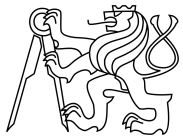
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Informační systém pro ukládání a zpracování dat z přistroje SATRAM na družici ESA Proba-V
Student:	Bc. Stanislav Zubal
Vedoucí:	doc. Ing. Carlos Humberto Granja, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem práce je analyzovat, navrhnout a implementovat informační systém pro zpracování a ukládání dat z vědeckého přístroje SATRAM obsahujícího detektor radiace Timepix spolu s navigačními záznamy z družice ESA Proba-V.
1. Proveďte analýzu požadavků na zpracování a ukládání dat z detektoru Timepix, která jsou korelovaná s navigačními záznamy z družice.
2. Prostudujte stávající knihovnu Pixelman vyvinutou na ÚTEF pro zpracování dat z Timepix a architekturu softwarového systému ROOT vyvinutého v CERN pro ukládání a zpracování dat.
3. Na základě analýzy navrhněte architekturu IS, která bude sloužit pro uživatelsky přívětivé a efektivní ukládání a následné zpracování dat z přístroje SATRAM spolu s navigačními záznamy z družice. Uvažujte o použití systému ROOT a knihovny Pixelman.
4. Analyzujte a navrhněte GUI.
5. Zanalyzujte požadavky dalších plánovaných aplikačních modulů vznikajícího IS na API.
6. Navržený informační systém implementujte a podrobte testům nad reálnými daty.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 7. října 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Informační systém pro ukládání a zpracování dat z přistroje SATRAM na družici ESA Proba-V

Bc. Stanislav Zubal

Vedoucí práce: doc. Ing. Carlos Humberto Granja, Ph.D., ÚTEF ČVUT

28. června 2016

Poděkování

Rád bych poděkoval vedoucímu své práce, panu doc. Ing. Carlos Humberto Granja, Ph.D. z ÚTEF ČVUT a jeho spolupracovníkům z ÚTEF ČVUT, panu Ing. Štěpánu Polanskému a panu Petrovi Mánkovi za jejich vedení, podporu a pomoc.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 28. června 2016 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Zubal, Stanislav. *Informační systém pro ukládání a zpracování dat z přístroje SATRAM na družici ESA Proba-V.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Obsahem práce je analýza a implementace informačního systému pro ukládání, zpracování a analýzu dat z přístroje SATRAM, umístěného na oběžné dráze, na palubě družice ESA Proba-V. Systém umožní spravovat a efektivně použít velké množství dat, zahrnující hodnoty a záznamy z vědeckého přístroje SATRAM-TIMEPIX, dále i navigační a stavová data z družice např. polohu, čas a orientaci. Po analýze požadavků a rešerši možností dostupných databázových řešení, byl pro tuto diplomovou práci vybrán a sestaven databázový systém založen na kombinaci relační databáze a systému ROOT, vyvinutém v CERN za účelem zpracování komplexních a rozsáhlých dat z velkých experimentů ATLAS a CMS na urychlovači částic LHC.

Klíčová slova Databázový systém, SATRAM, Timepix, ROOT, CERN.

Abstract

The aim of this projet is to analyze and implement infromation system for storing, processing and analyzing data from the SATRAM instrument placed in orbit on board ESA Proba-V satellite. The system allows to effectively use large amounts of data, including both values and records of scientific instruments SATRAM-TIMEPIX, as well as navigation and status data from

satellites such as Location, time and orientation. After analyzing of requirements and background research of options and available database solutions, we decided to create database system based on combination of relational database and system ROOT, developed at CERN to handle complex and large data sets from large experiments, ATLAS and CMS from the LHC particle collider.

Keywords Database system, SATRAM, Timepix, ROOT, CERN.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Při měření údajů z fyzikálních experimentů, vzniká velké množství dat, které je nutné efektivně ukládat a analyzovat. Pro tyto účely se využívají specializované informační systémy.

Předmětem této diplomové práce je vytvoření informačního systému pro ukládání, zpracování a analýzu komplexních a rozsáhlých dat z vědeckého přístroje SATRAM[1], vyvinutého v ÚTEF ČVUT a umístěného na palubě experimentální družice ESA PROBA-V.

Přístroj SATRAM je projekt realizován v ÚTEF ČVUT. Je to vědecky a technologicky inovativní přístroj, přizpůsoben pro práci na družicích. Přístroj obsahuje pokročilý a vysoce miniaturizovaný pixelový detektor radiace Timepix. Přístroj jako celek je také vysoce miniaturizovaný, což přináší velké výhody při využití ve vesmírných projektech. Navzdory svým malým rozměrům, dokáže tento přístroj s velkou přesností, citlivostí a dynamickým rozsahem zaznamenat komplexní radiační pole na oběžné dráze podél orbity družice. Cílem tohoto projektu je komplexně změřit a charakterizovat složení a intenzitu radiačních polí podél LEO orbity. Změřená data budou následně sloužit pro tvorbu map popisujících radiační prostředí této orbity.

