Nikola Kušlaković

Poređenje efikasnosti simuliranja epidemije na centralnoj procesorskoj jedinici i grafičkoj procesorskoj jedinici

Ispitivan je model epidemije koji koristi metod zasnovan na agentima, baziran na SIR modelu, jednom od osnovnih epidemioloških modela. Analiziran je način na koji simulacija ovog modela može da se modifikuje, tako da se pored izvršavanja na procesoru, ona izvršava i na grafičkoj kartici. Model je koristio 15 parametara koji su mogli da se menjaju, a testiran je tako da se vidi kakve sve tipove epidemije može da simulira. Simulacije sa malim brojem agenata su se nešto brže izvršavale na procesoru, a simulacije sa izuzetno puno agenata znatno brže su se izvršavale na grafičkoj kartici. Pored toga, kada su simulacije koje su izvršavane na grafičkoj kartici koristile atomične funkcije, izvršavala su se znatno brže od simulacija koje nisu koristile te funkcije. Povećanjem broja agenata, razlika u predviđanjima simulacija sve manje zavisi od računarske komponente na kojoj se one izvršavaju.

Uvod

Epidemija je prirodna pojava kod koje dolazi do širenja neke bolesti ili zaraze među ljudima ili životinjama unutar populacije. Epidemije se često događaju unutar malih zajednica. Većina njih nisu opasne i brzo prestaju da se šire, ali se događaju i slučajevi gde širenje epidemije nije lako zaustaviti. Širenjem na više kontinenata, država i zajednica, epidemija dobija status pandemije, i tada se može smatrati velikom pretnjom. Na širenje epidemije utiču prirodni faktori kao što su: vrsta zaraze ili virusa koji se prenosi, način prenošenja virusa ili zaraze, stepen imuniteta populacije, brzina prenošenja i stopa smrtnosti. Pored prirodnih faktora veliku ulogu igraju i ljudski, odnosno društveni faktori. Držanje fizičke distance, blagovremeni odlazak kod lekara, informisanost ljudi, redovna lična higijena i vakcinacija su faktori koji drastično utiču na to koliko će se neka epidemija brzo širiti.

Predviđanje toka epidemije predstavlja ključni elemenat za njeno zaustavljanje. Koliko će predviđanje biti dobro, zavisi od same simulacije, odnosno od modela koji se koristi. Dva najčešća načina za simuliranje epidemija su: metoda zasnovana na agentima i metoda zasnovana na populaciji (Jaffry i Treur 2008). Kod metode zasnovane na agentima, svaka osoba (agent) unutar populacije se tretira kao nezavisna jedinka, dok kod metode zasnovane na populaciji, cela populacija je podeljena na grupe koje se jasno izdvajaju, i prate se interakcije između tih grupa. Većina epidemioloških simulacija se zasniva na *SIR* modelu (Kermack i McKendrick 1927). Ovo je jednostavan matematički model koji povezuje tri skupa: skup podložnih (S – susceptible), skup zaraženih (I – infected) i skup oporavljenih (R – recovered) jedinki. Postoji i modifikovan oblik ovog modela, koji uključuje i skup preminulih. Model funkcioniše tako što osobe kada se zaraze, oporave, izgube imunitet ili umru, prelaze iz jednog skupa u drugi.

Pored modela i načina simuliranja epidemije, bitno je i na kojoj računarskoj komponenti se vrši simuliranje. Centralna procesorska jedinica (CPU) i grafička procesorska jedinica (GPU) su dva uređaja na kojima je moguće vršiti simuliranje. Ova dva uređaja se najvećim delom razlikuju po svojoj arhitekturi, što znači da na različite načine obrađuju podatke. Konkretno, procesor bolje obrađuje podatke redno (sekvencijalno), dok grafička kartica bolje radi sa podacima koji mogu paralelno da se obrađuju.

Nikola Kušlaković (2002), Novi Sad, učenik 3. razreda Gimnazije „Isidora Sekulić" u Novom Sadu

MENTOR:
Stefan Nožinić, student Fakulteta tehničkih nauka Univerziteta u Novom Sadu

U nekim ranijm radovima (Shekh et al. 2015), koristi se metoda zasnovana na agentima i jednostavan oblik *SIR* modela (Kovac et al. 2018). U tim radovima se ne porede rezultati predviđanja simulacija koje su izvršavale na dve različite računarske komponente, već se samo porede brzine izvršavanja. Korišćeni su i sistemi sa više udruženih grafičkih i procesorskih jednica (Aaby et al. 2010; Zou et al. 2013).

