvara hinna leidmisel; punktiiriga tähistatud absoluutne viga $0,05$

Sarnaselt Euroopa ja Ameerika optsiooni hinna arvutamise uurimisel, vaatame ka siin järgmisena arvutuse suhtelisi kiirendusi (joonis 16). Sellelt näeme, et saavutame kõikide paralleelsete meetoditega väga head kiirendused. Näeme, et kui kasutame 8 või 16 protsessi ja/või lõime, siis saavutatakse OMP ja MPI lahendustega lineaarsed kiirendused. Samuti, OMP lahenduse korral saavutatakse 32 lõime korral peaaegu lineaarne kiirendus.



Joonis 16. MC meetodi arvutuse suhtelised kiirendused Aasia ostuoptsiooni hinna arvutamisel, kasutades $M = 200$ ajasammu väärtust keskmise alusvara hinna leidmisel

Põhjus, miks saavutame nii MPI kui ka OMP lahendusega kiirendused on selles, et mõlema lähenemise korral suudeti edukalt optsiooni hinna arvutamise algoritm paralleliseerida. Lisaks mainime ära, et algoritmi paralleliseerimine nii OMP kui ka MPI lähenemisega ei olnud keeruline.

Kokkuvõttes saavutame nii MPI kui ka OMP lahendusega täpsed optsiooni hinnad suure ajalise võiduga. Selle põhjal saame väita, et Aasia optsiooni hinna arvutamine suudeti töö käigus edukalt paralleliseerida nii OMP kui ka MPI lähenemisega.

Kuigi mõlema paralleelse meetodiga suudeti optsiooni hinna arvutamine edukalt paralleliseerida, siis tekib küsimus, miks OMP lahendus saavutab paremaid kiirendusi. Selle põhjuseks

on paralleelsete protsesside/lõimede omavahelisele suhtlusele kuluv aeg. Kuna OMP lõimede omavaheliseks suhtluseks kulub vähem aega kui MPI protsessidel, siis seetõttu saavutame OMP lahendusega ka paremaid kiirendusi.

Kokkuvõte

Töös uurisime, kas optsiooni hinna arvutamist numbriliste meetoditega on võimalik paralleliseerida. Lähemalt uurisime binoomvalemi ja Monte Carlo meetodiga optsiooni hinna paralleelselt arvutamist. Binoomvalem on binoommudeli korral optsiooni hinna arvutamise valem, mis annab võrdse tulemuse binoommeetodiga leitud hinnale. Monte Carlo meetod on juhuslike arvude kasutamine simulatsioonides matemaatilise avaldise väärtustamiseks.

Paralleelarvutamist teostasime lähtudes mälumudelite jaotusest: jagatud, hajus- ja hübriidmälu. Keskendusime peamiselt jagatud ja hajusmäluga mudelitele. Numbriliste meetodite paralleliseerimiseks kasutasime programmeerimiskeelt C++. Kasutasime jagatud mäluga programmeerimiseks C++ täiendust OpenMP (OMP) ning hajusmäluga programmeerimiseks *openmpi* implementatsiooni sõnumite edastamise liidesest (ingl k *message passage interface (MPI)*).

Töö käigus leidsime, et optsiooni hinna arvutamist on võimalik edukalt paralleliseerida, v.a tavalise binoommeetodi korral. Parimad kiirendused saavutasime, kui kasutasime arvutis olevate tuumadega arvuga võrdset paralleelsusastet.

Binoomvalemit Euroopa optsiooni hinna arvutamiseks suutsime edukalt paralleliseerida nii OMP kui ka MPI lähenemisega. Kuna OMP lähenemine on lihtsam ja annab paremaid kiirendusi kui MPI lahendus, siis binoomvalemit on eelistatum OMP lähenemisega paralleliseerida.

Monte Carlo meetodit mitme alusvaraga Euroopa korvoptsiooni hinna arvutamiseks suutsime selle töö raames vaid MPI lähenemisega edukalt paralleliseerida. OMP lähenemisega ei suutnud me ära paralleliseerida mitmemõõtmelisest standardsest normaaljaotusest valimi genereerimist, mis oli ajaliselt kulukaim tegevus arvutuses. Kui suudaksime selle OMP lähenemisega ära paralleliseerida, siis oleks ka OMP lähenemisega lootust saavutada arvutuses kiirendusi.

