Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Softvérový rámec pre COR model výpočtov trajektórii častíc v magnetosfére Zeme

Diplomová práca

2019 **Daniel Gecášek**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Softvérový rámec pre COR model výpočtov trajektórii častíc v magnetosfére Zeme

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	RNDr. Pavol Bobík, PhD.
Konzultant:	doc. Ing. Ján Genči, PhD. , Ing. Michal Vrábel

Košice 2019 **Daniel Gecášek**

Názov práce:	Softvérový rámec pre COR model výpočtov trajektórii častíc v magnetosfére Zeme
Pracovisko:	Katedra počítačov a informatiky, Technická univerzita v Košiciach
Autor:	Daniel Gecášek
Školiteľ:	RNDr. Pavol Bobík, PhD.
Konzultant:	doc. Ing. Ján Genči, PhD. , Ing. Michal Vrábel
Dátum:	26. 4. 2019
Kľúčové slová:	kozmické žiarenie, magnetosféra Zeme, Tsyganenko 96, Tsyganenko 05, odrezávacia magnetická rigidita

Abstrakt: Práca bola zameraná na výrazné vylepšenie funkcionalít systému COR (Cut-Off Rigidity), ktorý je určený na simulácie trajektórií kozmického žiarenia v magnetosfére Zeme. V rámci práce sa hlavne vylepšili a pridali do systému nasledujúce funkcionality a moduly. Systém COR sme transformovali na distribuovaný systém ktorý je možné konfigurovať tak, aby bežal na viacerých nódoch. Boli pripravené dve verzie optimalizácie času výpočtov založené na odhade štartovacej rigidity simulácie. Ďalšou pridanou funkcionalitou bolo pridanie modelu výpočtov trajektórií kozmického žiarenia pre interval rokov od roku 0 až do roku 1968. Okrem tohto modelu bol pridaný aj model výpočtu, ktorý sa zastaví po výpočte prvej dovolenej trajektórie. COR systém sme upravili do podoby v ktorej je pridávanie nových modelov jednoduché. V rámci vylepšenia vizualizačnej časti systému bol pridaný modul pre výpočet a vizualizáciu jedinej trajektórie kozmického žiarenia v magnetosfére a modul vizualizácie funkcie priepustnosti. Pre vylepšenie presnosti výpočtu intenzity kozmického žiarenia bol systém rozšírený o knižnice spektier kozmického žiarenia vo vzdialenosti jednej astronomickej jednotky od Slnka.

Thesis title:	Software framework for COR model of particle trajectory calculations in Earth's magnetosphere
Department:	Department of Computers and Informatics, Technical University of Košice
Author:	Daniel Gecášek
Supervisor:	RNDr. Pavol Bobík, PhD.
Tutor:	doc. Ing. Ján Genči, PhD. , Ing. Michal Vrábel
Date:	26. 4. 2019
Keywords:	cosmic rays, Earth's magnetosphere, Tsyganenko 96, Tsyganenko 05, magnetic cutoff rigidity

Abstract: The work was focused on significant improvement of COR (Cut-Off Rigidity) system functionality, which is designed for simulation of cosmic ray trajectories in Earth's magnetosphere. The work has mainly improved and added to the system the following functionalities and modules. We have transformed the COR system into a distributed system that can be configured to run on multiple nodes. Two versions of computation time optimization were prepared based on the simulation's start rigidity estimation. Another added feature was the addition of a cosmic ray trajectory model for the years from 0 to 1968. In addition to this model, a model of calculation was added that stops after calculating the first allowed trajectory. We have modified the COR system into a form where adding new models is simple. To improve the visualization part of the system, a module for calculating and visualizing a single trajectory of cosmic rays in the magnetosphere and a visualization module of the transmission function was added. To improve the accuracy of cosmic ray simulation, the system was expanded to include libraries of cosmic ray spectra at a distance of one astronomical unit from the sun.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**

Študijný program: **Informatika**

Názov práce:

Softvérový rámec pre COR model výpočtov trajektórii častíc v magnetosfére Zeme

Framework for COR model for particles trajectory simulations in the Earth magnetosphere

Študent:	**Bc. Daniel Gecášek**
Školiteľ:	**RNDr. Pavol Bobík, PhD.**
Školiace pracovisko:	**Inštitút experimentálnej fyziky Slovenskej Akadémie Vied, Oddelenie kozmickej fyziky**
Konzultant práce:	**doc. Ing. Ján Genči, PhD., Ing. Michal Vrábel**
Pracovisko konzultanta:	**Katedra počítačov a informatiky**

Pokyny na vypracovanie diplomovej práce:

1. Študent aktualizuje model COR podľa odozvy prvých užívateľov. Pripraví nasadenie COR modelu ako distribuovaného systému.
2. Študent spracuje a vyhodnotí rôzne možnosti optimalizácie času multismerových výpočtov trajektórii kozmického žiarenia (KŽ) pre rôzne polohy na Zemi, respektíve vnútri magnetosféry.
3. Do modelu COR pridá model magnetického poľa umožňujúci výpočty trajektórii KŽ pre obdobia pred rokom 1950 až po začiatok nášho letopočtu (rok 0 n.l.).
4. Do modelu COR pridá modul pre výpočet jednej trajektórie KŽ v magnetosfére a jej vizualizácie.
5. Do modelu COR pridá katalógovú časť so spektrami kozmického žiarenia na TAU, a pripraví jej prepojenie na výpočet intenzít KŽ vnútri magnetosféry.
6. Podľa pokynov vedúceho práce pripraví dokumentáciu k realizovaným modulom.

Jazyk, v ktorom sa práca vypracuje: slovenský

Termín pre odovzdanie práce: 26.04.2019

Dátum zadania diplomovej práce: 31.10.2018

doc. Ing. Jaroslav Porubän, PhD.
vedúci garantujúceho pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 26.4.2019 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce a konzultantovy za ich čas a odborné vedenie počas riešenia mojej záverečnej práce.

Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

Obsah

1 Motivácia **1**

2 Systém COR **3**

2.1 Popis webového rozhrania 3

2.2 Opis databázy 4

2.2.1 Štruktúra tabuliek v databáze 6

2.3 Opis výpočtového systému 6

2.3.1 Opis modulu na spúšťanie úloh 8

2.4 Nedostatky používateľského rozhrania 10

2.4.1 Prístupnosť a administrácia 10

2.4.2 Ovládacie prvky rozhrania 11

2.4.3 Zabezpečenie prístupu na podstránky 11

2.5 Nedostatky výpočtového systému 12

2.5.1 Optimalizácia výpočtu viacerých vertikálnych trajektórií . . 12

2.5.2 Rozšíriteľnosť a spracovanie chýb 12

2.5.3 Chybové hlášky 13

2.5.4 Výber zdroja dát 13

3 Optimalizácia času výpočtov v systéme COR **15**

3.1 Odhad urýchlenia výpočtu zavedením optimalizácie spodnej odrezávacej rigidity 16

3.2 Spôsoby odhadu spodných odrezávacích rigidít 17

3.2.1 Výpočet spodných odrezávacích rigidít 17

3.2.2 Hľadanie vhodnej matematickej funkcie 17

3.2.3 Hľadanie presnejšej matematickej funkcie 18

4 COR ako distribuovaný systém **20**
4.1 Distribuované počítanie bez použitia vonkajších softvérových rámcov 21
4.2 Použitie BOINC alebo HTCondor 22
4.2.1 Použitie HTCondor 22
4.3 Porovanie možností nasadenia systému COR ako distribuovaného systému 23

5 Návrh implementácie používateľského rozhrania **24**
5.1 Pridanie vyhľadávania vo výpočtoch 24
5.1.1 Vyhľadávanie podľa času príchodu častíc 24
5.1.2 Vyhľadávanie podľa polohy dopadu častíc 25
5.1.3 Vyhľadávanie podľa kroku výpočtu 25
5.1.4 Vyhľadávanie podľa dátumu pridania simulácie 25
5.1.5 Kombinované vyhľadávanie 25
5.2 Vylepšenie zadávania nových výpočtov 25
5.2.1 Nové typy výpočtov 26
5.2.2 Zobrazenie duplikátnych výpočtov 26
5.3 Vylepšenie stránky na zobrazenie výpočtov 27
5.3.1 Stav výpočtu 27
5.3.2 Prístup k výsledkom 28
5.4 Administračná a editačná časť 28
5.4.1 Zavedenie rolí 28
5.4.2 Zmena používateľských údajov 29
5.5 Vizualizácia jedinej trajektórie 29

6 Návrh implementácie výpočtového systému **31**
6.1 Vylepšenia existujúcej funkcionality 31
6.1.1 Skrátenie času výpočtu vertikálnych trajektórií 31
6.1.2 Konfigurácia použitého modelu paralelizácie 32
6.1.3 Rozšírenie logovacieho podsystému 32
6.1.4 Zotavenie z chybových stavov 32
6.1.5 Indexovanie verzií 33
6.1.6 Modul na meranie trvania výpočtu a aktualizácia počtu trajektórií 33

6.1.7 Návrh implementácie zobrazenia chýbajúcich parametrov používateľom . . . 34
6.2 Návrh implementácie REST rozhrania a diskového rozhrania 35
6.2.1 Diskové rozhranie . . . 36
6.3 Pridanie spektier na 1AU pre výpočet intenzity kozmického žiarenia vnútri magnetosféry . . . 36
6.4 Návrh implementácie výpočtu a vizualizácie trajektórie 37
6.5 Funkcia pripustnosti magnetosféry . . . 37

7 Implementácia zmien a rošírení používateľského rozhrania 39
7.1 Implementácia vyhľadávania vo výsledkoch . . . 39
7.1.1 Vyhľadávanie podľa času pre ktorý bol výpočet realizovaný 39
7.1.2 Ostatné kritéria vyhľadávania . . . 40
7.2 Testovanie funkčnosti rozhrania . . . 41
7.3 Implementácia zobrazenia duplikátov po zadávaní výpočtu 42
7.3.1 Testovanie duplikátov pri viacerých zadávaných hodinách . 42
7.4 Implementácia riadenia prístupu k zdrojom a editačnej časti 43
7.4.1 Editačná časť . . . 43
7.5 Zmeny potrebné pre zadávanie a zobrazovanie výpočtov trajektórie 44
7.5.1 Archív výsledkov a prezentačná stránka . . . 44
7.6 Implementácia vkladania nových typov simulácií . . . 46
7.6.1 Implementácia prepočítavania spektra kozmického žiarenia 46
7.6.2 Pridanie úrovne optimalizácie . . . 48

8 Implementácia vylepšenia existujúcej funkcionality výpočtového systému a zásuvných modulov 49
8.1 Zmena štruktúry vstupného súboru pre generátor „infil“-ov 49
8.1.1 Zavedenie historického modelu . . . 50
8.2 Paralelizácia výpočtu vertikálnych trajektórií . . . 50
8.3 Výber paralelného modelu . . . 52
8.3.1 Zmena implementácie vlastného algoritmu spúštania 52
8.3.2 Abstrakcia na meranie času. . . . 53
8.4 Implementácia rozšírenia logovacieho podsystému . . . 53
8.5 Zotavenie z chybových stavov . . . 55
8.6 Zobrazenie chýbajúcich parametrov používateľovi . . . 56

9 Implementácia novej funkcionality do výpočtového systému a zásuvných modulov **58**
9.1 Implementácia rôznych zdrojov dát 58
9.1.1 Implementácia REST zdroja dát 59
9.1.2 Implementácia diskového zdroja dát 61
9.2 Implementácia prepočítavania spektra na strane výpočtového systému 61
9.2.1 Úprava modulu na spracovanie „outfil“-ov 62
9.3 Vizualizácia jednej trajektórie kozmického žiarenia v magnetosfére Zeme 63
9.4 Implementácia optimalizácie výpočtu viacsmerových simulácií . . 64
9.4.1 Implementácia prevodu geografických súradníc do geomagnetických súradníc 64
9.5 Implementácia funckie priepustnosti magnetosféry 65
9.5.1 Vizualizácia funkcie priepustnosti magnetosféry 66

10 Vyhodnotenie **67**
10.1 Aktualizácia systému COR 67
10.1.1 Nasadenie COR ako distribuovaného systému 67
10.1.2 Funkcia priepustnosti 68
10.2 Vyhodnotenie optimalizácie trvania viacsmerových simulácií 69
10.3 Vyhodnotenie nových modelov výpočtu a rozšírenia spracovania . 71
10.3.1 Vyhodnotenie pridania nových modelov výpočtu 71
10.3.2 Vizualizácia trajektórie častice kozmického žiarenia pre danú rigiditu 72
10.3.3 Vyhodnotenie prínosov zavedenia výberu spektier kozmického žiarenia na výpočet intenzity 73

11 Záver **75**

Literatúra **77**

Zoznam skratiek **81**

Zoznam príloh **83**

Zoznam obrázkov

2.1 Prihlasovací formulár. 4
2.2 Formulár na pridávanie simulácií. 4
2.3 Zobrazenie simulácií na veľkej obrazovke. 5
2.4 Entitno relačný diagram štruktúry databázy. 6
2.5 Schéma dátových prúdov v existujúcom systéme. 8

3.1 Mapa vertikálych odrezávacích rigidít vypočítaná pre 20. marec 2004 12:00 UTC [1]. 18
3.2 Smer magnetického poľa, zelená čiara určuje magnetický rovník [14]. 19

7.1 Zobrazenie webovej stránky formulára na pridávanie simulácií v nástroji Chrome DevTools. 41
7.2 Zobrazenie webovej stránky formulára na pridávanie simulácií v nástroji Chrome DevTools v mobilnom režime. 42
7.3 Formulár na zadávanie požiadavky na výpočet jedinej trajektórie častice. 45

10.1 Schéma nasadenia systému na počítače. 68
10.2 Vizualizácia priepustnosti magnetosféry pre Lomnický Štít pre 69. deň roka 2001. 69
10.3 Dolné odrezávacie rigidity vypočítané systémom COR. 72
10.4 Príklady vizualizácií trajektórií. 74

Zoznam tabuliek

2.1 Tabuľka meraní trvania behu vybraných simulácií 12

3.1 Tabuľka počtu zbytočne počítaných trajektórií pri viacsmerových výpočtoch 16

10.1 Tabuľka meraní trvania výpočtov na rôznych miestach na Zemi s rôznymi úrovňami optimalizácie. 70
10.2 Tabuľka meraní trvania výpočtov plného modelu, modelu na nájdenie dolnej odrezávacej rigidity a odhad minimálneho trvania výpočtu. 71
10.3 Tabuľka meraní trvania výpočtov trajektórií pre rovnakú polohu a smer príchodu častice. 72
10.4 Tabuľka hodnôt intenzity kozmického žiarenia použitím spektier AMS a 2D modelu modulácie kozmického žiarenia v heliosfére [37]. 73

11.1 Tabuľka meraní trvania výpočtov plného modelu, modelu na nájdenie dolnej odrezávacej rigidity a odhad minimálneho trvania výpočtu. 76

Zoznam výpisov

7.1 SQL príkaz na zistenie počtu dní roka. 40
7.2 PHP príkaz na zistenie počtu dní roka. 40
7.3 Inicializácie objektu povolení podľa kritérií. 43
7.4 Funkcia na overenie validity vstupného parametra. 44
7.5 Logika prevodu indexu na meno verzie. 45
7.6 Vytváranie formulára podľa typu simulácie. 47
7.7 Logika prevodu indexu verzie na typ výpočtu. 47
8.1 Nová štruktúra vstupného súboru pre generátor infilov. 50
8.2 Enumeračný typ verzií modelov. 50
8.3 Deklarácia funkcie na získanie vstupných dát. 51
8.4 Spôsob spúštania počítania simulácií. 51
8.5 Enumeračný typ stratégie paralelného spúštania výpočtov. 52
8.6 Meranie času použitím štandardnej knižnice jazyka C++. 53
8.7 Verejné rozhranie na konfiguráciu logovacieho podsystému. 54
8.8 Verejné rozhranie logovacieho podsystému. 54
8.9 Neverejné rozhranie k metódam, ktoré implementujú logovanie. . . 54
8.10 Implementácia vybranej metódy logovania so vzájomným vylúčením. 55
8.11 Riadok z konca „outfil“-u, za ktorým sú zapísané odrezávacie rigidity. 56
8.12 Enumeračný typ chýb a stavov. 57
9.1 Rozhranie zdrojov dát. 58
9.2 Inicializácia premennej dátového zdroja podľa konfigurácie. 59
9.3 Funkcia na spracovanie odpovede. 60
9.4 Príklady volania funkcie na spracovanie odpovede. 60
9.5 Rozhranie na spúšťanie výpočtov vo vedľajšom vlákne. 61
9.6 Enumeračný typ výpočtu simuacií a výber dát z dátového zdroja. . 62
9.7 Skript na konverziu súboru s IGRF dátami do samostatných súborov. 64

9.8 Prevod z geografickej do geomagnetickej súradnicovej sústavy . . . 65

1 Motivácia

Motiváciou diplomovej práce je úprava systému na výpočet trajektórií kozmického žiarenia do formy v ktorej by bol pre širšiu fyzikálnu komunitu zdrojom čo najkompletnejších, dobre prístupných a kvalitne vizualizovaných výsledkov o kozmickom žiarení v magnetosfére Zeme.

Táto diplomová práca nadväzuje na moju bakalársku prácu. V bakalárskej práci bolo mojou úlohou automatizovať spúštanie, beh a spracovanie výsledkov výpočtov v existujúcom systéme GeoMag opísaného v publikácii [1] (teraz COR (Cut-Off Rigidity)) a vytvoriť základné webové rozhranie na prístup k tomuto systému. Prax používania ukázala že systém je potrebné doplniť o rad nových funkcionalít a čo najviac urýchliť. V prvej verzii mal systém tieto vlastnosti:

- Možnosť vytvorenia používateľského konta.
- Zadávanie výpočtov a ich zálohovanie.
- Prístup k vizualizáciám výpočtov.

V tejto diplomovej práci sa sústredím na vylepšenie systému a to hlavne:

- Optimalizácia času výpočtov.
- Pridanie modelu výpočtov trajektórií pre obdobia od začiatku nášho letopočtu až do roku 1950.
- Pridanie katalógovej časti so spektrami kozmického žiarenia na magnetopauze.
- Pridanie modulu pre výpočet jedinej trajektórie a jej vizualizácie.
- Vylepšenie používateľského rozhrania:

  - Pridanie vyhľadávania vo výpočtoch.
  - Pridanie administračnej časti.
  - Pridanie lepšieho ukazateľa, ktorý by určoval v akom stave je simulácia počas výpočtu.
  - Zaručenie funkčnosti stránky pre mobilné zariadenia/pre prehliadače so zakázaným JavaSkript-om.
  - Nemožnosť prístupu pre neautorizovaných používateľov.

- Vylepšenie výpočtového systému:
  - Paralelné počítanie vertikálnych výpočtov.
  - Aktualizácia spracovania výpočtov po pridaní/úprave zásuvného modulu na spracovanie simulácií.
  - Chybové hlášky v prípade, že pre zadaný dátum neexistujú vstupné dáta.
  - Pridanie výpočtu vertikálneho smeru aj pre viacsmerové výpočty pre bohatšie možnosti vizualizácie.

Systém bude otestovaný porovnaním jednotlivých výstupov analýzy s výsledkami v odborných publikáciách.

2 Systém COR

V tejto kapitole je popísaný stav systému pred začiatkom práce. Systém COR slúži na automatizáciu výpočtov trajektórií kozmického žiarenia v magnetosfére Zeme [2], určenie vertikálnej odrezávacej rigidity [3] a odrezávacích rigidít v ostatných smeroch, na základnú analýzu týchto výsledkov, ich vizualizáciu a pre edukatívne účely pri štúdiu fyziky kozmického žiarenia a súvisiacich oblastí. Existujúci systém obsahuje tri základné časti a to webové rozhranie, databázu a výpočtový systém.

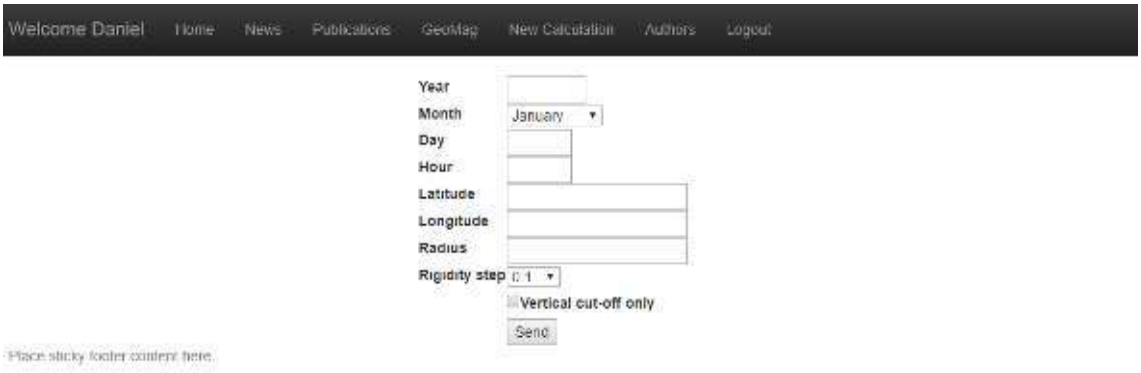
2.1 Popis webového rozhrania

Základná časť používateľského rozhrania je časť pre registráciu a prihlasovanie používateľov. Cieľom je identifikácia používateľa pre určenie, či ide o člena fyzikálnej komunity. Systém COR nie je určený pre širokú verejnosť, ale pre fyzikálnu komunitu zaoberajúcu sa časticami v magnetosfére. No aj bez registrácie sú pre záujemcov prístupné výsledky z už vypočítaných simulácií. Aktuálne riešenie prihlasovacieho formulára vidíme na obrázku 2.1.

Registrovaný a prihlásený používateľ dokáže zadávať výpočty. Zadávanie výpočtov vidíme na obrázku 2.2. Používateľ zadáva miesto a čas, pre ktoré bude simulácia realizovaná. Zadáva tiež, ktorý model externého geomagnetického poľa má byť pri simulácii použitý, krok výpočtu (krok v rigidite $\mathrm{GV(Giga\ Volt)}$) a či pôjde o výpočet pre vertikálne alebo pre všetky smery príchodu kozmického žiarenia.

Všetci používatelia si vedia zobraziť už existujúce záznamy výpočtov. Ako vidíme na obrázku 2.3, používateľ v starom systéme videl vstupné parametre každého zadaného výpočtu, číslo indikujúce stav výpočtu a odkazy na sumár výsledkov, a ich stiahnutie pri dokončených výpočtoch. Dokáže si taktiež výber zoradiť podľa jedného z atribútov alebo regulovať počet výpočtov, ktoré sa naraz na

Obr. 2.1: Prihlasovací formulár.



Obr. 2.2: Formulár na pridávanie simulácií.

