

SpaceJam

Изследване на скалируемостта на n-body спрямо различен брой тела и итерации

Изготвила: Ирина Куртева, ФН: 81831, Курс: 3, Група: 1



July 7, 2021

ФМИ, СУ ”Св.Климент охридски”

София, България

Съдържание

[1. Увод 2](#_Toc76547399)

[1.1. Формулировка на проблема 2](#_Toc76547400)

[1.2. История на проблема 2](#_Toc76547401)

[1.3. Математика зад проблема 2](#_Toc76547402)

[2. Анализ на n-body 3](#_Toc76547403)

[2.1. Различни алгоритми 3](#_Toc76547404)

[2.2. Източник [1] 3](#_Toc76547405)

[2.3. [3] 4](#_Toc76547406)

[2.4. Сравнение с източниците 4](#_Toc76547407)

[3. Използвани технологии 5](#_Toc76547408)

[3.1. UML диаграма 5](#_Toc76547409)

[3.2. Анимация 6](#_Toc76547410)

[4. Имплементация на алгоритъма 7](#_Toc76547411)

[5. Инструкции за употреба и параметри на програмата 7](#_Toc76547412)

[6. Анализ 8](#_Toc76547413)

[6.1. Харектеристики на тестовата машина 8](#_Toc76547414)

[6.2. Резултати 8](#_Toc76547415)

[6.3. Анализ на данните 10](#_Toc76547416)

[7. Заключение 18](#_Toc76547417)

[8. Източници 19](#_Toc76547418)

# Увод

## Формулировка на проблема

N-body се отнася за проблемът за движението на индивидуалните тела, при положение, че всички тела взаимодействат гравитационно помежду си. Търсенето на решение на този проблем е породено от желанието на хората да разберат движението на слънцето, планетите и видимите звезди. С откриването на системи от много звезди, намирането на решение на този проблем е станало все по-належащо, за да бъде възможно извличането на повече информация за тях. Проблемът е решен напълно за две тела, но има рестрикции, които пречат за откриването на точно решение при три и повече тела. Това, което пречи на откриването на общо решение е фактът, че гравитационното поле образувано от две тела има няколко особени точки (точки на Лагранж). Ако бъде поставено в тях тяло с достатъчно по-малка маса от тази на останалите 2 тела, то ще стои неподвижно и не влияе на движението на другите. Това пречи на стандартното решение на проблема, което съм имплементирала и аз. То се уповава на факта, че всеки 2 тела взаимодействат със себе си и колкото и малко тяло да имаме, то може да измени траекторията на някое друго тяло.

## История на проблема

Проблемът n-body се е зародил през 17 век, като първоначално е формулиран от Исак Нютон, който, изследвайки гравитационните взаимодействия, е установил, че не е достатъчно да знаем началната позиция, скоростта и масата на обектите, за да установим как ще взаимодействат помежду си с течение на времето. През 19 век, когато крал Оскар II на Швеция обявява награда за намиране на решение на проблема, интересът към него се засилва. Това е довело до успешното решение на проблема за две тела от Poincare и за ограничената версия за три тела от Karl Fritof.

## Математика зад проблема

Проблемът се състои в това да се пресметнат взаимодействията между n тела с маси mi за и начални координати xi и yi. Общата сила, която действа върхи дадено тяло е равна на сума от силите, които действат върху даденото тяло спрямо всяко от останалите тела, тоест Σ. Гравитационната сила, с която тяло i действа на тяло j, се задава чрез формулата на Нютон за гравитацията, тоест , където е разстоянието между телата, а γ е гравитационната константа. От това, че F = m \* a, тоест и формулата за гравитацията, получаваме , откъдето тъй като знаем масите на телата и разстоянието между тях, можем да намерим и ускорението ai. От него по формулата да намерим скоростта на тялото след момент . За да изчислим новата позиция на тялото i, можем да използваме формулата за движение на тяло със скорост v в дадена посока, а именно . Чрез този метод можем да получим движението на телата за една итерация с време , като чрез повторение на горните стъпки N на брой пъти, можем да изчислим движенията на телата за произволни интервали от време. Алгоритъмът, който съм имплементирала, разчита на това, че се намираме в 2д пространство и гравитационната сила на взаимодействие между 2 тела се пресмята по формулата . [2]

