Институт за математику и информатику
Природно-математички факултет
Универзитет у Крагујевцу

Завршни рад

Дистрибуирани и паралелни прорачуни у моделима таласне оптике

Ментор
др Милош Ивановић

Студент
Милица Стојановић, 95/2017

Јануар 2023.

Садржај

Сажетак

У реалним физичким и инжењерским системима наилазимо на веома комплексне проблеме до чијих решења на класичан начин, користећи персонални рачунар, није могуће доћи. Са појавом паралелних рачунара неки од тих проблема су постали решиви. Како паралелни рачунари постају све већи и бржи, постају способнији за решавање све више проблема чија су решења раније била недостижна. Неке од области у којима паралелно програмирање може бити корисно су: примењена физика, електротехника, финансијско и економско моделирање, вештачка интелигенција, квантна механика и још многе друге. У овом раду показаћемо пример примене паралелног и дистрибуираног рачунарства на проблеме који обухватају обраду велике количине података, као и коришћење Монте Карло методе код које је битан велики број покушаја како бисмо добили решење са што мањом грешком. Овде је конкретно обрађен пример из области таласне оптике, али се сличне методе могу применити на многе друге области. Оно што је најважније, а због чега је ова тема и обрађена, су резултати до којих смо дошли. Резултати су ту као доказ зашто је паралелно рачунарство веома битно и зашто би, без могућности паралелизације, многа открића била недостижна, чиме би се наш свет много спорије развијао и многе ствари које данас можемо да урадимо уз помоћ рачунара не би биле могуће.

1. Увод

Проблем добијања дифракционе слике на основу улазних података који укључују параметре светлости у свакој тачки прореза кроз који пролази светлост може да представља проблем за традиционално израчунавање које човек може и мануелно да реши. Како бисмо ефикасније дошли до решења користили смо програм који је написан у програмском језику Фортран. Код је у примарној верзији написан за секвенцијално извршавање. Како није могуће учитати вредности у свакој тачки прореза, као улазне податке узећемо средње вредности параметара светлости на сегментима прореза, тако да је резултат који добијамо са одређеном грешком. За добијање резултата са већом прецизношћу, улазни подаци су обимнији тако да су и израчунавања потребна за долазак до решења сложенија, а како рачунари не могу израчунавања да изведу у трену и сваки систем захтева неко време за извршавање одређених операција, што би значило да је за сложеније проблеме потребно и дуже време извршавања.

За проблеме као што је израчунавање дифракционе слике често је потребно много више од регуларног рачунара, на којима није могуће добити задовољавајуће решење, што имплицира решења са превеликом грешком или предугог израчунавања да би могло бити од реалне користи.

Дакле, долазимо до закључка да, иако далеко ефикаснији од мануелног израчунавања, чак ни рачунари нису довољно ефикасни уколико на њима израчунавања вршимо секвенцијално. Рачунари су ограничени у погледу брзине израчунавања операција, тако да морамо унапредити секвенцијално израчунавање на једном рачунару и пронаћи ново решење. Као два могућа решења за овај проблем користићемо следећа два приступа:

- MPI (*Message passing interface*) - Интерфејс за прослеђивање порука
- CUDA (*Compute Unified Device Architecture*) – Обједињена рачунарска архитектура уређаја

У наставку рада ћемо се прво позабавити таласном оптиком и самом појавом дифракције светлости, као и већ постојећим секвенцијалним решењем које је написано у програмском језику Фортран. Затим ћемо описати MPI и CUDA моделе које смо користили у експерименталном делу рада, које су предности и мане и шта их то разликује. Након теоријског осврта на дате моделе паралелизације, доћи ћемо до дела који се тиче саме примене на дати проблем. Биће представљени кључни делови кода уз њихов опис. На крају, као кључни део овог рада, биће представљени резултати до којих смо дошли експерименталним путем. Резултати ће бити представљени у виду поређења убрзања и квалитета слике добијених почетним кодом са секвенцијалним приступом и резултата добијених коришћењем два метода паралелног програмирања.

2. Таласна оптика и дифракција

2.1. Увод

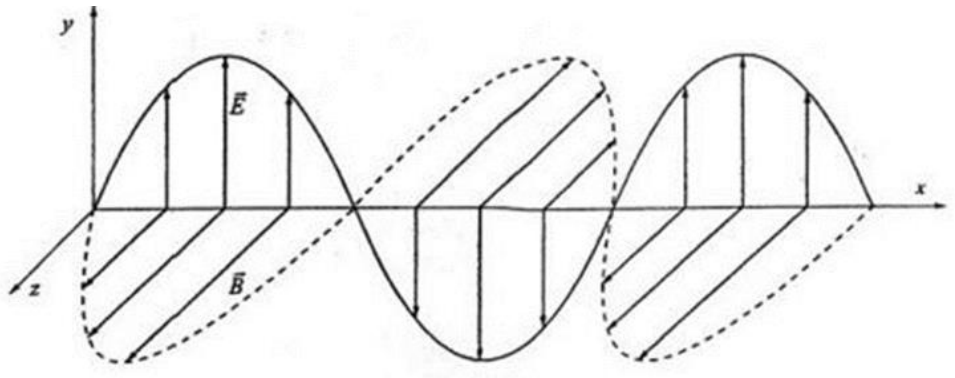
Простирање светлости кроз средине мањих димензија, реда таласне дужине светлости, се не може описати законима геометријске оптике, већ је неопходно разматрати њену таласну природу. По Хајгенс-Френеловом принципу, свака тачка погођена светлошћу постаје извор таласа.

2.2. Електромагнетни таласи

2.2.1. Природа електромагнетних таласа

Светлост је електромагнетни талас, па оно што важи за електромагнетне таласе, важи и за светлост. Електромагнетни талас има следеће особине (**Слика 1** – Вектори *E* и *B* представљају тренутну вредност електричног и магнетног поља таласа):

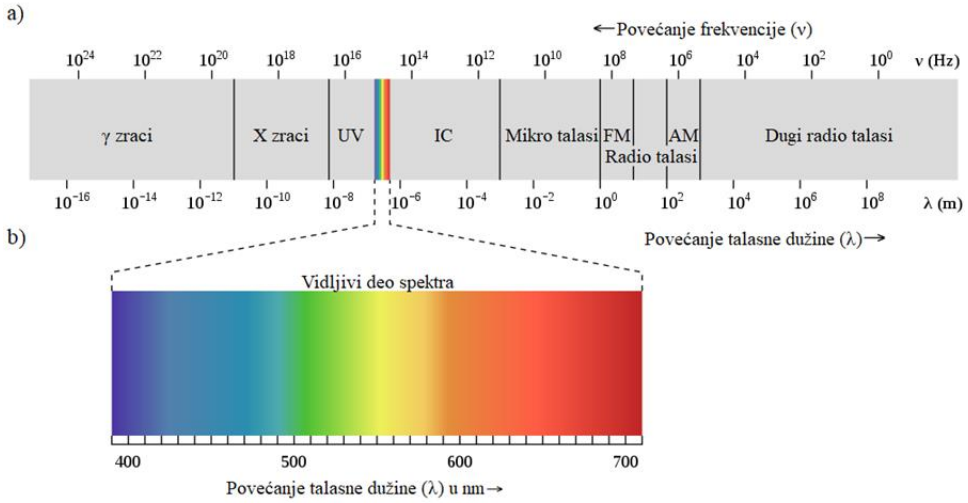
- Вектор *E* и вектор *B* истовремено достижу максимум и минимум, то јест налазе се у фази.
- Простире се дуж правца који се одређује на основу векторског производа *E* и *B,* нормално на оба вектора.



Слика 1: *E и B у прогресивном таласу налазе се у фази*

2.2.2. Подела електромагнетних таласа

Електромагнетни таласи обухватају веома широк опсег фреквенција, односно таласних дужина (**Слика 2**).



Слика 2: *а) Спектар електромагнетних таласа, b) Видљиви део спектра*

Видљив део спектра таласа обухвата најужи опсег таласних дужина и то је део електромагнетног спектра које људско око може да региструје. Региструје га као светлост различитих боја.