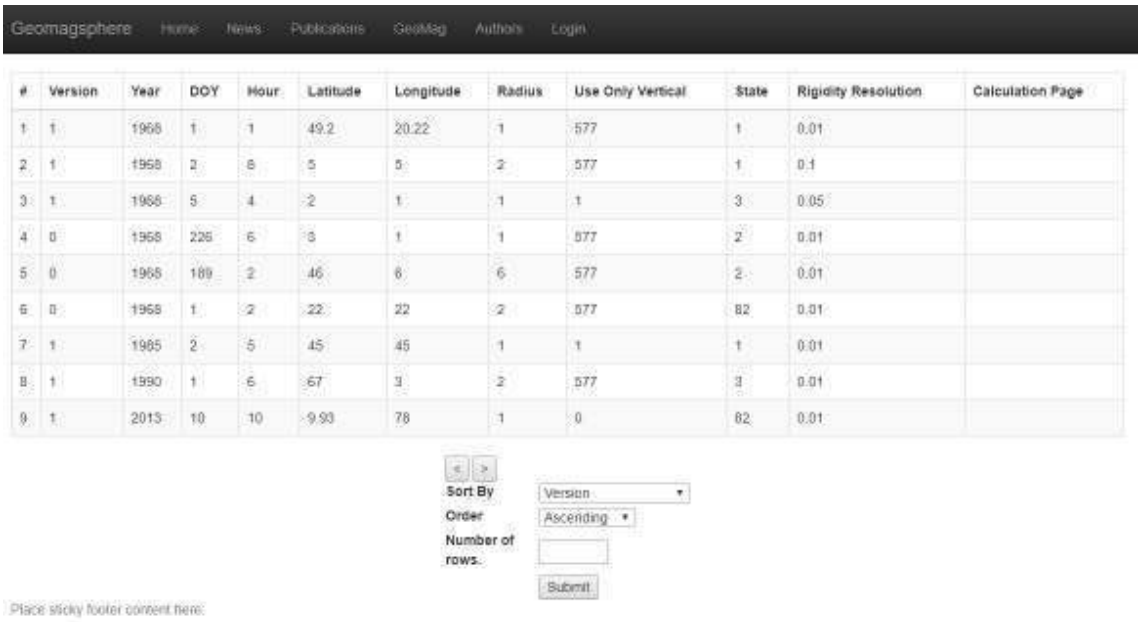
stránke zobrazia. Prihlásený používateľ si tiež dokáže vyfiltrovať iba svoje výsledky, čo na obrázku nevidíme.

Dokončenú simuláciu si prihlásený aj neprihlásený používateľ dokáže zobraziť. Na stránke zobrazenia vidí:

- Vizuálizáciu spektra zakázaných a dovolených rigidít pre všetky spočítané smery vo forme katalógu.
- Mapu oblohy odrezávacích rigidít.
- Mapy asymptotických smerov príchodu kozmického žiarenia [4].
- Kompletné údaje pre všetkých počítaných 576 smerov [2] príchodu kozmického žiarenia ktoré si používateľ môže stiahnuť v rámci komprimovaného súboru.

2.2 Opis databázy

Databáza typu SQL (Structured Query Language) obsahuje tabuľky so záznamami o používateľoch a simuláciách. Jej štruktúru vidíme na obrázku 2.4. Použitý data-



#	Version	Year	DOY	Hour	Latitude	Longitude	Radius	Use Only Vertical	State	Rigidity Resolution	Calculation Page
1	1	1968	1	1	49.2	20.22	1	577	1	0.01	
2	1	1968	2	8	5	5	2	577	1	0.1	
3	1	1968	5	4	2	1	1	1	3	0.05	
4	0	1968	226	6	3	1	1	577	2	0.01	
5	0	1968	189	2	46	6	6	577	2	0.01	
6	0	1968	1	2	22	22	2	577	82	0.01	
7	1	1985	2	5	45	45	1	1	1	0.01	
8	1	1990	1	6	67	3	2	577	3	0.01	
9	1	2015	10	10	-9.93	78	1	0	82	0.01	

Obr. 2.3: Zobrazenie simulácií na veľkej obrazovke.

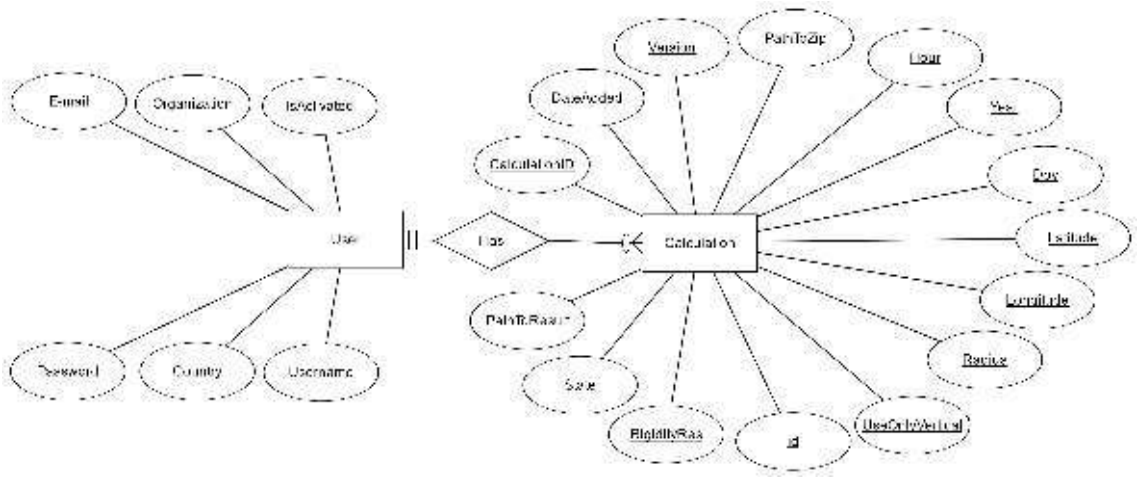
bázovy systém je MySQL[1] a systém úložiska je InnoDB[2] kvôli tomu, že ponúka zamykanie na úrovni riadkov.

Použitím úložiska typu InnoDB bol systém COR zčasti pripravený na nasadenie ako distribuovaný systém. Zamykanie riadku po výbere dát až pokiaľ rovnaký riadok nie je aktualizovaný zamedzuje problému, ktorý by mohol nastať ak viacero inštancií výpočtového systému pristupuje k databáze naraz. Kvôli kritériám výberu by sa mohlo stať že predtým, ako by jedna inštancia systému na riadenie výpočtu vybrala potrebné dáta a nastavila stav výpočtu na prebiehajúci, iná inštancia by pristúpila k rovnakým dátam a jeden výpočet by sa zbytočne počítal viackrát.

Databázový systém slúži ako rozhranie medzi webovým rozhraním a výpočtovou časťou. Požiadavka na realizáciu výpočtu zadaná používateľom je zapísaná do databázy, ktorú monitoruje výpočtový systém. Ak je simulácia vo výpočtovom systéme úspešne vypočítaná, výpočtový systém automaticky kontroluje, či sa v databáze nachádzajú nové požiadavky. Maximálny interval kontroly ke konfigurovateľný pričom dobrá hodnota bola 30 sekúnd. Ak žiadne požiadavky neexistujú, systém kontroluje databázu v pravidelných intervaloch a ak nájde novú požia-

[1] Dokumentácia MySQL na: `https://dev.mysql.com/doc/refman/5.6/en/introduction.html`

[2] Viac informácií na: `https://dev.mysql.com/doc/refman/5.6/en/innodb-introduction.html`



Obr. 2.4: Entitno relačný diagram štruktúry databázy.

davku, začne na nej pracovať.

2.2.1 Štruktúra tabuliek v databáze

V tabuľke so záznamami o používateľoch je uložené ich používateľské meno, ktoré používajú pri prihlásení. Aby bol ich účet zabezpečený, v databáze sa neukladá ich heslo ako reťazec, ale ukladá sa hash ich hesla vytvorený funkciou bcrypt [5]. Okrem spomenutých údajov sa ukladá aj e-mail, krajina ich pôsobenia a organizácia pre ktorú pracujú.

V tabuľke so záznamami o simuláciách je používateľské meno používateľa ktoré je cudzím kľúčom odkazujúcim sa na používateľské meno v tabuľke so záznamami o používateľoch, vďaka čomu si dokáže vyfiltrovať iba svoje simulácie. Taktiež tam sú fyzikálne dáta potrebné na výpočet, dátum pridania výpočtu (výpočtový systém ho používa na určenie nasledujúceho výpočtu, čim starší záznam, tým vyššia priorita) a stav výpočtu.

2.3 Opis výpočtového systému

Výpočtový systém je členený na tieto časti:

- Spúštač modulov.
- Generátor vstupých súborov pre výpočty v jednotlivých smeroch príchodu kozmického žiarenia.

- Fyzikálne modely interného a externého magnetického poľa.
- Zásuvné moduly.
- Skript na archiváciu.

Súbor „infil“ obsahuje všetky vstupné údaje potrebné pre realizáciu potrebných výpočtov. Súbor „outfil“ je výstupom fyzikálneho výpočtu a obsahuje údaje s rigiditymi dovolených trajektórií.

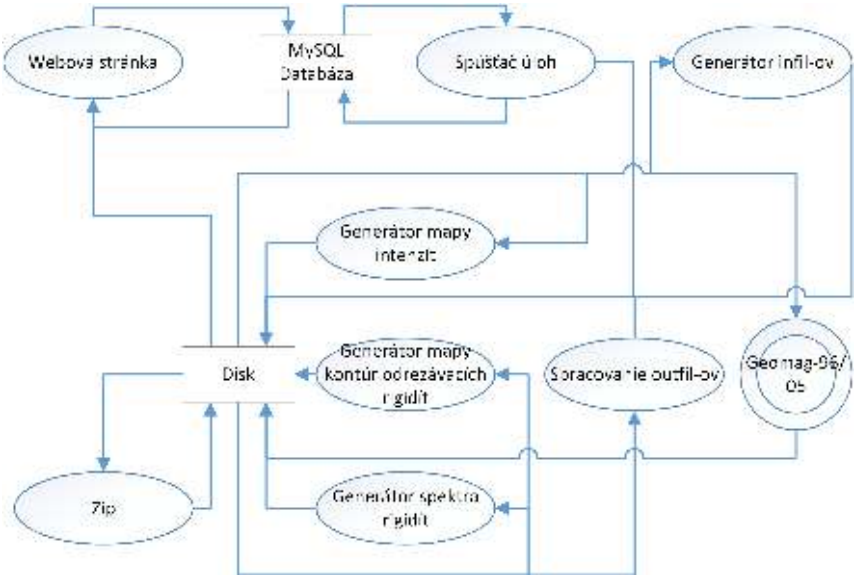
Ako vidíme na obrázku 2.5 spúšťač úloh riadi celý výpočet. Prvým krokom výpočtu je vygenerovanie vstupných súborov na základe údajov z databázy, ktoré sú neskôr použité ako vstup pre fyzikálne výpočty. Tento modul vyhľadáva údaje o stave magnetosféry v súboroch, ktoré sa nachádzajú v koreňovom priečinku aplikácie. V súboroch sú každej trojici - rok, deň v roku a hodina - priradené hodnoty potrebných údajov. Tieto súbory ale neobsahujú údaje pre všetky možné časy, ktoré môže používateľ zadať. Ak tento modul nenájde na základe času potrebné dáta, použije pre nich hodnotu 0.

Potom sú fyzikálne výpočty spúšťané paralelne tak, aby nehrozilo prepínanie procesov z jadra na jadro, čo negatívne ovplyvňuje výkon systému [6] a zachoval sa stály prísun nových vstupov ktoré majú v databáze stav nastavený na hodnotu ktorá hovorí že výpočet ešte nebol spracovaný.

Po dokončení výpočtu sa spustia zásuvné moduly, ktoré sú závislé na poradí v akom sú spustené, keďže výstup jedného zásuvného modulu je vstupom nasledujúceho zásuvného modulu. Tieto sú spúštané sériovo. V systéme COR je jediný modul, ktorý sa spúšta sériovo je modul na spracovanie výsledkov. Tento program paralelne načíta dáta z vytvorených súborov, analyzuje ich a vytvára súbory potrebné pre vizualizácie.

Nasledujú zásuvné moduly pri ktorých nezávisí na poradí v ktorom sú spúštané. Trvanie behu týchto zásuvných modulov je síce viacnásobne kratšie, ako beh výpočtov trajektórií častíc kozmického žiarenia vo všetkých smerov, no implementácia paralelného výpočtu nebola zložitá a poskytne malé urýchlenie, preto sú spúštané paralelne. Tieto moduly slúžia na vytvorenie vizualizácií, ktoré si používateľ dokáže prezrieť na stránke detailu výpočtu.

Na záver sa všetky výstupné súbory komprimujú spustením skriptu na komprimáciu a do databázy sa zapíšu cesty k výsledkom výpočtu a vytvorenému komprimovanému súboru tak aby, modul webovej stránky mal k dispozícii cestu



Obr. 2.5: Schéma dátových prúdov v existujúcom systéme.

k vypočítanému výsledku. Taktiež výpočtový systém zmení v databáze stav výpočtu z prebiehajúceho na dopočítaný.

2.3.1 Opis modulu na spúšťanie úloh

Modul na spúšťanie úloh je implementovaný v jazyku C++ a využíva princíp OOP [7]. Funkcionalita je rozdelená do nasledovných tried:

- `Config` - Načítanie konfigurácie.
- `Database` - Získavanie vstupných údajov z databázy.
- `InputData` - Štruktúra reprezentujúca vstupné údaje.
- `ProcessAdapter` - Adaptér procesov.
- `DirectoryManager` - Trieda na vytváranie priečinkov.
- `AbstractProcessManager` - Abstraktný manažér úloh.
- Konkrétne implementácie:
  - `ProcessManager` - Vlastná implementácia.
  - `ThreadedProcessManager` - Implementácia založená na OpenMP (Open Multi-Processing) [8] .

- Triedy výnimiek.

Po spustení sa zo súboru načíta konfigurácia, ktorá je prístupná po celý beh programu. Táto trieda obsahuje iba metódy na prístup k hodnotám načítaným z konfigurácie. Zmena súboru konfigurácie po spustení sa prejaví až po vypnutí celého programu, ktoré je možné odoslaním signálu `SIGUSR1`, a opätovnom spustení prgramu.

Trieda na získavanie vstupných údajov poskytuje rozhranie na získanie dát z databázy a aktualizáciu hodnôt v databáze. Metóda na získavanie údajov má návratovú hodnotu, ktorá obsahuje adresu na štruktúru reprezentujúcu vstupné údaje. Výber z databázy je chránený tak, že je zaručený atomický výber údajov a zmena ich stavu. Slovo atomický v tomto prípade znamená, že zložená operácia výberu a zmeny stavu buď prebehne celá alebo vôbec. Taktiež poskytuje metódu na zápis údajov do databázy po dokončení celého výpočtu.

Štruktúra na vstupné údaje má členské premenné na všetky potrebná dáta a metódy na prístup k nim, ich zmenu a prácu s nimy. Taktiež obsahuje enumerácie slúžiace na abstrakciu stavov výpočtu a verzií výpočtu.

Trieda `ProcessAdapter` slúži na dynamické vytváranie objektov procesov tak aby sa predišlo vyčerpaniu hardvérových zdrojov, konkrétne prekročenia maximálneho počtu možných otvorených súborov. Knižnica ktorá sa stará o vytváranie a riadenie procesov vytvára pre každý proces pomenovanú rúru ktorá je v systéme reprezentovaná ako súbor. Ak je naraz vytvorených príliš veľa rúr, pri vytváraní nasledujúcej nastane chyba `EMFILE`[3]. Vytváraním objektov procesov dynamicky a ich následnou dynamickou deštrukciou obmedzujeme počet naraz otvorených súborov procesom riadiaceho systému, počet maximálne bežiacich paralelných procesov nakonfigurovaných používateľom. Rozhranie obsahuje metódy, ktoré sú tenkou obálkou na prístup k spravovaným inštanciám procesov.

Manažér prečinkov má za úlohu vytvoriť potrebné priečinky pre výstup výpočtu a poskytovať relatívnu cestu k týmto priečinkom.

Abstraktný manažér úloh implementuje riadenie príprav a spracovanie výpočtu. Obsahuje jedinú neimplementovanú metódu `run`, ktorá slúži na postupné spúšťanie jednotlivých smerov.

Vlastná implementácia v pravidelných intervaloch kontroluje stav procesov a podľa potreby spúšta nové tak aby počet naraz spustených procesov nikdy nepre-

[3]Viac informácii na: `http://man7.org/linux/man-pages/man2/pipe.2.html`

siahol počet paralelných jednotiek procesora.

`ThreadedProcessManager` riadi spúštanie výpočtov použitím paralelného cyklu `for`.

2.4 Nedostatky používateľského rozhrania

Prvá verzia systému COR spĺňala základné kritériá zadania, no používateľské rozhranie nebolo dokončené tak, aby bolo plne použiteľné. Hlavné problémy sú prístupnosť stránky z mobilných zariadení, alebo zariadení so zakázaným JavaScript-om, administračné možnosti rozhrania, použiteľnosť ovládacích prvkov a zabezpečenie stránky.

2.4.1 Prístupnosť a administrácia

Moderné stránky využívajú JavaScript, no mnohí používatelia ho majú v prehliadači zakázaný. Dôvody prečo používatelia vo svojich prehliadačoch JavaScript zakazujú sú napríklad zachovanie súkromia [9] alebo zabránenie zneužitia používateľovho počítača na ťaženie kryptomien [10]. Aby sme oslovili čo najväčšiu základňu používateľov je potrebné, aby portál fungoval aj v prípade, keď používatelia vo svojom prehliadači zakázali interpretáciu JavaScript-u.

Počet používateľov, ktorí pristupujú na intenetové stránky z mobilných zariadení každým rokom stúpa. Preto, ak chceme zaručiť optimálny zážitok pre používateľa, je potrebné, aby používateľské rozhranie bolo použiteľné na čo najväčšom počte typov zariadení. Používanie štýlovacieho rámca Bootstrap[4] tento problém minimalizuje, no špecificky treba riešiť problémy s viditeľnosťou prvkov a jednoduchosť k ich prístupu.

Stará verzia nemala administračnú časť. Tá je dôležitá z ekonomických dôvodov. Viacsmerový výpočet trvá rádovo hodiny, pričom počas simulácie sa míňa značné množstvo energie. Pri zavedenej administácii musí zaregistrovaný používateľ počkať, kým mu administrátor aktivuje účet, aby mohol zadávať výpočty. Administrátor sa môže rozhodnúť, či je používateľ legitímny a podľa toho sa rozhodne, či mu aktivuje účet.

[4]Domovská stránka:`https://getbootstrap.com/`

2.4.2 Ovládacie prvky rozhrania

Na obrázku 2.3 vidíme rozhranie na prehľadávanie simulácií. Hlavné chyby tohto rozhrania sú:

- Zoradenie simulácií podľa atribútu je riešené cez formulár.
- Chýba ukazateľ terajšej/poslednej stránky.
- Nemožnosť preskočiť na ľubovoľnú stránku.
- Ikona na zobrazenie predchádzajúcej/ďalšej stránky je viditeľná aj pri zobrazovaní prvej/poslednej stránky.

Pri zobrazení záznamov z databázy neexistovalo vyhľadávanie vo výpočtoch, čo zapríčiňovalo neoptimálny používateľský zážitok. Do budúcej verzie je potrebné implementovať vyhľadávanie v simuláciách, alebo systém na ich filtráciu podľa vhodných kritérií.

Ako vidíme na obrázku 2.2 zadávanie výpočtov je možné iba pre konkrétnu hodinu, no pre praktické účely je častokrát vhodné simulovať trajektórie pre zadaný časový rozsah, väčšinou pre celé dni. Aby bolo zadávanie výpočtov pohodlnejšie bude potrebné implementovať možnosť zadávať výpočty pre interval hodín. Výpočet viacsmerových simulácií pre interval hodín trvá niekoľko dní. Z tohto dôvodu treba zaviesť role pre používateľov tak, aby iba používatelia, ktorých administrátor autorizuje, dokázali zadávať viacero výpočtov naraz.

2.4.3 Zabezpečenie prístupu na podstránky

V prvej verzii systému bolo možné, aby sa neprihlásený používateľ dostal na stránku, kde nemá mať prístup, ak poznal URL (Uniform Resource Locator) stránky. Toto je bezpečnostné riziko.

Aby neautorizovaný používatelia nedokázali pristúpiť k funkcionalite stránky, ktorá pre nich nemá byť prístupná, bude potrebné zaviesť do systému kontrolu prístupu k zdrojom.

2.5 Nedostatky výpočtového systému

Medzi nedostatky vo výpočtovej časti patria neefektívnosť spracovania vertikálnych simulácií, tažkosti v prípade pridania rozšírení, zotavenie po ukončení nedokončeného výpočtu a nezobrazovanie chýb spôsobených neexistenciou vstupných dát.

V tabuľke 2.1 vidíme približný čas trvania rôznych typov výpočtov. Trvanie výpočtu simulácie veľmi závisí od vstupných premenných, ale vo všeobecnosti tabuľka reprezentuje čas behu simulácie/výpočtov.

Počet smerov	Verzia	Čas
1	T96	4 minúty
577	T96	6 hodín
577	T05	15 hodín

Tabuľka 2.1: Tabuľka meraní trvania behu vybraných simulácií

2.5.1 Optimalizácia výpočtu viacerých vertikálnych trajektórií

V prechádzajúcom systéme boli vertikálne simulácie počítané sériovo. Takto sa počas výpočtu využívalo iba jediné jadro procesora. Jednoduchá optimalizácia spočíva v paralelizácií výpočtu vertikálnych simulácií. Paralelizácia je vhodná, keďže každá vertikálna simulácia je samostatný výpočet, ktorý nezávisí od ostatných simulácií.

Predstavme si, že máme v databáze požiadavky na výpočet 32 vertikálnych simulácií. Ak predpokladáme, že každá z nich trvá 4 minúty ako je zobrazené v tabuľke 2.1, neoptimalizovaný výpočet by trval $32 \times 4 = 128$ minút. Paralelizáciou sa tento čas dá teoreticky skrátiť maximálne P-krát, pričom P je počet procesorov. Prakticky očakávame skoro lineárne zrýchlenie.

2.5.2 Rozšíriteľnosť a spracovanie chýb

Systém ešte nie je vo finálnej verzii a očakáva sa stále aktualizovanie a rozširovanie systému v závislosti od vývoja fyzikálneho modelu [11]. Ak by bol pridaný nový zásuvný modul, ktorý dokáže vypočítané dáta lepšie spracovať je potrebné mať režim spustenia simulácií taký, aby sa zbytočne neprepočítavali získané vysledky, ale

spustilo sa iba ich spracovanie. Takto potrebujeme realizovať iba nové spracovania simulácií bez potreby opakovania celého výpočtu.

Na vývoji častí systému COR sa podieľajú aj ďalší študenti a to hlavne v časti vizualizácií. Na uskutočnenie bakalárskej práce potreboval Martin Vaško pridať počítanie vertikálneho smeru aj pre viacsmerové simulácie [12]. Aby sa existujúce simulácie nemuseli znova prepočítavať je potrebné doimplementovať možnosť detekcie a počítania neexistujúcich smerov v existujúcich simuláciách.

Možnosť zistenia, ktoré smery simulácie sú už vypočítané je výhodné aj v prípade, že počas výpočtu simulácie nastala chyba, systém prestal fungovať a výpočet sa zastavil. Po spustení by systém zistil, ktoré výstupné súbory už sú vypočítané a pokračoval by v počítaní chýbajúcich súborov.

2.5.3 Chybové hlášky

Nie pre všetky vstupné dátumy existujú dáta potrebné na výpočet (tlak slnečného vetra, By a Bz zložka medziplanetárneho magnetického poľa, Dst index, W1 až W6 parametre určujúce prehistóriu geomagnetického poľa). Ak táto situácia nastala v predchádzajúcej verzii používateľ na to nebol upozornený. Takto dostal nepresné výsledky bez varovaní, čo ohrozuje reputáciu ÚEF SAV (Ústav experimentálnej fyziky Slovenskej akadémie vied). Preto je v novej verzii potrebné zaviesť chybové hlášky, ak pre vstupný čas neexistujú potrebné dáta.