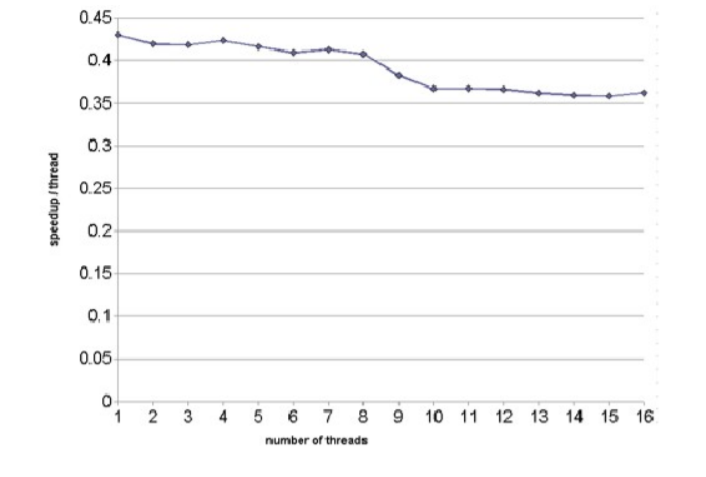
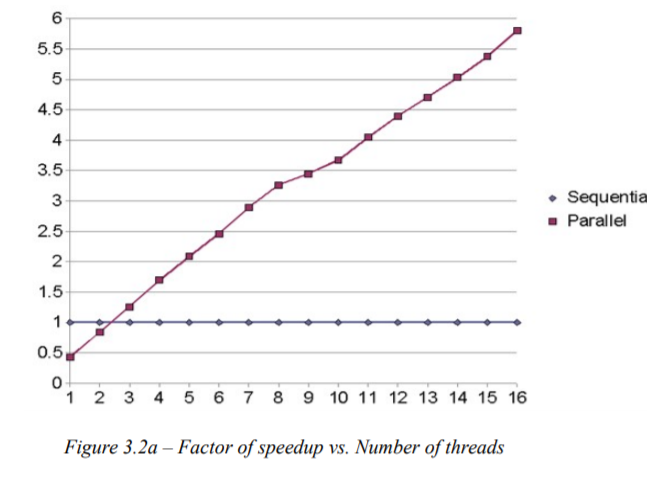
# Анализ на n-body

## Различни алгоритми

Има различни алгоритми, които решават n-body проблема. Този, който съм реализирала аз, е наивният, при който се изчислява силата на взаимодействие на всяко тяло с всяко от останалите. След това на база тези сили се изчислява скоростта на тялото в момента и съответно позицията му в следващия момент от време. Сложността на този алгоритъм е , което както очаквах не беше най-оптималният от времева гледна точка алгоритъм, но се оказа удобен за паралелизация. Други алгоритми, на които попаднах по време на проучването са Barnes-Hut, Fast Multipole Method и Parallel Multipole Tree algorithm. Същността на тези алгоритми е създаването и обработката на дървета, като сложностите им са съответно , и . И трите алгоритъма правят апроксимации при изчисленията. Това ги прави по-неточни, но ако имаме група тела на голямо разстояние от даденото, няма особено значение дали ги разглеждаме като отделни обекти или като един голям с маса равна на сумата на масите на всички тела и координати в центъра на тежестта им. Затова и тези алгоритми се използват по-често поради по-ниската им сложност.

## Източник [1]

Използваната машина за тестове в този случай е Sun Fire T2000 Server с 8 ядра и 16GB RAM памет. Този документ е от 2008, така че машината, на която е тествана реализацията е с доста по-слаби параметри от това, с което разполагам аз днес. Забелязва се, че при 16 нишки се достига ускорение 6, но това е при положение, че при една има забавяне от 2 пъти между паралелния и последователния алгоритъм. Ако приравня изчисленията към моите това ще съответсва на ускорение 12 при 16 нишки, което е малко по-малко от моето. Това е нормално, като се има предвид, че сървърът е с по-малко ядра и вероятно по-малко кеш от този, който аз използвах, е нормално. Силно впечатление прави плавното спадане на ефективността от 0.45 при една нишка до 0.35 при 16, което ако трябва да сравня с моите данни е доста по-малка разлика. Следователно ползването на език, специализиран за паралелно програмиране помага за оптимизацията на алгоритъма.