Pro vytvoření těchto map, je nutné záznamy o radiačním poli korelovat s navigačními údaji z družice např. polohou, časem a orientací družice. Proto je nutné data předzpracovat a následně již korelovaná data v této podobě ukládat. Tyto záznamy je také nutné rozšířit o další vědecké údaje, zjištěné během prvotní analýzy těchto dat.

Uložená data budou dále analyzována, proto je nutné vytvořit systém, který umožní jednoduchý, rychlý a efektivní přístup k těmto datům. Systém musí být optimalizován zejména pro požadavky, které vyžadují čtení dat z velkého množství záznamů a musí umožňovat jejich filtraci podle různých parametrů.

Systém má v budoucnu sloužit také pro jiné projekty ÚTEF ČVUT, proto je nutné analyzovat požadavky nejen projektu SATRAM, ale také jiných projektů a systém navrhnout tak, aby bylo možné jeho funkcionalitu dále rozšířit

pro práci s jinými projekty.

Kapitola 1

Cíl práce

Cílem práce je analyzovat, navrhnout a implementovat informační systém pro zpracování a ukládání dat z vědeckého přístroje SATRAM, obsahujícího detektor radiace Timepix, spolu s navigačními záznamy z družice ESA Proba-V. Systém bude sloužit jako databáze pro rychlý přístup a filtrovaní dat. Dílčí cíle jsou:

- Analyzovat vědecké a navigační údaje pocházející z přístroje SATRAM
- Analyzovat možnosti ukládaní dat, zejména prostudovat systém ROOT[2] vyvinutý v CERN a vybrat vhodný způsob ukládaní dat
- Analyzovat a navrhnou GUI
- Prostudovat software Pixelman[3]
- Navrhnout architekturu informačního systému pro zpracování a ukládání dat z vědeckého přístroje SATRAM
- Navrhnout databázový systém pro ukládání dat
- Implementovat databázový systém pro ukládání dat
- Otestovat databázový systém pro ukládání dat
- Vytvořit dokumentaci

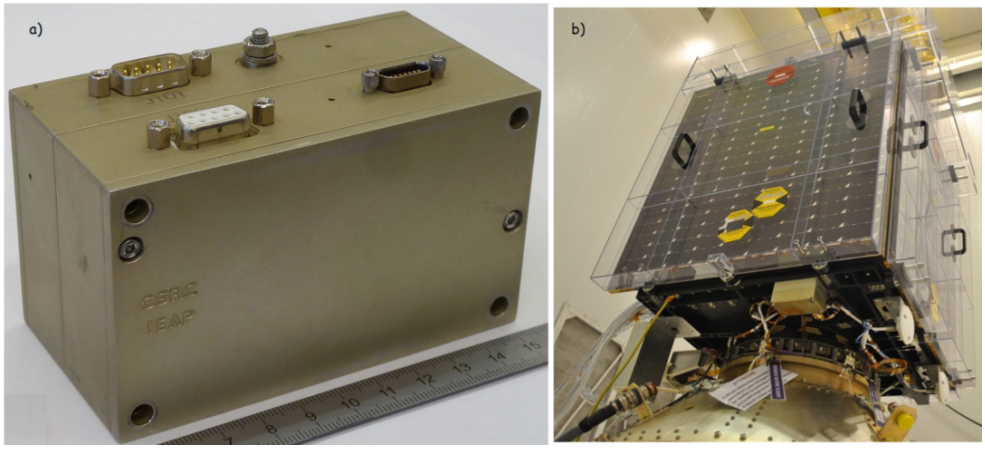
Kapitola 2

Co je to SATRAM

SATRAM[1] je vědecký přístroj, obsahující detektor radiace Timepix spolu s elektronikou potřebnou pro fungování tohoto detektoru, prvotního zpracování údajů, a komunikaci s družicí, na které je umístěn. Hlavním úkolem tohoto zařízení je zaznamenávání radiačního pole, kterému je vystaven. SATRAM je navržený tak, aby byl co nejvíce nezávislý na družici, na které je umístěn, z důvodu co největší flexibility při integraci tohoto přístroje na vybranou družici. SATRAM je tedy vytvořen jako samostatná jednotka s vlastním stíněním z hliníku o rozměrech 55,5 na 62,1 na 107,1 mm s celkovou hmotností 380 g, které můžete vidět na obrázku 2.1. V oblasti detektoru je stínění z hliníku ztenčené na 0,5 mm, čímž poskytuje detektoru stínění před přímým slunečním zářením. Celá jednotka nevyžaduje aktivní kontrolu teploty z družice. Přebytečné teplo je vyzařované pomocí povrchu přístroje, přičemž pracovní teplota se pohybuje od -40°C do +60°C a skladovací teplota se pohybuje od -50°C do +80°C. Pro komunikaci s družicí a pro napájení jsou využity tři konektory Cannon D-SUB9M, které poskytují přístroji SATRAM napětí 28 V DC z družice, přičemž maximální spotřeba tohoto přístroje je 3 W.