2.5.4 Výber zdroja dát

Keďže systém COR je modulárny, je dôležité aby sa jeho časti dali použiť samostatne. V predchádzajúcej verzii bolo možné výpočtový systém použiť bez časti webovej stránky no bolo potrebné mať nakonfigurovanú minimálne MySQL databázu z potrebnými tabuľkami. Pridaním konfigurovateľného zdroja dát sa táto požiadavka odstráni a používateľ bude mať na výber aký vstup bude výpočtový systém používať.

Okrem existujúceho zdroja dát MySQL databázy chceme pridať REST (Representational state transfer) rozhranie ktoré zariadi flexibilitu rôznych databáz a disk rozhranie pre používateľov ktorý chcú vykonať len pár výpočtov bez nastavovania komplikovaných databázových systémov. Ak si používateľ vyberie možnosť REST rozhrania, bude musieť rozhranie na poskytovanie dát implementovať tak aby bolo

kompatibilné z rozhraním systému.

3 Optimalizácia času výpočtov v systéme COR

Cieľom optimalizácie je skrátenie dĺžky výpočtov v systéme COR. V súčasnosti priemerný viacsmerový výpočet trvá približne 6 až 15 hodín, ako vidíme v tabuľke 2.1. V ideálnom prípade je očakávaným pozitívnym výsledkom skrátenie doby výpočtu o čas potrebný pre vypočítanie zakázaných trajektórií pod spodnou rigiditou, bez signifikantnej straty presnosti. V súčasnosti veľká časť výpočtov končí, ako zakázané trajektória.

Základným optimalizačným prístupom bude snaha o limitovanie výpočtov zakázaných trajektórií pod spodnou odrezávacou rigiditou. Cieľom je sa so štartovacou rigiditou výpočtu čo najviac priblížiť spodnej odrezávacej rigidite pre daný smer na danom mieste. V súčasnej verzii sa všetky simulácie začínajú počítať na rovnakej predom určenej hodnote štartovacej rigidity (0.01 GV). To znamená, že pre pozície s vysokými geografickými šírkami sa počíta celý potrebný rozsah rigidít, no pre strednošírkové a nízkošírkové pozície s vyššími odrezávacími rigiditami sa počíta zbytočne veľa trajektórií zakázaných rigidít [13].

Podobnú optimalizáciu je možné spraviť aj pre horné odrezávacie rigidity. Pri nich potom nie je potrebné počítať trajektórie s vyššími rigiditami, keďže horná odrezávacia rigidita určuje hranicu nad ktorou sú už všetky trajektórie povolené a nie sú potrebné na vyjadrenie intenzít dopadajúcich častíc. Vhodnejšie je sústrediť sa hlavne na optimalizáciu dolných odrezávacích rigidít pretože:

- Očakávaná úspora času je pri nízkych hodnotách rigidít väčšia.
- Vysoké hodnoty rigidít sa dajú využiť na určenie smerov príchodov častíc.

3.1 Odhad urýchlenia výpočtu zavedením optimalizácie spodnej odrezávacej rigidity

Na zistenie celkového počtu počítaných trajektórií použijeme vzorec:

$$P_{trajektorii} = (R_{posledna} - R_{prva})/\Delta R \tag{3.1}$$

Kde $P_{trajektorii}$ označuje počet trajektórií, R_{prva} je hodnota prvej rigidity, $R_{posledna}$ je hodnota poslednej rigidity a ΔR je krok výpočtu.

Keďže simulácia, ktorá má pre prax najväčší význam je s krokom 0.01 GV použijeme túto hodnotu. Počiatočná rigidita pri viacsmerových výpočtoch je 0.01 GV a konečná 100 GV. Po doplnení získame:

$$(100 - 0.01)/0.01 = 9999 \tag{3.2}$$

Celkový počet trajektórií počítaných pre jeden smer je teda 9999. Keďže chceme hodnotu pre všetky smery, musíme výsledok ešte vynásobiť počtom počítaných smerov teda číslom 577. Tak zistíme, že pri počítaní jedného viacsmerového výpočtu sa celkovo počíta $5\,769\,423$ trajektórií.

Na zistenie počtu trajektórií, ktoré sa zbytočne počítajú použijeme skript, ktorý zistí dolnú odrezávaciu rigiditu pre každý smer a dosadí ju do vzorca 3.1, ako parameter $R_{posledna}$. Parametre R_{prva} a ΔR majú hodnotu 0.01. Skript vypočítané hodnoty pre všetky smery spočíta.

V tabuľke 3.1 vidíme počty zbytočne počítaných trajektórií pre príklad nízkej strednej a vysokej zemepisnej šírky. Percentuálne zníženie počtu počítaných trajektórií nevyjadruje presnú hodnotu času trvania výpočtu.

Zemepisná šírka	Zemepisná dĺžka	Počet	Percento z maxima($5\,769\,423$)
$9.93°$	$78.00°$	$1\,420\,147$	24.62 %
$49.20°$	$20.22°$	$269\,966$	4.68 %
$68.36°$	$226.28°$	$27\,194$	0.47 %

Tabuľka 3.1: Tabuľka počtu zbytočne počítaných trajektórií pri viacsmerových výpočtoch

Trajektórie častíc s nižšou rigiditou sú väčšinou komplikovanejšie a dlhšie, ako tie s vyššou rigiditou. Čas výpočtu trajektórií s nižšou rigiditou je tak dlhší ako

pri trajektóriách s vyššou rigiditou. To znamená, že aj keď pri stredných zemepisných šírkach je počet trajektórií, ktoré touto optimalizáciou vylúčime z výpočtu relatívne nízky, percentuálne zníženie času výpočtu môže byť väčšie, ako percento vylúčených trajektórií. Optimalizáciou pri ktorej sa snažíme vylúčiť z výpočtov trajektórie pod prvou dovolenou rigiditou vylučujú tie najkomplikovanejšie a preto najdlhšie počítané trajektórie.

3.2 Spôsoby odhadu spodných odrezávacích rigidít

V tejto časti práce popisujeme rôzne metódy odhadu spodných odrezávacích rigidít ktoré neskôr použijeme na implementáciu optimalizácie výpočtov.

3.2.1 Výpočet spodných odrezávacích rigidít

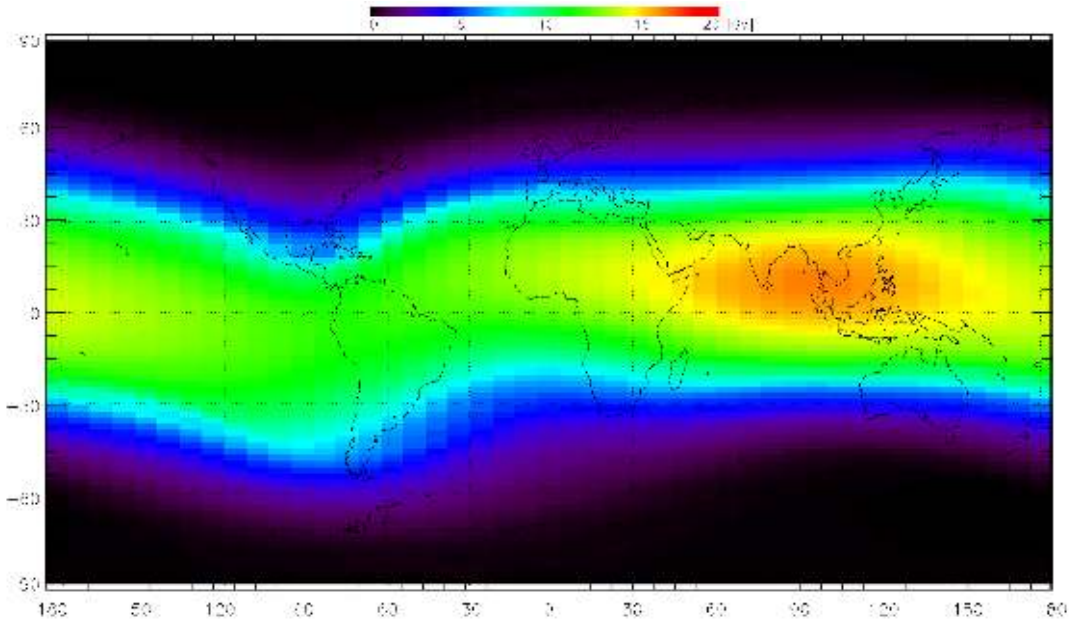
Primárnym podkladom pre nájdenie štartovycích odrezávacích rigidít budú vypočítané hodnoty spodných odrezávacích rigidít. Tieto určíme pre sieť bodov rovnomerne rozmiestnených na povrchu Zeme. V prípade nájdenia vhodnej matematickej funkcie na popis spodných odrezávacích rigidít počítanej siete bodov použijeme pre odhad štartovacej rigidity túto funkciu.

3.2.2 Hľadanie vhodnej matematickej funkcie

V prvej použitej metóde vyrátame pre vybranú geomagnetickú dĺžku sériu výpočtov s rôznymi geomagnetickými šírkami s krokom $20°$. Vhodná geomagnetická dĺžka je $0°$, pretože približne na tejto geomagnetickej dĺžke sú odrezávacie rigidity minimálne vzhľadom na ostatné regióny na planéte. Na obrázku 3.1 je uvedená mapa vertikálnych odrezávacích rigidít povrchu Zeme. Z mapy vidíme, že na geomagnetickom rovníku je minimálna odrezávacia rigidita približne na polohe $-70°$Z ($290°$ východnej dĺžky). Táto poloha približne zodpovedá $0°$ geomagnetickej dĺžky.

Krok $20°$ výpočtov je vybraný s ohľadom na dĺžku výpočtov, ktoré môžeme prvému odhadu venovať. Rádovo potrebujeme, aby výpočet trval desať a nie sto dní a takýto krok je realizovateľný v priateľnom časovom intervale.

Pre každý bod vypočítame odrezávacie rigidity pre 577 smerov. Pre každý bod vyberieme smer s minimálnou spodnou odrezávacou rigiditou. Tieto hodnoty



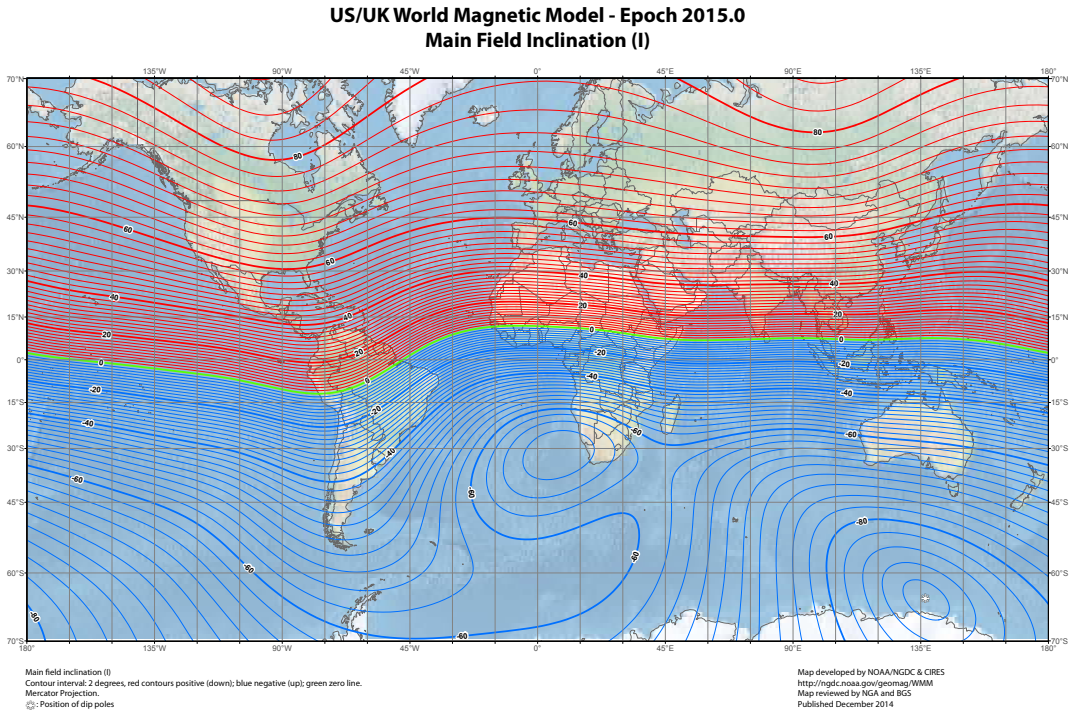
Obr. 3.1: Mapa vertikálych odrezávacích rigidít vypočítaná pre 20. marec 2004 12:00 UTC [1].

použijeme pre odhad závislosti štartovacej rigidity od geomagnetickej šírky, ktorú popíšeme vhodnou matematickou funkciou. V odhade stanovíme koeficient o ktorý bude štartovacia rigidita menšia od spodnej odrezávacej rigidity. Jeho vhodné nastavenie zároveň zabezpečí, že odhad bude platný pre akúkoľvek geomagnetickú dĺžku.

Pri riešení chceme overiť, či fitovaním týchto hodnôt dokážeme nájsť funkciu jedného parametra, geomagnetickej šírky, ktorá popisuje závislosť dolnej odrezávacej rigidity od geomagnetickej polohy.

3.2.3 Hľadanie presnejšej matematickej funkcie

Po otestovaní prvého prístupu zvolíme všeobecnejší postup, ktorý zahrnie aj vplyv geomagnetickej dĺžky. Keďže odrezávacie rigidity sa menia aj s geomagnetickou dĺžkou, je možné pri použití tohto spôsobu očakávať ďalšie urýchlenie výpočtov. Pre túto optimalizáciu si vypočítame globálnu sieť bodov s šírkovým krokom $20°$ a dĺžkovým krokom $30°$. Celkovo tak spočítame trejktórie príchodu kozmického žiarenia pre 72 bodov, ktoré sú rovnomerne rozmiestnené vzhľadom na geomagnetické koordináty. Pre každú zo 72 polôh nájdeme minimálnu spodnú odrezávaciu



Obr. 3.2: Smer magnetického poľa, zelená čiara určuje magnetický rovník [14].

rigiditu. Túto sieť 72 spodných odrezávacích rigidít aproximujeme vhodnou fitovacou funkciou závislou od geomagnetickej šírky a geomagnetickej dĺžky.

Tieto optimalizačné postupy zrýchľujú výpočty realizované pre body na povrchu Zeme. Na nízkej orbite postačí malá korekcia získaných optimalizačných funkcií, ktorá zabezpečí správnosť získaných výsledkov do výšky približne 500 kilometrov nad Zemským povrchom. Pre väčšie výšky je pre modifikáciu optimalizačných funkcií potrebná ďalšia analýza. Väčšina požívateľov bude mať záujem o výpočty na povrchu Zeme alebo nízkej orbite.

4 COR ako distribuovaný systém

Systém COR bol vyvíjaný tak, aby nebolo veľmi komplikované nasadiť ho, ako distribuovaný systém. Distribuovaný systém je systém, ktorého procesy sú spustené na viacerých počítačoch, no navonok sa javí, ako systém spustený na jednom počítači [15].

Používateľské rozhranie a databáza si nevyžadujú škálovanie, keďže pre očakávaný počet používateľov je výkon, ktorý poskytuje jeden počítač postačujúci. Potreba škálovania výpočtového systému je však očividná, hlavne z dôvodu, že pre fyzikálnu prax je potrebné vykonať veľké várky výpočtov.

Predstavme si, že používateľ potrebuje vypočítať charakter povolených a zakázaných rigidít pre jedno miesto počas dvoch dní použitím modelu Tsyganenko 2005. Ak predpokladáme dĺžku jedného výpočtu v trvaní 15 hodín, ako ilustruje tabuľka 2.1 a budeme počítať pre dvojdňový interval s jednohodinovým krokom, to znamená pre 48 časov, tento výpočet by zabral 720 hodín, čo je 30 dní. Jedným riešením tohto problému je optimalizácia samotného výpočtu, druhou možnosťou je škálovanie výpočtového systému nasadením na ďalší počítač. Keďže na výpočet siete bodov ako je to popísané v 3.2.3 bude potrebné vykonať veľa výpočtov, nasadenie systému na viac počítačov skráti čas hľadania presnejšej optimalizačnej funkcie.

Možností, ako nasadiť COR, ako distribuovaný systém je viacero, napríklad:

- Vlastná implementácia.
- Použitie nástroja Grid Computing [16] napríklad HTCondor [17].
- Použitie nástroja na verejné počítanie BOINC (Berkeley Open Infrastructure for Network Computing) [18].
- Použitie SLURM (Simple Linux Utility for Resource Management) [19].

4.1 Distribuované počítanie bez použitia vonkajších softvérových rámcov

Nasadenie na nový počítač realizujeme kopírovaním výpočtovej časti na vybraný počítač a zmenou konfigurácie. Konkrétne v sekcii `[Database]` nastavíme hodnotu `IPAddress` na IP (Internet Protocol) adresu počítača s databázou a v časti `[General]` nastavíme hodnotu `MachineID` na číslo v intervale 0 až 255, ktoré nemá priradený žiaden spustený systém.

Po takejto konfiguráii má výpočtový systém prístup k databáze a výpočty majú priradené ID počítača na ktorom sa nachádzajú. No keďže časť webovej stránky je na inom stroji, ako časť výpočtového systému, webová stránka nemá k vypočítaným dátam prístup. Tento problém vyriešime nakonfigurovaním NFS (Network File System) [20]. V NFS rozoznávame:

- Server - poskytuje prístup k disku svojím klientom.
- Klient - dokáže používať diskový priestor servera, ako svoje lokálne úložisko

V našej situácii môžeme prístup k dátam zariadiť dvoma základnými prístupmi:

- Server bude jeden a bude slúžiť na ukladanie dát pre všetky spustené inštancie, ktoré sú klienti.
- Server bude každá inštancia výpočtového systému a klient bude pre všetky servery jeden a to počítač s databázou a webovou stránkou.

Z hľadiska konfigurácie je jednoduchšie nakonfigurovať jeden server, no výpočtový systém vyžaduje zapisovanie dát na disk. V prípade, že viacero klientov vyžaduje prístup k dátam naraz, ich výpočet by závisel iba od rýchlosti disku teda výpočet by sa zoserializoval a tak by sa stratil benefit distribuovaného systému.

Pre naše použitie by bola najvhodnejšia hybridizácia oboch prístupov. Kvôli jednoduchej konfigurácii mať iba jeden server kde spustené inštancie sú klienti, no počas výpočtu používať lokálne pamäte, ako dočastné úložisko a až po dokončení celý výpočet presunúť.

4.2 Použitie BOINC alebo HTCondor

BOINC je systém na distribuované počítanie na počítačoch dobrovoľníkov. Aby bolo možné COR model nasadiť do systému BOINC, je potrebné upraviť systém tak, aby každá úloha mohla bežať oddelene na rôznych počítačoch. Toto nie je problém, keďže každý smer je samostatný výpočet, ktorého výsledok je závislý len od jeho vstupu.

V systéme BOINC voláme takúto uzavretú výpočtovu jednotku WU (Work Unit). Táto jednotka obsahuje vstupy pre aplikáciu (v našom systéme súbor infil) a samotnú aplikáciu (v našom systéme výpočet jedného smeru). Výsledok obsahuje referenciu na druh aplikácie, keďže vo všeobecnosti môže jeden BOINC projekt pracovať na riešení rôznych úloh a výstupné súbory (v našom systéme outfil).

Rozdelenie výpočtového systému by tak bolo nasledovné:

- Server, ktorý by vytváral infily pre rôzne počítače.
- BOINC klient na ktorom by prebiehal výpočet jedného alebo viacerých smerov.

Po vypočítaní výsledkov by sa tieto preniesli na server, kde by sa ďalej spracovali a boli publikované na webovej stránke.

4.2.1 Použitie HTCondor

Konfigurácia projektu v systéme HTCondor spočíva v pripravení úlohy tak, aby mohla bez dozoru prebiehať, vybratia vhodného Condor „vesmíru“ a vytvorenia súboru, ktorý popisuje detaily úlohy [21]. Obsah súboru môže obsahovať nasledovné údaje:

- Názov úlohy.
- Názov „vesmíru“.
- Platformu a typ procesora vhodný pre danú úlohu.
- Názvy súborov do ktorých presmerovať `stdin`, `stdout` a `stderr`.
- Adresu na ktorú bude zaslaný e-mail po ukončení výpočtu.

4.3 Porovanie možností nasadenia systému COR ako distribuovaného systému

Štruktúra a konfigurácia HTCondor projektu je podobná, ako pri BOINC projekte, no chýba pri nej centrálny server, ktorý by vytváral úlohy pre počítače v skupine. Vytváranie úloh je zodpovednosťou používateľov systému, HTCondor služi iba na ich automatické spúštanie, preto je menej vhodný a vyžadoval by si väčšie úpravy, ako BOINC. HTCondor je určený na uzavreté, dobre kontrolovateľné siete, kde účastníci sú členmi danej organizácie, preto je menej kladený dôraz na zabezpečenie proti neautentickým výsledkom a zároveň viac kladený dôraz na mechanizmy zaručenia vhodnoti výpočtovej úlohy pre daný počítač.

Nasadenie systému COR na systém BOINC alebo HTCondor by vyžadovalo značné úpravy a konfiguráciu existujúceho systému, no v prípade potreby sú to dobré spôsoby na zníženie trvania času výpočtu. Výhoda systému BOINC spočíva v teoreticky neobmedzenej výpočtovej kapacite pri dostatku dobrovoľníkov, no zároveň pri nízkom záujme zo strany verejnosti sa táto výhoda vôbec nemusí prejaviť.

Oba popísané systémy dokážu spolupracovať. HTCondor sa dá použiť na vytvorenie jednotného distribuovaného systému z viacerých počitačov v organizácii a tento je možné nakonfigurovať tak, aby v čase nečinnosti počítal BOINC úlohy [1].

Kvôli potrebe úpravy systému v prípade použitia BOINC alebo HTCondor bude najlepšou možnosťou nasadiť systém COR ako distribuovaný systém ako je popísané v sekcii 4.1.

[1] Viac informácií na: `http://research.cs.wisc.edu/htcondor/manual/v7.8/3_12Setting_Up.html#SECTION004129000000000000000`

5 Návrh implementácie používateľského rozhrania

Existujúca webová stránka nebola dobre použiteľná a bolo potrebné opraviť jej chyby, vylepšiť použiteľnosť a pridať novú funkcionalitu. Hlavná chýbujúca funkcionalita je filtrovanie existujúcih výpočtov, vylepšenia zadávania výpočtov, prístup k nim a ich poskytnutie používateľom. Pre splnenie cieľov diplomovej práce bolo najviac potrebné pridať nexistujúcu funkcionalitu, ako výpočet vizualizácie trajektórie s danou energiou a implementácia administračnej časti.

5.1 Pridanie vyhľadávania vo výpočtoch

V systéme COR sa simulácie po vypočítaní archivujú a ich počet narastá. Aby používateľ dokázal jednoducho nájsť existujúci výpočet je potrebné implementovať systém vyhľadávania/filtrácie výpočtov podľa kritérií.

5.1.1 Vyhľadávanie podľa času príchodu častíc

Tento typ vyhľadávania poskytuje používateľovi možnosť vyhľadávať simuláciu podľa času, pre ktorý bola rátaná. Tento čas je daný rokom, dňom v roku, hodinou a intervalom prípustných hodín. Valídny vstup pre toto vyhľadávanie je teda napríklad: piaty deň roka 2000, nultá hodina, odchylka ± 24 hodín. Takýto vstup nájde všetky výpočty trajektórií, ktoré boli počítané v systéme COR pre čas dopadu častíc od $1.4.2000\ 00:00$ do $2.4.2000\ 23:59$.