Način simuliranja koji je korišćen u ovom radu inspirisan je pristupom zasnovanom na agentima, po uzoru na referentni rad (Shekh et al. 2015), gde imamo dve vrste agenata (odrasle i decu), pri čemu je svakom agentu dodeljeno mesto stanovanja i mesto gde odlazi na posao. Naš model predstavlja modifikaciju originalnog modela, tako da su određeni delovi uprošćeni, neki poboljšani, a uvedene su i neke nove stavke. Najbitnije izmene su sledeće: tok simulacije je drugačiji, imamo samo jednu vrstu agenata (odrasle), dodata je mogućnost da agenti posle određenog vremenskog perioda izgube imunitet i ponovo postanu podložni zarazi, osim mesta stanovanja i mesta gde agenti odlaze na posao imamo i popularne lokacije (npr. kafiće, restorane i klubove). Takođe, u našoj varijanti simulacije korišćen je drugačiji način generisanja slučajnih vrednosti, i bilo je moguće menjati veliki broj parametara simulacije (uvedeno je 15 različitih parametara), pa su na taj način mogli da se simuliraju različiti tipovi epidemija. U okviru analize modela upoređena su vremena izvršavanja simulacije na centralnoj procesorskoj jedinici i grafičkoj procesorskoj jedinici. Takođe, ispitana je validnost modela kao i razlike u predviđanjima simulacija u zavisnosti od toga da li se izvršavaju na centralnoj procesorskoj jedinici ili grafičkoj procesorskoj jedinici.

Metod

Opis simulacije

Simulacija je zasnovana na metodi agenata. Svakom agentu je dodeljeno mesto gde stanuje i mesto gde odlazi na posao. Pored toga svaki agent je imao atribut koji govori da li on želi da smanji kontakte sa drugim agentima, odnosno ovakav agent će samo da odlazi na posao i da se vraća kući. Agenti koji ne paze na svoje zdravlje će odlaziti na popularne lokacije, kao što su kafići, bioskopi ili klubovi.

Pre početka same simulacije potrebno je izgenerisati sve agente i dodeliti im lokacije na kojima žive i odlaze na posao. Nakon toga, određeni procenat agenata iz populacije biva prebačen u stanje zaraženosti. Ovi agenti će započeti širenje zaraze kroz populaciju, pa se nazivaju inicijalni slučajevi epidemije.

Jedan dan u simulaciji se odvija tako što bi ujutru svi agenti odlazili na posao i tamo provodili određeni vremenski period. Po završetku radnog vremena, disciplinovani agenti se odmah vraćaju kući, dok bi agenti koji se ne čuvaju zaraze odlazili na dve popularne lokacije i tamo provodili određeni vremenski period, i tek posle toga bi se vraćali svojim kućama gde bi ostajali do sledećeg jutra.

Agent se može zaraziti tako što na nekoj od lokacija ostvari interakciju sa drugim agentom koji je već zaražen. Verovatnoća da se agent zarazi je jedan od parametara sumilacije. Bitno je napomenuti da agent može imati interakciju sa samim sobom, čime se modeluju slučajevi kada neko na nekoj lokaciji provodi vreme sam. Takođe, agent može više puta sa istim agentom da stupi u interakciju. Na kraju svakog dana agent koji je zaražen ima izvesnu verovatnoću da umre. Nakon izvesnog vremena, agenti koji su zaraženi a nisu umrli, postaju imuni, a taj imunitet je vremenski ograničen, odnosno traje samo određeni vremenski period, tokom kojeg ne mogu da se zaraze, niti da zaraze druge agente. Kada mu prestane imunitet, agent postaje ponovo podložan zarazi. Na ovaj način agenti prelaze iz jednog skupa u drugi, a u slučaju da umru, više ne učestvuju ni u kakvim interakcijama sa drugim agentima.