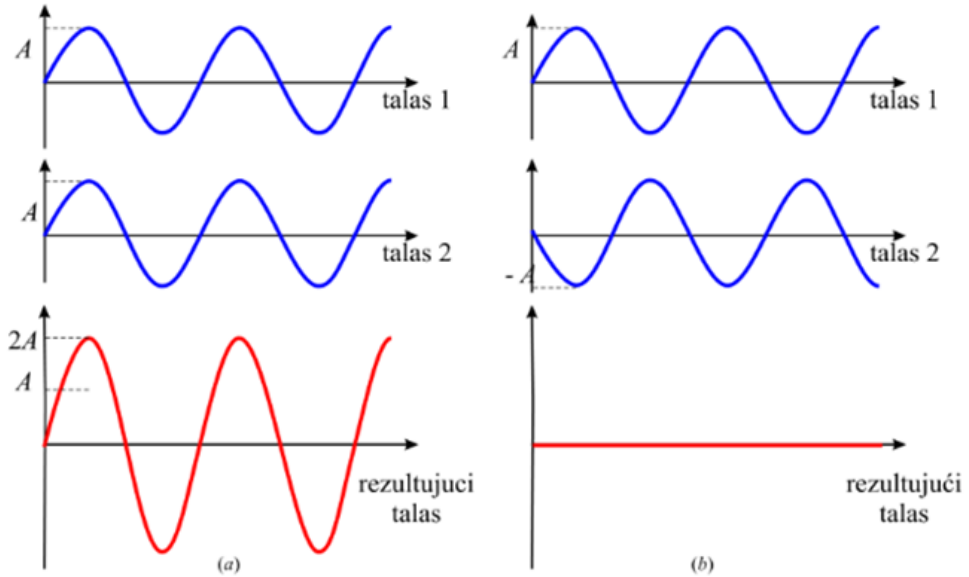
2.3. Интерференција светлости

2.3.1. Интерференција светлосних таласа

Интерференција светлосних таласа је слагање два или више светлосних таласа, при чему резултујући талас има у неким тачкама мањи, а у неким тачкама већи интензитет у односу на збир интензитета појединих таласа. На тај начин при слагању таласа долази до слабљења или до појачања интензитета таласа у односу на првобитни интензитет таласа.

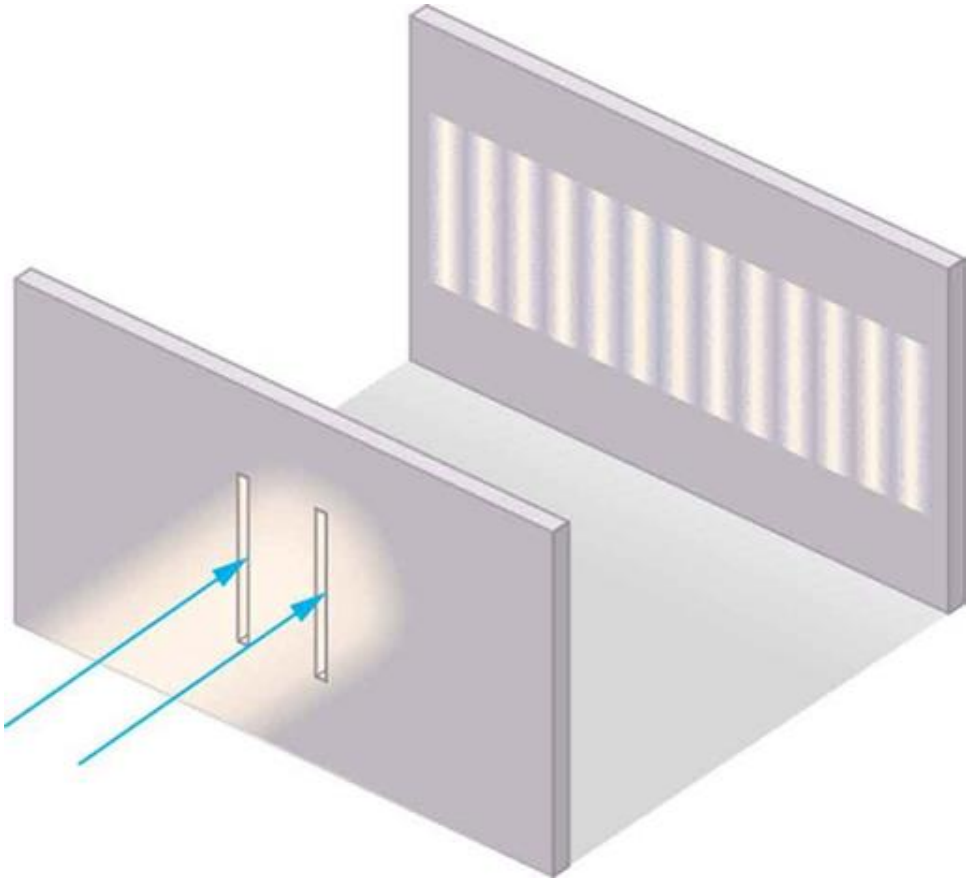
Ова расподела минималних и максималних интензитета је на одређени начин правилна и представља ефекат интерференције (интерференциону слику).

Фазна разлика таласа емитованих са природних светлосних извора се мења брзо и непредвидиво, па су такви таласи кохерентни само за време трајања једне таласне поворке, па је толико трајање и интерферентне слике која се затим мења. Наше око, као ни остали детектори светлости не може да прати тако брзе промене, већ усредњава те промене интензитета и уместо екрана на коме постоји стално променљива интерферентна слика види равномерно осветљен екран. Интерференцијом два или више светлосних таласа настаје нови талас. Интерференција може бити конструктивна и деструктивна у зависности од разлике у фазама два снопа светлости (**Слика 3**).



Слика 3: *а) Конструктивна интерференција, b) Деструктивна интерференција*

Интензитет светлости на екрану није једнак збиру интензитета појединачних светлосних таласа, већ зависи и од њихове фазне разлике. Отуда се на екрану виде наизменично поређане светле и тамне пруге, што представља интерференциону слику (**Слика 4**).



Слика 4: *Интерференциона слика*

2.3.2. Услов за интерференцију

Ако имамо две сијалице постављене једну поред друге, неће се формирати интерференциона слика на екрану. Интерференције неће бити зато што су таласи једне сијалице емитовани независно од таласа из друге сијалице. Светлосни таласи из та два светлосна извора немају константну фазну разлику у времену, већ важи да је фазна разлика функција времена, и мења се стохастички, па је средња вредност једнака нули.

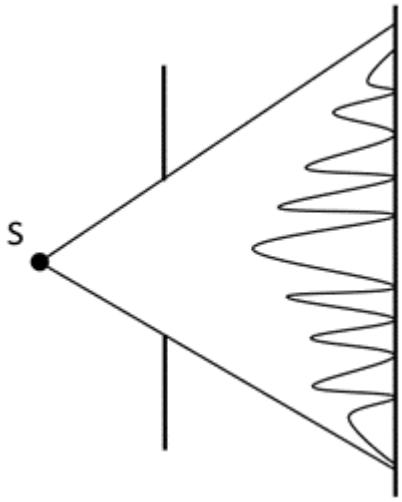
Да би имали одрживу интерференцију коју можемо размотрити, морају бити задовољени следећи услови:

- Извор мора бити кохерентан (разлика у фази мора бити константна, једна у односу на другу).
- Извор би морао бити монохроматски (извор једне таласне дужине).

Да бисмо имали стабилну интерференциону слику морамо имати таласе између којих је фазна разлика константна. Светлост код које је фазна разлика константна назива се кохерентна светлост. Извори који дају такву светлост називају се ласери.

2.4. Дифракција

Дифракција је појава скретања светлосних зрака са праволинијске путање када падају на препреке или прорезе малих димензија реда таласне дужине светлости. Постојање дифракције је и доказ о таласној природи светлости. Дифракција постоји и код звучних таласа. Захваљујући њој звук се чује иза препрека, јер је таласна дужина звучних таласа око једног метра, па су димензије препрека упоредиве са њом. Код светлосних таласа, таласна дужина је реда од *400-700nm*, па се ова појава уочава на објектима малих димезнија.



Слика 5: *Дифракциона слика при проласку светлости кроз прорез*

Ако се посматра монохроматска светлост која пролази кроз правоугаони прорез малих димензија као на слици (**Слика 5**), иза прореза на неком екрану, према законима геометријске оптике, појавиће се светли круг, оивичен зрацима који се праволинијски простиру од извора кроз прорез. Оваква ће се слика формирати уколико је прорез великих димензија. Међутим, уколико је прорез малих димензија, слика на екрану је другачија. Појављује се шири осветљени круг него што то предвиђају закони геометријске оптике и расподела интензитета светлости унутар круга није равномерна. Ако светлост наиђе на малу препреку као што је длака или танка жица, на екрану иза препреке ће се такође појавити тамне и светле пруге и опет ће светла пруга бити у средини.

Када талас наиђе на мали отвор или мало тело, све тачке отвора као и ивице отвора и тела постају извори секундарних сферних таласа. При свом простирању ови таласи интерферирају и на неким местима се међусобно слабе, а на неким појачавају. Што је отвор или препрека мања, скретање зрака је веће, тј. ефекти су јаче изражени.

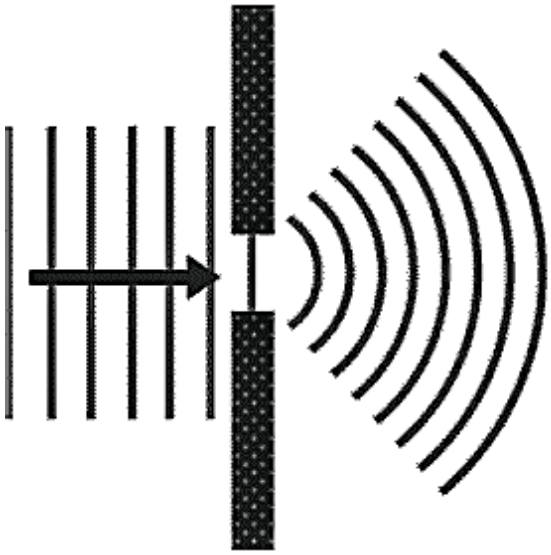
Појава ефективног скретања светлости и интерференције светлосних таласа од континуално распоређених извора назива се **дифракција**. Слика на екрану која се састоји од правилно распоређених тамних и светлих пруга или концентричних кругова, а која настаје услед дифракције, назива се **дифракциона слика**.