*Фигура 1: Ускорение спрямо броя нишки Фигура 2: Ефективност спрямо броя нишки*

## [3]

Този източник не е най-подходящият за сравнение, но съм го използвала при проучването, затова реших да го спомена. Тук са използвани доста по-сложни математически методи за работа в 3D пространството и Message Passing, а не нишки, каквито съм използвала аз. Освен това, тъй като са използвани алгоритмите със сложност и , тестовете са правени с 20000000 тела, докато SpaceJam е тестван с най-много 80000 тела.

## Сравнение с източниците

За сравнение основно съм използвала [1], тъй като там реализацията на алгоритъма е същата като моята. Разликата е използваната машина за тестовете и езикът за реализация – Farrell е използвал ZPL, което е език, специализиран за паралелно програмиране, докато аз използвах Java и нишки. В документацията липсват данни за точното време за итерации. Има представени само такива за ускорението и ефективността, затова и съответната клетка в таблицата е празна. Също така времето за една итерация на SpaceJam е толкова малко, тъй като е за 8000 тела, понеже това е броят тела най-близък до този на [1].

|  |  |  |
| --- | --- | --- |
|  | [1] | SpaceJam |
| Машина | Sun Fire T2000 Server | 40-intel |
| # тела | 2048 | 8000 |
| Разпределение | Равномерно | Равномерно |
| Време за изчисление на една итерация | - | 0.004223 секунди |
| Време за изчисления | 97% | 97% |
| Останало време | 3% | 3% |

*Фигура 3: Таблица сравняваща проучените алгоритми*

# Използвани технологии

SpaceJam е реализиран на езика Java с използването на нишки за параралелизацията и mutex, counter и семафор за синхронизацията на нишките на всяка итерация, тъй като в противен случай изчисленията на новите координати на всяка стъпка няма да са коректни.

За визуализацията използвах езика R и библиотеките ggplot и gganimate, като кодът ми генерираше svg файл с позициите на обектите, теглата им, номер на обекта и итерация. По тези данни генерирам анимация, а цветът на точките съответства на теглата им.