Parametri simulacije

Simulaciju je moguće podešavati kroz 15 parametara, pri čemu je važno kako će parametri biti podešeni. Parametri su uvek bili podešeni tako da odgovaraju stvarnoj situaciji, osim u slučaju kada je trebalo testirati simulaciju, kada su korišćene ekstremne vrednosti parametara kako bi se utvrdila validnost simulacije. Na primer, nije se dešavalo sledeće: da u jednoj kući stanuje više od 15 agenata, da radno vreme traje manje od 5 sati ili da period kada je agent zaražen traje manje od 3 dana. Menjanje jednog parametra prouzrokuje da drugi parametar

Algoritam 1 Preudokod simulacije

```
GenerišiAgente()
PostaviAgentePoKućama()
InficirajAgente()
while VremeSimulacije < DužinaTrajanjaSimulacije do
    PremestiAgenteNaLokacije()
    while Brojač < BrojInterakcija do
        OstvariInterakcije()
        Brojač ← Brojač + 1
    Brojač ← 0
    VremeDana ← VremeDana + 1
    if VremeDana = DužinaTrajanjaDana then
        PromeniStatusAgentima()
        VremeSimulacije ← VremeSimulacije + VremeDana
        VremeDana ← 0
```

mora da se menja. Tako, povećanjem broja agenata povećava se i broj domova i broj radnih mesta. Parametri simulacije koje je bilo moguće menjati su sledeći:

- Parametri definisani celobrojnom vrednošću:
    1. Broj agenata
    2. Broj kuća gde agenti žive
    3. Broj radnih mesta
    4. Broj popularnih lokacija
    5. Broj interakcija koje agent ostvari po jednom satu
- Parametri definisani realnom vrednošću:
    6. Verovatnoća da se agent zarazi kada dođe u kontakt sa zaraženim agentom
    7. Verovatnoća da agent na kraju svakog dana simulacije umre (ako je zaražen)
    8. Procenat ljudi u populaciji koji su zaraženi na početku simulacije (inicijalni slučajevi)
    9. Procenat ljudi u populaciji koji osim kuće odlaze samo na posao
- Parametri definisani u danima:
    10. Dužina trajanja zaraze kod zaraženog agenta
    11. Dužina trajanja perioda imuniteta kod agenta koji je preležao zarazu
    12. Dužina trajanja cele simulacije
- Parametri definisani u satima:
    13. Dužina trajanja jednog dana simulacije
    14. Dužina trajanja radnog vremena
    15. Dužina trajanja vremena koji agenti provode na popularnim lokacijama

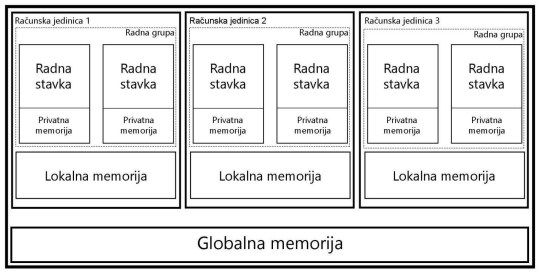
Simulacija na centralnoj procesorskoj jedinici

Algoritam simluacije nije morao da se prilagođava da bi se simulacija izvršavala na procesoru. Ceo program se prevodio u mašinski kod i predavao procesoru na izvršavanje; procesor je alocirao delove radne memorije (RAM) kako bi se privremeno čuvali podaci simulacije, a rezultati i neki bitni podaci o simulaciji beleženi su u trajnu memoriju, u tekstualni fajl. Svaka komanda na procesoru se izvršavala redno (sekvencijalno) i isključivo jedno procesorsko jezgro je bilo u upotrebi.

Bitno je napomenuti da je za većinu delova simulacije bilo potrebno izgenerisati određene slučajne brojeve. Konkretno, ako uzmemo jednog agenta i želimo da mu nađemo drugog agenta sa iste lokacije sa kojim će imati interakciju, to ćemo da uradimo tako što ćemo nasumično izabrati nekog agenta iz skupa svih agenata sa te lokacije. Korišćenjem slučajnih vrednosti, možemo približno da oponašamo situacije koje se događaju u stvarnosti.

Pošto računari ne mogu da izgenerišu „prave" slučajne brojeve, koristiti se generator pseudoslučajnih brojeva. Za realizaciju ovog projekta korišćen je Mersenne Twister pseudogenerator (Matsumoto i Nishimura 1998), pošto je jedan od najkorišćenijih i najpouzdanijih. Pseudogenerator se inicijalizuje na osnovu jedinstvene vrednosti koja predstavlja seme generatora (seed). U svim simulacijama izvršenim na procesoru bilo je korišćeno isto seme, zato što je korišćen samo jedan generator objekat.