2.1 Umístnění

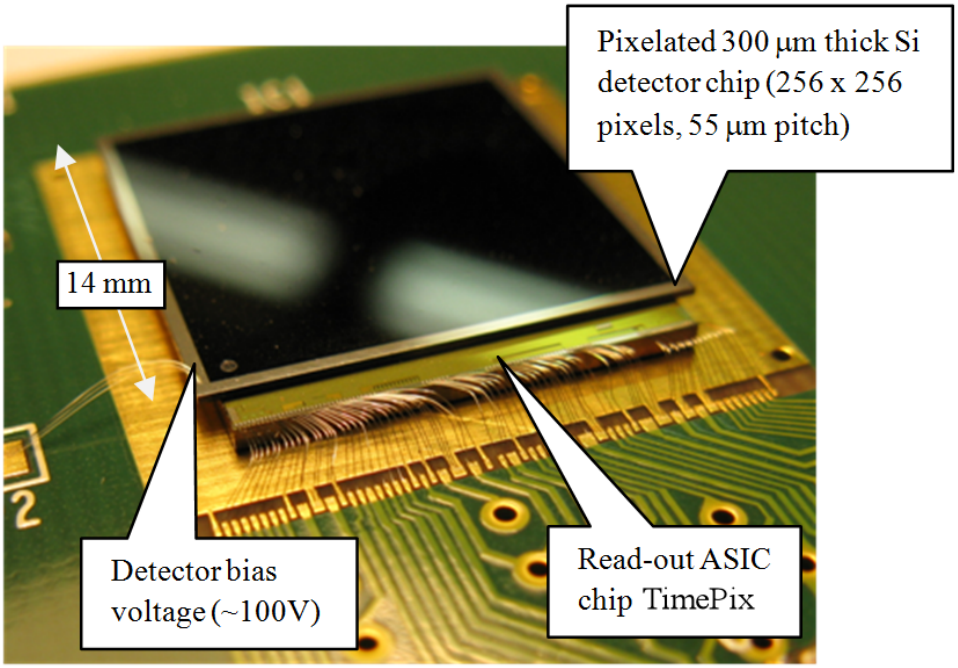
SATRAM je umístnění na trupu družice ESA Proba-V, která byla vypuštěna na orbitu 7. května 2013. Tato družice se pohybuje po nízké oběžné dráze okolo Země ve výšce 820 km po polární orbite s inklinací 98°. Tohle umístnění umožňuje přístroji SATRAM sběr dat o radiačním poli v okolí Země, které jsou užitečné pro mnoho studií jako například studování geomagnetické a solární aktivity slunce, sledování interakce solárního a galaktického kosmického záření s magnetosférou, sledování distribuce radiačních pásů Země, systematické sledování a předpovídání vesmírného počasí a studování radiační fyziky vyšších vrstev atmosféry.



Obrázek 2.1: Na obrázku vlevo se nachází přístroj SATRAM, na obrázku vpravo se nachází družice ESA Proba-V, která má ve své spodní části připevněný přístroj SATRAM

2.2 Timepix

Detektor radiace Timepix, který je součástí přístroje SATRAM, je hybridní polovodičový pixelový detektor s vysokým dynamickým rozsahem a nízkým elektronickým šumem, odvozený od detektoru Medipix2. Hybridní architektura umožňuje využití různých materiálů a různé tloušťky pro senzor. Čip je vytvořený pomocí 0,25µm CMOS technologie. Čip m rozměry 14×14 mm a obsahuje matici pixelů s rozlišením 256×256. Detektor Timepix můžete vidět na obrázku 2.2. Každý z pixelů je připojený k vlastnímu předzesilovači, diskriminátoru a 14-bitovému počítadlu. Všechny pixely tohoto čipu pracují souběžně během expozice. Je možné si zvolit libovolnou délku expozice avšak běžně se používají expoziční časy v řádu milisekund až sekund, ne delší. Po expozici jsou data z čipu pročtena a čip je vyčištěn. S využitím vysokofrekvenčního časovače a počítadel v jednotlivých pixelech je možné využit tři módy pro tento detektor a to mód *Time over Treshold* kde čip registruje pro každý pixel čas, po který v daném pixely síla signálu překračuje předem danou hranici. Dále mód *Time of Arrival* kde se registruje čas, ve kterém částice interagovala s čipem od začátku expozice. A nakonec je možné využít *Counting* mód, ve kterém každý pixel počítá kolik částic interagovalo s daným pixelem během expozice. Každý z pixelů je možné nezávisle nastavit tak, aby pracoval v jednom z těchto módů.