Разликујемо две врсте дифракције:

- **Френелова дифракција** – дифракција код које су светлосни извори или заклон или оба на коначном растојању од препреке. У овом случају зраци који долазе на препреку, као и они који се иза препреке простиру ка заклону, нису паралелни.
- **Фраунхоферова дифракција** – дифракција када су зраци који долазе на препреку и зраци који одлазе са препреке ка заклону паралелни. У овом случају светлосни извор и заклон се сматрају да су на ефективно бесконачном растојању од препреке. Ова паралелност зрака може се постићи и када су светлосни извор или заклон на коначном растојању уз помоћ сабирног сочива.

2.4.1. Хајгенс-Френелов принцип

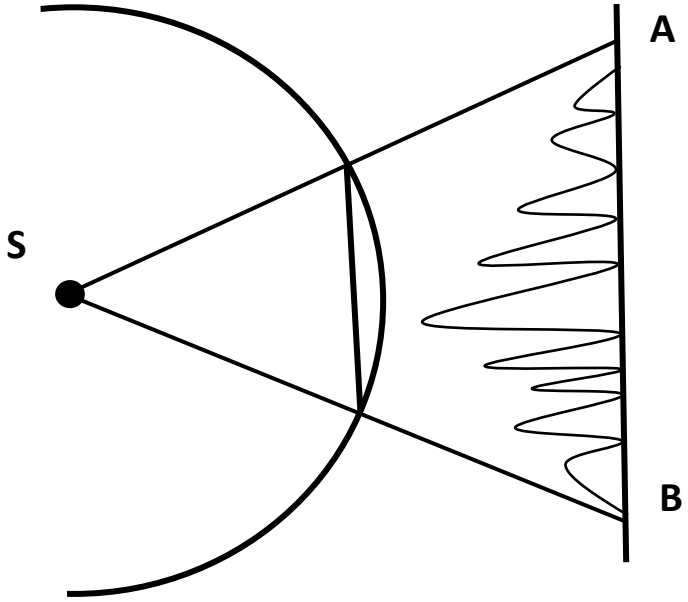
Простирање светлосног таласа за хомогене средине се може описати Хајгенс-Френеловим принципом. Хајгенс је геометријски описао простирање таласа, а математички формулисао Френел. Нека имамо један светлосни извор који обасјава један мали прорез (**Слика 6**). Свака тачка тог прореза је погођена светлосним таласом. Услед тога, свака та тачка постаје извор сферног светлосног таласа који се шири – **Хајгенсов принцип**. Новоформирани таласи називају се секундарни таласи. Ако се електрично поље светлостног таласа на површини прореза *dS* мења *као* $\boldsymbol{E}\cdot\cos(\omega t)$, тада у тачки на растојању $\boldsymbol{r}$ од сегмента електрично поље ће бити једнако $a(\boldsymbol{E}/r)\cdot\cos(\omega t-\boldsymbol{kr})$, где је *a* функција од угла којом правац таласа заклапа са сегметном прореза *dS*.



Слика 6: *Простирање таласа кроз прорез*

2.4.2. Френелова дифракција

Посматрајмо Френелову дифракцију на округлом диску. Нека имамо светлосни извор *S* и на неком растојању непрозирну препреку (**Слика 7**).



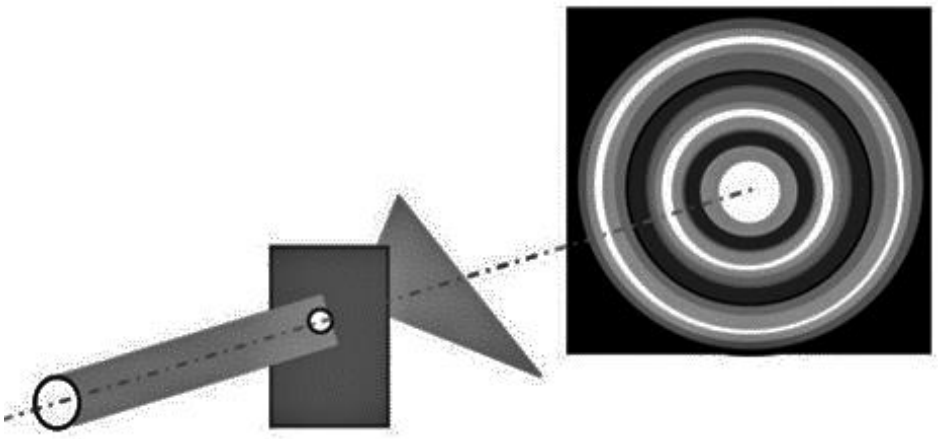
Слика 7: *Френелова дифракција*

Према законима геометријске оптике, на екрану би се очекивала сенка између тачака *A* и *B*, а остатак екрана би био осветљен. Међутим, у том делу екрана јављају се наизменично распоређене светле и тамне пруге.

2.4.3. Фраунхоферова дифракција на једном прорезу

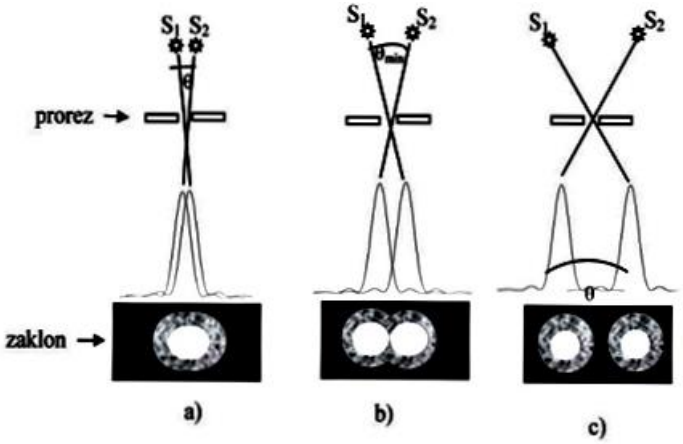
Фраунхоферова дифракција се јавља када се извор и екран налазе бесконачно далеко од препреке (прореза). То је услов да би упадни сноп светлосних таласа био паралелан, као и да би паралелан био сноп светлости након проласка кроз препреку.

Ако је постављен мали отвор у облику круга, тада дифракциона слика има облик концентричних тамних и светлих прстенова као на слици (**Слика 8**).



Слика 8: *Дифракција на кружном прорезу*

Дифракција се јавља и код посматрања удаљених тела оптичким инструментима, због коначне ширине отвора објектива тих инструмената. Такође, она се јавља и када светлосни талас не пада нормално на раван отвора или препреке већ под неким углом. У том случају средиште централног максимума није у пресеку симетрале система и заклона, већ је померено (**Слика 9**).



Слика 9: *Моћ разлагања*

2.4.4. Дифракциона решетка

Оптички елемент који садржи велики број прореза назива се **дифракциона решетка** (**Слика 10**).

Слика 10: *Дифракциона решетка*

Величина прореза је на слици означена са *a,* док је растојање између прореза означено са *d* и представља константу решетке. Ако светлост пролази кроз *N* паралелних светлих отвора, дифракциона слика се мења у односу на ону која настаје при проласку светлости кроз један отвор. У овом случају се јављају јасно изражени главни максимуми између којих постоји *N*-2 наизменично постављена максимума знатно мањег интензитета. Што је број *N* већи, главни максимуми су све већег интензитета и све ужи, тако да је дифракциона слика све јаче изражена.

Дифракционе решетке се користе за анализу светлости, одређивање таласне дужине монохроматске светлости, као и за разлагање сложене светлости на основне боје.

Ако на дифракциону решетку пада бела светлост, тада је централни максимум такође бела светлост, међутим остали максимуми вишег реда се виде у облику спектра, тј. дуге.

Да би се дифракциона решетка користила за разликовање блиских таласних дужина, линије на заклону које одговарају овим таласним дужинама треба да буду на што већем међусобном растојању.

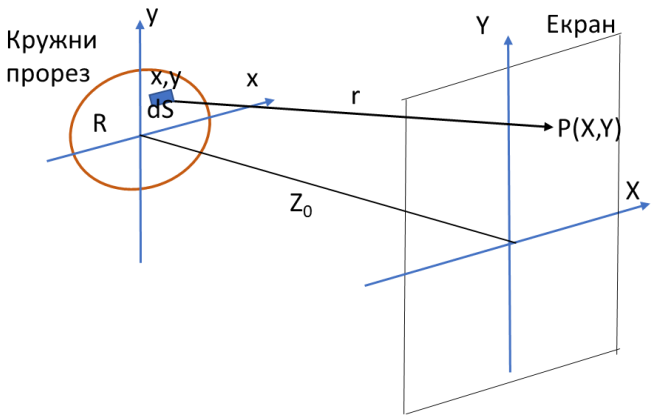
Да би се разликовале линије блиских таласних дужина, поред довољне разлике у углу под којим се виде морају бити и уске, да би се што боље разликовале.