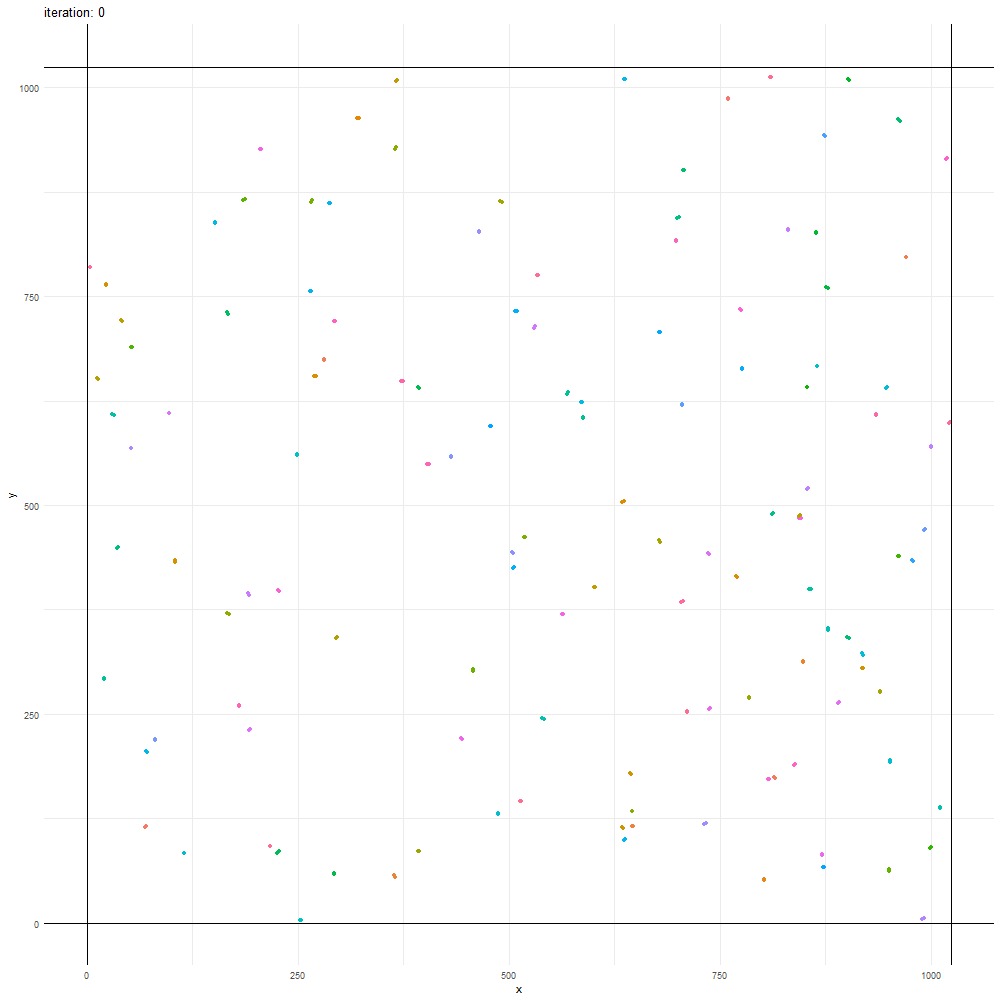
## UML диаграма

Diagram

Description automatically generated

*Фигура 4: UML диаграма на алгоритъма*

## Анимация



*Фигура 5: Анимация*

# Имплементация на алгоритъма

Първата стъпка при реализацията на проекта беше да проуча математиката зад n-body проблема, въпреки че тя не беше нова за мен, тъй като в училище съм се занимавала с астрономия, което ме подтикна към избора именно на този проект. След като вече се бях уверила, че разбирам как работи всичко, преминах и към самата имлементация. Първо реализирах алгоритъма последователно, като по-надолу съм направила сравнение с паралелния вариант на алгоритъма. След това направих разделянето на телата между нишките, като това се случва по следния начин: на всяка нишка задаваме начален и краен индекс в масива с тела, като това са телата, за които той ще пресмята взаимодействията с останалите. Броят тела на нишка намираме като (общ брой тела/брой нишки), като ако това деление дава някакъв остатък m този остатък се разпределя като броят тела на първите m нишки се увеличава с единица. След самото разделяне, всяка нишка има цикъл по броя итерации, като в началото на всяка прави локално копие на глобалния масив с тела. В края на всяка итерация нишките се изчакват и след това всяка копира новите координати и скорости на своите тела в глобалния масив. От него в началото на следващата итерация всички нишки ще вземат обновените координти. Първоначално се опитах да направя реализация, използваща само споделена памет и един общ масив с тела, но това води до редица проблеми – достъпът до даден елемент в масива се осъществява само от една нишка едновременно, тоест нишките си пречат взаимно. Иначе може някоя нишка да прочете вече обновените от друга координати за дадено тяло, вместо текущите, и това да доведе до грешки в изчисленията.

# Инструкции за употреба и параметри на програмата

За да може да работи програмата коректно, е необходимо на машината, на която се изпълнява, да има инсталирана Java, версия поне 14, тъй като съм я компилирала с такава версия и не тръгва с по-ниска версия. Кодът, обаче, би трябвало да може да се компилира и с по-стара версия на Java, тъй като не използва неща, специфични за новите версии.