5.1.2 Vyhľadávanie podľa polohy dopadu častíc

Tento typ vyhľadávania poskytuje používateľovi možnosť vyhľadávať simuláciu podľa miesta dopadu, ktoré je špecifikované zemepisnou šírkou, dĺžkou, vzdialenosťou od stredu Zeme a ich odchýlkami. Valídny vstup pre toto vyhľadávanie je teda: zemepisná šírka $0°$ z odchylkou $\pm 10°$, zemepisná dĺžka $0°$ z odchylkou $\pm 10°$, vzdialenosť od zeme 2 zemské polomery z odchylkou ± 1 zemský polomer. Takýto vstup nájde všetky výpočty pre polohu dopadu v rozsahu zemepisnej šírky -$10°$ až $10°$, zemepisnej dĺžky -$10°$ až $10°$ a vzdialenosť od jadra zeme jeden polomer Zeme až tri polomery Zeme.

5.1.3 Vyhľadávanie podľa kroku výpočtu

Tento filter vyhľadá iba výpočty, ktoré majú presne špecifikovaný krok výpočtu. Valídny vstup je vybraný z vysúvacieho menu z hodnotami: 0.01, 0.02, 0.05 alebo 0.1 GV.

5.1.4 Vyhľadávanie podľa dátumu pridania simulácie

Tento filter vyhľadá iba výpočty, ktoré boli do systému pridané v intervale od prvého dátumu do druhého dátumu. Valídny vstup je napríklad od 1.1.2018 do 1.1.2019.

5.1.5 Kombinované vyhľadávanie

Pri tomto type vyhľadávania si používateľ dokáže špecifikovať všetky doteraz spomenuté údaje.

5.2 Vylepšenie zadávania nových výpočtov

Pri zadávaní výpočtov chýbal popis, ktorý hovorí o tom, čo vstupné parametre určujú a aké sú ich prípustné rozsahy. Taktiež chýbal popis modelov a oznam o priemernom trvaní výpočtov simulácií pre dané modely. Popisy bude potrebné pridať ku každému parametru tak, aby bolo jednoznačné ku ktorému patria a boli zobrazené aj v prípade použitia mobilného zariadenia na zadávanie simulácie.

Taktiež je potrebné implementovať možnosť zadania rozsahu hodín no takáto možnosť by mala byť prístupná len pre špecifických používateľov, ktorých administrátor považuje za výhodných. Preto sa pri generovaní formulára musí zisťovať, či daný používateľ má privilégium pridávania výpočtov pre rozsah hodín. Ak má, formulár bude obsahovať dva parametre: prvá hodina intervalu a posledná hodina intervalu. Aby bolo používateľovi jasné, pre ktoré hodiny budú výpočty zadané a má povolený JavaScript, budú mu vypísané všetky hodiny, ktoré budú do systému zadané. Ak používateľ nemá povolený JavaScript, zadávanie bude stále fungovať, no rozsah špecifikujúci zadávané hodiny mu nebude vypísaný.

Ak používateľ nemá privilégium pridávať interval hodín, formulár bude obsahovať iba jeden parameter špecifikujúci konkrétnu hodinu, ktorá bude do systému zadaná.

5.2.1 Nové typy výpočtov

Realizáciou tejto diplomovej práce sa zvýši počet typov výpočtov ktoré môže používateľ zadať. Typy výpočtu ktoré predtým neexistovali a budú pridané sú výpočet ktorý sa zastavý na rigidite prvej priepustnej trajektórie a historický model pre roky 0 až 1950.

Historický model je pridávaný pretože to je jedna z požiadaviek splnenia zadania diplomovej práce, model na zistenie dolnej odrezávacej rigidity bude užitočný pre výpočet optimalizácie popísanej v 3.2.1 a je možné že takýto typ výpočtu bude atraktívny aj pre budúcich používateľov.

Pre zvýšenie prehľadnosti formulára je dobré tieto typy rozdeliť do viacerých formulárov. Keď používateľ klikne na odkaz na formulár na zadávanie výpočtu, bude mu zobrazený formulár pre klasický typ výpočtu, no nad ním budú odkazy aj na formuláre na výpočet dolnej odrezávacej rigidity a historického modelu.

5.2.2 Zobrazenie duplikátnych výpočtov

Ak používateľ zadá požidavku na výpočet so vstupnými dátami, ktoré v tabuľke už existujú v terajšom systéme, nedostane o tom žiadnu spätnú väzbu. Do novej verzie je potrebné implementovať systém, ktorý o existujúcich dátach používateľa po ich zadaní upovedomí a v prípade, že je simulácia dokončená, bude používateľovy odkaz na výsledok na tej istej stránke sprístupnený.

5.3 Vylepšenie stránky na zobrazenie výpočtov

Pri prehľadávaní archívu výpočtov je potrebné vylepšiť prepínanie stránok obsahu. Tento komponent bude obsahovať:

- Odkaz na prvú stránku, ak terajšia stránka nie je prvá.
- Odkaz na predchádzajúcu stránku, ak existuje.
- Číslo stránky, ktoré označuje terajšiu stránku s možnosťou zmeny na ľubovoľnú stránku.
- Celkový počet stránok.
- Odkaz na nasledujúcu stránku, ak existuje.
- Odkaz na poslednú stránku, ak terajšia stránka nie je posledná.

Keďže výpočtov na jednej stránke môže byť veľa a používateľ môže používať túto stránku na mobilnom zariadení kde je nepohodlné „posúvať“ sa. Je potrebné, aby bol tento prvok stále pre používateľa viditeľný a nebolo potrebné, aby používateľ „posúval“ stránku až na spodok, kde sa tento prvok nachádza. Ak ale je používateľ na spodku stránky, kde tento prvok patrí, usadí sa na svoje miesto.

5.3.1 Stav výpočtu

Stav výpočtu je potrebné používateľovi indikovať. Základný spôsob je vypísanie stavu slovom, pri jednosmerových výpočtoch tento spôsob stačí, keďže dĺžka trvania tohoto výpočtu nie je dlhá. Pri viacsmerových výpočtoch je dobré, aby používateľ videl, že jeho výpočet naozaj prebieha. Toto sa dá zariadiť zobrazovaním počtu už vypočítaných smerov. Na túto zmenu bude potrebné pridať do databázy stĺpec s týmto údajom, upraviť modul stránky tak, aby k tomuto údaju z databázy vedel pristupovať a zobraziť ho na stránke a upraviť spúšťač úloh, tak aby tento údaj po každom ukončenom výpočte jedného smeru aktualizoval.

Aby bolo pre každého používateľa vizuálne jasné v akom stave je daný výpočet, je vhodné vizuálne rozlíšiť riadky s dokončenými simuláciami od riadkov s prebiehajúcimi/nezačatými simuláciami. Toto sa dá dosiahnuť podfarbením riadkov, kde zelená farba označuje dokončené výpočty a červená nezačaté/prebiehajúce

výpočty. Výpočet môže byť v stave, kde práca na ňom už začala, no ešte ani jeden smer nebol vypočítaný. Aby si bol používateľ istý, že jeho simulácia sa počíta, je potrebné mú túto informáciu zobraziť tak, že na mieste, kde vypočítaná simulácia má odkaz na detail, bude špecifikované, že práca na simulácii už začala.

Ak práca na výpočte ešte nezačala a používateľ si uvedomí, že zadaný výpočet obsahuje nesprávne vstupné údaje, je z hľadiska trvania výpočtu vhodné, aby mal možnosť takýto výpočet zmazať. Aby si používatelia navzájom nemazali výpočty, je potrebné zaručiť, že používateľ dokáže odstrániť len ten výpočet, ktorý on sám zadal.

5.3.2 Prístup k výsledkom

Ak sú simulácie dokončené, používateľ si dokáže zobraziť ich detail a tam si ich stiahnuť. Ak používateľa detail nezaujíma a potrebuje si stiahnuť viacero výpočtov, je pre neho nepohodlné zobraziť si detail, stiahnuť si dáta, vrátit sa na zoznam so simuláciami a tento postup opakovať. Preto je potrebné už do zoznamu pridať odkaz na stiahnutie. Po kliknutí na tento odkaz sa bez zmeny stránky začne súbor hneď sťahovať.

5.4 Administračná a editačná časť

Administračná časť slúži na správu používateľov. Administrátor v administračnej časti dokáže meniť stav aktivácie používateľov, role používateľov a mazať používateľov.

5.4.1 Zavedenie rolí

Na kontrolu prístupu k zdrojom je vhodné použiť ACL (Access control list), ktorý by sme v implementovanej podobe mohli nazvať aj RBAC (Role-based access control), pretože spĺňa kritéria pre isté podskupiny oboch technológií riadenia prístupu ako píše [22].

Role používateľov v systéme sú nasledovné:

- Administrátor - správca stránky.

- Neprivilegovaný používateľ - používateľ, ktorý dokáže pridávať iba jeden výpočet odoslaním formulára.
- Privilegovaný používateľ - používateľ, ktorý dokáže zadávať rozsah hodín pre daný čas a polohu pre počítanie trajektórií.
- Hosť - neregistrovaný používateľ.

Pre definíciu našej hierarchie prístupu je výhodné použiť metódu čiernej listiny (blacklist) kde všetci používatelia majú prístup ku všetkým zdrojom okrem výnimiek, ktoré zadefinujeme bezpečnostnými pravidlami [23]. Na rozdiel od metódy bielej listiny (whitelist), kde používatelia s danými rolami môžu pristupovať iba k zdrojom, ktoré im sú bezpečnostnými pravidlami sprístupnené.

Metóda čiernej listiny je vo všeobecnosti menej bezpečná, ako metóda bielej listiny, keďže vo veľkom systéme je jednoduché pozabudnúť zadefinovať bezpečnostné pravidlo a systém je v tom prípade nezabezpečený. Metódu čiernej listiny je ale jednoduchšie zadefinovať a kvôli malému počtu zdrojov v našej aplikácii ju je vhodné použiť [23].

5.4.2 Zmena používateľských údajov

Pri registrácii používateľ zadáva svoje osobné údaje a to e-mail, organizáciu do ktorej patrí a krajinu kde pôsobí. Taktiež zadáva aj svoje heslo. Po registrácii môže nastať potreba tieto údaje zmeniť. Napríklad môže nastať situácia, kedy používateľ prestane pôsobiť pre zadanú organizáciu, presťahuje sa do inej krajiny alebo bude mať pocit, že heslo ktoré používa už nie je bezpečné.

Ak by bol jeho účet napadnutý, napríklad získaním jeho identifikátora HTTP relácie [24], útočník by dokázal používateľove údaje editovať. Aby sme zabránili takémuto typu útoku, je potrebné od používateľa pri zmene údajov vyžadovať jeho terajšie heslo.

5.5 Vizualizácia jedinej trajektórie

Pre výpočet vizualizácie jednej trajektórie sú vstupné dáta podobné, ako pri simulácii nájdenia odrezávacích rigidít. Je potrebné zadať fyzikálny model výpočtu, rok, deň v roku, hodinu, zemepisnú šírku, zemepisnú dĺžku a vzdialenosť od stredu

Zeme v polomeroch Zeme. Okrem týchto údajov je potrebné zadať koordináty smeru v lokálnej sústave a rigiditu častice ktorej trajektóriu počítame.

Pri výpočtoch trajektórie nebola kladená požiadavka na ich archiváciu no rovnaký princíp akým sa ukladajú viacsmerové simulácie, je možné uplatniť aj na tento typ výpočtu. Taktiež očakávaná veľkosť výstupu je rádovo v kilobajtoch takže veľký počet výpočtov nebude zaberať na disku veľa miesta.

Z hľadiska databázy bude potrebné pridať novú tabuľku, ktorá bude obsahovať stĺpce na spomenuté vstupné údaje a tiež indikátor stavu výpočtu a cestu k výsledku.

Samotný výpočet bude prebiehať v module spúštania úloh, ktorý bude musieť pravidelne kontrolovať databázu a pridané výpočty načítať, vypočítať a cestu k výsledkom zapísať do databázy.

6 Návrh implementácie výpočtového systému

Pre implementáciu viacerých úprav spomenutých v 2.5 je potrebné upraviť aj modul výpočtového systému alebo zásuvné moduly.

6.1 Vylepšenia existujúcej funkcionality

Zmeny, ktoré budú popísané v tejto sekcii nepridávali do systému zásadne novú funkcionalitu, ale pridávali malé vylepšenia alebo optimalizácie času výpočtu.

6.1.1 Skrátenie času výpočtu vertikálnych trajektórií

Paralelizáciu výpočtov vertikálnych trajektórií dosiahneme zmenou algoritmu výberu dát výpočtov z databázy tak, aby boli najprv vybrané všetky vertikálne trajektórie a až potom sa systém pokúsil získať viacsmerové simulácie. Taktiež bude potrebné určiť jednosmerovým simuláciám prioritu, keďže oproti viacsmerovým je dĺžka samotného výpočtu optimalizovaná a jeden výpočet trvá viacnásobne kratšie, ako pri viacsmerových simuláciách. Preto je vhodné, aby boli po ukončení každého výpočtu z databázy najprv vybrané zadania pre všetky vertikálne výpočty, následne boli vypočítané a až potom sa začnú počítať viacsmerové simulácie.

Doteraz boli z databázy vybrané údaje potrebné pre jediný výpočet reprezentovaný inštanciou triedy `InputData`, pre túto zmenu treba kód upraviť tak, aby existoval zoznam vstupných dát, ktorý bude paralelne počítaný.

6.1.2 Konfigurácia použitého modelu paralelizácie

V existujúcom systéme sú implementované dva modely paralelizácie, vlastná implementácia, kde sa v pravidelných intervaloch kontroluje koľko výpočtov je spustených v danom okamihu a podľa potreby sa spúšťajú nové a implementácia automatickej paralelizácie cyklu `for` použitím OpenMP. Podľa výsledkov meraní času výpočtov bolo zistené, že vlastná implementácia dokončí simulácia rýchlejšie no tento výsledok mohol biť špecifický pre daný stroj. Používateľ v minulej verzii systému nemal možnosť vybrať konkrétnu implementáciu, automaticky bola použitá vlastná implementácia. Do novej verzie je vhodné implementovať konfiguračný atribút ktorým by používateľ dokázal vybrať aký model bude použitý.

6.1.3 Rozšírenie logovacieho podsystému

Logovanie v systéme COR slúži na identifikáciu chybových stavov po tom, ako nastanú. Logované informácie sú užitočne pri implementácii opráv kódu alebo opravy konfiguračného súboru. Doteraz bolo logovanie realizované na štandardný výstup. Zavedením výpočtu trajektórie jednej častice sa ukazuje potreba logovať rôzne diagnostické hlášky do viacerých výstupov. Preto je potrebné do konfigurácie zaviesť názvy súborov do ktorých sa budú informačné hlásenia relevantné k daným typom výpočtov zapisovať.

Keďže k funkciám ktoré budú implementovať logovanie sa bude pristupovať z kontextov viacerých paralelne spustených vláken, je potrebné zaručiť vzájomné vylúčenie prístupu k zdrojom [25].

6.1.4 Zotavenie z chybových stavov

Ak nastane vnútorná alebo vonkajšia chyba počas behu simulácie, ktorá spôsobí, že sa aplikácia vypne, je veľmi pravdepodobné, že sa tak stane počas prebiehajúcej simulácie. Aby systém po opätovnom zapnutí nestrácali čas na výpočet smerov, ktoré sú už vypočítané, je potrebné implementovať procedúru, ktorá podľa mien súborov existujúcich vo výstupnom priečinku zistí indexy dokončených výpočtov. Podľa indexov už dokončených výpočtov dokážeme zistiť nedopočítané výpočty.

Počas výpočtu môže ale nastať prípad, kde výstupný súbor smeru už existuje, ale neobsahuje celý rozsah povolených rigidít. V tomto prípade je potrebné buď

vymyslieť spôsob kontroly obsahu súborov alebo posledných N existujúcich indexov vymazať a prepočítať ich odznova. Pričom vhodné N je stupeň súbežnosti pri ktorom bol program spustený pred neočakávaným ukončením.

6.1.5 Indexovanie verzií

Aby bola možnosť jednoducho pridávať verzie fyzikálneho modelu bez potreby zmeny kódu spúšťača úloh, je potrebné implementovať systém konfigurácie, ktorý to bude podporovať. Informácia o tom, ktorú verziu je potrebné pre daný výpočet použiť je v databáze v tabuľke s výpočtom v stĺpci `Version`. Doteraz tento stĺpec obsahoval presný názov spustiteľného súboru a spúštač identifikoval tieto názvy a podľa tohto údaju určil, ktorú verziu spustí. To znamenalo, že pre každú novú verziu bolo potrebné upraviť kód a rekompilovať aplikáciu.

Je potrebné, aby informácia v tabuľke bola jednoznačne priradená k potrebnej verzii. Jeden zo spôsobou ako to dosiahnuť je ukladať do databázy index, ktorý určuje na ktorom mieste v zozname možných fyzikálnych modelov v konfigurácii sa nachádza ten správny. Takto je potrebné na strane stránky priradiť každému modelu index, ktorý je potrebné v konfigurácii spúštača správne nakonfigurovať.

Pre rôzne verzie vyzerá vstupný súbor rôzne, preto je potrebné informáciu o verzii zapísať do vstupného súboru pre generátor infilov.

6.1.6 Modul na meranie trvania výpočtu a aktualizácia počtu trajektórií

Pre používateľa a aj pre túto diplomovú prácu je zaujímavé poznať trvanie výpočtov. Tento modul zistí čas pri začatí počítania a pri jeho úspešnom ukončení a rozdiel týchto časov zapíše do databázy. Preto je potrebné okrem implementácie tohto modulu pridať do tabuľky výpočtu aj položku `Runtime`. Aby mal používateľ kompletný prehľad o výpočte je možné pridať do databázy aj čas začatia výpočtu pridaním položky `StartTime`, hodnota tejto položky sa aktualizuje v čase získania dát z databázy.

Na implementáciu aktualizácie počtu dopočítaných trajektórií je potrebné do šruktúry tabuľky pre simulácie pridať stĺpec `DirectionsCalculated`, po dopočítaní každého smeru simulácie aktualizovať počet dokončených výpočtov v objekte typu `InputData` a do triedy `Database` pridať funkciu na komunikáciu

s databázou a aktualizáciu tejto hodnoty v databáze po každej zmene.

6.1.7 Návrh implementácie zobrazenia chýbajúcich parametrov používateľom

Chybové hlášky musia byť generované programom, ktorý generuje vstupné súbory pre fyzikálny model, pretože ten číta súbory obsahujúce potrebné údaje. Tieto súbory nemusia daný údaj obsahovať. To, že chyba nastala tiež musí byť zapísané do databázy tak, aby boli údaje prístupné pre webové rozhranie. Aby tento údaj bolo možné do databázy zapísať, bude ho musieť generátor vstupných súborov odovzdať výpočtovému systému. Typické možnosti, ako to dosiahnuť sú soket [1], pipe (rúra) [2], zdieľaná pamäť. [3], zápis do externého súboru alebo návratová hodnota aplikácie. Okrem týchto možností by sa dala implementovať komunikácia s databázou priamo do programu na generovanie vstupných údajov bez odovzdávania stavu riadiacemu programu. Zo spomenutých spôsobov je z hľadiska implementácie najjednoduchšie použiť návratovú hodnotu.

Chybových hlášok môže byť viacero, no v rámci efektivity by bolo dobré, aby zaberali, čo najmenej miesta. Aby sme dokázali vyjadriť všetky možné základné chyby a aj odvodené, ktoré sú ich rôznymi kombináciami, potrebujeme ich nejakým spôsobom zakódovať. Počet základných chýb je 4 z čoho vieme vypočítať počet všetkých kombinácií $2^4 = 16$.

Dobrým riešením tohto problému je využitie bitového pola [26], kde každá pozícia bitu v rámci bajtu má priradenú chybu. Pozícia môže mať jednu z dvoch hodnôt, nulu alebo jednotku. Táto hodnota nesie informáciu o tom, či sa daná chyba vyskytuje alebo nie. Problémom tejto metódy môže byť nekompatibilita z dôvodu rôzneho bitového endiánu [27] na rôznych platformách.

Príklad programu, ktorý používa bitové pole, ako návratovú hodnou je fsck[4]. Bitové pole taktiež rieši jednoduchý výpis chyby vo webovom rozhraní, kde výsledná chyba bude vytvorená súčtom čiastkových chýb.

[1] Viac informácií na: `http://man7.org/linux/man-pages/man2/socket.2.html`

[2] Viac informácií na: `http://man7.org/linux/man-pages/man2/pipe.2.html`

[3] Viac informácií na: `http://man7.org/linux/man-pages/man7/shm_overview.7.html`

[4] Viac informácií na: `http://man7.org/linux/man-pages/man8/fsck.8.html`

6.2 Návrh implementácie REST rozhrania a diskového rozhrania

REST rozhranie je realizované cez internet, použitím jedného z protokolov HTTP alebo HTTPS (Hypertext Transfer Protocol Secure). Na zaručenie bezpečnosti je lepšie použiť protokol HTTPS. Na reprezentáciu dát posielaných na a zo servera použijeme formát JSON. Tento formát je podporovaný knižnicami pre C++ ako aj rôznymi serverovými technológiami preto je jeho použitie vhodné.

Samotné REST rozhranie musí mať vstupné údaje v rovnakom tvare ako metódy rozhrania na získavanie dát v rozhraní databázy. URL sa skladá z časti hostiteľa a časti cesty ku zdroju. Na získavanie alebo aktualizáciu potrebných dát definujeme nasledujúce cesty k zdrojom:

- `calculations/` - Požiadavka na tento zdroj obsahuje ID stroja, počet požadovaných výsledkov, informáciu či sú žiadané nevypočítané alebo dopočítané riadky, informáciu či sú žiadané všetky už dokončené výpočty na opätovné spracovanie a informáciu či má byť rozhraním vrátená simulácia ktorú používateľ vybral na opätovné spracovanie. V odpovedi je očakávané pole záznamov objektov riadkov z databázy alebo JSON objekt z premennou `error` obsahujúcou text chyby.

- `update-calculation/` - Požiadavka na tento zdroj obsahuje ID simulácie, stav na ktorý sa má výpočet nastaviť, čas ako dlho výpočet bežal, cesty k výsledkom, informáciu či je chcené aktualizovať čas behu a počet dopočítaných smerov. V odpovedi je očakávaný počet aktualizovaných riadkov alebo JSON objekt z premennou `error` obsahujúcou text chyby.

- `update-directions/` - Požiadavka na tento výpočet obsahuje ID simulácie, informáciu či je chcené aktualizovať čas behu, aktuálnu dĺžku výpočtu a počet aktuálne spočítaných smerov. V odpovedi je očakávaný počet aktualizovaných riadkov alebo JSON objekt z premennou `error` obsahujúcou text chyby.

- `trajectories/` - Požiadavka na tento zdroj obsahuje ID stroja a počet požadovaných výsledkov. V odpovedi je očakávané pole záznamov objektov

riadkov z databázy alebo JSON objekt z premennou `error` obsahujúcou text chyby.

- `update-trajectory/` - Požiadavka na tento zdroj obsahuje ID trajektórie, stav na ktorý sa má výpočet nastaviť, čas ako dlho výpočet bežal a cesty k výsledkom. V odpovedi je očakávaný počet aktualizovaných riadkov alebo JSON objekt z premennou `error` obsahujúcou text chyby.

6.2.1 Diskové rozhranie

Diskové rozhranie nemusí byť plnohodnotné rozhranie. Keďže ide iba o krátkodobé riešenie, nie je potrebné implementovať aktualizáciu údajov. Bez vedomosti o stave jednotlivých výpočtov by bolo nemožné určiť, ktoré výpočty vo vstupnom súbore už sú vypočítané a ktoré nie. Preto musí systém v prípade diskového rozhrania zaručit, že všetky výpočty budú načítané naraz a spustené iba raz. Po vypočítaní zadaných simulácií sa program musí vypnúť keďže po opätovnom načítaní vstupných dát by sa všetky začali počítať čo by bolo zbytočné.