2.5. Нумеричке симулације простирања светлости

Методологија за одређивање дифракционе слике дата у овом раду може се применити за било коју произвољну површину дефинисану једначином. Сегментацијом отвора и екрана на коме се посматра слика, укупни интензитет у некој тачки екрана може се добити сумирањем доприноса електомагнетног поља од свих елемената отвора. Услов који треба испунити је да сегменти отвора морају бити довољно мали.

На основу модела, написан је програм који се користи за добијање дифракционе слике светлости која пролази кроз један, два и *N* прореза (дифракциона решетка). У програму се могу задати позиције прореза и екрана на коме се формира дифракциона слика. Као резултат програма добија се дифракциона слика светлости након проласка кроз наведене елементе.

Алгоритам за одређивање дифракционе слике се састоји у томе да се у датој тачки *P* на екрану рачуна јачина електромагнетног поља од сегмента прореза. Укупна јачина електромагнетног поља у тачки *P* од целог прореза добија се сумирањем јачина електромагнетних поља од свих сегмента. Даљим рачуном долазимо до интензитета светлости у тачки P на екрану. Наведени поступак се даље понавља и за остале тачке на екрану, све док се не одреди интензитет светлости у свим тачкама екрана.



Слика 11: *Геометрија прореза и екрана*

Поступак је следећи. На екрану се изабере тачка P (X,Y), у којој се ређује интензитет светлости. На кружном прорезу радијуса P изабере се сегмент површи *dS*, дефинисан координатама (x,y). Јачина електричног поља светлосног таласа који од сегмента dS допире то тачке P је према Хајгенс-Френеловом принципу једнака

$$dE_P = \frac{E}{r} dS \cos\left(k\, r + \phi\right),$$

где је *E*, амплитуда електричног поља на сегменту *dS*, $r = \sqrt{\left(X - x\right)^2 + \left(Y - y\right)^2 + Z_0^2}$ је растојање тачке P од сегмента *dS*, а $k=2\pi/\lambda$ је таласни број. У овом изразу је изостављен члан ωt, јер не доприноси фазној разлици таласа свих сегмената прореза. ϕ је почетна фаза светлости на сеегменту *dS*. Ако је $\phi=0$ за сваки сегмент, тада имамо кохерентну светлост са прореза. Уколико је $\phi=\pi \cdot u$, тада имамо некохерентну (природну) светлост, при чему је *u* случајан број од 0 до 1.

Укупна вредност електричног поља у тачки P од свих сегмената прореза је једнака

$$E_P = \sum_i dE_P \,,$$

где се сумирање врши по свим сегментима прореза. Интензитет светлости у тачки P је једнак

$$I = E_P^2 .$$

3. Паралелни програмски модели

3.1. MPI

Интерфејс за прослеђивање порука (MPI) је стандардна спецификација интерфејса за преношење порука за паралелно рачунање у системима са дистрибуираном меморијом. MPI није програмски језик већ је библиотека функција које се могу позивати из кода написаног на неком од многих језика, као што су C, C++, Пајтон, Јава, Фортран,..., како би се програм паралелизовао и извршавао на више чворова, као и како би ти чворови могли да комуницирају и размењују податке. У паралелном рачунарству, више рачунара, а понекад и више процесорских језгара у оквиру истог рачунара, називају се чворови. Сваки чвор у систему може да ради на делу укупног рачунарског проблема. Изазов је онда синхронизовати радње сваког паралелног чвора, размењивати податке између чворова и обезбедити команде и контролу над целим паралелним кластером.

Комуникатор дефинише групу процеса који имају способност да међусобно комуницирају. У овој групи процеса, сваком се додељује јединствени ранг и они експлицитно комуницирају једни са другима позивајући се на своје рангове. MPI комуникатор се може динамички креирати и имати више процеса који се истовремено покрећу на одвојеним чворовима кластера. Сваки процес има јединствени MPI ранг као своју идентификацију, сопствени меморијски простор и извршава се независно од других процеса. Процеси комуницирају једни са другима прослеђивањем порука за размену података и синхронизацију. Интерфејс за прослеђивање порука дефинише стандардни скуп функција за ове задатке. Паралелизам се јавља када се програмски задатак подели на мање делове и дистрибуира те делове међу процесима, у којима сваки процес обрађује свој део.

MPI није одобрен као званични стандард од стране било које организације за стандардизацију, као што је Институт инжењера, електротехнике и електронике (IEEE – *The institute of Electrical and Electronics Engineers*) или Међународна организација за стандардизацију (ISO – *International Organization for Standardization*), али се генерално сматра индустријским стандардом и чини основу за већину комуникационих интерфејса које усвајају програмери паралелног рачунарства. Софтверске и хардверске компаније су креирале и разне имплементације MPI стандарда.

Основна идеја дистрибуираног паралелизма је имати више инстанци истог програма које раде на различитим подацима. Програм би могао бити покренут на истој машини или групи машина.

Један од главних недостатака MPI паралелног оквира је то што су перформансе ограничене комуникационом мрежом између чворова, тако да је потребно бити опрезан приликом креирања оваквог кода јер би неоптимизованом комуникацијом међу чворовима могло да се изгуби значајно на перформансама. У екстремним случајевима може доћи до тога да се паралелни код извршава и дуже од секвенцијалног.

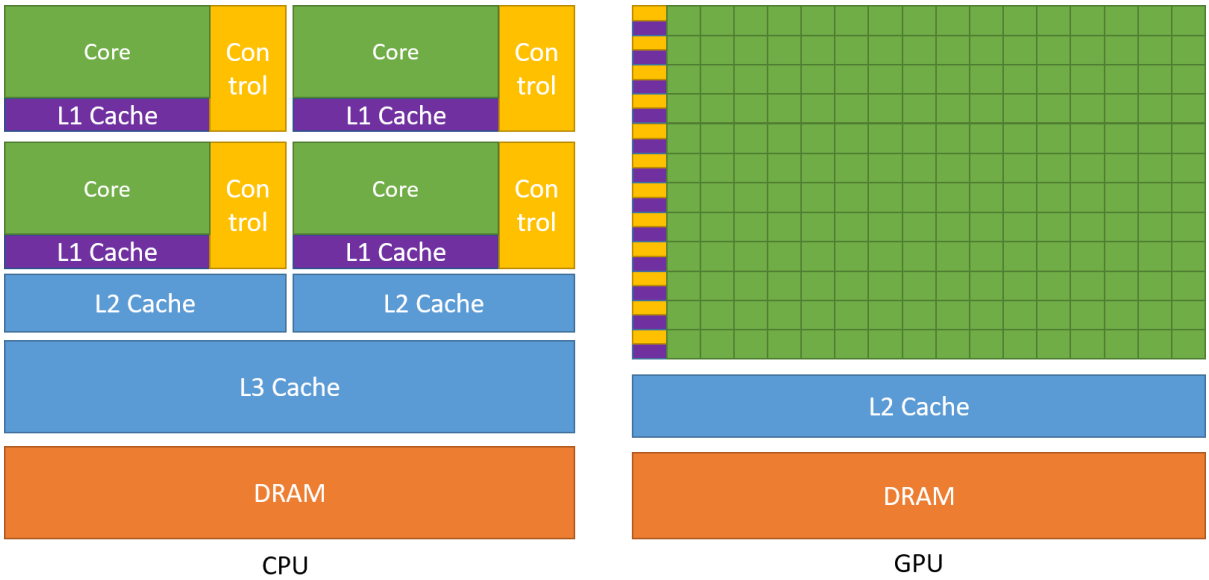
3.2. CUDA

Обједињена рачунарска архитектура уређаја (CUDA) је паралелна рачунарска платформа и модел програмирања који је развила Nvidia за опште рачунарство на сопственим јединицама за обраду графике (GPU - *graphics processing unit*). CUDA омогућава програмерима да убрзају рачунарски интензивне апликације тако што ће искористити снагу графичких картица за део рачунања који се може паралелизовати.

Графичке картице пружају много већу пропусност инструкција и меморијски пропусни опсег од процесора (CPU - *Central Processing Unit*) у оквиру сличне цене и снаге. Многе апликације користе ове веће могућности да раде брже на графичким интерфејсима него на процесору.

Ова разлика у могућностима између графичких картица и процесора постоји јер су дизајнирани са различитим циљевима на уму. Док је процесор дизајниран да буде ефикасан у извршавању низа операција, названих нит, што је брже могуће и може да изврши неколико десетина ових нити паралелно, графички процесор је дизајниран да буде ефикасан у извршавању хиљада нити паралелно жртвујући перформансе једне нити ради постизања веће пропусности. То би значило да је извршавање једне нити на графичком процесору знатно спорије него на процесору, али то се надокнађује огромним бројем нити које може да обрађује паралелно.

Графички процесор је специјализован за високо паралелна израчунавања и стога је дизајниран тако да је више транзистора посвећено обради података, а не кеширању података и контроли тока. Следећа слика (**Слика 12**) приказује пример расподеле ресурса чипа за процесор у односу на графички процесор.