Obrázek 2.2: Detektor radiace Timepix

Kapitola 3

Analýza

3.1 Vstupní data

3.1.1 Popis vstupních dat

Vstupní data pocházejí z přístroje SATRAM a družice ESA Proba-V. Vědecká data jsou získávána z detektoru Timepix, který je součástí přístroje SATRAM. Po expozici a pročtení dat z detektoru Timepix jsou data uložená v rámci přístroje SATRAM, kde jsou korelována spolu s daty o nastavení detektoru a navigačními a stavovými daty z družice. Tyto data jsou každých přibližně 90 minut odeslána na Zem pomocí komunikačních kanálů družice.

Tyto data se skládají ze třech souborů, a to surových dat z čipu, indexového souboru a souboru s popisem dat.

Soubor se surovými daty obsahuje naměřená vědecká data. Jedná se o textový soubor kde každý řádek, kromě řádku, které oddělují jednotlivé snímky, obsahuje dvojici čísel. První číslo je číslo pixelu, kde pixely jsou očíslované od 0 do 65535. Druhé číslo představuje hodnotu čítače daného pixelu po pořízení daného snímku. Pixely s hodnou čítače 0 jsou vynechány. Pro oddělení jednotlivých snímků jsou použité řádky obsahující znak # .

Indexový soubor obsahuje index pro všechny snímky v souboru se surovými daty pro rychlejší přístup k jednotlivým snímkům.

Soubor s popisem dat obsahuje data o nastavení a stavu detektoru spolu s navigačními a stavovými údaji družice pro každý snímek. Mezi těmito daty jsou zejména data o délce expozice, čase začátku expozice, interní teplotě družice, minimálním a maximálním napětí vybraných součástek, frekvenci časovače, poloze a orientaci družice ve vesmíru a magnetickém poli, ve kterém se družice nachází.

Korelace dat z detektoru spolu s navigačními a stavovými daty z družice poskytuje kontext naměřeným datům, díky kterému je možné provádět přesnější analýzy naměřených dat.

3.1.2 Množství dat

Data jsou z družice zpátky na Zem odesílána jednou za den. Množství dat získaných za den se může lišit v závislosti na nastavení parametrů měření, zejména na délce expozice a frekvenci snímkování. Při krátkých expozicích je nízká pravděpodobnost, že detektor zasáhne mnoho částic, a tak každý snímek obsahuje méně částic a tedy menší množství dat na jeden snímek. Při delších expozicích na druhou stanu vzrůstá pravděpodobnost, že se stopy dvou či více částic překryjí a tedy budou muset být vyřazeny z analýzy, čímž se také sníží množství dat na snímek. Na délce expozice také závisí frekvence snímkování. Čím delší je expozice tým menší je maximální možná frekvence snímkování. Protože mezi každými dvěma snímky musí dojít k přečtení dat a vyčistění čipu detektoru, které trvá fixní množství času. Při snižování délky expozice se tedy rychlost snímkování nezvyšuje lineárně, ale když se délka expozice začne blížit délce čtení a čištění, začne se značně zvětšovat poměr času kdy detektor neměří data k času kdy data měří, což snižuje množství získaných dat za jednotku času.

Po dobu tří let, během kterých přístroj SATRAM měří data, už nasbíral velké množství dat. Z těchto dat je možné odhadnout přibližné množství dat, které bude vyprodukováno tímto systémem za jednotku času. Během jednoho dne je vytvořeno v průměru 3 000 snímků z nichž každý obsahuje v průměru 100 clusterů. Tyto data mají ve velikost přibližně 300 MB v nekomprimované textové podobě popsané v podsekci 3.1.1. Po cluster analýze a při využití formátu a způsobu uložení dat popsaného níže v podsekci 3.4.2, zabírají data za jeden den přibližně 55 MB.

Za rok je to tedy přibližně 1 000 000 snímků a 100 000 000 clusterů. To je přibližně 19,6 GB dat za rok po zpracování.

3.2 Požadavky na systém

Systém bude sloužit především pro ukládání a správu dat z přístroje SATRAM. Data uložená v systému budou také využívána pro další analýzu. Proto je potřebné systém optimalizovat i pro tento účel. Dále je nutné aby systém splňoval tyto funkční požadavky:

- Systém bude poskytovat standardní webové aplikační rozhraní pro přístup k datům
- Data budou načítána ze souborů produkovaných pomocí *Software pro cluster analýzu a korelaci s navigačními údaji* t.j. z *rootfile* souborů
- Systém bude poskytovat možnost přímého stažení souborů *rootfile*, které byli použité pro načítání dat do systému
- Data budou organizovaná pomoci indexu

- Data budou tříděna pomoci filtrů
- Software bude možné konfigurovat pro použití s jinými projekty

Systém bude také splňovat tyto nefunkční požadavky:

- Software bude kompatibilní s platformou Linux
- Software bude používat programy pro cluster analýzu vyvinuté na ÚTEF ČVUT
- Data budou přenositelná mezi systémy

Další požadavky na systém vyplývají zejména z vlastností dat.