6.3 Pridanie spektier na 1AU pre výpočet intenzity kozmického žiarenia vnútri magnetosféry

Na výpočet intenzity kozmického žiarenia sa doteraz používalo spektrum získané z experimentu AMS-01 [28]. Toto spektrum bolo merané v roku 1998 pri prekurzor lete AMS experimentu na palube raketoplánu Discovery. Ide o spektrum z oblastí s najväčšou geomagnetickou šírkou pokrytých trajektóriou letu AMS-01, kde sa predpokladá minimálny vplyv magnetosféry na získané údaje. Spektrá intenzity kozmického žiarenia v medziplanetárnom priestore vo vzdialenosti jednej astronomickej jednotky od Slnka (mimo magentosféry) ale existujú aj pre iné roky. Ide o spektrá z meraní experimentov PAMELA, AMS-02, CAPRICE, BESS, Tiger a o spektrá z modelov modulácie kozmického žiarenia v heliosfére. Spektrum na 1AU sa mení so slnečnou aktivitou. Použitie viacerých spektier, ktoré v ideálnom prípade pokryjú celý rozsah slnečnej periódy, preto povedie k zvýšeniu presnosti výpočtov intenzít kozmického žiarenia v magnetosfére.

Aby systém COR dokázal tieto dáta používať, je potrebné aby program na spracovanie výsledkov ako argument dostal priečinok ktorý obsahuje vybrané

spektrum a názov súboru ktorý obsahuje dáta v tvare (rok).(percento určujúce deň) (názov súboru z konkrétnym spektorm). Program na spracovanie takto vyberie dve vhodné spektrá, z nich načíta hodnoty energie a intenzity, podľa časových súradníc vybraného súboru vypočíta percento interpolácie a na základe tejto hodnoty načítane dáta interpoluje.

6.4 Návrh implementácie výpočtu a vizualizácie trajektórie

Na implementáciu výpočtu a následnej vizualizácie trakektórie častice kozmického žiarenia sa využíva upravený modul výpočtu, ktorý je dodaný ÚEF SAV. Generovanie vstupných dát je podobné ako pri normálnej simulácii, no smer nie je generovaný, ale dodaný vo vstupnom súbore a začiatočná a konečná rigidita sú rovnaké. Takto dosiahneme, že modul na výpočet spracuje iba jedinú trajektóriu s danou rigiditou. Na dosiahnutie týchto kritérií je potrebné upraviť generátor infilov tak aby v prípade že je potrebné počítať trajektóriu, vytvoril infil v potrebnom formáte.

Po výpočte sú v priečinku dva súbory „trasfer“ a „tragsm“. Pre túto vizualizáciu je relevantný súbor „tragsm“, ktorý potrebuje vizualizačný modul [12] ako vstupný parameter. Súbor tragsm obsahuje súradnice trajektórie v GSM (Geocentric Solar Magnetospheric) [29] sústave. Celý priečinok sa po vytvorení vizualizácie komprimuje.

6.5 Funkcia pripustnosti magnetosféry

Z hľadiska popisu spektra zakázaných a dovolených rigidít a neskoršieho vyjadrenia intenzít vnútri magnetosféry je užitočné zaviesť takzvanú funkciu priepustosti. Funkcia priepustnosti popisuje pravdepodobnosť prechodu častice magnetosférou na dané miesto v jej vnútri, pre časticu s energiou v danom rozsahu energií.

Výhoda jej zavedenia je v tom, že umožnuje jednoduché vyjadrenie spektra v nútri magentosféry ktoré je konvolúciou funkcie priepustnosti a medziplanetárneho spektra na jednej astronomickej jednotke. Takto je možné pre určenú priepustosť magnetosféry vyjadrenú v podobe funkcie priepustnosti určiť spekturm vnútri

magnetosféry pre rôzne modely, ktoré simulujú spekturm na jednej astronomickej jednotke na hranici magnetosféry alebo pre merania spektier v medziplanetárnom priestore v okolí Zeme.

Na implementáciu funkcie priepustnosti magnetosféry je potrebné zmeniť modul spracovania súborov tak aby generoval vstupný súbory obsahujúce dáta na vizualizáciu priepustnosti magnetosféry pre počítanú simuláciu a implementovať zásuvný modul ktorý vygenerovaný súbor načíta a podľa jeho štruktúry vageneruje vyzualizácie. V prípade výpočtu vertikálnej simulácie vykreslí iba charakteristiku vertikálneho smeru a v prípade vacsmerovej simulácie vykreslí aj charakteristiku vertikálnehosmeru aj charakteristiku sčítania všetkých smerov.

7 Implementácia zmien a rošírení používateľského rozhrania

V tejto kapitole je popísaná implementácia kľúčovej funkcionality, ktorá bola pridaná do časti webová stránka. Pridaná funkcionalita spočíva v pridaní vyhľadávania vo výsledkoch, vylepšení spätnej väzby pri pridávaní duplicitných výpočtov a pridaní riadenia prístupu ku zdrojom a editácie profilu. Používateľské rozhranie bolo testované na funkčnosti pre mobilné zariadenia.

7.1 Implementácia vyhľadávania vo výsledkoch

Keďže údaje, v ktorých sa vyhľadáva, sú uložené v databáze, je potrebné na implementáciu vyhľadávania napísať vhodné SQL „select“ príkazy. Funkcia na vyhľadávanie potrebuje informáciu o tom, aký typ vyhľadávania používateľ zadal na to, aby bol použitý vhodný „select“.

7.1.1 Vyhľadávanie podľa času pre ktorý bol výpočet realizovaný

Výstupné údaje pre tento typ vyhľadávania sú rok, deň roka, hodina a odchýlka v hodinách. V databáze sú rovnaké údaje okrem odchýlky. Na zistenie, ktoré záznamy sú v intervale zadaného dátumu $\pm$ odchýlka je obe hodnoty potrebné konvertovať na hodiny a rozdiel v absolútnej hodnote porovnať s maximálnou odchýylkou. Konverzia roka na hodiny je vyjadrená vo vzťahu 7.1 a konverzia dňa roka na hodiny vo vzťahu 7.2.

$$R_{hodiny} = R * DOY_{max} * 24 \tag{7.1}$$

```
DAYOFYEAR(CONCAT(`Year`,'-12-31'))'
```

Výpis 7.1: SQL príkaz na zistenie počtu dní roka.

```
$daysInYear = date('z', strtotime('31 December '.$dbA[FConst::YEAR]))+1;
```

Výpis 7.2: PHP príkaz na zistenie počtu dní roka.

$$DOY_{hodiny} = DOY * 24 \tag{7.2}$$

Sčítaním R_{hodiny} a DOY_{hodiny} získame časový údaj v hodinách, rovnakým spôsobom je potrebné vytvoriť hodnoty z databázy. Odčítaním týchto hodnôt získame rozdiel, ktorý môže byť kladný alebo záporný. Pre naše účely znamieko nie je dôležité, preto použijeme absolútnu hodnotu rozdielu, ktorú porovnáme zo zadanou hodnotou. Takto vyberieme údaje z databázy, ktoré sa od vstupnej hodnoty odchylujú o dovolenú hodnotu.

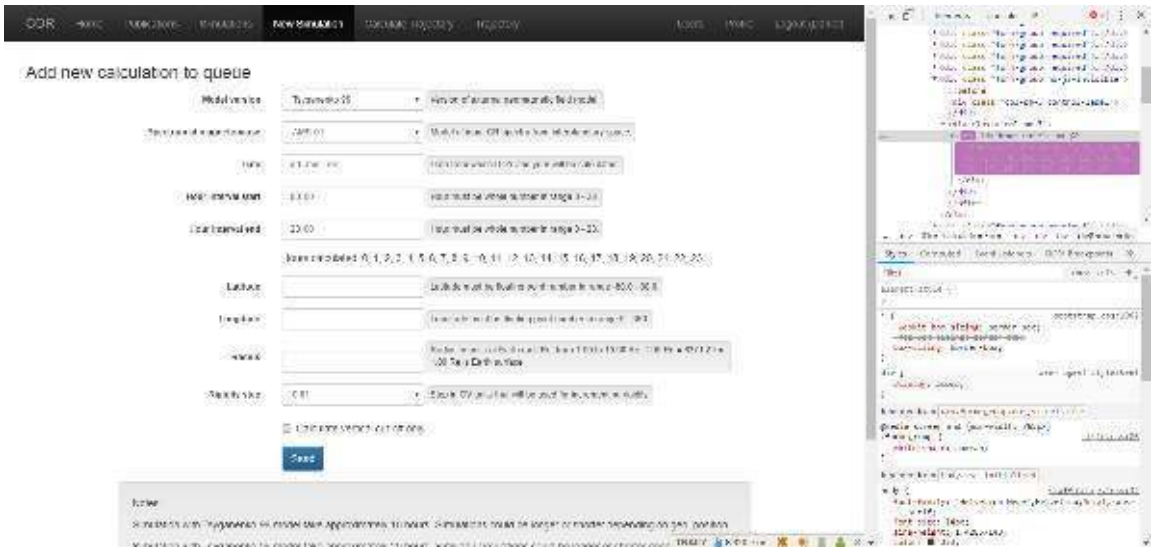
Na zistenie počtu dní v roku v SQL použijeme kód na výpise 7.1 kde `Year` je stĺpec v tabuľke, kde je skladovaný rok ako číslo. Na rovnakú konverziu v PHP (PHP: Hypertext Preprocessor) použijeme kód z výpisu 7.2. Pri konverzii v PHP ešte priátavame jednotku, pretože táto funkcia začína „doy“ počítať od nuly ale my od jednotky.

7.1.2 Ostatné kritéria vyhľadávania

Pri vyhľadávaní podľa polohy dopadu sa zisťuje, či je absolútna hodnota rozdielu menšia ako zadaná odchýlka.

Pri vyhľadávaní podľa času pridania sa porovnáva dátum v databáze zo zadaným dátumom použitím príkazu `BETWEEN`. Keďže v databáze je dátum skladovaný aj z časom pridania, pre databázový dátum sa použije čas zo začiatku dňa a pre zadávaný dátum čas z konca dňa.

Pri vyhľadávaní podľa kroku sa zisťuje hodnota v databáze, ktorá je v intervale zadaná hodnota $\pm$ desatina hodnoty. Dôvodom je vnútorná reprezentácia čísel v plávajúcej čiarke [30].



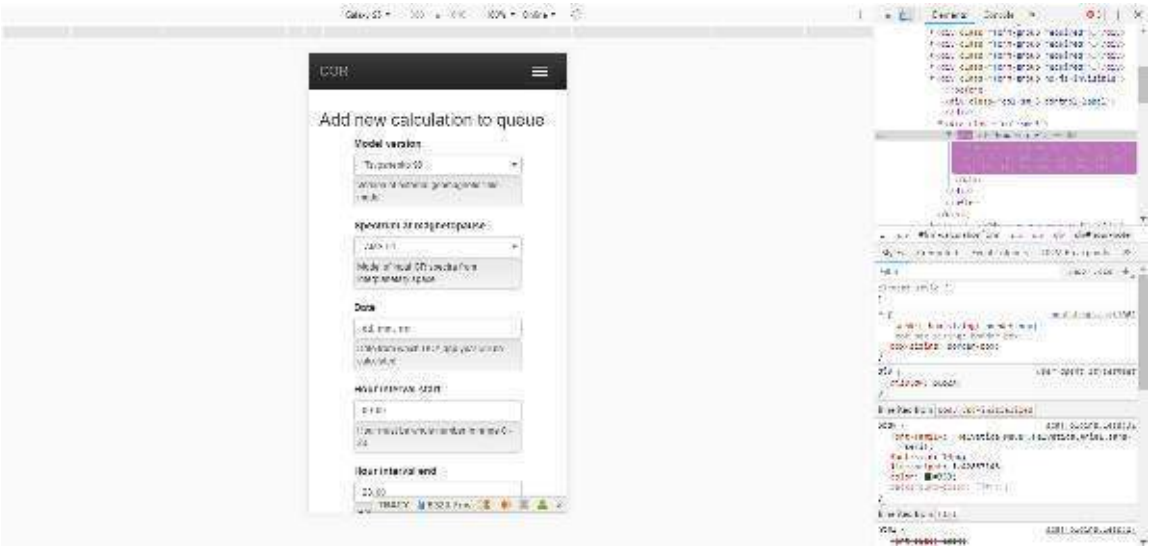
Obr. 7.1: Zobrazenie webovej stránky formulára na pridávanie simulácií v nástroji Chrome DevTools.

7.2 Testovanie funkčnosti rozhrania

Rozhranie má byť funkčné ako na monitoroch tak aj na mobilných telefónoch. Aby sme funkčnosť zaručili je potrebné rozhranie pri pridávaní funkcionality rozhrania testovať. Na testovanie klasického rozhrania pri vývoji je možné použiť ľubovoľný prehliadač webových stránok. Testovanie na mobilnom telefóne je možné v prípade, ak je webová stránka prístupná na internete. Takéto testovanie pri pridávaní každej novej funkcionality by bolo nepohodlné.

Lepším nástrojom na otestovanie funkčnosti rozhrania pre mobilné zariadenia je nástroj prehliadača Chrome DevTools[1]. Na obrázku 7.1 vidíme zobrazenie stránky na veľkých obrazovkách a na obrázku 7.2 zobrazenie stránky na mobilnej obrazovke, kde je demonštrovaná funkčnosť návrhu.

[1]Viac o tomto nástroji na: `https://developers.google.com/web/tools/chrome-devtools/`



Obr. 7.2: Zobrazenie webovej stránky formulára na pridávanie simulácií v nástroji Chrome DevTools v mobilnom režime.

7.3 Implementácia zobrazenia duplikátov po zadávaní výpočtu

Na zobrazenie duplikátov je potrebné zistiť, či sa v zadávaných simuláciách vyskytujú už existujúce zadania. Stĺpce v databáze, ktorých kombinácia hodnôt určuje unikátnu simuláciu sú: „Version“, „Year“, „DOY“, „Hour“, „NumberOfDirections“, „RigidityResolution“, „Latitude“, „Longitude“, „Radius“ a „OptimizationLevel“.

Medzi týmito hodnotami sú aj čísla v plávajúcej čiarke, preto je potrebné ich porovnávať operátorom „BETWEEN“ v intervale s odchýlkou ϵ napríklad $\epsilon = G*0.1$ kde G je najmenšia jednotka rozdielu medzi dvoma hodnotami. V tomto prípade sú do vzorca zadávená údaje zaokrúhľované na dve desatiné miesta, preto $G = 0.01$ a teda $\epsilon = 0.001$.

Po vybraní duplikátov zadaných simulácií je potrebné zo zoznamu zadaných simulácií existujúce dáta odstrániť a ostatné dáta vložiť jedným „insertom“.

7.3.1 Testovanie duplikátov pri viacerých zadávaných hodinách

V prípade, že používateľ zadal simulácie intervalom hodín, je potrebné aby boli z databázy vybrané všetky výpočty, ktoré sa v ostatných parametroch zhodujú a sú v danom intervale hodín. Po výbere treba vytvoriť pole rozdielov zavolaním

```
public static function create()
{
    $acl = new Permission();
    foreach (self::ALL_ROLES as $role)
    {
        $acl->addRole($role);
    }
    foreach (self::ALL_RESOURCES as $resource)
    {
        $acl->addResource($resource);
    }
    $acl->allow(self::ALL_ROLES, self::ALL_RESOURCES);
    $acl->deny([self::GUEST, self::USER], ['Admin:default','Admin:activate']);
    $acl->deny([self::ADMIN, self::USER], ['Register:default','Register:success','Login:
        default']);
    $acl->deny([self::GUEST], ['Logout:default','NewCalc:default','NewCalc:feedback','
        Profile:default','Profile:edit']);
    return $acl;
}
```

Výpis 7.3: Inicializácie objektu povolení podľa kritérií.

PHP funkcie `array_diff`. Toto pole obsahuje N hodnôt hodín.

Nakoniec sa do databázy vloží N simulácii s rovnakými hodnotami pre všetky stĺpce okrem stĺpca `Hour`. Hodnota stĺpca `Hour` je vybraná z poľa rozdielov intervalov zadaných a existujúcich hodín pre zadané simulácie.

7.4 Implementácia riadenia prístupu k zdrojom a editačnej časti

Na výpise kódu 7.3 vidíme inicializáciu objektu povolení ku zdrojom. Najprv sú pridané všetky role a zdroje, ktoré sú deklarované v poliach. Implementácia čiernej listiny je realizovaná tak, že je najprv prístup udelený všetkým roliam ku všetkým zdrojom a potom sú pridané zákazy prístupu pre vybrané zdroje vybraným roliam.

7.4.1 Editačná časť

Formulár na editáciu údajov má pre každý údaj rovnakú štruktúru. Jeden vstup určuje novú hodnotu údaja, ktorý upravujeme a druhý je heslo používateľa na

```
public static function isValid($valueName)
{
    return strtoupper($valueName) === 'EMAIL'
        || strtoupper($valueName) === 'PASSWORD'
        || strtoupper($valueName) === 'ORGANIZATION'
        || strtoupper($valueName) === 'COUNTRY';
}
```

Výpis 7.4: Funkcia na overenie validity vstupného parametra.

autorizáciu. Prezenter preto dostáva URL parameter, ktorý označuje meno upravovaného údaja. Takéto priame odovzdávanie parametra dáva útočníkom veľký priestor na ohrozenie bezpečnosti. Preto je potrebné vykonať kontrolu vstupného parametra ktorá je implementovaná funkciou 7.4.

Ak parameter nie je valídny, používateľovi je vypísaná správa o chybe. Ak parameter bol valídny, je poslaný triede na vytvorenie formulára a následne je vytvorený správny formulár.

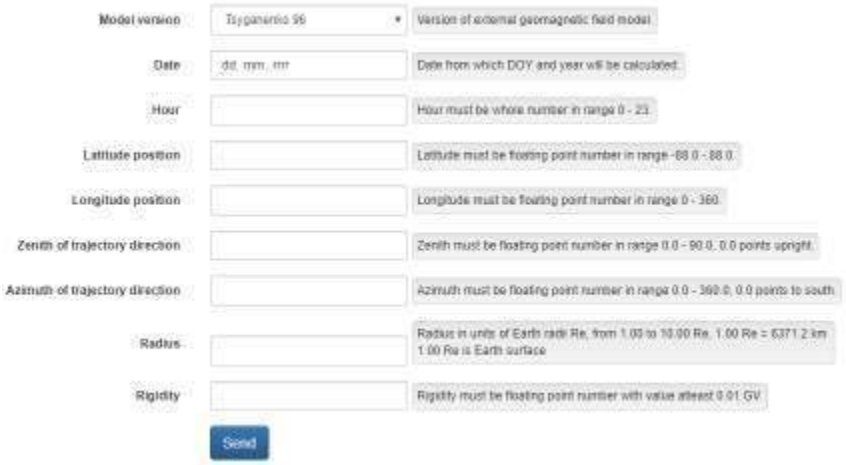
Po odoslaní formulára je volaná funkcia na úpravu údaja v databáze. Pri všetkých typoch údajov je proces rovnaký, iba heslo nie je ukladané ako reťazec ale ako hash.

7.5 Zmeny potrebné pre zadávanie a zobrazovanie výpočtov trajektórie

Hlavnou zmenou je pridanie formulára, ktorý vidíme na obrázku 7.3. Dôležité je zadávanie smeru príchodu častice v horizontálnom súradnicovom systéme popísanom v [31], kde namiesto uhlovej vzdialenosti od horizontu používame uhlovú vzdialenosť od zenitu. Systém na výpočet ale akceptuje súradnicu iba v systéme geografických súradníc. Na prevod použijeme upravenú funkciu z modulu Generátora Infilov, ktorá tento prevod implementuje.

7.5.1 Archív výsledkov a prezentačná stránka

Na implementáciu archívu výsledkov trajektórií použijeme podobný prístup ako na implementáciu archívu výsledkov simulácií. Prezenter bude obsahovať premennú určujúcu stĺpec podľa ktorého sa výsledky zobrazujú, smer zoradenia zostupne



Obr. 7.3: Formulár na zadávanie požiadavky na výpočet jedinej trajektórie častice.

```
foreach (CSConst::VERSIONS_INTEGERS as $key => $value)
{
    if($key === $version)
    {
        $returnValue = $value;
        break;
    }
}
```

Výpis 7.5: Logika prevodu indexu na meno verzie.

alebo vzostupne, index prvého výsledku kvôli stránkovaniu a limit výsledkov na jednu stránku.

Na stránke výsledku sú zobrazené vstupné dáta, vizualizácia trajektórie a odkaz na stiahnutie. Keďže verzia je v databáze implementovaná ako index, no my chceme používateľovi zobraziť jej meno, bola implementovaná funkcia na prevod indexu do mena, ale aj na prevod mena do indexu. Na implementáciu bola použitá vlastnosť PHP, kde polia sú vlastne štruktúry podobné mapám pričom jeden záznam je tvorený párom kľúč a hodnota. Pri iterácií cez pole je možné vybrať jeden pár kľúč hodnota. Hodnoty páru sú reprezentované dvoma premennými ako vidíme na výpise 7.5. Takto dokážeme jednoducho implementovať všetky prevody, ktoré sa dajú implementovať ako pár v oboch smeroch.

7.6 Implementácia vkladania nových typov simulácií

Okrem existujúcich modelov na výpočet trajektórií kozmického žiarenia boli do systému pridané ďalšie tri modely. Dve slúžia na výpočet prvej odrezávacej rigidity a jeden na výpočet pre roky 0 až 1968. Modely na výpočet najnižšej odrezávacej rigidity sú reprezentované konštantou `LCR` a historický model konštantou `HISTORICAL`.

Implementácia spočíva v predávaní konštanty typu formulára ako URL parametra. Ak je parameter nulový, nastaví sa typ formulára na klasický model. Keďže formuláre rôznych typov simulácií sú podobné, nebolo nutné pre každý formulár vytvárať novú triedu a stačilo do konštruktora pridať parameter typu formulára.

Logiku na vytváranie formulára vidíme na výpise 7.6. Všetkým modelom, ktoré majú vlastný typ spekra je tento typ nastavený. Ak spektrá nemajú, je hodnota spektra nastavená na `null` a možnosť na výber spektra do formulára nie je pridaná. Rovnako v prípade historického modelu sa do formulára nepridáva možnosť na výber úrovne optimalizácie.

7.6.1 Implementácia prepočítavania spektra kozmického žiarenia

Implementácia prepočítavania v časti webová stránka spočíva v pridaní kolonky na výber spektra do formulára na zadávanie simulácie, pridanie formulára na prepočítanie už dopočítanej simulácie na stránku výsledného výpočtu a zmena stavu v databáze tak, aby časť výpočtového systému dokázala zistiť výpočty, ktoré je potrebné prerátať. Logika implementácie vytvárania formulára je rovnaká ako na výpise 7.6, kde parameter odovzdaný konštruktoru `$simulationCategory` rozhoduje, aké spektrum bude do formulára pridané. Funkcia na prevod indexu verzie na správnu kategóriu simulácie je na výpise 7.7.