Слика 12: *Расподела ресурса*

3.3. MPI и CUDA

MPI и CUDA су потпуно различите архитектуре. MPI нам омогућава да дистрибуирамо своју апликацију на неколико чворова, док CUDA омогућава да користимо графичке картице унутар локалног чвора. Предности CUDA приступа најлакше можемо да приметимо у задацима као што су извршавање простих операција над много података. Уколико имамо много сложенијих операција као што су разни упити тада можемо да приметимо опадање перформанси. MPI приступ се добро показује уколико имамо много сложенијих операција у коду. То је очекивано с обзиром да је примарна примена графичких процесорских јединица израчунавање координата које се добијају са неколико примарних операција, док су процесори дизајнирани да решавају знатно сложеније проблеме.

Реалне апликације најчешће имају мешавину паралелних и секвенцијалних делова, тако да су најјачи супер-рачунарски системи дизајнирани са мешавином графичких картица и процесора, како би се максимизовале укупне перформансе. Апликације са високим степеном паралелизма могу да искористе ову масивно паралелну природу графичких картица да би постигле веће перформансе него на процесорима.

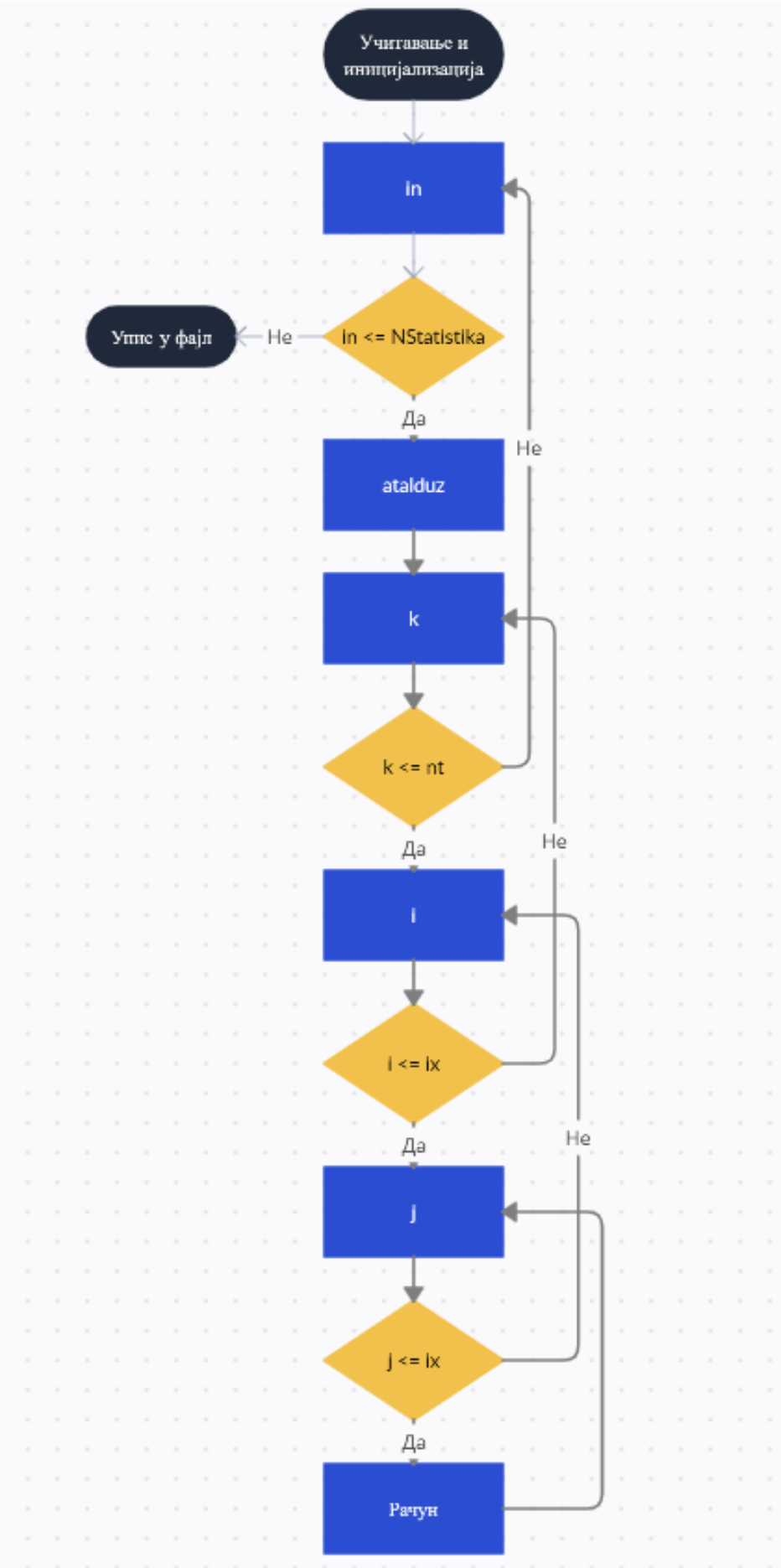
4. Програмска решења

Као почетну тачку за креирање овог рада имали смо изворни код написан у Фортрану који се извршава секвенцијално. Код је састављен од више угњеждених петљи и садржи знатан број података за обраду, који се могу обрађивати паралелно. Подаци се могу поделити на више начина. Такође, комуникација између чворова се може извршити са различитим комбинацијама. У овом одељку ћемо видети делове кода који су кључни за поделу података који ће се обрађивати у паралелним деловима кода. Биће представљени дијаграми тока извршавања за оригинални изворни код, као и за кодове написане коришћењем MPI и CUDA приступа. Поред дијаграма видећемо и сам изворни код, који ћемо описати како би цео процес био јаснији.

Главни програм написан у Фортрану добијене резултате на крају обраде само уписује у текстуални фајл. Саму дифракциону слику добијамо из помоћног програма написаног у *Python*-у. Тај програм учитава податке из фајла и на основу њих исцртава прорачунату дифракциону слику.

4.1. Секвенцијални приступ

На следећем дијаграму дат је ток извршавања програма секвенцијалним приступом на једном процесору (**Дијаграм 1**).



Дијаграм 1: *Секвенцијално извршавање*

На почетку је потребно учитати све потребне податке и поставити почетне вредности променљивих. У коду имамо неколико петљи. Петље *i* и *j* представљају сегменте (апроксимација тачака) на дифракционој слици. У зависности од *ix* зависи и резолуција резултујуће слике коју ћемо на крају да добијемо. У коду имамо и петљу *atalduz* кроз коју пролазимо кроз различите таласне дужине и њих укључујемо у приказ дифракционе слике. За потребе овог пројекта смо креирали дифракциону слику служећи се само једном таласном дужином, тачније користећи само монохроматску светлост. Како бисмо могли да дођемо до решења, као улаз је потребно да имамо учитане параметре светлости на сегментима прореза, где је број сегмената означен са *nt*. Потребан нам је и насумично одабран број *u* јер је за методу израчунавања дифракционе слике у овом коду коришћена *Монте Карло* метода, одатле *NStatistika* представља број проласка. Што је тај број већи, решење има нижу грешку, тј. приближније је стварној дифракционој слици. То ћемо и демонстрирати на сликама које смо добили експерименталним путем.

У наставку је дат део изворног кода.