Algoritam simulacije koji se izvršavao na procesoru bio je napisan u programskom jeziku C++, gde je za sve pomoćne strukture podataka bila korišćena standardna biblioteka ovog jezika.



Slika 1. Šematski prikaz grafičke procesorske jedinice.

Figure 1. chematic representation of a graphics processing unit.

Simulacija na grafičkoj procesorskoj jedinici

Arhitektura i način funkcionisanja grafičke kartice se dosta razlikuju od arhitekture i načina funkcionisanja procesora, pa je zbog toga bilo potrebno prilično modifikovati algoritam simulacije. Svaka grafička kartica poseduje više računskih jedinica (jezgara), od kojih svaka jedinica može da pozove više radnih stavki odjednom (work-item, thread). Radne stavke su organizovane u radne grupe, gde se svaka radna grupa izvršava na jednoj računskoj jedinici, paralelno. Broj računskih jedinica, broj radnih stavki koje mogu da se pozovu odjednom i broj radnih stavki unutar jedne radne grupe zavise od same grafičke kartice. Svrha svake radne stavke je izvršavanje specijalne funkcije koja se naziva jezgro (kernel). Jezgro je program koji se, kao i kôd namenjen za izvršavanje na procesoru, prevodi u mašinski kôd, i šalje grafičkoj kartici.

Memorija grafičke kartice je takođe drugačije struktuirana u odnosu na radnu memoriju sistema. Imamo tri tripa memorije: privatnu, lokalnu i globalnu memoriju. Globalna memorija je najveća, i njoj mogu da pristupe sve računske jedinice, što znači da joj može pristupiti i svaka radna stavka. Globalna memorija je zapravo radna memorija grafičke kartice. Lokalna memorija je memorija same računske jedinice, i ona je brža od globalne memorije, ali je zato manja po veličini. Njoj mogu samo da pristupe radne stavke iz iste radne grupe. Privatna memorija je memorija radnih stavki. Ona je najmanja po veličini i njoj može samo da pristupi radna stavka kojoj je ta memorija dodeljena.

Na slici 1 dat je šematski prikaz grafičke procesorske jedinice. Konkretno, u ovom slučaju imamo tri računske jedinice, od kojih svaka ima aktivne po dve radne stavke, koje ujedno pripadaju istoj radnoj grupi.

Prilikom adaptiranja simulacije za grafičku karticu najpre su generisani slučajni brojevi, za šta je, takođe, korišćen pseudogenerator Mersenne Twister. Pošto radne stavke ne smeju da koriste isti generator objekat, svaka od njih je morala da ima svoj generator koji je drugačije inicijalizovan, što znači da je svaki generator generisao drugačiji niz pseudoslučajnih brojeva. Na ovaj način ne dolazi do narušavanja konzistentnosti resursa, odnosno ne može se desiti da više radnih stavki upisuje podatke u istu memorijsku adresu (Feautrier 2011), te se na taj način ne remeti tok simulacije. Još jedan od načina rešavanja ovog problema, koji je takođe razmatran, jeste da se slučajni brojevi generišu unapred, a potom upišu u globalnu memoriju grafičke kartice. Ovaj način je brzo odbačen, jer se pokazao kao izuzetno neefikasan: zahteva jako puno memorije, usled čega ne mogu da se upišu drugi bitniji podaci simulacije.

Sledeći korak bio je adaptiranje algoritma simulacije, da bi se određeni delovi algoritma mogli izvršavati paralelno. Funkcije koje služe za premeštanje agenata na različite lokacije, ostvarivanje interakcija između agenata i menjanje statusa agentima (premeštanje agenata u skupove oporavljenih, zaraženih i podložnih) napisane su u obliku jezgra,

i na taj način su mogle da se izvršavaju paralelno. Najkompleksnija za implementaciju bila je funkcija koja premešta agente na različite lokacije. Za to su korišćene specijalne atomične funkcije, koje su sprečavale da više radnih stavki menja isti podatak u memoriji. One funkcionišu tako što „zaključaju" podatak. Dok jedana radna stavka menja podatak, druga stavka, koja želi da pristupi tom istom podatku, mora da čeka da prva završi sa radom nad tim podatkom. Funkcija za ostvarivanje interakcije između agenata prilagođena je tako da svakoj radnoj stavci dodeli po jednu lokaciju koju će da obrađuje. Pošto agenti koji nisu na istim lokacijama ne mogu da budu u kontaktu, radne stavke neće morati da pristupaju drugim lokacijama, osim onim koje su im dodeljene. Što se tiče funkcije koja menja status agentima, ona je izmenjena tako da svakoj radnoj stavci dodeli po jednog agenta. Na ovaj način je simulacija paralelizovana.