След като вече имаме jar файла, програмата може да се изпълни като се извика от командния ред по следния начин:

java -jar path bodies threads iterations maxX maxY minStartVelocity maxStartVelocity minMass maxMass deltaT,

където:

1. bodies – брой тела
2. threads – брой нишки
3. iterations – брой итерации на програмата
4. maxX – максимална координата по X
5. maxY – максимална координата по Y
6. minStartVelocity – минимална начална скорост на дадено тяло
7. maxStartVelocity – максимална начална скорост на дадено тяло
8. minMass – минимална маса на дадено тяло
9. maxMass – максимална маса на дадено тяло
10. deltaT – времето, което е изминало при една итерация на алгоритъма; колкото по-малко е това число, толкова повече итерации ще са необходими, за да обхванем даден интервал от време, но от друга страна самите изчисления ще станат по-точни
11. path – път до jar файла с кода на програмата

# Анализ

## Харектеристики на тестовата машина

Машината, която използвах за тестване на проекта, има 2 процесора Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz, всеки от които има по 10 ядра, всяко от които поддържа до 2 нишки. Това прави общо 20 ядра и 40 нишки, което означава, че е смислено да тествам програмата си с до 40 нишки, тъй като оттам насетне машината няма да има достатъчно хардуер за истински паралелното изпълнение на всички нишки.

Тестовата машина има 640KB за L1i и L1d кеш, 5MB за L2 кеш и 50MB за L3 кеш. Откъм RAM тя разполага с 64GB памет.

## Резултати

Засичах времето за работата на програмата от пускането на първата нишка до приключването на последната. Правех по 5 опита за всеки брой тела и итерации, като от тях избирах трите най-добри, тъй като поради натовареността на сървъра част от опитите отнемаха много повече време от останалите опити със същите параметри. Премахнах колонката с грануларност, тъй като при мен грануларността се определя еднозначно от броя тела и броя нишки, а поради хомогенността на изчисленията, няма как да се получи дисбаланс в работата на отделните нишки, за да е смислено изследването на грануларност. Така параметрите, които съм изследвала и представила в таблицата са следните:

1. p – брой нишки
2. It – брой итерации
3. N – брой тела
4. Tp(1), Tp(2), Tp(3) – трите резултата от опитите
5. Tp – най-малкото време от трите
6. Sp – ускорението на програмата при изпозването на p нишки
7. Ep – ефикасността на програмата при използването на p нишки