Do databázy bolo potrebné pridať stĺpec `CatalogueIndex`, ktorý označuje index spektra, ktoré je potrebné použiť. Na signalizáciu toho, že je potrebné vykonať spracovanie dopočítaného výpočtu je použitý stĺpec `State`, kde hodnota 83 označuje stav `REPROCESS`. Keď je výpočet v stave `REPROCESS` nie je možné, aby sa používateľ dostal na stránky výsledku výpočtu. Keďže k simulácii dokážu pristupovať aj iný používatel je potrebné, aby zmenu stavu mohol vykonať iba zadávateľ simulácie. Preto sa formulár na výber iného spektra zobrazuje iba ak je prihlásený rovnaký používateľ ktorý výpočet zadal.

```
if($simulationCategory === CSConst::NORMAL)
{
    $this->simulationCategory = CSConst::NORMAL_MODELS;
    $this->spectrumIndices = CSConst::CATALOGUE_INDICES;
    $this->optimizationLevel = CSConst::OPTIMIZATION_LEVEL;
}
else if($simulationCategory === CSConst::LCR)
{
    $this->simulationCategory = CSConst::LCR_MODELS;
    $this->spectrumIndices = null;
    $this->optimizationLevel = CSConst::OPTIMIZATION_LEVEL;
}
else if($simulationCategory === CSConst::HISTORICAL)
{
    $this->simulationCategory = CSConst::HISTORICAL_MODELS;
    $this->spectrumIndices = CSConst::HISTORICAL_CATALOGUE_INDICES;
    $this->optimizationLevel = null;
}
...
if($this->optimizationLevel !== null)
{
    $oLevel = $this->form->addSelect(FConst::OPTIMIZATION_LEVEL, FConst::
        OPTIMIZATION_LEVEL_LAB);
    $oLevel->setItems($this->optimizationLevel,true);
    ...
}
if($this->spectrumIndices !== null)
{
    $catalogue = $this->form->addSelect(FConst::CATALOGUE_INDEX, FConst::
        CATALOGUE_INDEX_LAB);
    $catalogue->setItems($this->spectrumIndices, true);
    ...
}
```

Výpis 7.6: Vytváranie formulára podľa typu simulácie.

```
if($version === CSConst::T05_IDX || $version === CSConst::T96_IDX)
{
    $this->simulationCategory = CSConst::NORMAL;
}
else if($version === CSConst::T05_LCR_IDX || $version === CSConst::T96_LCR_IDX)
{
    $this->simulationCategory = CSConst::LCR;
}
else if($version === CSConst::H_IDX)
{
    $this->simulationCategory = CSConst::HISTORICAL;
}
```

Výpis 7.7: Logika prevodu indexu verzie na typ výpočtu.

7.6.2 Pridanie úrovne optimalizácie

Ako vidíme na výpise 7.6, okrem spektra intenzít bola do zadávacieho formulára pridaná pre niektoré typy výpočtov aj úroveň optimalizácie. Na odovzádavanie všetkých parametrov výpočtovému systém je použitá databáza, preto bolo potrebné upraviť štruktúru tabuľky `calculation` pridaním stĺpca `OptimizationLevel`.

8 Implementácia vylepšenia existujúcej funkcionality výpočtového systému a zásuvných modulov

V tejto kapitole sú popísané zmeny, ktoré boli implementované do novej verzie výpočtového systému a modulov. Hlavným bodom je zvýšenie konfigurovateľnosti systému, ktoré je realizované pridaním konfigurácie zdrojov vstupných dát pre simulácie. Pridaná bola aj možnost výberu paralelného modelu výpočtu a konfigurácia logovania stavu výpočtu. Popísaná je aj optimalizácia výpočtu vertikálnych simulácií, zmena rozhrania na odovzdávanie dát modulu na generovanie „infil"-ov a zotavenie po spustení výpočtu prerušeného náhlym vypnutím aplikácie.

8.1 Zmena štruktúry vstupného súboru pre generátor „infil"-ov

V predchádzajúcej verzii bola štruktúra vstupného súboru jednoriadková, pričom vstupný súbor obsahoval rok, deň, hodinu, zemepisnú šírku, zemepisnú dĺžku a rádius. Pre podporu novej funkcionality bol pridaný druhý riadok, v ktorom je prepínač, ktorý rozhoduje či generátor vygeneruje infily alebo len vráti chybový stav. Ďalšie údaje v riadku sú stav výpočtu, index verzie výpočtu, počet smerov, krok výpočtu a úroveň optimalizácie. Informácie o verzii výpočtu, počte smerov a kroku výpočtu boli v predchádzajúcej verzii odovzdávané ako argument príkazového riadka pri spustení, no takto bolo možných argumentov príliš veľa a v rámci jednoduchšieho používania dávalo zmysel zmeniť spôsob odovzdávania týchto informácií. Pri načítavaní verzie výpočtu prebehne konverzia indexu na enumeračný

```
2010 1 0 -40.000000 190.000000 1.000000
1 82 1 5 0.100000 2
```

Výpis 8.1: Nová štruktúra vstupného súboru pre generátor infilov.

```
enum Version : int16_t
{
    T96 = 0,
    T05 = 1,
    T96_LCR = 2,
    T05_LCR = 3,
    H = 4,
};
```

Výpis 8.2: Enumeračný typ verzií modelov.

typ na výpise 8.2. Konkrétny vygenerovaný vstupný súbor pre generátor „infil"-ov je na výpise 8.1.

8.1.1 Zavedenie historického modelu

Na zavedenie historického modelu bolo potrebné upraviť hlavne modul na generovanie „infil"-ov, keďže vstupný súbor pre historický model neobsahuje riadok s dátami o magnetosfére. Ďalším rozdielom je, že pre historický model neexistuje spoľahlivá optimalizácia pre vertikálny smer ani pre viacsmerový výpočet, preto sa žiaden druh optimalizácie pre tento typ výpočtu nevykonáva. Tieto zmeny boli implementované jednoduchou kontrolou verzie použitého modelu a v prípade, že bola použitá verzia historického modelu, nebol riadok s informáciami o stave magnetosféry do „infil"-u zapísaný a nebola vykonaná žiadna optimalizácia.

8.2 Paralelizácia výpočtu vertikálnych trajektórií

Prvým krokom k paralelizácii je zmena rozhrania funkcie na získavanie dát s databázy. Namiesto funkcie, ktorá vráti dynamicky rezervovanú inštanciu triedy vstupných dát dostane táto funkcia vstupno-výstupný parameter typu smerník na inštanciu štandartnej C++ triedy `vector` vstupných dát. Takto vieme použitím jednej funkcie získať nula a viac výsledkov a podľa ich počtu sa ďalej rohodnúť. Deklarácia tejto funkcie je na výpise 8.3. Táto funkcia taktiež dostane parameter

```
void getInputData(int concurrency,std::vector<SimulationData> *inputDataVector,
    SimulationType simulationType);
```

Výpis 8.3: Deklarácia funkcie na získanie vstupných dát.

```
#pragma omp for nowait
for(unsigned int i=0;i<pmVector.size();i++)
{
    ...
}
```

Výpis 8.4: Spôsob spúštania počítania simulácií.

`concurrency`, ktorý určuje stupeň paralelnosti ktorý je nakonfigurovaný používateľom, tento parameter určuje maximálny počet vstupných dát vertikálnych simulácií. Parameter `simulationType` určuje stav výpočtov, ktoré sú relevantné pre výber, bližší popis je v sekcii 9.1.

Ak sa nepodarí získať viac ako nula výsledkov, kontroluje sa databáza v pravidelných intervaloch. Ak získame jeden výsledok, či už to je simulácia vertikálneho smeru alebo viacsmerová simulácia, počíta sa sériovo. Pre zaručenie správnej funkcie musíme mať záruku, že viac ako jednu mnohosmerovú simulácia touto funkciou nikdy nezískame. Ak získame viacero vertikálnych simulácií, dokážeme ich počítať paralelne. Problém ale nastáva pri generovaní vstupných súborov pre generátor „infil"-ov a pre samotný výpočet, keďže vytvorené súbory majú rovnaký názov. Jednoduché riešenie spočíva v pridaní indexu výpočtu z vektora na koniec vstupných súborov. „Outfil"-y nemusia obsahovať rôzne indexy na konci keďže každá simulácia má vlastný priečinok.

Samotnú paralelizáciu je možné implementovať použitím rozhrania OpenMP ako je znázornené na výpise 8.4. Implementácia takýmto spôsobom zaručí, že v prípade viacerých vertikálnych simulácií budú vertiakálne simulácie počítané na optimálnom počte vláken, keďže hodnota `pmVector.size()` hovorí o počte požiadaviek vybraných z dátového zdroja. Ako bolo spomenuté, táto hodnota je zhora ohraničená hodnotou `concurrency`.

```
enum PM_IMPLEMENTATION
{
    MANUAL,
    OPEN_MP,
};
```

Výpis 8.5: Enumeračný typ stratégie paralelného spúštania výpočtov.

8.3 Výber paralelného modelu

Už v predchádzajúcej verzii boli implementované dve paralelné modely výpočtu. Hlavný model založený na spúšťaní várok výpočtov a kontroly počtu výpočtov z várky, ktoré sú už dokončené v pravidelných intervaloch a následnom spúšťaní výpočtov tak, aby počet v momente spustených výpočtov nepresahoval maximálny počet paralelných vláken. Druhý model bol založený na využití rozhrania OpenMP na automatickú paralelizáciu. V prechádzajúcej verzii bol model založený na OpenMP vyhodnotený ako menej efektívny. Test ale prebiehal iba na jednom počítači a pre iné počítače tento výsledok nemusí platiť. Preto sme zaviedli možnosť nastavenia modelu v konfiguračnom súbore.

Druh paralelného modelu je načítaný z konfiguračného súboru a pomocou funkcie `stringToPM` prebehne konverzia načítaného reťazca do enumeračného typu zobrazenom na výpise 8.5.V prípade nerozpoznaného reťazca metóda `stringToPM` vypíše chybovú hlášku, ktorá používateľa oboznámi o platných reťazcoch a ukončí program. Ak je nakonfigurovaný paralelný model `MANUAL` použije sa vlastná implementácia, ináč sa spustí OpenMP implementácia. Vytváranie objektu správneho paralelného modelu je realizované implementáciou návrhového vzoru Továreň popísaného v [32].

8.3.1 Zmena implementácie vlastného algoritmu spúštania

Implementácia spúštania bola zmenená z algoritmu aktívneho čakania na algoritmus založený na synchronizačných objektoch a vzájomnom vylúčení. Hlavné vlákno programu slúži na iteráciu cez zoznam úloh a ich postupnom spúštaní. Ak je ale dosiahnutá maximálna úroveň paralelnosti, je stav tohto vlákna prepnutý do stavu čakania použitím objektu typu `std::condition_variable`.

V každom vlákne je po ukončení výpočtu vyslaný signál na prebudenie hlav-

```
auto start_time = std::chrono::high_resolution_clock::now();
...
auto end_time = std::chrono::high_resolution_clock::now();
runtime = (float)std::chrono::duration_cast<std::chrono::milliseconds>(end_time -
    start_time).count();
```

Výpis 8.6: Meranie času použitím štandardnej knižnice jazyka C++.

ného vlákna ktoré, ako bolo popísané vyššie do dosiahnutia maximálnej úrovne paralelizmu spúšťa výpočty ďalších smerov. Vysielanie signálu na prebudenie hlavného vlákna je zamknuté použitím objektu zámku typu `std::unique_lock` chráneného objektom vzájomného vylúčenia typu `std::mutex`. Taktiež aj spúšťanie procesu je vo vzájomnom vylúčení s vytváraním vláken tak aby bolo zaručené, že počet naraz spustených výpočtov nepresiahol maximálnu úroveň paralelnosti.

Po spustení všetkých výpočtov je ešte implementovaný cyklus na zaručenie, že všetky spustené výpočty boli dokončené. V cykle sa naprv kontroluje počet už dokončených výpočtov a ak je tento počet menší ako počet smerov, je hlavné vlákno uspané rovnakou podmienkovou premennou ako v cykle spúštania. Ak je nejaká simulácia dokončená, je poslaný signál na prebudenie hlavnému vláknu. Po dopočítaní posledného smeru je cyklus ukončený.

8.3.2 Abstrakcia na meranie času.

Na meranie času výpočtov bolo potrebné implementovať jednoduché rozhranie. Na zistenie aktuálneho momentu je v štandardnej knižnici jazyka C++ implementovaná metóda `std::chrono::high_resolution_clock::now`. Na zistenie času od počiatočného momentu po konečný slúži kód na výpise 8.6. Keďže meranie času je výhodné nie len na zistenie trvania výpočtu ale aj na meranie iných časov, bola do systému COR implementovaná jednoduchá trieda, ktorá je abstrakciou výpisu 8.6. Jej rozhranie obsahuje metódy na začiatok merania času, ukončenie merania a získanie časového intervalu od začiatku po zastavenie merania.

8.4 Implementácia rozšírenia logovacieho podsystému

Logovanie sa týka buď klasickej simulácie alebo vizualizácie konkrétnej trajektórie. Logovanie nemusí byť vôbec povolené a môže byť vypisované do súboru alebo

```
void setDoLog(bool setDoLog);
void setLogToFile(bool setLogToFile);
void setLogPath(const std::string path);
void setTrajectoryLogPath(const std::string path);
```

Výpis 8.7: Verejné rozhranie na konfiguráciu logovacieho podsystému.

```
void log(const std::string &message, bool logDatetime);
void logNoNewline(const std::string &message);
void logTrajectory(const std::string &message, bool logDatetime);
void logNoNewlineTrajectory(const std::string &message);
```

Výpis 8.8: Verejné rozhranie logovacieho podsystému.

na štandardný výstup. Aby bola navigácia v logoch jednoduchšia, v prípade, že používateľ nakonfiguroval logovanie do súboru každý typ výpočtu sa loguje do samostatného súboru.

Na výpise 8.8 je verejné rozhranie k metódam na logovanie správ. V týchto metódach sú volané interné metódy, ktoré samotný výpis implementujú, a tých rozhranie vidíme na výpise 8.9. Parametre interných metód rozhodujú o texte vypísanej správy, súboru do ktorého je text vypísaný (ak je nakonfigurované vypisovanie do súboru) a vlajku, ktorá rozhoduje o tom či sa pred znenie výpisu pripojí dátum a čas momentu, počas ktorého je text vypísaný. Metódy ktoré implementujú globálne nastavenia logovacieho podsystému sú na výpise 8.7. Verejné metódy na logovanie ktorých názov nekončí na `Trajectory`, volajú interné metódy z parametrom cesty nastaveným metódou `setLogPath`, ináč je použitá cesta k súboru nastavená metódou `setTrajectoryLogPath`.

Keďže logovanie môže biť volané z viacerých vláken, bolo potrebné zaručit vzájomné vylúčenie prístupu. To sa dá zariadiť globálnym objektom vzájomného vylúčenia v implementačnom súbore a jeho využitím na zamykanie prístupu ako je to ilustrované na výpise 8.10. Všetky metódy verejného rozhrania na logovanie používajú rovnaký princíp vzájomného vylúčenia.

```
void _log(const std::string &message, std::string *currentLogPath, bool logDatetime);
void _logNoNewline(const std::string &message, std::string *currentLogPath);
```

Výpis 8.9: Neverejné rozhranie k metódam, ktoré implementujú logovanie.

```
void log(const std::string &message, bool logDatetime)
{
    std::unique_lock<std::mutex> lk(mtx);
    _log(message, &simulationLogPath, logDatetime);
}
```

Výpis 8.10: Implementácia vybranej metódy logovania so vzájomným vylúčením.

8.5 Zotavenie z chybových stavov

Na určovanie počtu už dokončených smerov používame vektor boolovských hodnôt o veľkosti počtu počítaných smerov získaného z databázy, kde každý „outfil“ považujeme za neexistujúci. Na zistenie či „outfil“ existuje sa používa regulárny výraz `outfil_([0-9]+)` pre ktorý vyhovujú všetky mená súborov, ktoré sa začínajú na `outfil_` za čím nasleduje 0 a viac čísel. Súbory, ktoré v priečinku vyhovujú tomuto regulárnemu výrazu sú následne testované na veľkosť. Ak je súbor väčší ako 0 bajtov, je považovaný za potenciálne platný „outfil“.

Pri ukončení programu výnimočným stavom je možné, že čiastočný výsledok je v súbore zapísaný a preto má súbor viac ako 0 bajtov. Preto je dôležité poslednú paralelne počítanú várku výpočtov dôkladnejšie skontrolovať. Na zistenie indexu prvého potenciálne falošného pozitíva použijeme vzorec 8.1, kde $I_{zaciatocne}$ je index prvého potenciálne neplatného „outfil“-u, $I_{potencialne\_platne}$ je maximálny index „outfil“-u s nenulovou veľkosťou a $U_{paralelnosti}$ je nakonfigurovaná úroveň paralelizácie. Keďže počítanie prebieha po indexoch od najnižšieho po najvyšší, môžeme považovať „outfil“-y s indexom menším ako $I_{potencialne\_platne}$ za bezpečne dokončené aj v prípade náhleho ukončenia spôsobeného napríklad výpadkom elektrickej energie.

$$I_{zaciatocne} = I_{potencialne\_platne} - U_{paralelnosti} + 1 \tag{8.1}$$

Pre všetky existujúce „outfil“-y s indexom väčším ako $I_{zaciatocne}$ sa testuje obsah súboru. Ak súbor obsahuje riadok, ktorý používateľ nakonfiguroval je „outfil“ považovaný za dokončený. Ak nie, je považovaný za nedopočítaný spolu s „outfil“-my, ktoré v priečinku neexistovali alebo nemali veľkosť väčšiu ako 0. Ako riadok ktorý zaručuje dopočítanie „outfil“-u je vhodné zvoliť riadok z konca súboru, ktorý majú všetky „outfil“-y spoločné. Tento riadok je na výpise 8.11. Tento riadok začína

```
CUTOFF s~rigidities P(S),P(C),P(M) are:
```

Výpis 8.11: Riadok z konca „outfil“-u, za ktorým sú zapísané odrezávacie rigidity.

dvoma medzerami, ktoré sú pri načítaní s konfigurácie kvôli použitej knižnici odstránené. Aby bol celý riadok zachovaný a teda pri porovnávaní bola zhoda je potrebné ho zabaliť do úvodzoviek, ktoré sú neskôr odstránené. Pri type výpočtu iba najnižších dovolených rigidít „outfil“ neobsahuje tento riadok. To nie je problém, keďže kvôli implementácii algoritmu budú považované za neplatné „oufil“-y také, ktoré majú index väčší ako $I_{zaciatocne}$. V najhoršom prípade sa opakovane spočíta už dokončených $U_{paralelnosti}$ „outfil“-ov.

8.6 Zobrazenie chýbajúcich parametrov používateľovi

Na implementáciu generovania chybových hlášok pri načítavaní vstupných dát pre výpočty v module generátora „infil“-ov bolo potrebné upraviť rozhrania funkcií na čítanie vstupných dát pridaním vstupno-výstupného parametra `error`. Pri zistení chýbajúceho alebo nevyhovujúceho parametra je v tejto premennej nastavený chybový stav z enumerácie chýb a stavov 8.12 použitím operácie bitového súčtu.

Najnižšia hodnota, ktorá predstavuje písmeno v ASCII (American Standard Code for Information Interchange) je 65 pre písmeno `A` čo je `01000001` v binárnej forme. To znamená, že na reprezentáciu chybových stavov je možné použiť 6 základných hodnôt čo predstavuje 64 možných kombinácií. Na reprezentáciu stavov výpočtu je možné použiť hodnoty `01000001` až `11111111` čo je 64 stavov. Takáto implementácia zaručuje minimálnu spotrebu dát s priestorom na pridanie dvoch chybových stavov a 60 stavov výpočtu.

Na poslanie týchto dát volajúcemu procesu sa využíva návratová hodnota, ktorá je výsledkom binárneho súčtu výsledných hodnôt všetkých chybových stavov. Po ukončení výpočtu je táto hodnota zapísaná do dátového zdroja. Ak je nakonfigurovaný ako zdroj dát disk, je návratová hodnota ignorovaná.

```
enum State : int
{
    NO_ERROR = 0,
    W_INDICES = 1 << 0,
    PDYN = 1 << 1,
    IMF_Y = 1 << 2,
    IMF_Z = 1 << 3,
    DST = 1 << 4,
    UNUSED_ERROR_STATE_1 = 1 << 5,
    UNUSED_ERROR_STATE_2 = 1 << 6,
    PROCESSING = 'P',
    WAITING = 'W',
    READY = 'R',
    REPROCESS = 83,
};
```

Výpis 8.12: Enumeračný typ chýb a stavov.

9 Implementácia novej funkcionality do výpočtového systému a zásuvných modulov

V tejto kapitole sú popísané všetky vylepšenia systému, ktoré v prechádzajúcej verzii neexistovali. Implementovaná bola vizualizácia jednej trajektórie kozmického žiarenia v magnetosfére Zeme, konfigurovateľné zdroje dát, optimalizácia výpočtu a vizualizácia funkcie priepustnosti magentosféry.

9.1 Implementácia rôznych zdrojov dát

Keďže dáta ktoré potrebujeme získať sú rovnaké a očakávame ich v rovnakom formáte, vieme vytvoriť rozhranie, ktoré budú implementovať všetky konkrétne implementácie zdrojov dát. Definícia takého rozhrania je na výpise 9.1.

Na výpise kódu 9.2 vidíme metódu na vytváranie konkrétnych tried imple-

```
class DataInterface
{
public:
    virtual void getInputData(int c,std::vector<SimulationData> *inputDataVector,
        SimulationType simulationType) = 0;
    virtual void updateCalcRecord(std::vector<SimulationData> *inputDataVector) = 0;
    virtual void updateDirectionsCalculated(SimulationData *data) = 0;
    virtual void getTrajectoryData(int c, std::vector<TrajectoryData> *output) = 0;
    virtual void updateTrajectoryRecord(std::vector<TrajectoryData> *output) = 0;
    virtual ~DataInterface() {}
};
```

Výpis 9.1: Rozhranie zdrojov dát.

```
DataInterface *dataInterfaceFactory(DataSource datasource, Config *cfg)
{
    DataInterface *result = nullptr;
    if(datasource == DataSource::DATABASE)
    {
        result = new Database(cfg);
    }
    else if(datasource == DataSource::REST)
    {
        result = new RestService(cfg);
    }
    else if(datasource == DataSource::DISK)
    {
        result = new DiskDataSource(cfg);
    }
    else
    {
        utils::log("Configured data source doesn't exist!");
        exit(-1);
    }
    return result;
}
```

Výpis 9.2: Inicializácia premennej dátového zdroja podľa konfigurácie.

mentujúcich rozhranie `DataInterface` podľa konfigurácie. Táto metóda je implementáciou návrhového vzoru Továreň. Získavanie dát z MySQL databázy bolo implementované už v minulej verzii. Využíva oficiálnu knižnicu od dodávateľov MySQL [1]. Novým zdrojom dát je REST rozhranie a diskové rozhranie.

9.1.1 Implementácia REST zdroja dát

Na implementáciu REST rozhrania bola potrebná knižnica na vytváranie JSON objektov, a HTTPS klient na zabezpečený prenos dát. Z knižníc na prácu sme vybrali „JSON for Modern C++“ [2]. Na posielanie HTTP požiadaviek sme vybrali „HTTPRequest“ [3].