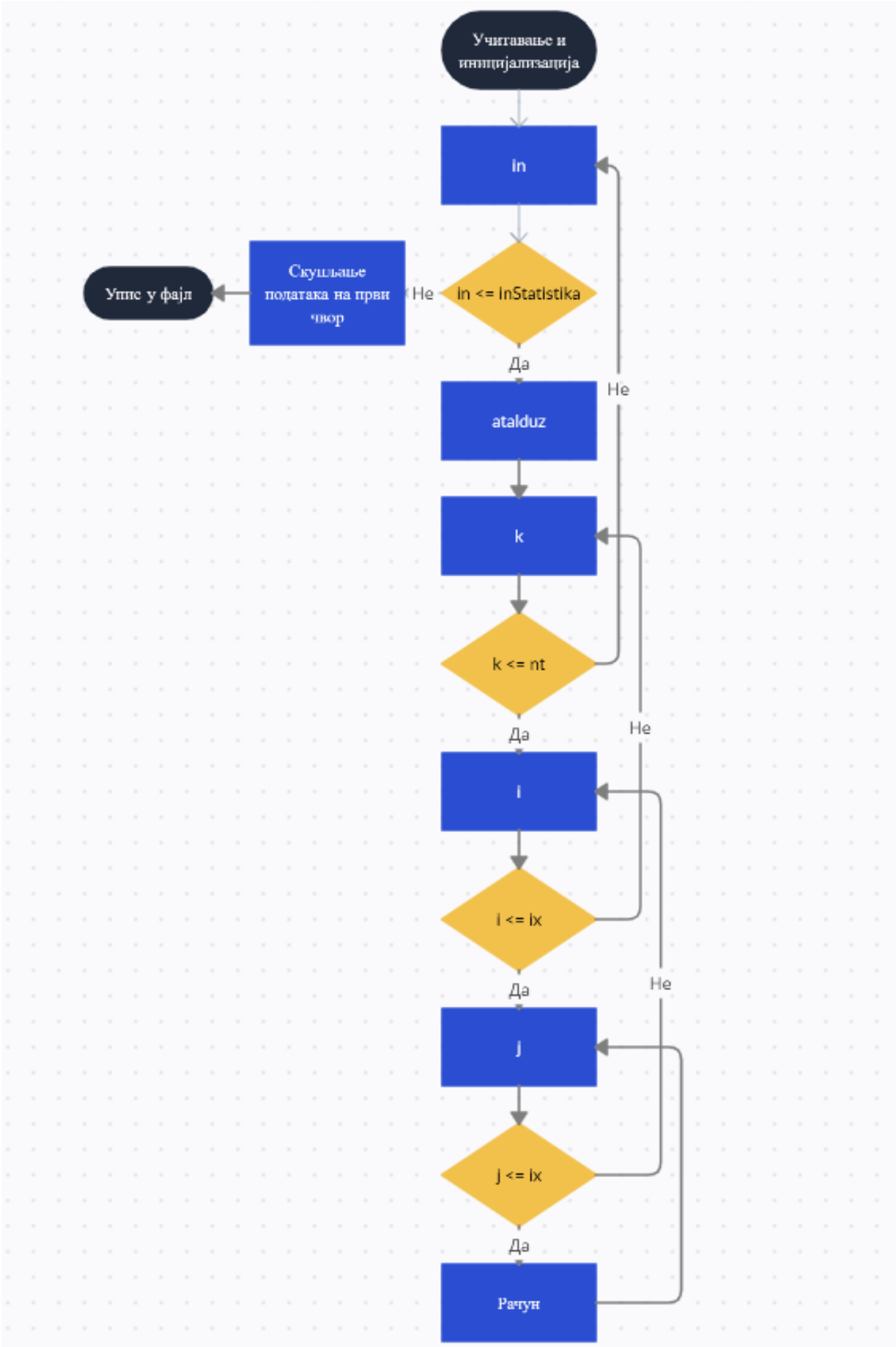
Листинг 1: JedanProcesor.f90

```
1   DO in = 1, Nstatistika
2         do atalduz = 600, 600  ! Razlicite talasne duzine
3            do k = 1, nt   ! segmenti traga
4
5               call random_number(u)
6               fazasl = pi*u
7
8               AK = 2.*PI/atalduz        ! tal broj
9
10              do i = 1, ix
11              do j = 1, ix
```

Видимо да имамо укупно 5 петљи у којима се извршава рачун. Резултат након сваке петље додајемо на збир и на крају акумулиране вредности уписујемо у фајл. Видимо да је проблем упрошћен тиме да посматрамо само монохроматску светлост, тако да *atalduz* петља практично ни не постоји, јер кроз њу пролазимо само једном. Уколико хоћемо да посматрамо сложенију светлост онда бисмо имали још сложенији проблем. Дакле, овде се указује додатни простор за паралелизацију. Свака итерација је независна од претходне тако да је могуће извршавати их паралелно и на крају сабрати резултате свих делова, што нам омогућава да знатно убрзамо програм.

4.2. MPI

Применом MPI приступа добијамо следећи ток извршавања (**Дијаграм 2**).



Дијаграм 2: *Извршавање MPI приступа на једном чвору*

Део кода из решења коришћењем MPI приступа дат је у наставку.

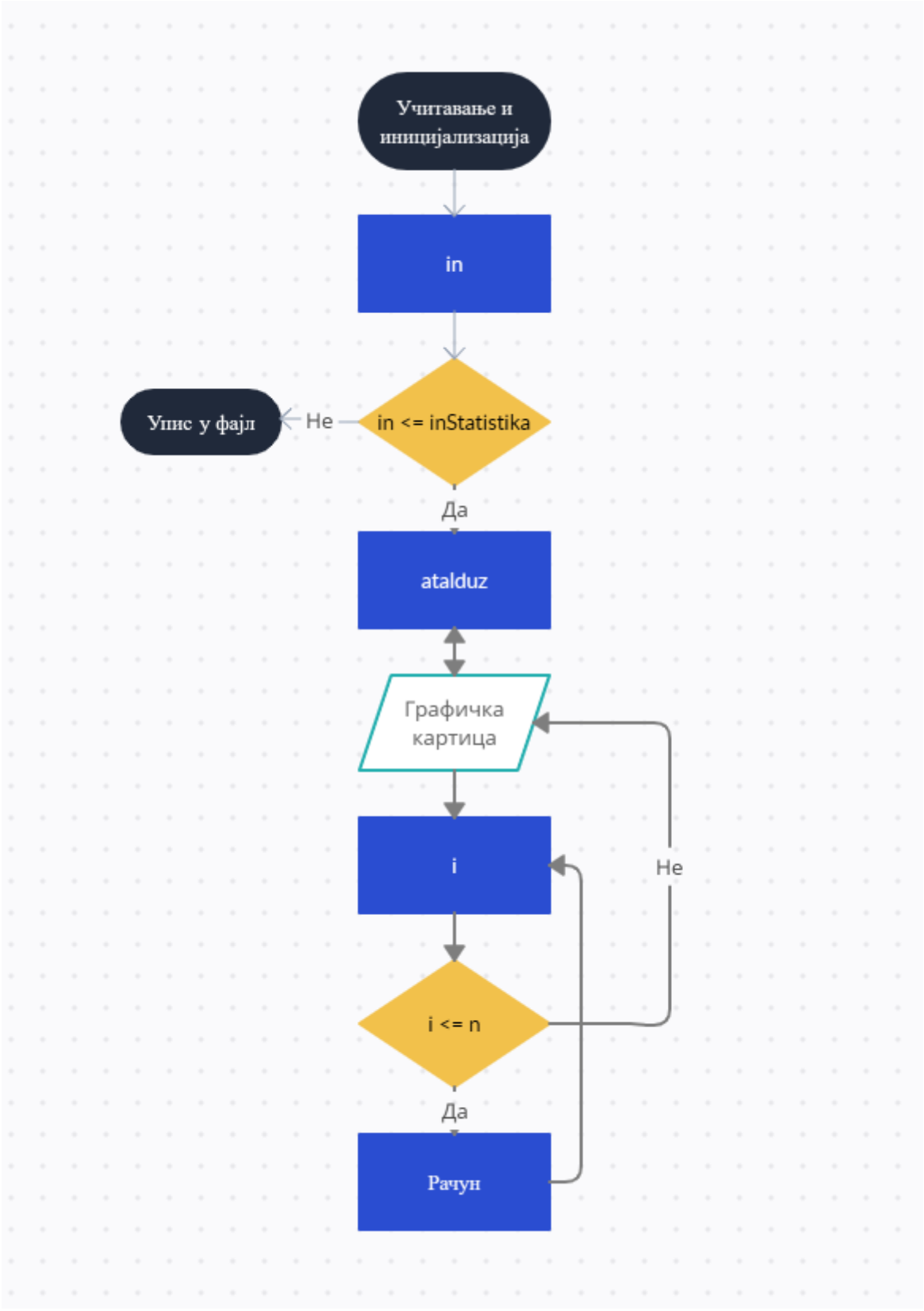
Листинг 2: MPI.f90

```
1   inStatistika = NStatistika / (num_of_processes)
2      modStatistika = MOD(NStatistika, num_of_processes)
3      if(num_of_processes - process_rank <= modStatistika) then
4         inStatistika = inStatistika + 1;
5      end if
6
7      SVETR = 0
8      SVETIM = 0
9
10     DO in = 1, inStatistika
11        do atalduz = 600, 600  ! Razlicite talasne duzine
12           do k = 1, nt   ! segmenti traga
13
14              call random_number(u)
15              fazasl = pi*u
16
17              AK = 2.*PI/atalduz        ! tal broj
18
19              do i = 1, ix
20              do j = 1, ix
```

Задатак смо поделили тако да сваки чвор пролази део петље *Nstatistika,* што је прва спољашња петља у коду. Пошто су за сваки пролаз кроз *Nstatistika* петљу потребни сви подаци, први чвор ће учитати улазни фајл и затим послати све осталим чворовима. Након тога ће се посао поделити и сваки чвор ће уместо кроз петљу *Nstatistika* проћи кроз петљу *inStatistika* тако да је збир *inStatistika* свих чворова једнак *Nstatistika*. Сваки чвор ће добити свој интерни резултат који ће на крају послати првом чвору, први чвор ће сакупити све резултате и израчунати коначни резултат који ће затим сачувати у фајл у виду матрице (дифракционе слике).

4.3. CUDA

CUDA приступ карактерише нешто другачији дијаграм тока, што се може видети (**Дијаграм 3**).



Дијаграм 3: *Извршавање коришћењем графичког процесора*

За CUDA метод имамо креирану подрутину која рачуна део резултата. Део кода који распоређује посао је дат у наставку.