3.2.1 Vlastnosti dat

Data produkovaná přístrojem SATRAM a jinými zařízeními využívajícími detektory Timepix nebo Medipix mají z principu fungování těchto detektorů velice podobný charakter, podobný jiným datům produkovaným v rámci experimentů v částicové fyzice. Jde o velké množství dat záznamů s předem danou strukturou, tyto data můžeme obecně rozdělit na data o experimentu a data o událostech.

Data o experimentu jsou data, která popisují podmínky měření, jako nastavení detektorů, vlastnosti prostředí a podobně. V případe přístroje SATRAM tyto data odpovídají navigačním datům z družice, stavovým datům přístroje SATRAM a družice a nakonec nastavení detektoru Timepix. Každý snímek tak lze pokládat za samostatný experiment s vlastním nastavením.

Data o událostech jsou data, která popisují jednu naměřenou událost, která proběhla během experimentu. V případe přístroje SATRAM tyto data odpovídají jednomu clusteru a datům popisující všechny jeho vlastnosti.

Data v těchto skupinách dat tvoří seskupení velkého množství atributů, které tak popisují experiment či událost. Tyto seskupení typicky není možné rozumně dělit na menší logické celky, protože data mezi jednotlivými experimenty nebo jednotlivými událostmi nesdílí předem dané nastavení podmnožiny atributů. Například o žádných dvou clusterech nemůžeme s určitostí říci, že budou mít stejný azimut na základě jiných vlastností těchto clusterů.

V případech kdy je možné data více rozdělit do logických celků, jde vetšinou pouze o specializaci dat. V takovémto případě mají tyto data stromovou strukturu, kde k obecným datům přidáváme další specifická data. Data z těchto experimentů netvoří složitější datové struktury, kde by se nacházeli cyklické vztahy a podobně.

3.2.2 Přístup k datům

K datům se bude přistupovat zejména pro účely analýzy nebo vizualizace dat.

V případe analýzy dat se bude přistupovat k velkému množství dat současně. Požadavky budou obsahovat filtry, které specifikují vlastnosti požadovaných záznamů, čímž omezí množství vybraných záznamů a také budou obsahovat možnost filtrovat atributy, které jsou pro danou analýzu potřebné aby se omezilo množství posílaných dat na minimum.

V případě vizualizace dat se bude většinou přistupovat k menšímu množství dat, ale požadavky na vizualizaci dat budou přicházet častěji. V případě dat z přístroje SATRAM lze předpokládat přístup k základním informacím o jednotlivých snímcích, kde budou požadované informace o jednom snímku a následně informace o jiném snímku.

Z pohledu běžného uživatele systému nebude možné data v systému měnit, mazat ani přidávat. Data budou také dostupná pro odbornou i širokou veřejnost v rámci spolupráce a popularizace vědy. Díky tomu není nutné v rámci systému implementovat žádnou správu přístupu k datům.

3.2.3 Komunikační API

Systém bude poskytovat REST API pro komunikaci s jinými aplikacemi. Komunikace bude probíhat pomocí HTTP protokolu, kde data budou posílaná převážně ve formátu JSON. Komunikační rozhraní musí poskytovat možnost pokročilého filtrování dat, přímo na serveru, aby bylo omezeno množství odesílaných dat na co nejnižší možnou míru.

3.2.4 Konfigurace informačního systému

Informační systém bude možné jednoduše konfigurovat za účelem přizpůsobení aktuálním požadavkům a hardwarové, či softwarové konfiguraci serveru, na kterém je používán.

Takovými nastaveními jsou zejména nastavení komunikačního rozhraní, jako například komunikační port, nastavení příslušících složek na serveru pro přijímaní a ukládaní dat, nebo nastavení pro komunikaci s jinými aplikacemi potřebnými pro běh programu.

Konfigurace bude také umožňovat přizpůsobení, či přidávání funkcionalit informačnímu systému, tak aby bylo možné tento systém použít nejen pro účely správy dat z přístroje SATRAM, ale také pro použití s jinými projekty, jejichž data se vyznačují podobným charakterem. Systém tedy musí poskytovat dostatečnou flexibilitu, aby byl schopen přizpůsobit se požadavkům na ukládaní jiných dat, než pro které je primárně určen.

3.2.5 Uživatelé systému

Systém bude sloužit zejména pracovníkům ÚTEF ČVUT pro rychlÝ přístup k datům sesbíraným pomocí přístroje SATRAM. Tyto data budou dále analyzována za účelem vědeckého bádání. Systém bude poskytovat data také širší odborné i laické veřejnosti, zejména za účelem popularizace vědy obecně a také ÚTEF ČVUT.