Keďže na prenos dát používame HTTPS je potrebné nakonfigurovať cestu k lokálnemu páru kľúčov tak, aby dáta boli zašifrované. Taktiež je možné nakonfigurovať vlastnú certifikačnú autoritu. To je potrebné v prípade, že server na ktorý

[1] Stiahnutie na: `https://dev.mysql.com/downloads/connector/cpp/`
[2] Prístupná na: `https://github.com/nlohmann/json`
[3] Prístupná na: `https://github.com/elnormous/HTTPRequest`

```
template <typename T> T RestService::handleResponse(nlohmann::json *jsonResponse, std::
    string key )
{
    T v8{};
    try
    {
        v8 = key.size()==0 ? jsonResponse->get<T>() : jsonResponse->find(key)->get<T>();
    }
    catch(std::exception &e)
    {
        std::string errorString = jsonResponse->find("error")->get<std::string>();
        if(!errorString.empty())
        {
            throw std::runtime_error(errorString);
        }
    }
    return v8;
}
```

Výpis 9.3: Funkcia na spracovanie odpovede.

```
auto trajectories = handleResponse<std::vector<nlohmann::json::object_t>>(&jsonResponse,
     "");
auto updatedRows = handleResponse<int>(&jsonResponse, "updatedRows");
```

Výpis 9.4: Príklady volania funkcie na spracovanie odpovede.

sa pripájame nie je overený globálnou certifikačnou autoritou, no administrátor systému danému serveru verí.

Požiadavka smerujúca na webovú adresu je v tvare `POST` a obsahuje vstupné údaje [4]. Odpoveďou na požiadavku je pole typu `nlohmann::json::object_t` alebo objekt s jediným kľúčom a hodnotou. Ak je očakávaná odpoveď v tvare objekt s kľúčom a hodnotou, je potrebné tento kľúč špecifikovať. Ak je dĺžka kľúča 0, predpokladá sa, že odpoveď je v tvare poľa. Ak odpoveď nie je v tvare ani jednej z týchto možností, nastala chyba a je očakávaná odpoveď v tvare JSON objektu s jediným kľúčom `error` s hodnotou chybového hlásenia. V prípade chybového stavu volanie vyhodí výnimku, ktorú volacia metóda odchytí a jej text vypíše do správneho logu. Ak chybový stav nenastal, analyzuje sa odpoveď a vyberú sa z nej potrebné dáta. Implementácia metódy je na výpise 9.3. Príklady na volanie tejto metódy sú na výpise 9.4.

[4] Popísane v 6.2

```
class SecondaryCalculator
{
public:
    ...
    void start();
    void stop();
protected:
    ...
    virtual void runInParallel() = 0;
};
```

Výpis 9.5: Rozhranie na spúšťanie výpočtov vo vedľajšom vlákne.

9.1.2 Implementácia diskového zdroja dát

Vo výpise 9.1 sú aj metódy, ktoré v tomto type zdroja dát nemajú význam a to metódy na aktualizáciu stavu výpočtu. Tieto metódy sú implementované iba ako informačné výpisy do logov.

Cesty k vstupným súborom sú získané z konfigurácie. Na získavanie dát zo súboru používame štandardné funkcie na prácu so súbormi. Pri načítavaní vstupu sú ignorované riadky začínajúce znakom `#`. Ostatné riadky sú po jednom načítané a z načítaných riadkov sú vytvárané inštancie objektov vstupných dát.

9.2 Implementácia prepočítavania spektra na strane výpočtového systému

Na strane výpočtového systému je potrebné pridať systém detekcie stavov výpočtu v databáze, ktorý bude stav monitorovať v pravidelných intervaloch nezávisle od toho, či výpočtový systém pracuje na výpočte simulácie alebo nie. Preto musí byť tento systém spustený vo vedľajšom vlákne. Tento princíp fungovania je vhodný aj pre iné funkcionality, preto je vhodné ho oddeliť do samostatnej triedy.

Na 9.5 je fragment kódu rozhrania na spúšťanie úloh vo vedľajších vláknach. Funkcie `start` a `stop` sú implementované v tejto triede, pričom metóda `runInParallel`, ktorá je v metóde `start` volaná na vedľajšom vlákne, je implementovaná až v konkrétnych implementáciách.

Konkrétna implementácia metódy `runInParallel` sa už nachádza v konkrétnej triede implementujúcej prepočítavanie spektier. Prvý krok spočíva v pravidelnej

```
enum SimulationType
{
    WHOLE,
    POST_PROCESSING
};
```

Výpis 9.6: Enumeračný typ výpočtu simuacií a výber dát z dátového zdroja.

kontrole stavov výpočtov v databáze. Na fragmente kódu 9.6 vidíme dva typy výberov dát, ktoré špecifikujú o ktoré stavy v zdroji dát má výpočtový systém záujem. Ak sa funkcii na získanie dát ako argument posunie `WHOLE`, vyberajú sa z databázy riadky, ktoré nemajú stav `PROCESSING` a počet vypočítaných smerov sa nerovná nule. Ak je hodnota argumentu `POST_PROCESSING`, sú vyberané riadky ktoré sú v stave `REPROCESS`.

Ak existujú výpočty, ktoré sa majú prepočítať, je spustený generátor „infil"-ov v režime kde negeneruje žiadne súbory, iba vráti chybové stavy. Tento krok je nutný, pretože pred výpočtom bol stav nastavený na `REPROCESS`, pričom stav mohol byť rovnako v stave `READY` ako aj v chybovom stave. Po zistení stavu sa spustí samotné spracovanie „outfil"-ov, kde je programu na spracovanie ako argument poslaný index katalógu ktorý zadal používateľ. Po dokončení spracovania je záznam v databáze aktualizovaný nastavením správneho stavu.

9.2.1 Úprava modulu na spracovanie „outfil"-ov

Prvým krokom je pridanie dvoch argumentov z príkazového riadka a to cestu k priečinku so spektrami a názov hlavného súboru so štruktúrov popísanou v 6.3. Ak je ako argument cesty k priečinku použitý reťazec „null", je použité ako spektrum AMS, na ktorého hodnoty sú polia obsahujúce energiu a intenzitu primárne inicializované.

Keďže hodnoty spektier sú platné pre rozsah rokov a pre každé spektrum existujú hodnoty pre viacero rokov, treba zistiť rok pre ktorý bol výpočet realizovaný. Tento údaj je v každom súbore „outfil", je potrebné ho načítať. Na premenu dátumu v tvare deň mesiac na hodnotu dňa v roku je použitá trieda knižnice Boost `boost::gregorian::date` [5]. V prípade, že vytváranie tejto triedy spôsobí vý-

[5]Dokumentácia prístupná na: `https://www.boost.org/doc/libs/1_39_0/doc/html/date_time/gregorian.html`

nimku v dôsledku neplatného dátumu, čo sa deje pri historickom modeli, je chyba vypísaná na štandardný výstup procesu a ako hodnota pre deň roka je použitá 0.

Po získaní dátumu je potrebné podľa dátumu výpočtu nájsť záznamy, ktoré sa čo najviac približujú k načítanému dátumu. Ak je takýto záznam nájdený a existuje druhý záznam pre ktorý platí, že hľadaný dátum patrí do časového intervalu medzi týmito bodmi, je hľadaný aj ten. Vzdialenosť medzi týmito bodmi normalizujeme a nájdeme bod interpolácie, ktorý vyjadríme hodnotou v intervale od 0 do 1. Najprv načítame hodnoty z prvého súboru a ak existoval aj druhý, interpolujeme medzi oboma hodnotami na základe vypočítaného podielu a výsledok zapíšeme na správny index.

9.3 Vizualizácia jednej trajektórie kozmického žiarenia v magnetosfére Zeme

Na výpočet trajektórie bolo potrebné implementovať metódu `runInParallel` v triede `TrajectoryCalculator`, ktorá rozširuje abstraktnú triedu `SecondaryCalculator` popísanú v 9.2. V metóde `runInParallel` sa pravidelne kontroluje stav požiadaviek v zdroji dát. Ak je v zdroji dát nevypočítaná požiadavka o výpočet, dáta požiadavky sú z databázy vybraté a začne sa ich spracovanie. Ináč sa pokračuje v pravidelných kontrolách každé 3 sekundy. Táto hodnota nie je konfigurovateľná.

Pre všetky načítané dáta je spustený Generátor „Infil“-ov v režime generovania „Infil“-ov pre výpočet jednej trajektórie. Rozdiel oproti normálnemu režimu spočíva v pridaní konkrétneho smeru a jednej hodnoty rigidity na koniec prvého riadka vo vstupnom súbore. Keďže v subore „infil“ je hodnota začiatočnej aj koncovej rigidity, pre obe tieto údaje sa použije rovnaká hodnota.

Po dopočítaní „infil“-ov je spustený výpočet, ktorý prebieha na jednom vlákne. Spracovanie vytvorených dát spočíva v spustení skriptu na vizualizáciu. Tento skript si vyžaduje hodnotu rigidity, preto bola do konfigurácie zavedená nový zástupný reťazec`RIGIDITY_VALUE` ktorý je pri spúštaní skriptov na spracovanie výsledkov nahradený konkrétnou hodnotou rigidity. Po ukončení výpočtu sú údaje v databáze aktualizované.

```
for i in {1..24}
do
  ((idx = i * 5 + 1895))
  ((idx2 = i + 3))
  echo "`sed 's/[\t ][\t ]*/ /g'  < igrf.dat | cut -d' ' -f 1,2,3,$idx2`" > igrf_$((idx)
      ).dat
done
```

Výpis 9.7: Skript na konverziu súboru s IGRF dátami do samostatných súborov.

9.4 Implementácia optimalizácie výpočtu viacsmerových simulácií

O vytvorenie vstupných súborov pre výpočtový modul sa stará modul generátora „infil"-ov, preto je potrebné aby sa úroveň optimalizácie zapisovala do jeho vstupného súboru. Samotnú optimalizáciu vykonáva modul generátora „infil"-ov, ktorý na základe geomagnetických súradníc ako vstupného parametra do vybraného optimalizačného vzorca vypočíta vhodné počiatočné hodnoty rigidity.

9.4.1 Implementácia prevodu geografických súradníc do geomagnetických súradníc

Z možností ako uskutočniť prevod z geografickej do geomagnetickej dipólovej sústavy sme sa rozhodli prepísať relevantné časti kódu `Geopack 2008` [6], konkrétne Fortran rutinu `RECALC_08`, ktorej úlohou je inicializovať vzorec popisujúci interné geomagnetické pole[7]. Dáta [8] pre model IGRF, ktoré sú do vzorca dosadzované, sú vo formáte kde každý stĺpec obsahuje dáta pre daný rok. Pre nás je ale vhodnejšie aby každý rok bol v samostatnom súbore s menom daného roka. Na vytvorenie samostatného súboru pre každý rok bol použitý skript 9.7.

Okrem prepisu sme zmenili spôsob načítavania dát. V originálnom kóde boli dáta zapísané do textu programu. Takýto prístup má nevýhodu v tom, že pri každej zmene dát je nutné dáta do programu doplniť a skompilovať ho. Pri prepise sme zaviedli načítavanie dát zo súborov podľa vstupného roka, čo bolo možné

[6] Kód prístupný na: `http://geo.phys.spbu.ru/~tsyganenko/Geopack-2008.html`

[7] Vzorec popísaný na: `https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html`

[8] Dostupné na: `https://www.ngdc.noaa.gov/IAGA/vmod/coeffs/igrf12coeffs.txt`

```
void geoToMagCorrected(int year, int day,  float th, float fi, float r, float *thMag,
    float *fiMag, float *rMag)
{
    double radToDeg = 180.0 / PI;
    double degToRad = PI / 180.0;
    thi = (90.0 - th) * degToRad;
    fi = (fi) * degToRad;
    float XGEO = r * cos(fi) * sin(th);
    float YGEO = r * sin(fi) * sin(th);
    float ZGEO = r * cos(th);
    double CL0, SL0, CTSL,CTCL, ST0, STCL, STSL, CT0;
    RECALC_08(year, day, &CL0, &SL0, &CTSL, &CTCL, &ST0, &STCL, &STSL, &CT0);
    float XMAG = XGEO * CTCL + YGEO * CTSL - ZGEO * ST0;
    float YMAG = YGEO * CL0 - XGEO * SL0;
    float ZMAG = XGEO * STCL + YGEO * STSL + ZGEO * CT0;
    *rMag = sqrt(XMAG * XMAG + YMAG * YMAG + ZMAG * ZMAG);
    *thMag = 90.0 - acos(ZMAG / (*rMag)) * radToDeg;
    *fiMag = atan2(YMAG, XMAG) * radToDeg;
}
```

Výpis 9.8: Prevod z geografickej do geomagnetickej súradnicovej sústavy

vďaka vyššie popísanej konverzii formátu dát.

Na implementáciu prevodu použijeme fragment kódu funkcie `GEOMAG_08`. Problém použitia tohoto fragmenu ale spočíva v tom, že vstupné súradnice musia byť v karteziánskej súradnicovej sústave a výstup je rovnako v karteziánskej súradnicovej sústave. Náš očakávaný vstup aj výstup je v sférických súradniciach. Preto je potrebné vstupné dáta konvertovať zo sférických súradníc do karteziánskych a výstup opačne z karteziánskych súradníc do sférických ako je znázornené na fragmente kódu 9.8. Taktiež zemepisná šírka sa udáva v intervale $-90°$ až $90°$ no v sférických súradniciach je tento interval $0°$ až $180°$. Ako poslednú konverziu treba ešte stupne previesť na radiány a naopak.

9.5 Implementácia funckie priepustnosti magnetosféry

Na implementáciu je potrebné si určiť šírku binu v rigidite. Vhodná hodnota je desaťnásobok kroku rigidity. Do každého binu je pre každý smer potrebné zaradiť všetky rigidity, ktorých hodnoty sa nachádzajú v intervale binu. Poloha binu je reprezentovaná ako index v poli, pri každom zaradení rigidity do konkrétneho binu sa počet rigidít v bine zvýši o jeden.

Po zaradení rigidít do binov pre všetky smery je potrebné špecificky spracovať vertikálny smer. Pri vertikálnom smere sa do vstupného súboru generuje počiatočná hodnota rigidity a hodnota vrchnej rigdity. Z tohto dôvodu budú mať vrchné biny nad touto rigiditou hodnotu priepustnosti 0. Preto je potrebné nájsť počiatočný index od ktorého sú všetky hodnoty priepustnosti nulové a zmeniť ich hodnoty na jednotky.

V tomto štádiu máme pole funkcií priepustnosti pre všetky smery, no my potrebujeme jedinú funkciu priepustnosti, preto je potrebné vytvoriť nové pole pre viacsmerovú funkciu priepustnosti. Do tohto poľa sčítame hodnoty priepustnosti jednotlivých binov všetkých smerov a vydelíme ich počtom smerov.

Vo vizualizačnom skripte chceme mať už všetky dáta spracované, preto ešte iteráciou výsledným poľom zistíme indexy na ktorých začínajú a končia nenulové a nejednotkové hodnoty.

Výstupný súbor obsahuje vstup pre funkciu prenosu vertikálneho smeru, a ak je počítaný viacsmerový výpočet, tak aj vstup pre viacsmerovú funkciu prenosu. Jedna jednotka dát pre funkciu prenosu obsahuje dva riadky. Prvý riadok obsahuje:

- `startIndex` - Prvý index na ktorom je priepustosť vačšia ako nula.
- `endIndex` - Prvý index od ktorého je priepustosť jednotková.
- `binWidth` - Šírka binu v rigidite.
- `maxRigidity` - Maximálna hodnota rigidity.

Druhý riadok obsahuje hodnoty priepustnosti pre všetky biny a aj tie kde je hodnota nulová alebo jednotková.

9.5.1 Vizualizácia funkcie priepustnosti magnetosféry

Vizualizačný skript je naprogramovaný v jazyku `Python` a využíva knižnicu `matplotlib` [9]. Skript v Python-e najprv prečíta riadky s dátami funkcie prenosu pre vertikálny smer. Pri načítaní hodnôt odstráni nulové a jednotkové hodnoty a pre každý bin určí x-ovú súradnicu ktorú, uloží do poľa. Načítané dáta vizualizuje použitím funkcie `bar` [10].

[9] Prístupná na: `https://matplotlib.org/`

[10] Dokumentácia funkcie: `https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html`

10 Vyhodnotenie

V tejto kapitole vyhodnotíme prínosy jednotlivých bodov zadania diplomovej práce.

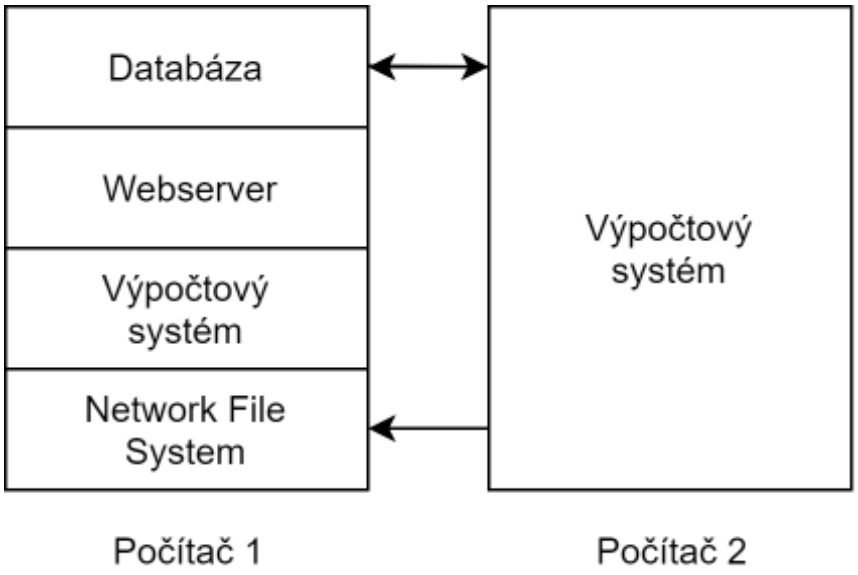
10.1 Aktualizácia systému COR

Pridaním vyhľadávania vo výpočtoch sa zvýšila použiteľnosť systému pri veľkom počte archivovaných výsledkov. Doplnená administračná časť priniesla administrátorovi systému možnosť používateľom priradzovať privilégia. Pridanie ukazateľa stavu v podobe počtu už dokončených výpočtov zvýši informovanosť používateľa o stave jeho výpočtu. Zaručenie funkčnosti stránky pre prehliadače so zakázaným JavaSkript-om rozšíri počet potenciálnych používateľov systému v prípade, že pravidlá ich organizácie JavaSkript nedovoľujú.

Paralelné počítanie vertikálnych výpočtov značne zrýchľuje efektivitu systému a podáva nedočkavým používateľom systému dôkaz o funkčnosti systému. Pridanie možnosti automatickej aktualizácie už existujúcich výpočtov zjednodušuje administrátorom systému pridávanie nových zásuvných modulov do systému. Chybové hlášky informujú používateľa systému o nepresnostiach výsledkov.

10.1.1 Nasadenie COR ako distribuovaného systému

Systém COR bol nasadený ako distribuovaný systém, čo prispelo k urýchleniu výpočtov v prípade, že boli zadané dva a viac výpočtov naraz. Vďaka tomu, že dve inštancie modulu spúštača dokážu pracovať bez toho, aby medzi nimi prebiehala priama komunikácia, nebolo potrebné komplikovane riešiť ich synchornizáciu. Celá synchronizácia prebieha v časti databázového systému, kde je využitá funkcia zamykania riadkov až do ich aktualizácie.



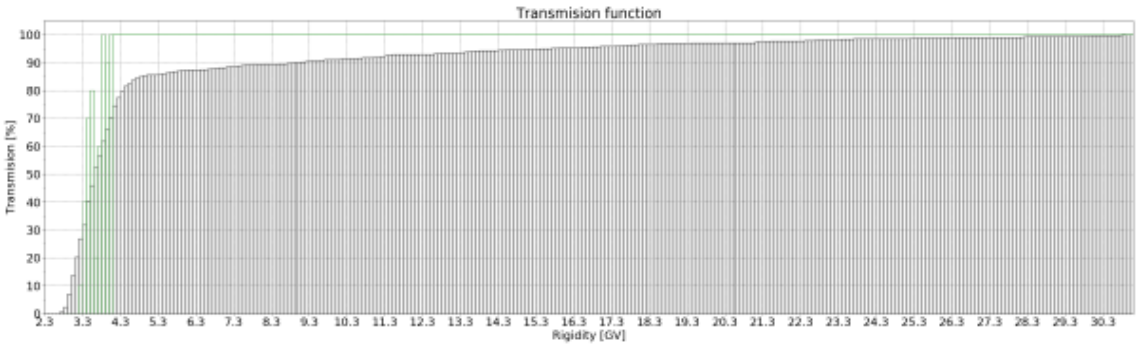
Obr. 10.1: Schéma nasadenia systému na počítače.

Reálne bol systém nasadený v konfigurácii 10.1. Lepšia konfigurácia by bola taká, kde je každý výpočtový systém nasadený na jednom počítači a časti databázy, webového servera a súborového systému by boli na inom. Takáto konfigurácia je možná, systém ale takto nasadený nebol kvôli nedostatku hardvérových prostriedkov.

Súčastné nasadenie systému sa na dĺžke výpočtov prejaví len málo, keďže webový server ani databáza pri terajšom využívaní systému nepotrebujú na svoju činnosť veľa systémových zdrojov a prenos výsledkov z sekundárneho stroja na primárny taktiež nie je náročný na zdroje.

10.1.2 Funkcia priepustnosti

Na obrázku 10.2 je zobrazený príklad vizualizácie funkcie priepustnosti vypočítanej systémom COR. Zelenou čiarou je znázornená funkcia priepustnosti pre vertikálny smer, čiernou je znázornená funkcia priepustnosti pre všetky smery príchodu častíc. Aplikácia správnosti vypočítania a vizualizácie funkcie priepustnosti magnetosféry je kvalitatívne overená porovnaním s funkciou priepustnosti uvedenou v článku [33]. Naša funkcia pripustnosti a funkcia v článku vyzerajú podobne, majú podobnú počiatočnu rigiditu a dosahujú maximálnu priepustnosť (hodnota funkcie priepustnosti 1) na podobných hodnotách rigidity. To ukazuje na veľmi podobné výsledky získané v oboch prípadoch. Úplne identické výsledky sme



Obr. 10.2: Vizualizácia priepustnosti magnetosféry pre Lomnický Štít pre 69. deň roka 2001.

neočakávali, keďže v citovanom článku je použitý starší model geomagnetického poľa Tsyganenko 89 [34].

10.2 Vyhodnotenie optimalizácie trvania viacsmerových simulácií

V tabuľke 10.1 sú informácie o výpočtoch pre rôzne geografické polohy na povrchu Zeme a úrovne optimalizácie počítané pre nultú hodinu desiateho dňa roka 2010 modelom Tsyganenko 05 s krokom 0.01GV. Ako vidíme v tabuľke, prvá optimalizácia konzistentne skracuje čas trvania výpočtu, pričom správnosť výsledku nie je ovplyvnená. Druhá optimalizácia, na rozdiel od prvej, v niektorých prípadoch skracuje čas výpočtu menej ako prvá úroveň optimalizácie.

Výsledky sú v tabuľke hodnotené ako správne aj v prípade, že sa dáta po spracovaní výsledkov nie sú úplne identické. Rozdiel výsledkov je spôsobený nedokonalosťou reprezentácie desatiných čísel v binárnej sústave [35].