Листинг 3: CUDA.cu

```
1   DO in = 1, Nstatistika
2
3            do atalduz = 600, 600 ! 400, 700, 50 ! Razlicite talasne duzine
4                   !ataladuz=600
5
6                   do k = 1, nt
7                         call random_number(u(k))
8                   end do
9
10                  d_ix = ix
11                  d_u = u
12                  d_nt = nt
13                  d_x0 = x0
14                  d_y0 = y0
15                  d_z0 = z0
16                  d_atalduz = atalduz
17                  d_zdoekr = zdoekr
18                  d_zdubina = zdubina
19                  d_p0 = p0
20                  d_ds = ds
21                  d_ax = ax
22                  d_ay = ay
23                  d_az = az
24                  d_svetim = svetim
25                  d_svetr = svetr
26
27                  d_countPerThread = countPerThread
28
29                  call calculate<<<numOfBlocks,
    threadsPerBlock>>>(d_countPerThread, d_ix, d_nt, d_u, d_x0, d_y0, d_z0,
    d_atalduz, d_zdoekr, d_zdubina, d_p0, d_ds, d_ax, d_ay, d_az, d_svetim,
    d_svetr)
30
31                  svetim = d_svetim
32                  svetr = d_svetr
33
34                  !end do ! prebrojava tal duz
35
36           END DO ! PETLJA ZA TAL DUZINU
37      END DO ! PETLJA ZA STATISTIKU
```

Овде смо за поделу користили унутрашње петље *i, j* и *k,* које су знатно веће од спољашње јер, за разлику од броја чворова, број нити у графичким процесорима је вишеструко већи и броји стотине. Као што видимо свака нит ће извршити приближно *x/(numberOfBlocks*threadPerBlock)* итерација, где је *x* укупан број итерација у изворном коду што износи *x=ix*ix*nt*. Оно што нам знатно олакшава сакупљање резултата је постојање CUDA методе *atomicadd()* којом можемо да додајемо нове резултате на постојећи збир на нивоу целе графичке картице. Ово нам омогућава да не морамо да водимо рачуна о том делу конкурентности и не морамо да бринемо да ће се резултат сваке нити урачунати у крајње решење, за разлику од MPI приступа где је сваки чвор слао свој резултат првом чвору и онда је он сабирао те резултате како би добио крајњи

резултат. Решење које смо добили на графичкој картици се враћа у главни део програма где се памти и користи за даље проласке кроз спољашње петље све до краја програма, а затим се последње добијено решење са графичке картице исписује у фајл.

5. Резултати и дискусија

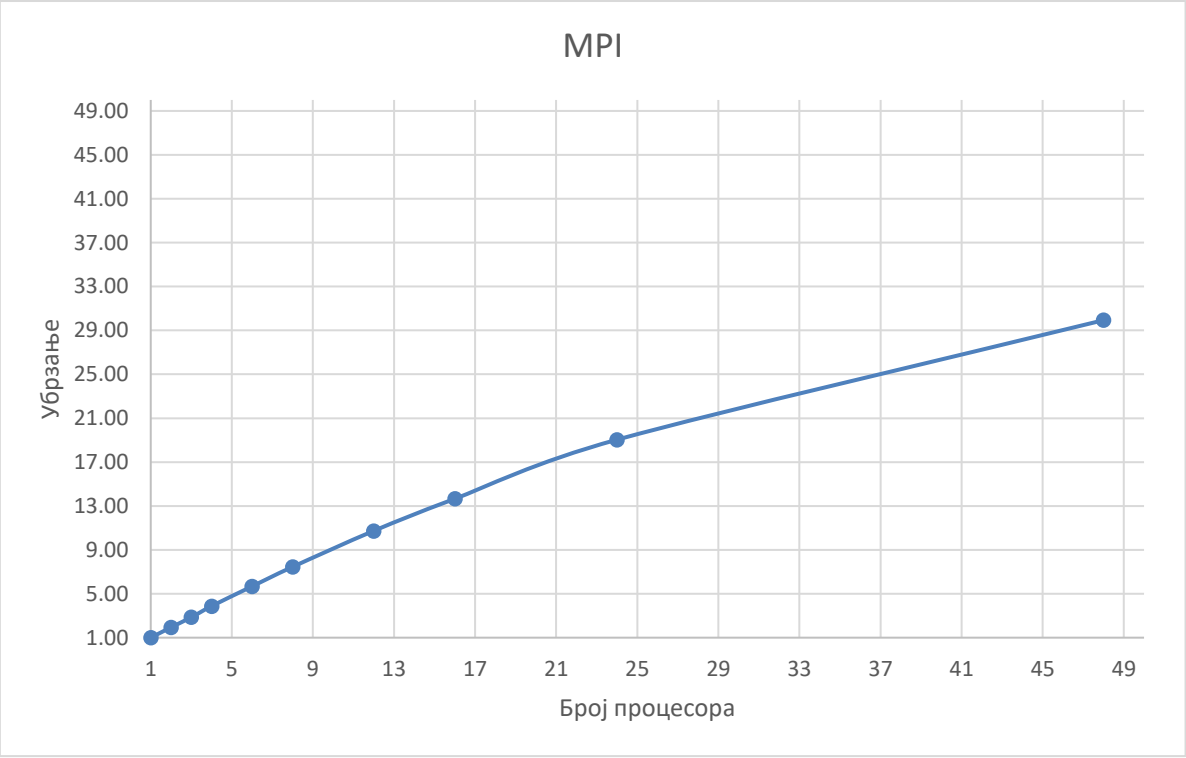
5.1. Убрзање

Разлике које смо добили коришћењем поменутих метода су очигледне уколико упоредимо време које је потребно за извршавање програма, тј. убрзања која добијамо.

Убрзање представља величину која описује колико пута је време извршавања програма краће у односу на време програма секвенцијалног програмирања.

За потребе тестирања је постављена вредност променљиве *Nstatistika* на 100, чиме се не добија довољно квалитетна дифракциона слика, али је за упоређивање убрзања била практичнија због предугог времена извршавања програма приликом коришћења већих вредности.

На следећем графикону (**Графикон 1**) је резултат убрзања који смо добили MPI методом.



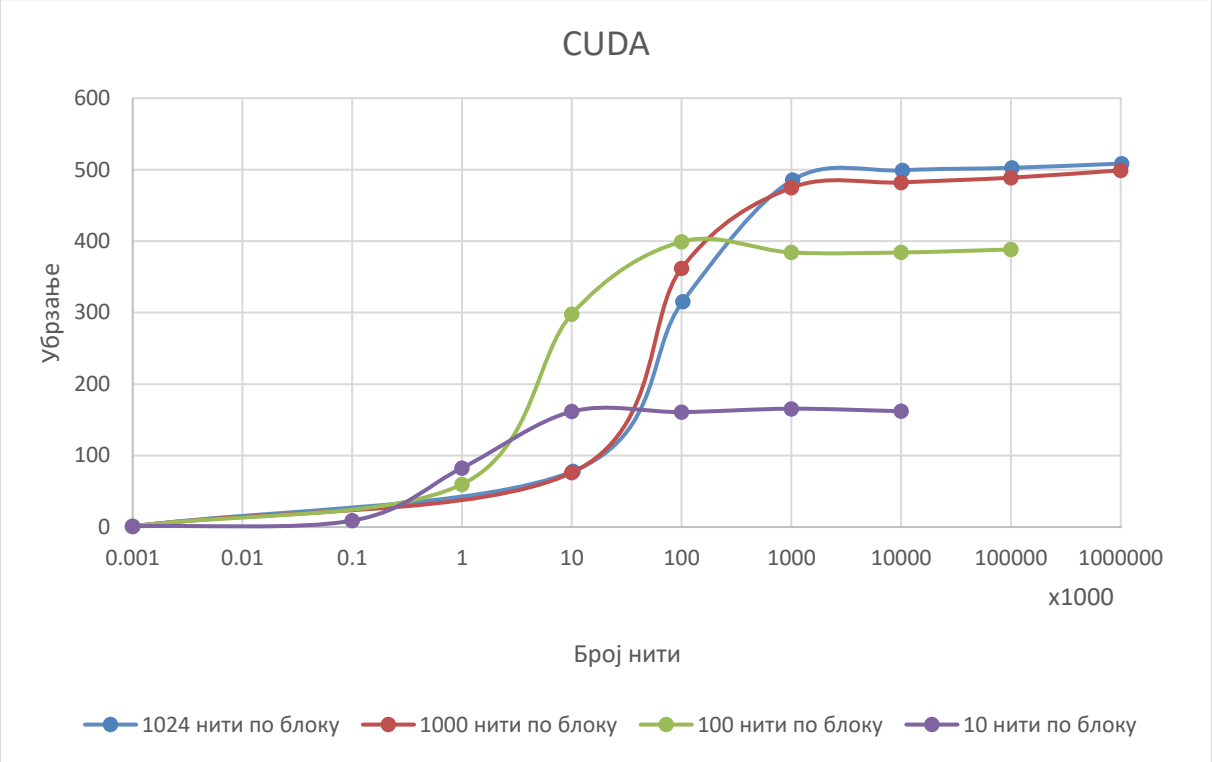
Графикон 1: *Убрзање добијено применом MPI методе*

Као што видимо, са повећањем броја чворова добијамо и веће убрзање. Због ограничености у броју чворова ово је резултат који смо добили експерименталним путем. Највеће убрзање које смо могли да добијемо у овом случају је приближно 30. Иако нисмо могли да испробамо шта би се десило са већим бројем чворова, то можемо да закључимо. Као што видимо, раст убрзања није линеаран што значи да са повећањем броја процесора убрзање све спорије расте, тако да ће у једном тренутку доћи до засићења и тада ће убрзање да опада са повећањем броја чворова. То се дешава због

брзине мреже, што више података шаљемо кроз мрежу спорије ће сваки чвор да дође до потребних података, тако да ће у једном тренутку доћи до тога да је спорије достављање потребних података него што би један чвор могао да обради те податке.