3.3 Způsob uložení dat

V této sekci analyzujeme různé způsoby uložení dat na serveru. Projdeme běžně používané technologie pro správu dat a také specializované technologie vyvinuté pro zpracovaní velkých dat. Zaměříme se především na jednotlivé technologie jako na celek, ne na konkrétní implementace. Budeme zvažovat výhody a nevýhody jednotlivých technologií vzhledem k charakteru dat produkovaných systémem SATRAM a jemu podobným systémům.

Technologie, které jsme vybrali pro hlubší analýzu musely splňovat několik podmínek, aby bylo možné o těchto technologiích vůbec uvažovat. Tyto technologie musí být přizpůsobené především pro efektivní zpracování velkého množství dat a mít možnost jejich rychlé filtrace. Jelikož ukládaná data mají být veřejně přístupná a současně systém nebude dovolovat běžným uživatelům možnosti přidávání, úpravy či mazání dat přímo pomocí komunikačního rozhraní, nejsou důležité funkce jako transakční zpracování, řízení přístupu uživatelů k datům nebo různé uživatelské pohledy na data.

Vybrané technologie pro hlubší analýzu[4][5] jsou relační databáze, protože se jedná o nejčastěji používanou technologii pro pokročilou správu dat a je vyvíjená již mnoho let. Poté se podíváme na technologii objektových databází jako na zajímavou alternativu k relačním databázím. Další technologií je Column base store, který byl vyvinutý speciálně pro účely data warehousingu a pro big data. Následovat bude technologie grafových databází, která se využívá převážně pro účely vytváření sociálních sítí. Nakonec se podíváme na systém ROOT, který obsahuje hierarchickou objektově orientovanou databázi.

3.3.1 Relační databáze

Relační databáze[6] jsou jednou z nejstarších a nejhojněji využívaných forem databázových systémů. Tyto systémy jsou v praxi nejrozšířenějšími díky své univerzálnosti, léty prověřené spolehlivosti a faktu, že mnohé aplikace, kde je nutné vyžívat nějakou formu správy dat, jsou relační databáze ideální volbou.

Relační databáze jsou založené na relačním modelu dat, kde jsou data organizována v relacích , které si můžeme představit jako tabulky. Každá z relací v relačním modelu dat je definovaná svým vlastním schématem. Schéma relace se skládá z názvu relace, atributů relace, které si můžeme představit jako sloupce a tabulky, a z domén jednotlivých atributů, které předepisují, jaký typ

dat je možné uložit jako daný atribut. Prvky relace jsou reprezentovány jako n-tice, které si můžeme představit jako řádky tabulky, kde každá n-tice musí splňovat formát předepsaný schématem relace.

V relační databázi se na jednotlivé relace díváme jako na matematické množiny dat, z čeho vyplývá, že každý prvek relace musí být v rámci relace jedinečný. Z tohoto důvodu je v rámci relačních databází zavedený pojem primárních klíčů. Primární klíč tvoří podmnožina atributů relace taková, že pro všechny prvky relace je primární klíč jedinečný, všechny prvky relace jsou přímo nebo tranzitivně identifikačně závislé na primárním klíči a současně neexistuje nevlastní podmnožina atributů primárního klíče taková, že tato podmnožina by mohla být primárním klíčem.

Pro modelování propojení dat mezi jednotlivými relacemi relačního modelu databáze se využívají cizí klíče. Cizí klíč je taková podmnožina atributů relace, která umožňuje jednoznačně identifikovat konkrétní prvek v jiné relaci. Typicky se jedná o jeden z možných primárních klíčů jiné relace, který je součástí dané relace. Tento konstrukt dovoluje zmenšovaní relací na množiny menších navzájem propojených relací, díky čemuž je možné zmenšit množství dat potřebných k uložení všech námi požadovaných informací, nebo zabránit aktualizačním anomáliím, které se vyskytují, když jsou v databázi redundantní data a při jejich změně se aktualizují pouze některé z výskytů těchto redundantních dat.

Pro dotazovaní nad relační databází se využívá převážně dotazovací jazyk SQL. Tento jazyk má vetší vyjadřovací schopnost než relační algebra, která je matematickým nástrojem pro dotazování nad relacemi. Jedná se o deklarativní jazyk. To znamená, že při psaní dotazu deklarujeme jaký výsledek požadujeme, ne jak se má tohoto výsledku dosáhnout.

Z důvodu zajištění odolnosti relačních databází před ztrátou či poškozením dat a poskytnutí rychlého a asynchronního přístupu k datům, je v rámci relačních databází zavedené transakční zpracováni příkazů. Transakce je logická jednotka práce, která má svůj začátek a konec, která se provede buďto celá, nebo se při chybě některé z částí transakce celá transakce neprovede. Transakce musí splňovat takzvané ACID vlastnosti.