Monte Carlo vähimruutude meetodit Ameerika optsiooni hinna arvutamiseks suutsime edukalt paralleliseerida vaid MPI lähenemisega tingimusel, et arvutusvea suudame viia väiksemaks kui $0,05$. OMP lähenemisega ei suutnud me Ameerika optsiooni hinna arvutamises ära paralleliseerida alusvara teekonna simuleerimist, mis oli üks aeganõudvamaid tegevusi

arvutuses. Seetõttu ei olnud me Ameerika optsiooni hinna arvutamise paralleliseerimises OMP lähenemisega edukad. Kui OMP lähenemisega õnnestub alusvara hinna teekondade paralleelne simuleerimine, siis leidub võimalus, et OMP lähenemisega on Ameerika optsiooni hinna arvutamine võimalik edukalt paralleliseerida.

Monte Carlo meetodit fikseeritud täitmishinnaga Aasia optsiooni hinna arvutamiseks suutsime töö raames edukalt paralleliseerida nii OMP kui ka MPI lähenemisega. Sarnaselt binoomvalemiga on eelistatud OMP lähenemine, kuna sellega oli paralleliseerimine kergemini teostatav ning saadavad kiirendused on, võrreldes MPI omadega, paremad.

Kokkuvõtvalt saame öelda, et MPI lähenemisega olime rohkemates optsiooni hinna arvutamise paralleliseerimistes edukamad, kuid tihtipeale olid MPI lahenduse arvutusvead väiksemate iteratsioonide arvu korral suuremad. Nendel juhtudel, kus olime paralleliseerimises edukad OMP lähenemisega, täheldasime, et saavutame OMP lahendusega paremad kiirendused kui MPI lahendusega. Selle töö kontekstis ei täheldanud me hübriidse lähenemisega täiendavat kiirendust võrreldes OMP ja MPI lähenemistega.

Antud teemal järgmine samm võiks olla diferentsiaalmeetodite paralleliseerimise uurimine. Täiendavalt võiks uurida, et kuidas käesoleva töö raames MPI lähenemisega paralleliseeritud meetodites vähendada väiksemate iteratsioonide arvu korral arvutusvigu ning kuidas paralleliseerida mitmemõõtmelisest normaaljaotusest valimi ja alusvarade teekondade genereerimist OMP lähenemisega.

Viited

Gentle, J.E. (2003) *Random number generation and monte carlo methods*, 2nd ed ed., Statistics and computing. Springer.

Glasserman, P. (2003) *Monte carlo methods in financial engineering: V. 53*, 1st ed., Applications of mathematics. Springer.

Hesham El-Rewini, M.A.-E.-B. (2005) *Advanced computer architecture and parallel processing*, 1st ed., Wiley series on parallel and distributed computing. Wiley.

Kroese, D.P., Brereton, T., Taimre, T., & Botev, Z. (2014) Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, **6**, 386–392.

Lapeyre, B. & Temam, E. (2000) Competitive monte carlo methods for the pricing of asian options. *Journal of Computational Finance*, **5**, 39–57.

Longstaff, F. & Schwartz, E. (2001) Valuing american options by simulation: A simple least-squares approach. *Review of Financial Studies*, **14**, 113–147.

Mehrdoust, F., Vajargah, K., & Rahimi, A. (2013) Numerical simulation for multi-asset derivatives pricing under black-scholes model. *Chiang Mai Journal of Science*, **40**.

OpenMPI koduleht (2021). URL https://www.open-mpi.org/.

OpenMP juhend (2021). URL https://computing.llnl.gov/tutorials/openMP/#ProgrammingModel.

OpenMP koduleht (2021). URL https://www.openmp.org/.

Popuri, R., S. K. & Gobbert, M.K. (2017) Parallelizing computation of expected values in recombinant binomial trees. *Journal of Statistical Computation and Simulation*, **88(4)**, 657–674.

Rabenseifner, R. (2003) Hybrid parallel programming on hpc platforms. ed.

Roman Trobec, P.B., Boštjan Slivnik (2018) *Introduction to parallel computing: From algorithms to programming on state-of-the-art platforms*, 1st ed. ed., Undergraduate topics in

computer science. Springer International Publishing.

Seghiouer, H., Lidouh, A., & Nqi, F. (2011) Parallel monte carlo method for pricing asian options using trapezium scheme. *Applied Mathematical Sciences (Ruse)*, **5**.

Seydel, R.U. (2012) *Tools for computational finance*, 5th ed., Universitext. Springer.

Thomas Rauber, G.R. (2010) *Parallel programming: For multicore and cluster systems*, 1st Edition. ed. Springer Berlin Heidelberg.

Lihtlitsents lõputöö reproduktseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Meelis Utt,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Paralleelarvutused optsiooni hinna leidmise numbrilistes meetodites", mille juhendajad on Toomas Raus ja Eero Vainikko, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktis 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Meelis Utt
25.05.2021