Тестирање је извршено на кластеру који се састоји из 5 чворова опремљених са по два процесора AMD Opteron 6272 на 2.1 GHz са по 16 језгара, нешто старије генерације. Иако је проблем релативно једноставан за паралелизацију, нисмо добили убрзања близу идеалних вредности. Разлог томе је сама архитектура поменутог процесора који заправо има 16 јединица за рад са целим бројевима, али само 8 јединица за рад у покретном зарезу, што је за наш проблем од круцијалног значаја.

Резултат који смо добили CUDA приступом дат је на следећем графикону (**Графикон 2**).



Графикон 2: *Убрзање добијено применом CUDA методе*

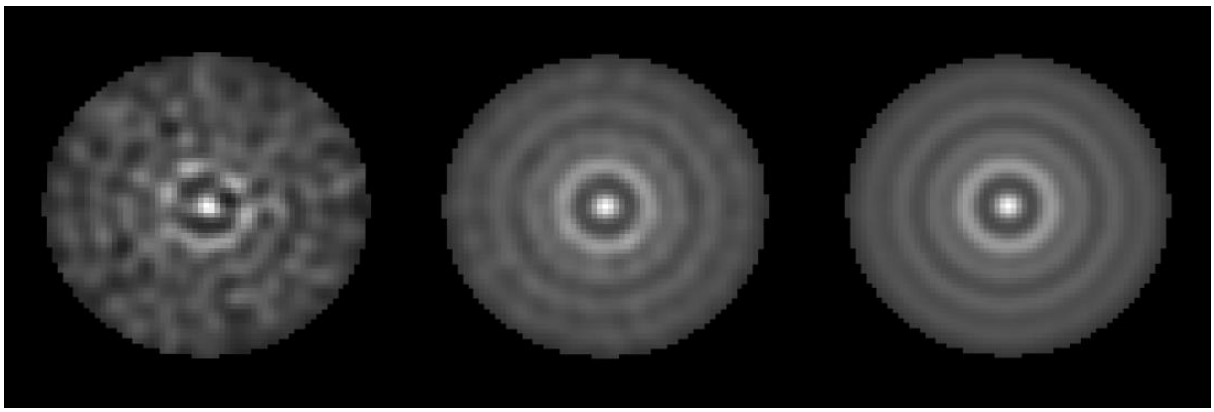
Видимо да је убрзање добијено методом која укључује графички процесор знатно озбиљније. За ове резултате била нам је потребна само једна графичка картица. Убрзање које смо добили је приближно 500 пута, што је знатно више од резултата добијених MPI методом. Као што видимо, ни овде убрзање није линеарно, јер у време треба да урачунамо и време потребно за копирање потребних података на графичку картицу. То би значило да са малим бројем нити не добијамо неко велико убрзање управо зато што је време уштеђено на израчунавању приближно времену копирања на графичку картицу. Са великим бројем нити убрзање све спорије расте, јер време утрошено на извршавање паралелног кода постаје мање од времена потребног да се сви потребни подаци прекопирају из главне у меморију графичке картице и обрнуто.

Одавде видимо да смо применом паралелизације омогућили извршавање програма више од 500 пута брже, што смо добили коришћењем CUDA приступа. MPI

методом смо такође успели да убрзамо извршавање, али ограничење у броју чворова је било велико јер је највећи број који смо могли да извучемо 48, тиме је убрзање било приближно 30.

5.2. Квалитет слике

Најбоља демонстрација реалне примене је резултат који можемо да добијемо за исто време извршавања програма коришћењем различитих приступа. На следећим сликама можемо упоредити резултате добијене на једном процесору, затим MPI приступом и на крају CUDA методом за приближно *35 минута* рачунања (**Слика 13**). Оно што се разликује је *Nstatistika* тако да се разликује број пролазака кроз прву спољашњу петљу у коду. Као што смо у претходном поглављу рекли, *Nstatistika* одређује квалитет дифракционе слике, са повећањем њене вредности слика се изоштрава. За исто време на једном процесору је могуће проћи кроз ту петљу само 3 пута, применом MPI методе кроз исту петљу је могуће проћи 50 пута, а уколико за исти задатак користимо CUDA методу тај број је чак 1000 (дате вредности за *Nstatistika* су приближне).



Слика 13: *Дифракционе слике (Један процесор, MPI и CUDA)*

Погледом на слику, можемо рећи да секвенцијалним извршавањем нисмо добили употребљив резултат, тј. оно што желимо да добијемо се ни не назире.

За резултат који смо добили MPI приступом већ можемо да кажемо да смо ближи решењу и да је слика доста јаснија, али опет није оно што желимо да добијемо.

CUDA метода се за дати проблем показала као најбоље решење, што се и очекивало на почетку, јер је потребан огроман број пута извршити исто израчунавање које се састоји од више простих операција за огроман број података, а ограничење у броју чворова је отежавајући фактор при коришћењу MPI методе, јер је највећи број чворова које смо могли да искористимо 48, што и није велики број у односу на огроман број нити графичке картице које смо имали на располагању. При израчунавању на графичкој картици смо скоро сваки пролаз петљи које смо користили за паралелизацију могли да рачунамо паралелно.

6. Закључак

У овом раду смо добијање дифракционе слике учинили приступачнијим, јер је могуће за разумно време добити добре резултате за разлику од оригиналног кода којим бисмо такве резултате чекали предуго. За неко време које нам је прихватљиво не бисмо имали жељене резултате, па чак ни приближне резултате који би нам били и донекле корисни. Оно што би могло даље да се уради је да се комбинују MPI и CUDA приступ, тако би се додатно убрзао код. За тај приступ би нам требале машине које имају графичке картице и повезане су у кластер.

Један од проблема које сам имала приликом израде пројекта био је тај да су чворови на кластеру често били заузети тако да је извршавање програма за већи број чворова било одложено некад и неколико дана, а коришћењем мањег броја чворова извршавање би се продужило. Када би посао (j*ob*) стигао на ред, заузимао би скоро све ресурсе, што би значило да други пројекат не бисмо могли да покренемо већ бисмо морали да сачекамо да се све заврши и да се ослободе ресурси.

Са већим бројем чворова на кластерима могли смо да добијемо боље резултате. Такође нам и брзина преноса података игра велику улогу тако да би се бржа мрежа показала боље, али то би било скупље решење. Са друге стране, графичке картице су се показале веома добро за решавање датог проблема, тј. добре резултате смо добили већ коришћењем само једне графичке картице. Комбинацијом ова два приступа, тачније коришћењем више чворова са више графичких картица могли бисмо да решавамо знатно сложеније проблеме којих реално има много више.

Литература

[1] Завршни рад - Развој модела простирања светлости по законима таласне оптике (Марко Милошевић, Индекс: 1037/19)

[2] https://curc.readthedocs.io/en/latest/programming/MPI-Fortran.html (10.2022.)

[3] https://imi.pmf.kg.ac.rs/moodle/file.php/127/Predavanja/IS1-predavanja-1.pdf (12.2022.)

[4] https://docs.nvidia.com/hpc-sdk/compilers/cuda-fortran-prog-guide/index.html (10.2022.)

Кратка биографија кандидата

Милица Стојановић рођена у Јагодини 20.04.1998. године. Основну школу ''Светозар Марковић'' завршила је у Лоћики, општина Рековац. Током основне школе учествовала је на републичким такмичењима из Физике, Хемије и Математике. Након чега је похађала Прву Крагујевачку Гимназију у СМ одељењу. Први пројекат, који је и мотивисао за упис информатике на Природно Математичком факултету у Крагујевцу, био је програмирање електронике за моделе авиона. Током студија учествовала је на такмичењу ''RoboMac'' у области роботике одржаном у Скопљу, Северна Македонија. Након завршене друге године студија била је на пракси у компанији ''Quantox'', а након треће године у компанији ''DM Dokumenten Management GmbH''. Затим у другом семестру четврте године била је на пракси у компанији ''Enjoying'', која је касније променила назив у ''Createq'', где је по завршеној пракси наставила са радом као стално запослена на ''Ifolor'' пројекту. Планови за будућност су одлазак у Шпанију где би наставила са даљим развијањем каријере.

Универзитет у Крагујевцу
Природно-математички факултет
Институт за математику и информатику

Завршни рад под називом

Паралелни и дистрибуирани прорачуни у моделима таласне оптике

одбрањен је _____.

МЕНТОР:

_____
др Милош Ивановић, ванр. професор, ПМФ Крагујевац

ЧЛАНОВИ КОМИСИЈЕ:

_____
___
др Ненад Стевановић, ванр. професор,
ПМФ Крагујевац

_____
___
др Бранко Арсић, доцент, ПМФ Крагујевац

Завршни рад је оцењен оценом _____.