Atomicity transakce proběhne buď celá nebo vůbec

Consistency data jsou před a po ukončení transakce v konzistentním stavu

Independence efekty transakce nejsou viditelné jiným transakcím, dokud transakce neproběhne celá

Durability efekty úspěšně provedené transakce jsou trvale uloženy

Transakce tedy zajišťují možnosti sdíleného přístupu k datům bez obavy o konzistenci dat a pomocí transakčního žurnálu, ve kterém se uchovávají změnové vektory databáze, také zotavení po chybě.

3.3.1.1 Výhody relačních databází

Relační databáze poskytují možnost jednoduše psát složité dotazy pomocí jazyka SQL, který má velmi silnou vyjadřovací schopnost a současně je jednoduchý na pochopení. Tento typ databáze taktéž poskytuje značnou flexibilitu při návrhu databázového modelu, dovolují přesně modelovat vztahy mezi daty ze skutečného světa a poskytují mnoho nástrojů a pravidel, díky kterým je možné zabezpečit, že datový model neobsahuje chyby, které by mohly vést k nekonzistenci dat. Tento typ databáze taktéž umožňuje účinnou správu přístupu k datům, práv uživatelů, a celkové bezpečnosti dat. Transakční zpracování taktéž umožňuje efektivní práci databáze s dotazy, které mění obsah databáze tj. přidávání, změna a mazání dat. Z těchto důvodů jsou tyto databáze vhodné pro použití například v bankovnictví a jiných oblastech, kde je důležité zabezpečení dat, neustála konzistence dotazů, propracované řízení přístupu a požaduje se zvládání velkého množství paralelně přijímaných krátkých dotazů, které mění data tj. takzvaný On-line Transaction Processing.

3.3.1.2 Nevýhody relačních databází

Relační databáze neumožňují flexibilní přizpůsobení změně definice dat. Dále tyto databáze mají značné nároky na místo pro uložení dat. Kvůli fragmentaci dat do mnoha relací, jsou dlouhé a komplexní dotazy na data pomalé, kvůli nutnosti zpětné kompozice dat. Taktéž dotazy, které ovlivňují velké množství dat můžou blokovat přístup k datům pro ostatní uživatele. Kvůli tomu, že dotazovací jazyk SQL je deklarativním jazykem, je nutné aby databáze před vyhodnocením dotazu sestavila prováděcí plán daného dotazu, pokud jsou dotazy velice komplexní tyto prováděcí plány nemusí být optimální a pro jejich optimalizaci je potřebné hluboké pochopení vnitřního fungování dané implementace databáze, a proto je potřebný při těchto optimalizacích zásah experta na danou implementaci databáze.

3.3.2 Objektově orientované databáze

Objektově orientované databáze[7] vznikly s nástupem objektově orientovaných programovacích jazyků. Tento typ databází přináší objektově orientovaný pohled na data, který je stejný jako u objektově orientovaných programovacích jazyků, což umožňuje snazší implementaci těchto databázi do aplikací napsaných pomocí objektově orientovaných programovacích jazyků.

Objektově orientované databáze jsou založené na přímém ukládání struktury objektů. To znamená, že data ukládána ve formě objektů, kde každý objekt je instancí třídy, která předepisuje formát dat uloženích ve objetu ve formě atributů, a to jak a jaké operace je objekt schopný vykonávat. Objekt je tedy instancí třídy kde atributy představují stav objektu a metody představují komunikační rozhraní s daným objektem. Implementovány jsou také

další rysi objektově orientovaného paradigma jako zapouzdření, nebo dědičnost. Zapouzdření je koncept, kdy třída nedovoluje přímí přístup k atributům a některým metodám, čím chrání data před před nepřípustnými změnami. Dědičnost je koncept, kdy třída, která je potomkem jiné třídy zdědí všechny vlastnosti své rodičovské třídy, které může redefinovat, nebo může přidat další vlastnosti. To umožňuje postupnou specializaci tříd.

Objektově orientované databáze mohou implementovat možnost transparentní persistence dat, kde k datům uložením persistentně v databázi je možné přistupovat přímo, volat jejich metody a tím měnit jejich vnitřní stav. Pomocí této funkce je možné objekty v databázi přecházet a ovlivňovat obdobným způsobem, jako kdyby to byli objekty uložené v lokální paměti programu. Objektově orientované databáze mohou také implementovat OQL dotazovací jazyk, které vychází z SQL. OQL je tedy SQL přizpůsobené pro práci s objekty, stále se však jedná o deklarativní dotazovací jazyk.