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | ***p*** | ***It*** | *N* | ***Тp* (1)** | ***Тp* (2)** | ***Тp* (3)** | ***Тp=*min(*Tp*(*i*))** | ***Sр* = *T*1*/Tp*** | ***Eр* = *Sp/p*** |
| **1** | 1 | 50 | *8000* | ***210534*** | ***211205*** | ***211038*** | **210534** | 1 | 1 |
| **2** | 2 |  |  | *107662* | *107599* | *106588* | **106588** | 1,975213 | 0,98760648 |
| **3** | 4 |  |  | *53866* | *54226* | *53845* | **53845** | 3,9100009 | 0,97750023 |
| **4** | 8 |  |  | *28148* | *28356* | *28902* | **28148** | 7,4795367 | 0,93494209 |
| **5** | 10 |  |  | *22922* | *23085* | *22951* | **22922** | 9,1848006 | 0,91848006 |
| **6** | 16 |  |  | *16836* | *16151* | *16294* | **16151** | 13,035354 | 0,81470962 |
| **7** | 20 |  |  | *16898* | *17657* | *16388* | **16388** | 12,846839 | 0,64234196 |
| **8** | 32 |  |  | *15039* | *14914* | *15465* | **14914** | 14,116535 | 0,44114171 |
| **9** | 40 |  |  | *13740* | *13510* | *14002* | **13510** | 15,583568 | 0,38958919 |
| **10** | 1 | 100 | *8000* | ***422553*** | ***420581*** | ***421542*** | **420581** | 1 | 1 |
| **11** | 2 |  |  | *215038* | *210778* | *210642* | **210642** | 1,9966626 | 0,99833129 |
| **12** | 4 |  |  | *107250* | *107489* | *107078* | **107078** | 3,9278003 | 0,98195007 |
| **13** | 8 |  |  | *57400* | *56881* | *57367* | **56881** | 7,3940507 | 0,92425634 |
| **14** | 10 |  |  | *46086* | *45280* | *45737* | **45280** | 9,2884496 | 0,92884496 |
| **15** | 16 |  |  | *32761* | *33569* | *32559* | **32559** | 12,917504 | 0,80734398 |
| **16** | 20 |  |  | *33335* | *33986* | *33511* | **33335** | 12,616799 | 0,63083996 |
| **17** | 32 |  |  | *28276* | *29398* | *28246* | **28246** | 14,889931 | 0,46531035 |
| **18** | 40 |  |  | *27717* | *27859* | *26820* | **26820** | 15,681618 | 0,39204045 |
| **19** | 1 | 200 | 8000 | ***852652*** | ***844662*** | ***845123*** | **844662** | 1 | 1 |
| **20** | 2 |  |  | *430123* | *430135* | *404012* | **404012** | 2,0906854 | 1,04534271 |
| **21** | 4 |  |  | *215534* | *215894* | *214910* | **214910** | 3,9303057 | 0,98257643 |
| **22** | 8 |  |  | *114223* | *114689* | *111126* | **111126** | 7,6009395 | 0,95011743 |
| **23** | 10 |  |  | *91601* | *92855* | *90602* | **90602** | 9,3227743 | 0,93227743 |
| **24** | 16 |  |  | *64870* | *65515* | *66173* | **64870** | 13,020842 | 0,81380261 |
| **25** | 20 |  |  | *67020* | *67978* | *67020* | **67020** | 12,603133 | 0,63015667 |
| **26** | 32 |  |  | *57391* | *60254* | *61820* | **57391** | 14,717674 | 0,4599273 |
| **27** | 40 |  |  | *53962* | *54574* | *54581* | **53962** | 15,652904 | 0,3913226 |
| **28** | 1 | 50 | 12000 | ***477862*** | ***476401*** | ***478734*** | **476401** | 1 | 1 |
| **29** | 2 |  |  | *235303* | *213278* | *238223* | **213278** | 2,2337091 | 1,11685453 |
| **30** | 4 |  |  | *120013* | *120346* | *120151* | **120013** | 3,9695783 | 0,99239457 |
| **31** | 8 |  |  | *57264* | *53476* | *52403* | **52403** | 9,0911017 | 1,13638771 |
| **32** | 10 |  |  | *47014* | *46528* | *45413* | **45413** | 10,49041 | 1,04904102 |
| **33** | 16 |  |  | *33079* | *33934* | *31936* | **31936** | 14,917366 | 0,93233537 |
| **34** | 20 |  |  | *34027* | *33371* | *34239* | **33371** | 14,275898 | 0,71379491 |
| **35** | 32 |  |  | *27529* | *27567* | *27039* | **27039** | 17,619032 | 0,55059474 |