Simulacija na grafičkoj kartici tekla je na sledeći način:

1. Generisanje agenata i njihovo čuvanje u radnoj memoriji
2. Inficiranje agenata (inicijalni slučajevi epidemije)
3. Generisanje pseudo-slučajnih brojeva i njihovo čuvanje u radnoj memoriji
4. Prevođenje jezgra
5. Slanje svih podataka iz radne memorije sistema u globalnu memoriju grafičke kartice
6. Simuliranje na grafičkoj kartici (izvršavanje svih jezgara)
7. Prepisivanje rezultata i podataka o simulaciji iz globalne memorije grafičke kartice u radnu memoriju sistema

Bitno je napomenuti da je najsporiji deo simulacija bio pripremna faza, odnosno deo gde se generišu agenti i pseudogenerator objekti i dalje šalju u memoriju grafičke kartice. Prepisivanje iz globalne memorije u radnu memoriju je izuzetno sporo, zato je najbolje bilo poslati sve podatke odjednom.

Za pisanje programa, koji se većim delom izvršavao na grafičkoj kartici, bio je korišćen programski okvir OpenCL. Program zadužen za inicijalizaciju, slanje podataka i merenja se izvršavao na procesoru, dok su se jezgra (kerneli) izvršavala na grafičkoj kartici i bila su pisana u programskom jeziku baziranom na modifikaciji C99 standarda.

Rezultati i diskusija

Svi rezultati navedeni u ovom radu su dobijeni korišćenjem procesora AMD FX-8300 i grafičke kartice AMD R7 360.

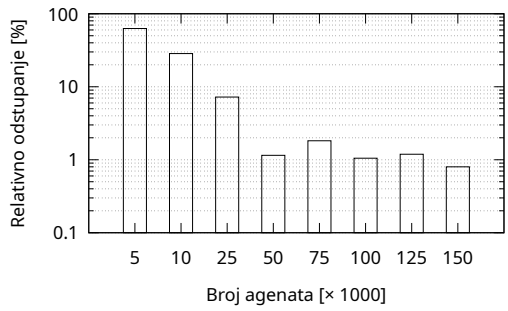
Analiza modela

Prvo je testiran sam model, kako bi se proverilo da li simulacija daje podatke koji imaju smisla. Parametri su menjani tako da mogu da se simuliraju različiti tipovi epidemija. Uspešno su simulirani sledeća četiri slučaja:

- Zaraza koja ima visoku smrtnost. Tada epidemija posle nekoliko dana brzo prestaje da se širi, jer zaraženi prebrzo umiru, pa ne dolazi do širenja zaraze.
- Zaraza kod koje je visoka stopa prenošenja i mala stopa smrtnosti, a agenti se brzo oporavljaju i imaju dugačak imuni period. Tada posle određenog vremenskog perioda dolazi do naglog rasta zaraženih i onda do naglog pada broja zaraženih (pik epidemije). Epidemija nakon određenog perioda prestaje da se širi, i potpuno nestaje iz populacije.
- Zaraza koja se nikada neće istrebiti iz populacije. Ovde je takođe visoka stopa prenošenja zaraze i mala stopa smrtnosti, ali je zato dugačak period zaraženosti agenta i imunitet je kraći. Ovde takođe dolazi do naglog rasta i pada broja novozaraženih, ali se posle određenog perioda zaraza ne gubi iz populacije već je svaki dan približno konstantan broj novozaraženih.
- Imamo više grupa unutar populacije i te grupe su međusobno izolovane, odnosno agenti iz jedne grupe ne ostvaruju interakcije sa agentima iz

druge grupe. Ako se zaraze samo agenti iz jedne grupe, druge grupe se neće zaraziti, jer nema mešanja grupa.