Okrem toho v tabuľke vidíme, že skrátenie trvania výpočtu je veľmi silno ovplyvnené miestom na Zemi pre ktorý bol výpočet zadaný. V rovníkovej oblasti udanej geografickou šírkou $-10.05°$ a geografickou dĺžkou $289.79°$ trvá výpočet bez optimalizácie len 5 hodín a skrátenie približne o pol hodinu predstavuje skrátenie o približne desatinu celkového času, aj keď sa štartovacia rigidita zvýšila z pôvodnej hodnoty 0.01GV na 6.29GV. Na rozdiel od toho, v stredných geografických šírkach, ako napríklad na mieste s geografickou šírkou $-60.15°$ a geografickou dĺžkou $289.79°$, spôsobí optimalizácia skrátenie zo 14 hodín na približne 9 hodín, čo je

Latitude	Longitude	Optimalizácia	Štartovacia rigidita	Trvanie	Výsledok
−60.15	287.79	Žiadna	0.01	14:04:31	Správny
−60.15	287.79	Prvá	1.73	09:12:33	Správny
−60.15	287.79	Druhá	1.61	09:34:13	Správny
−10.05	287.79	Žiadna	0.01	05:05:36	Správny
−10.05	287.79	Prvá	6.29	04:24:15	Správny
−10.05	287.79	Druhá	5.59	04:28:12	Správny
40.21	287.79	Žiadna	0.01	14:11:44	Správny
40.21	287.79	Prvá	1.28	10:28:57	Správny
40.21	287.79	Druhá	1.43	09:55:10	Správny
45	0	Žiadna	0.01	14:23:45	Správny
45	0	Prvá	1.61	11:37:39	Správny
45	0	Druhá	2.07	11:12:05	Správny

Tabuľka 10.1: Tabuľka meraní trvania výpočtov na rôznych miestach na Zemi s rôznymi úrovňami optimalizácie.

skrátenie o približne tretinu celkového času pri zvýšení štartovacej rigidity na 1.73GV.

Trvanie výpočtov zobrazené v tabuľke bolo namerané systémom COR použitím abstrakcie na meranie času popísanej v 8.3.2. Jednotlivé série výpočtov pre danú súradnicu boli merané na jednom stroji tak, aby rozdiely v trvaniach výpočtov nemohli byť spôsobené rôznym výkonom počítačov, na ktorých výpočty prebiehali.

V tabuľke 10.2 vidíme trvania výpočtu plného modelu a modelu na nájdenie prvej odrezávacej rigidity pre Lomnický Štít 10. januára 2010 vypočítané modelom Tsyganenko 2005. Rozdielom týchto dvoch časov dokážeme vypočítať teoretické minimum trvania výpočtu pre danú polohu v danom čase. Porovnaním najkratšieho času dosiahnutého prvou optimalizáciou a dĺžkou minimálneho trvania vidíme, že naša optimalizácia je v tomto prípade iba o dve hodiny pomalšia ako teoretické minimum. Malé rozdiely v minimálnom trvaní výpočtu sú spôsobené spúšťaním výpočtov vo várkach.

Úroveň optimalizácie	Plný model	Nájdenie dolnej rigidity	Minimálne trvanie
Žiadna	16:25	9:49	6:36
Prvá	08:30	2:07	6:23
Druhá	11:15	4:45	6:30

Tabuľka 10.2: Tabuľka meraní trvania výpočtov plného modelu, modelu na nájdenie dolnej odrezávacej rigidity a odhad minimálneho trvania výpočtu.

10.3 Vyhodnotenie nových modelov výpočtu a rozšírenia spracovania

V rámci diplomovej práce bolo do systému COR pridaných viacero modelov magnetického poľa a to historický model a dva modely, ktoré ukončia výpočet po nájdení prvej rigidity častice s dovolenou trajektóriou. Tiež bola pridaná katalógová časť pre všetky modely na výpočet trajektórií okrem modelu na výpočet spodnej rigidity.

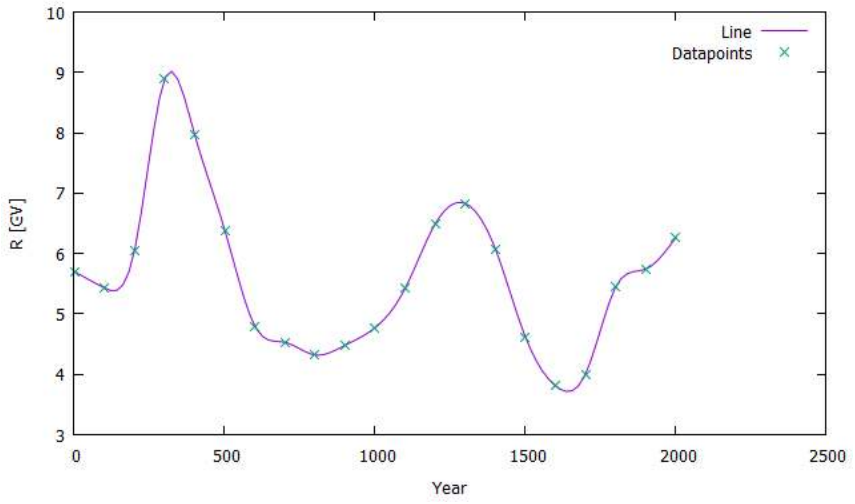
10.3.1 Vyhodnotenie pridania nových modelov výpočtu

Výhoda zmeny formátu vstupných dát určujúcich verziu modelu na index spočíva v jednoduchosti implementácie nových modelov. Jediné zmeny pri pridaní nového modelu do výpočtového systému sú zmeny programu na generovanie „infil"-ov tak, aby generovali súbory v potrebnom formáte.

Správnosť výsledkov historického modelu je vidno pri porovnaní obrázka 10.3 s obrázkom v štúdii [36]. Pre vybraný strednošírkový bod na povrchu Zeme sme spravili sériu výpočtov od roku 1 až po rok 2001 s krokom 100 rokov. Vizualizácia bola vytvorená použitím programu Gnuplot [1].

Modely na zistenie najnižšej rigidity častice s dovolenou rigiditou značne urýchlili čas výpočtu bodov potrebných pre zistenie rovníc, ktoré vykonávajú odhad hodnoty dolnej odrezávacej rigidity.

[1] Viac informácií na: `http://www.gnuplot.info/`



Obr. 10.3: Dolné odrezávacie rigidity vypočítané systémom COR.

Rigidita v GV	Trvanie v sekundách
0.01	21
3.63	12
3.38	6
5.00	4
100.00	4

Tabuľka 10.3: Tabuľka meraní trvania výpočtov trajektórií pre rovnakú polohu a smer príchodu častice.

10.3.2 Vizualizácia trajektórie častice kozmického žiarenia pre danú rigiditu

Samotná vizualizácia bola implementovaná v práci [12]. V tejto diplomovej práci sme zautomatizovali priebeh výpočtu, umožnili používateľom prístup k výsledku a jeho archiváciu. Zadávanie výpočtu aj archivácia je prístupná aj pre neregistrovaných používateľov, keďže samotný výpočet nie je hardvérovo náročný, ako vidíme v tabuľke 10.3. Podobne ako v prípade merania času trvania viacsmerových výpočtov, meranie trvania výpočtu bolo vykonané automaticky systémom COR.

Na obrázku 10.4 vidíme dve vizualizácie trajektórií častíc kozmického žiarenia vytvorené v systéme COR. Na obrázku 10.4a je zakázaná trajektória. Toto vieme povedať, pretože trajektória nekončí na magnetopauze, alebo v chvoste magnetosféry

Rok	AMS	2D
2006	2355.61	1586.19
2009	2355.45	2648.5

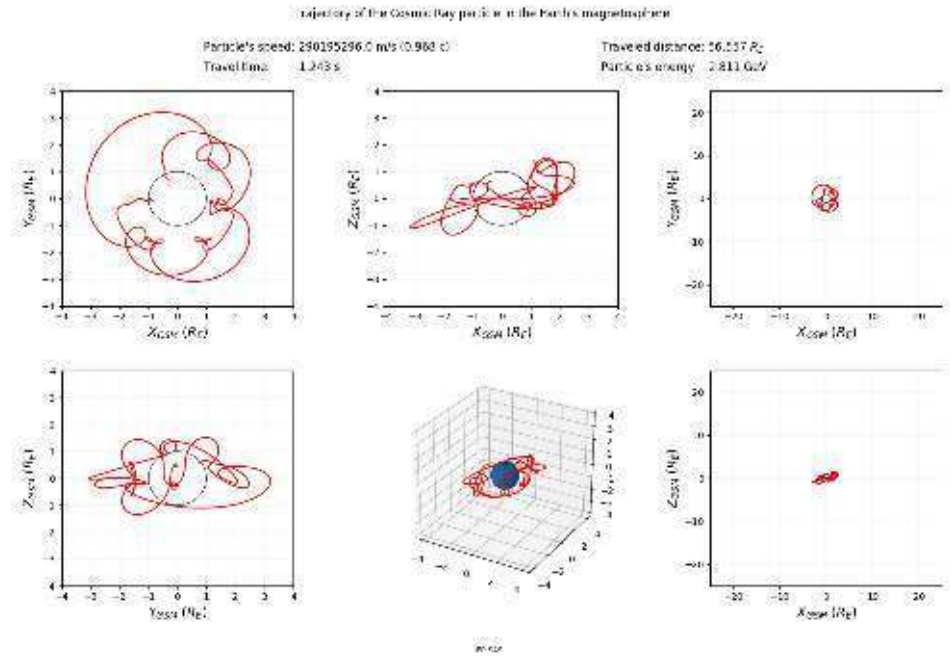
Tabuľka 10.4: Tabuľka hodnôt intenzity kozmického žiarenia použitím spektier AMS a 2D modelu modulácie kozmického žiarenia v heliosfére [37].

ale pretína Zem. Na obrázku 10.4b, je zobrazená povolená trajektória unikajúca von z magentosféry a pretínajúca magnetopauzu.

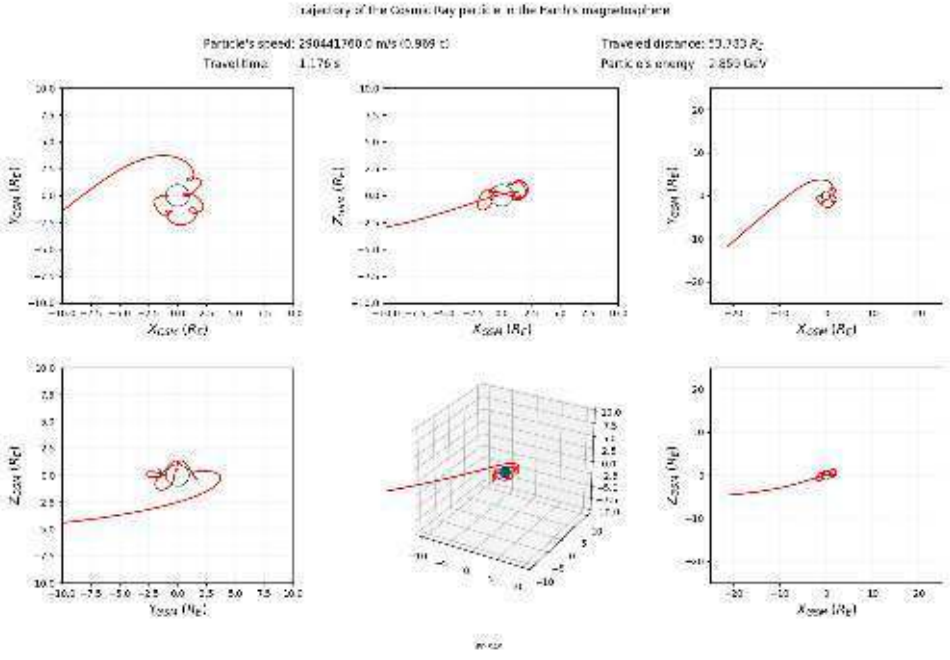
10.3.3 Vyhodnotenie prínosov zavedenia výberu spektier kozmického žiarenia na výpočet intenzity

V súčastnosti použitý katalóg spektier kozmického žiarenia na jednej astronomickej jednotke správne ukazuje nárast intenzity kozmického žiarenia medzi rokmi 2006 a 2009 pre neutrónový monitor stanice Hermanus v Južnej Afrike čo je zobrazené v tabuľke 10.4. Kvalitatívne to popisuje vývoj intenzity kozmického žiarenia na prelome 23 a 24 cyklu slnečnej aktivity.

Do výpočtového systému bola zavedená aj možnosť jednoduchého pridávania nových spektier konfiguráciou. Takto je vytvorený priestor pre pridanie nových spektier pre klasické výpočty aj pre historický model magnetického poľa Zeme.



(a) Vizualizácia trajektórie častice s rigiditou 3.63GV.



(b) Vizualizácia trajektórie častice s rigiditou 3.68GV.

Obr. 10.4: Príklady vizualizácií trajektórií.

11 Záver

V diplomovej práci sme rozšírili systémCOR o sériu nových funkcionalít, konkrétne:

- Pridanie vizualizácie funkcie priepustnosti magnetosféry pre kozmické žiarenie.
- Pridanie nových typov výpočtov a to výpočtu prvej odrezávacej rigidity pre modely Tsyganenko 96 a Tsyganenko 05 a modelu pre výpočet trajektórií kozmického žiarenia v období rokov 0 až 1968.
- Pridanie modulu na výpočet a vizualizáciu jednej trajektórie.
- Pridanie katalógovej časti spektier kozmického žiarenia mimo magnetosféry na magnetopauze pre výpočet hodnôt intenzít kozmického žiarenia.

Hlavným výsledkom diplomovej práce je pridanie optimalizácie dĺžky výpočtu viacsmerových simulácií trajektórií kozmického žiarenia v magnetosfére Zeme. Implementované boli dve úrovne optimalizácie, no systém je pripravený aj na pridanie nových typov optimalizácií.

Prvá úroveň optimalizácie spočíva v minimalizácii výpočtového času na základe odhadu štartovacej rigidity výpočtu, čo vylučuje zo simulácií výpočty nepotrebných zakázaných trajektórií. Pri prvej úrovni optimalizácie sa štartovacia rigidita odhaduje zo série výpočtov pozdĺž magnetického poludníka, kde profil odrezávacích rigidít dosahuje minimálne hodnoty. Na každej geomagnetickej šírke vybraného poludníka pritom do všetkých vypočítaných smerov vyberáme ten s minimálnou rigiditou. Minimálne rigidity sú ako funkcia geomagnetickej šírky popísané polynómom štvrtého stupňa.

Druhá úroveň optimalizácie je založená na sérii výpočtov pre sieť bodov pokrývajúcich celú planétu. Sieť bodov má ekvidistančné vzdialenosti bodov v geomagnetickej šírke a dĺžke. Pre každý bod tejto siete bolo nájdené minimum zo všetkých

Úroveň optimalizácie	Plný model	Nájdenie dolnej rigidity	Rozdiel
Žiadna	16:25	9:49	6:36
Prvá	08:30	2:07	6:23
Druhá	11:15	4:45	6:30

Tabuľka 11.1: Tabuľka meraní trvania výpočtov plného modelu, modelu na nájdenie dolnej odrezávacej rigidity a odhad minimálneho trvania výpočtu.

smerov trajektórií prichádzajúcich na dané miesto a výsledná sieť bola popísaná dvojrozmerným polynómom šiesteho stupňa.

Prvý typ optimalizácie skrátil výpočet až o tretinu trvania neoptimalizovaného výpočtu v závislosti od polohy dopadu simulovaných častíc a stavu magnetosféry v čase, pre ktorý bola simulácia počítaná. Druhý typ optimalizácie je v zrýchlení výpočtu menej konzistentný a v závislosti od polohy dopadu je častokrát pomalší ako prvý typ optimalizácie. Aj v prípadoch, kde druhý typ optimalizácie prekoná skrátenie času výpočtu voči času výpočtu pri prvej úrovni optimalizácie, je toto skrátenie rádovo v desiatkach minút, čo je zanedbateľný čas v porovnaní s viachodinovým trvaním celkového výpočtu.

V tabuľke 11.1 je prezentovaný príklad účinnosti optimalizácie pre výpočet pre stanicu Lomnický Štít 10. januára 2010.

Budúcim vylepšením systému COR bude pridanie tretej úrovne optimalizácie kde každému smeru trajektórie bude vypočítaná hodnota blízka dolnej odrezávacej rigidite. Systém bude doplnený o možnosť výpočtov trajektórií iných častíc kozmického žiarenia než protónov. Ďalšími budúcimi vylepšeniami výpočtového systému je pridanie konfigurovateľného formátu dát načítaných z databázy, konfigurovateľného formátu súborov na odovzdávanie dát zásuvným modulom a optimalizácia kódov fyzikálnych modelov na výpočet trajektórií kozmického žiarenia v magnetosfére Zeme.

Literatúra

[1] P Bobik et al. „GeoMag and HelMod webmodels version for magnetosphere and heliosphere transport of cosmic rays“. In: *arXiv preprint arXiv:1307.5196* (2013).

[2] D Gecášek. „Automatizácia výpočtov trajektórií kozmického žiarenia v magnetosfére Zeme“. Technická univerzita v Košiciach, 2017.

[3] DJ Cooke et al. „On cosmic-ray cut-off terminology“. In: *Il Nuovo Cimento C* 14.3 (1991), s. 213–234.

[4] P Bobik et al. „Transport of cosmic rays in magnetosphere and heliosphere: GeoMag and HelMod webmodels“. In: *ASTROPARTICLE, PARTICLE, SPACE PHYSICS AND DETECTORS FOR PHYSICS APPLICATIONS: Proceedings of the 14th ICATPP Conference*. World Scientific. 2014, s. 189–195.

[5] Niels Provos a David Mazieres. „A Future-Adaptable Password Scheme.“ In: *USENIX Annual Technical Conference, FREENIX Track*. 1999, s. 81–91.

[6] Chuanpeng Li, Chen Ding a Kai Shen. „Quantifying the cost of context switch“. In: *Proceedings of the 2007 workshop on Experimental computer science*. ACM. 2007, s. 2.

[7] Mark Stefik a Daniel G Bobrow. „Object-oriented programming: Themes and variations“. In: *AI magazine* 6.4 (1985), s. 40–40.

[8] Leonardo Dagum a Ramesh Menon. „OpenMP: an industry standard API for shared-memory programming“. In: *IEEE computational science and engineering* 5.1 (1998), s. 46–55.

[9] Dongseok Jang et al. „An empirical study of privacy-violating information flows in JavaScript web applications“. In: *Proceedings of the 17th ACM conference on Computer and communications security*. ACM. 2010, s. 270–283.

[10] Jan Rüth et al. „Digging into Browser-based Crypto Mining“. In: *arXiv preprint arXiv:1808.00811* (2018).

[11] P Bobik et al. „Magnetospheric transmission function approach to disentangle primary from secondary cosmic ray fluxes in the penumbra region“. In: *Journal of Geophysical Research: Space Physics* 111.A5 (2006).

[12] M Vaško. „Vizualizácia atmosférických časticových spŕšok“. Technická univerzita v Košiciach, 2018.

[13] MA Shea, DF Smart a KG McCracken. „A study of vertical cutoff rigidities using sixth degree simulations of the geomagnetic field“. In: *Journal of Geophysical Research* 70.17 (1965), s. 4117–4130.

[14] Arnaud Chulliat et al. *The US/UK world magnetic model for 2015-2020*. Tech. spr. BGS a NOAA, 2015.

[15] Andrew S Tanenbaum a Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[16] Ian Foster. „What is the grid?-a three point checklist“. In: *GRIDtoday* 1.6 (2002).

[17] Douglas Thain, Todd Tannenbaum a Miron Livny. „Distributed computing in practice: the Condor experience.“ In: *Concurrency - Practice and Experience* 17.2-4 (2005), s. 323–356.

[18] David P Anderson. „Boinc: A system for public-resource computing and storage“. In: *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*. IEEE. 2004, s. 4–10.

[19] Andy B Yoo, Morris A Jette a Mark Grondona. „Slurm: Simple linux utility for resource management“. In: *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer. 2003, s. 44–60.

[20] Spencer Shepler et al. *Network file system (NFS) version 4 protocol*. Tech. spr. 2003.

[21] Todd Tannenbaum et al. „Condor – A Distributed Job Scheduler“. In: *Beowulf Cluster Computing with Linux*. Ed. Thomas Sterling. MIT Press, okt. 2001.

[22] John Barkley. „Comparing simple role based access control models and access control lists“. In: *Proceedings of the second ACM workshop on Role-based access control*. ACM. 1997, s. 127–132.

[23] Dave Shackleford. „Application Whitelisting: Enhancing Host Security“. In: *SANS Institute InfoSec Reading Room* (2009).

[24] David Kristol a Lou Montulli. *HTTP state management mechanism*. Tech. spr. 2000.

[25] Edsger W Dijkstra. „Solution of a problem in concurrent programming control“. In: *Pioneers and Their Contributions to Software Engineering*. Springer, 2001, s. 289–294.

[26] Steve Oualline. *Practical C programming*. "O'Reilly Media, Inc.", 1997, s. 104.

[27] Danny Cohen. „On holy wars and a plea for peace“. In: *Computer* 14.10 (1981), s. 48–54.

[28] J Alcaraz et al. „Protons in near earth orbit“. In: *Physics Letters B* 472.1-2 (2000), s. 215–226.

[29] Christopher T Russell. „Geophysical coordinate transformations“. In: *Cosmic Electrodynamics* 2.2 (1971), s. 184–196.

[30] William Kahan. „IEEE standard 754 for binary floating-point arithmetic“. In: *Lecture Notes on the Status of IEEE* 754.94720-1776 (1996), s. 11.

[31] Archie E Roy a David Clarke. *Astronomy: Principles and Practice, (PBK)*. CRC Press, 2003.

[32] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[33] P Bobik, K Kudela a I Usoskin. „Geomagnetic cutoff Penumbra structure: Approach by transmissivity function“. In: *International Cosmic Ray Conference*. Zv. 10. 2001, s. 4056.

[34] NA Tsyganenko. „A magnetospheric magnetic field model with a warped tail current sheet“. In: *Planetary and Space Science* 37.1 (1989), s. 5–20.

[35] David Goldberg. „What every computer scientist should know about floating-point arithmetic“. In: *ACM Computing Surveys (CSUR)* 23.1 (1991), s. 5–48.

[36] K Kudela a P Bobik. „Long-term variations of geomagnetic rigidity cutoffs“. In: *Solar Physics* 224.1-2 (2004), s. 423–431.

[37] P Bobik et al. „Systematic investigation of solar modulation of galactic protons for solar cycle 23 using a Monte Carlo approach with particle drift effects and latitudinal dependence“. In: *The Astrophysical Journal* 745.2 (2012), s. 132.

Zoznam skratiek

ACL Access control list.

AMS Alpha Magnetic Spectrometer.

ASCII American Standard Code for Information Interchange.

BOINC Berkeley Open Infrastructure for Network Computing.

COR Cut-Off Rigidity.

GSM Geocentric Solar Magnetospheric.

GV Giga Volt.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IGRF International Geomagnetic Reference Field.

IP Internet Protocol.

JSON JavaScript Object Notation.

NFS Network File System.

OOP Objektovo Orientované Programovanie.

OpenMP Open Multi-Processing.

PHP PHP: Hypertext Preprocessor.

RBAC Role-based access control.

REST Representational state transfer.

SLURM Simple Linux Utility for Resource Management.

SQL Structured Query Language.

URL Uniform Resource Locator.

WU Work Unit.

ÚEF SAV Ústav experimentálnej fyziky Slovenskej akadémie vied.

Zoznam príloh

Príloha A Používateľská príručka.

Príloha B CD médium.

Príloha BA Systémová príručka webovej stránky v angličtine.

Príloha BB Systémová príručka generátora „infil“-ov v angličtine.

Príloha BC Systémová príručka systému na riadenie výpočtov v angličtine.

Príloha BD Systémová príručka programu na spracovanie výpočtov v angličtine.

Príloha BE Systémová príručka generátora „infil“-ov v slovenčine.

Príloha BF Systémová príručka systému na riadenie výpočtov v slovenčine.

Príloha BG Systémová príručka programu na spracovanie výpočtov v slovenčine.

Príloha BH Zdrojové kódy, spustiteľné súbory kompilované pre Ubuntu 18.04.1 LTS a gcc 7.3.0-27 a skripty.