| **36** | 40 |  |  | *26394* | *26180* | *26947* | **26180** | 18,197135 | 0,45492838 |
| **37** | 1 | 50 | 12000 | ***947480*** | ***949253*** | ***955351*** | **947480** | 1 | 1 |
| **38** | 2 |  |  | *508323* | *454273* | *480101* | **454273** | 2,0857062 | 1,04285309 |
| **39** | 4 |  |  | *239736* | *241593* | *234131* | **234131** | 4,0467943 | 1,01169858 |
| **40** | 8 |  |  | *113786* | *135417* | *112243* | **112243** | 8,4413282 | 1,05516602 |
| **41** | 10 |  |  | *90941* | *87089* | *87505* | **87089** | 10,879445 | 1,08794452 |
| **42** | 16 |  |  | *64257* | *63483* | *62932* | **62932** | 15,055616 | 0,94097597 |
| **43** | 20 |  |  | *68447* | *69131* | *71245* | **68447** | 13,842535 | 0,69212676 |
| **44** | 32 |  |  | *53827* | *54335* | *53495* | **53495** | 17,711562 | 0,55348631 |
| **45** | 40 |  |  | *52435* | *51779* | *55574* | **51779** | 18,298538 | 0,45746345 |
| **46** | 1 | 200 | 12000 | ***1890685*** | ***1866811*** | ***1903762*** | **1866811** | 1 | 1 |
| **47** | 2 |  |  | *811950* | *918117* | *868498* | **811950** | 2,2991699 | 1,14958495 |
| **48** | 4 |  |  | *478912* | *479431* | *479405* | **478912** | 3,8980251 | 0,97450628 |
| **49** | 8 |  |  | *214394* | *204110* | *215320* | **204110** | 9,1461026 | 1,14326282 |
| **50** | 10 |  |  | *178755* | *174495* | *173474* | **173474** | 10,76133 | 1,07613302 |
| **51** | 16 |  |  | *131619* | *125038* | *127926* | **125038** | 14,929949 | 0,93312183 |
| **52** | 20 |  |  | *138295* | *137881* | *138347* | **137881** | 13,539291 | 0,67696456 |
| **53** | 32 |  |  | *109021* | *108802* | *109884* | **108802** | 17,157874 | 0,53618356 |
| **54** | 40 |  |  | *113490* | *111160* | *105518* | **105518** | 17,691872 | 0,44229681 |
| **55** | 1 | 25 | 20000 | ***659415*** | ***658754*** | ***658680*** | **658680** | 1 | 1 |
| **56** | 2 |  |  | *306599* | *264714* | *284778* | **264714** | 2,4882704 | 1,24413518 |
| **57** | 4 |  |  | *167485* | *166577* | *167134* | **166577** | 3,9542074 | 0,98855184 |
| **58** | 8 |  |  | *70387* | *70869* | *70548* | **70387** | 9,357978 | 1,16974725 |
| **59** | 10 |  |  | *58403* | *59238* | *57642* | **57642** | 11,427084 | 1,14270844 |
| **60** | 16 |  |  | *38732* | *40126* | *40666* | **38732** | 17,006093 | 1,06288082 |
| **61** | 20 |  |  | *39483* | *35758* | *36679* | **35758** | 18,420493 | 0,92102467 |
| **62** | 32 |  |  | *30868* | *30988* | *33829* | **30868** | 21,338603 | 0,66683135 |
| **63** | 40 |  |  | *28470* | *28972* | *32186* | **28470** | 23,135933 | 0,57839831 |
| **64** | 1 | 5 | 80000 | *2114424* | *2106690* | *2107234* | **2106690** | 1 | 1 |
| **65** | 2 |  |  | *834613* | *830629* | *965000* | **830629** | 2,5362587 | 1,26812933 |
| **66** | 4 |  |  | *485971* | *495440* | *510890* | **485971** | 4,3350118 | 1,08375294 |
| **67** | 8 |  |  | *221161* | *221604* | *226914* | **221161** | 9,5255945 | 1,19069931 |
| **68** | 10 |  |  | *181444* | *189749* | *181092* | **181092** | 11,633258 | 1,16332582 |
| **69** | 16 |  |  | *116615* | *121658* | *116443* | **116443** | 18,092028 | 1,13075174 |
| **70** | 20 |  |  | *97242* | *131709* | *97023* | **97023** | 21,713305 | 1,08566525 |
| **71** | 32 |  |  | *83351* | *93021* | *88127* | **83351** | 25,274922 | 0,7898413 |
| **72** | 40 |  |  | *86552* | *75925* | *77551* | **75925** | 27,746987 | 0,69367468 |