Bitno je istaći da u simulaciji nisu implementirane bolnice niti neki vidovi izolacije koje agenti moraju da poštuju kada se otkrije da su zaraženi. Kada se agent zarazi, on nastavlja sa svojim dnevnim obavezama, a oni agenti koji ne žele da smanje kontakte, odlaze i na popularne lokacije, bez obzira da li su zaraženi ili ne. Ovo je jedan od nedostataka modela. Pored toga, ne modeluju se ni ponašanja različitih starosnih grupa. Neće svi ljudi u populaciji da se bave istim poslom i da odlaze na iste lokacije u isto vreme. Deca će da odlaze u škole i tamo će provoditi vreme u smenama, odrasli će da odlaze na posao u različito vreme na različite lokacije, a stariji ljudi će većinu dana provoditi u kućama.



Slika 2. Relativno odstupanje simulacija izvršavanim na grafičkoj kartici u odnosu na simulacije izvršavane na procesoru. Vremenski period pokriven simulacijama iznosi 60 dana.

Figure 2. Relative deviation between simulations performed on the graphics card compared to simulations performed on the processor. The time period covered by the simulations is 60 days.

Razlike u predikcijama simulacija

Pošto se implementacija simulacije za grafičku karticu u nekim delovima dosta razlikovala od implementacije za procesor, bilo je potrebno međusobno uporediti rezultate dobijene simulacijama na ova dva uređaja.

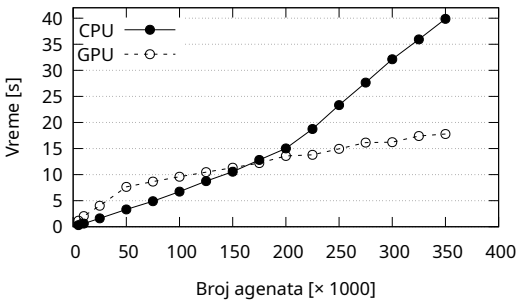
Na slici 2 prikazane su relativna odstupanja simulacija u zavisnosti od broja agenata. Relativna odstupanja su dobijena deljenjem razlike između vrednosti dobijenih na procesoru i na grafičkoj kartici vrednošću dobijenom na procesoru. Svaka od ovih simulacija, i na grafičkoj kartici i na procesoru, imala je iste parametre, osim broja agenata i broja lokacija, a sve su obuhvatale period od 60 dana.

Vidimo da su za simulacije sa 50 hiljada i više agenata razlike u predviđanjima vrlo male, ispod 2%. Ove razlike su posledica različitog načina generisanja slučajnih vrednosti koje se koriste u simulaciji. Kod simulacija koje su se izvršavale na procesoru korišćen je samo jedan pseudogenerator objekat, dok je kod simulacija na grafičkoj kartici korišćeno više pseudogeneratora. Simulacije izvršavane na grafičkoj kartici nisu uvek davale iste rezultate, pa je zbog toga simulacija sa potpuno istim parametrima pokrenuta više puta, i za konačnu vrednost rezultata je uzeta srednja vrednost. Ove varijacije u predviđanjima kod grafičke kartice posledica su korišćenja atomičnih funkcija, zbog kojih se svaki put drugačije raspoređuju podaci između radnih stavki. Osim toga, radna stavka neće uvek istom brzinom obrađivati iste podatke.

Vreme izvršavanja simulacije

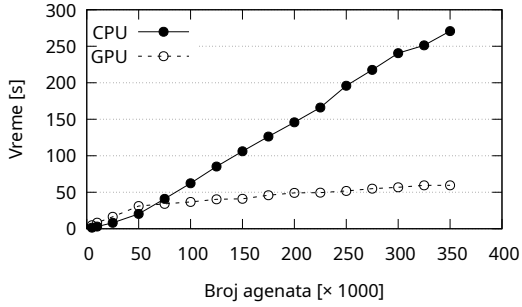
Vreme na procesoru mereno je od početka do kraja simulacije, pri čemu nije uračunato vreme potrebno da se generišu i pripreme podaci koji će se koristiti u simulaciji. Pritom, vreme izvršavanja simulacija na grafičkoj kartici mereno je nešto drugačije – uračunato je samo vreme potrebno da se izvrši svako jezgro, ali nije uzeto u obzir vreme potrebno da se podaci pošalju na memoriju grafičke kartice, kao ni vreme potrebno da se rezultati simulacije pročitaju nazad iz memorije. Osim broja agenata, parametar simulacije koji je takođe uzet u obzir je broj interakcija koje agent ostvari po satu (K).

Na slici 3 prikazano je vreme izvršavanja simulacija na grafičkoj kartici i procesoru. Ove simulacije su, za razliku od simulacija sa slike 2, obuhvatale period od 20 dana. Vidimo da se izvršavanje simulacije do otprilike 160 hlijada agenata brže odvija na procesoru nego na grafičkoj kartici, a da je za dosta veći broj agenata, vreme izvršavanja na grafičkoj kartici bitno manja od vremena izvršavanja na procesoru. Povećanjem broja agenata, vreme iz-



Slika 3. Zavisnost vremena izvršavanja simulacije od broja agenata u slučaju malog broja interakcija (K=4) za jedno procesorsko jezgro i za grafičku karticu. Vremenski period pokriven simulacijama iznosi 20 dana.

Figure 3. Execution time of simulation with little interactions (K=4) for different number of agents on one processor core and graphics card. The time period covered by the simulations is 60 days.



Slika 4. Zavisnost vremena izvršavanja simulacije od broja agenata u slučaju velikog broja interakcija (K=20) za jedno procesorsko jezgro i za grafičku karticu. Vremenski period pokriven simulacijama iznosi 20 dana.

Figure 4. Execution time of simulation with many interactions (K=20) for different number of agents on one processor core and graphics card. The time period covered by the simulations is 20 days.

vršavanja simulacije brže raste za procesor nego za grafičku karticu.

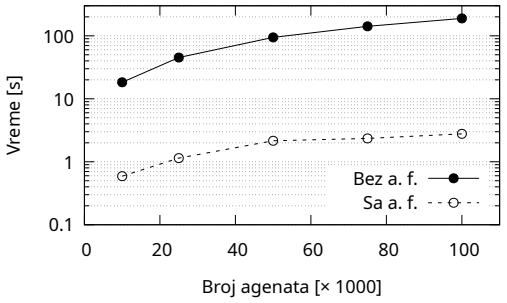
Kao i na slici 3, na slici 4 je prikazano vreme izvršavanja simulacija na grafičkoj kartici i procesoru, samo što je u drugom slučaju znatno povećan broj interakcija između agenata, tako što je broj interakcija povećan 5 puta (K=20). To znači da su ove simulacije bile računski znatno zahtevnije od simulacija prikazanih na slici 3. Do otprilike 65 hiljada agenata, procesor je nešto brži od grafičke kartice, ali za mnogo veći broj agenata, vreme izvršavanja na grafičkoj kartici je osetno manja od brzine izvršavanja na procesoru. Razlika u vremenu izvršavanja simulacija na grafičkoj kartici i procesoru je ovde znatno izraženija nego kod simulacija sa malim brojem interakcija (slika 3).

Jedno procesorsko jezgro je mnogo brže i efikasnije od jedne računske jedinice grafičke kartice, a još brže i efikasnije od jedne radne stavke. Kada se izvršavaju simulacije sa malim brojem agenata, računske jedinice grafičke kartice nisu sve podjednako iskorišćene, pa je tada izvršavanje na procesoru brže. Povećanjem broja agenata, povećava se i upotreba svake računske jedinice, i tada su sve računske jedinice podjednako iskorišćene. Tada paralelno izvršavanje komandi grafičke kartice postaje znatno efikasnije od rednog izvršavanja komandi procesora. Ovim se može objasniti zašto su se simulacije sa manjim brojem agenata izvršavale kraće na procesoru nego na grafičkoj kartici i zašto grafik grafičke kartice posle određenog broja agenata naglo menja tok.

Inače, moguće je optimizovati simulacije sa malim brojem agenata, tako da se brže izvršavaju na grafičkoj procesorskoj jedinici. Jedna od mogućnosti jeste bolja organizacija radnih stavki po radnim grupama. Ovim bi svaka računarska jedinica bila podjednako iskorišćena, ali ova optimizacija ne bi bitno ubrzala izvršavanje simulacije.

Osim što je upoređeno vreme izvršavanja simulacija između grafičke kartice i procesora, vreme izvršavanja je upoređene i između simulacija sa i bez atomičnih funkcija. Ove simulacije su se isključivo izvršavale na grafičkoj kartici. Na slici 5 prikazani su rezultati ovog poređenja na logaritamskoj skali.