*Фигура 6: Таблица с резултатите*

## Анализ на данните

Първата стъпка при анализа беше да генерирам графика на ускорението и да видя докъде то е линейно и как се влияе от различния брой тела. За прегледност съм поставила и отделни графики, а не само обща, тъй като тя е доста претрупана с информация и е трудно да се изведат всички заключения. Както се забелязва от трите графики по-долу, ускорението е линейно или почти линено до около 16 нишки, а след това започва да спада и да се приближава към права, успоредна на абсцисата, както и се очаква според закона на Амдал.

*Фигура 7*

*Фигура 8*

*Фигура 9*

*Фигура 10*

Реших да проуча дали има връзка между ускорението и броя итерации, каквато при моята реализация не би трябвало да съществува, тъй като изчисленията на всяка стъпка са едно и също количество и би трябвало при тройно повече итерации, просто да има тройно повече изчисленията, но пропорциите във времето между различния брой нишки да остават константни. Въпреки това изследвам и този аспект на програмата, за да съм убедена, че изводите, които правя, са коректни, защото те са на база различен брой тела и различен брой итерации. Ако пусна твърде малко тела за твърде малко итерации, може точността, с която пресмятам времето (милисекунди) да не е достатъчна. Както се вижда от графиките за 8000 и 12000 тела и съответно 50, 100 и 200 итерации, няма особена разлика в ускорението при различния брой итерации (различното време в света на SpaceJam). Разминаванията в графиките по-скоро се дължат на различната натовареност на сървъра в дадените моменти и на факта, че не само кодът и ускорението му влияят на времето за изпълнение на програмата, а има и фактори, независещи от нас, както се вижда и от таблицата, времето за изпълнение при фиксиран брой тела, нишки и итерации не е константно, а варира.

*Фигура 11*

*Фигура 12*

След като вече установих, че броят итерации не влияе на ускорението, насложих графиките за ускорението при различен брой тела една върху друга (тук броят итерации е различен, но както казах, това не влияе) и лесно се забелязва, че при повече тела ускорението е линейно по-дълго. Това, според мен, се получава, защото при малко тела, като ги разпределим между много нишки, не се постига достатъчно натоварване на всяка, за да се забележи ползата от наличието на повече нишки.

*Фигура 13*

Интересен феномен, който наблюдавах, е спадът на ускорението при 20 нишки, който е най-ясно изразен при 12000 тела. Това, според мен, се получава, защото машината, на която тествах, е с 2 процесора с по 10 ядра и 2 нишки на ядро и когато броят нишки стане 20 и повече, вече нишките почват да се разпределят между двата процесора и започва да се използва NUMA. Комуникацията между двата процесора води до забавяне на синхронизацията. Въпреки че 20 нишки все още се побират в капацитета на един процесор, предполагам, че диспечерът на процеси се опитва да разпредели натоварването между двата процесора и съответно избягва ситуации, в които единият процесор е натоварен на 100%, а другият на 0%.

Другият параметър на програмата, който изследвах, е ефектовността при различен брой тела, тъй като щом ускорението не се влияе от броя итерации, то и ефективността няма да се влияе. Отново имам отделни графики за прегледност и обща, от която направих изводите. Вижда се тенденция за линеен спад на ефективността, независимо от броя тела.

*Фигура 14*

*Фигура 15*

*Фигура 16*

*Фигура 17*

От долната графика, на която е показана ефективността съответно при 8000, 12000, 20000 и 80000 тела се забелязва, че както ускорението остава линейно или почти линейно за повече нишки при по-голям брой тела, така и ефиктивността при повече тела е по-добра. При 8000 тела, тя спада до 0.39, докато при 80000 тела най-ниската стойността на ефективността при 40 нишки е 0.69.

*Фигура 18*

# Заключение

Цялостно се вижда, че алгоритъмът постига добро ускорение при много тела. Получава се свръхлинейна аномалия между 1 и 2 нишки, което вероятно се дължи на факта, че при повече нишки процесорът успява да събере данните в кеша и съответно достъпът до тях е по-бърз, докато при една нишка не успява да се възползва напълно от кеша. Оптимизация, която мога да направя за в бъдеще е процесорът да смята взаимодействията между една група тела с размер, събиращ се в кеша и чак след това да преминава към другите тела с цел по-бързо пресмятане при много тела и малко нишки.

# Източници

[1] Brandon Farrell, Parallel N-Body Simulation Using Problem Space Promotion, University of Washington, CSE 497 - Undergraduate Research Seminar - Spring Quarter 2008 ( <https://s3-us-west-2.amazonaws.com/www-cse-public/ugrad/thesis/TR08-02_Farrell.pdf> )

[2] Damon Binder, Simulating Gravity, ( <http://djbinder.com/articles/NBody/> )

[3] P. E. Kyziropoulos, C. K. Filelis-Papadopoulos, and G. A. Gravvanis, Parallel N-Body Simulation Based on the PM and P3M Methods Using Multigrid Schemes in conjunction with Generic Approximate Sparse Inverses, 2015, (<https://www.hindawi.com/journals/mpe/2015/450980/> )

[4] Sverre J. Aaeseth, Gravitational N-Body simulations, Institute of Astronomy, University of Cambridge, 2003