Vidimo da su simulacije koje nisu koristile atomične funckije izuzetno spore u odnosu na one koje su koristile takve funkcije. Ovo se dešava jer se deo simulacije u kom se agenti premeštaju sa jedne lokacije na drugu, kada se atomične funkcije ne koriste, izvršava samo na jednoj radnoj stavci. Ovaj deo simulacije nije paralelizovan, i zato je najspori-



Slika 5. Vreme izvršavanje simulacija koje su koristile i koje nisu koristile atomične funkcije.

Figure 5. Execution time of simulations that used and did not use atomic functions.

ji, što prouzrukuje da cela simulacija bude izuzetno spora.

Zaključak

Iako jednostavna, razvijena verzija simulacije bazirana na *SIR* modelu epidemije i agentskom pristupu, pokazala se zadovoljavajućom. Izborom odgovarajućih vrednosti za 15 parametara modela, moguće je simulirati različite vrste i oblike epidemija. Razlike u predikcijama, dobijene od simulacija koje su se izvršavale sa identičnim parametrima na grafičkoj kartici i procesoru, su male kada je broj agenata veći od 50 hiljada. Simulacije sa izuzetno mnogo agenata (preko 250 hiljada) se znatno brže izvršavaju na grafičkoj kartici nego na procesoru. Što se tiče vremena izvršavanja simulacija sa malim brojem agenata (manje od 60 hiljada), one će se brže izvršavati na procesoru nego na grafičkoj kartici, ali razlika u vremenu izvršavanja nije velika. Na osnovu ovoga zaključujemo da ako hoćemo da simuliramo epidemiju, možemo koristiti model predstavljen u ovom radu i najbolje je da simulaciju koja ima puno agenata i puno interakcija izvršavamo na grafičkoj kartici, i da se tada koriste atomične funkcije.

Literatura

Aaby B. G., Perumalla K. S., Seal S. K. 2010. Efficient simulation of agent-based models on multi-GPU and multi-core clusters. U *Proceedings of the 3rd international icst conference on simulation tools and techniques* (ur. F. Perrone i G. Stea). Brisel: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), str. 1-10.

Feautrier P. 2011. Bernstein's conditions. U *Encyclopedia of Parallel Computing* (ur. A. Padua). Berlin: Springer, str. 130-134.

Jaffry S. W., Treur J. 2008. Agent-based and popula-tion-based simulation: A comparative case study for epidemics. U *Proceedings of the 22nd European Conference on Modelling and Simulation*. ECMS (The European Council for Modelling and Simulation), str. 1-10.

Kermack W. O., McKendrick A. G. 1927. A contribution to the mathematical theory of epidemics. U *Proceedings of the royal society of london*. Series A. **115** (772): 700-21.

Kovac T., Haber T., Van Reeth F., Hens N. 2018. Heterogeneous computing for epidemiological model fitting and simulation. *BMC bioinformatics*, **19** (1): 101-11

Matsumoto M., Nishimura T. 1998. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, **8** (1): 3-30.

Shekh B., De Doncker E., Prieto D. 2015. Hybrid multi-threaded simulation of agent-based pandemic modeling using multiple GPUs. U *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, str. 1478-85.

Zou P., Lü Y. S., Wu L. D., Chen L. L., Yao Y. P. 2013. Epidemic simulation of a large-scale social contact network on GPU clusters. *Simulation*, **89** (10): 1154-72.

Nikola Kušlaković

Comparison of epidemic simulation efficiency on a central processing unit and a graphic processing unit

In this paper, an epidemic simulation model using an agent based method, based on the SIR model is examined. The way in which the algorithm of such a model can be modified is analyzed, so that in addition to execution on the processor, the simulation can also be executed on the graphics card. The simulation model had 15 parameters that could be changed and it was tested to see what types of epidemics it could simulate. The execution time of performing such a simulation on the processor and graphics card was also examined. In addition, it was concluded that simulations (performed exclusively on a graphics card) that used atomic functions proved to be extremely better than those simulations that did not use those functions. Simulations with a small number of agents (smaller than 60 thousand) were executed a little faster on the processor, and simulations with an extremely large number of agents (more than 250 thousand) were executed much faster on the graphics card. Also, the results of the simulations were compared, so that we can see the difference between the predictions obtained from the simulations that were performed on two different computer components. Increasing the agents in the simulation reduces the difference in the prediction of the simulations. Relative deviation is less than 2% for simulations that have more than 50 thousand agents.