3.3.2.1 Výhody objektově orientovaných databází

Objektově orientované databáze umožňují přímé ukládání objektů, to umožňuje využít datový model aplikace také jako datový model pro persistentní data. Jelikož je možné s daty uloženými v těchto databázích pracovat stejně jako kdyby byly tyto data uložené v lokáni paměti, je integrace těchto databází velice jednoduchá. Přímé ukládání objektů také umožňuje efektivní ukládání a načítaní komplexních objektů, protože není potřebné složité zpracování těchto objektů. Dědičnost umožňuje těmto databázím taktéž za běhu přizpůsobovat datový model. Objektově orientované databáze jsou vhodné pro použití, kde je potřeba vysoký výkon při práci s komplexními daty.

3.3.2.2 Nevýhody objektově orientovaných databází

Objektově orientované databáze mají značné nároky na místo pro uložení dat, protože v nich uložená data mají velkou míru redundance. Kvůli zapouzdření je velice složité vytvářet nad daty indexy. Z těchto důvodů nejsou tyto databáze vhodné pro uchovávání velkého množství jednoduchých dat. Pro práci s těmito databázemi je téměř nutností pracovat s objektově orientovaným programovacím jazykem.

3.3.3 Column base store

Column base store[8][9] vznikly, aby vyřešili problém s ukládáním a zpracovaným velkého množství dat, za účelem analýzy těchto dat. Data v těchto databázích jsou organizována do sloupců, kde každý sloupec má svůj název a typ dat, které jsou v něm uloženy. K datům v sloupcích se dá přistupovat pomocí klíče, kde každý záznam v sloupci má svůj vlastní klíč. Tyto sloupce můžou být dále organizované do rodin sloupců. Rodiny sloupců můžou obsahovat jakékoliv množství sloupců. Záznamy v rámci jedné rodiny sloupců

mezi sloupci sdílejí přístupoví klíč a tím vzniká konstrukce podobná tabulce, s tím rozdílem, že není vyžadováno, aby všechny sloupce v rodině sloupců obsahovali záznam pro všechny klíče v dané rodině sloupců. To umožňuje jednoduché přidávaní dalších sloupců do rodiny sloupců, bez nutnosti zásahu do již uložených dat.

3.3.3.1 Výhody column base store

Ukládáni dat po sloupcích přináší těmto databázím velké množství výhod v oblasti zpracování velkého množství dat. Hlavní výhodou je rychlá agregace nebo filtrování dat z jednotlivých sloupců, protože databázový systém nemusí načítat všechna data která náleží danému klíči, ale pouze data ze sloupců, které nás zajímají. Další výhodou je snadná a účinná komprese dat v databázi, protože je možné data komprimovat po sloupcích, kde je velká pravděpodobnost výskytu podobných dat, které pak lze dobře komprimovat. Nakonec organizace do sloupců umožňuje snadné přidávání a odebírání sloupců bez ovlivnění dat z jiných sloupců.

3.3.3.2 Nevýhody column base store

Kvůli absenci koncepty cizích klíčů mezi jednotlivými rodinami sloupců není v těchto databázích možné explicitně vytvářet vztahy mezi jednotlivými rodinami sloupců. Z toho důvodu není v těchto databázích možné explicitně modelovat složité datové závislosti a jsou tak vhodné spíše pro jednoduší datové modely. Tyto databáze taktéž poskytují slabý výkon při práci s daty po řádkách tj. když přistupujeme k datům ze všech sloupců z dané rodiny sloupců, které náleží danému klíči.

3.3.4 Grafové databáze

Grafové databáze[10] vznikly pro ukládání dat, kde jsou podstatné především vztahy mezi entitami, které ukládáme. Grafové databáze jsou založené na prvcích z teorie grafů, kde jednotlivé entity odpovídají vrcholům grafu a vztahy mezi entitami odpovídají hranám grafu. Každá entita uložená v grafové databázi má jedinečný identifikátor, dále má svou sadu atributů, které určují vlastnosti dané entity, typ entity a nakonec příslušnou kolekci hran. Kolekce hran obsahují hrany, kde hrana může být orientovaná nebo neorientovaná, může mít svůj název nebo typ a může také obsahovat vlastní atributy, které dále specifikují vztah mezi entitami, které tato hrana propojuje. Kolekce hran taktéž umožňují propojení dvou entit více než jednou hranou, což umožňuje vytváření multigrafů. Protože jednotlivé entity a vztahy mezi nimi jsou implementačně nezávislé, datový model těchto databází je velmi flexibilní. Kdykoliv můžeme přidat nebo odebrat druh entity, druh hrany nebo atribut entity a atribut hrany.

Grafové databáze mají vztahy mezi jednotlivými entitami spolu s parametry těchto vztahů uložené, není proto nutné je při každém dotazu znova propočítávat. To umožňuje rychlé a efektivní zpracovávat dotazů, kde je nutné propojit data z velkého množství entit.

Pro ukládání dat grafových databází můžeme použít relační nebo objektově orientované databáze, tím však nevyužijeme plný potenciál grafových databází. Proto se často vyžívají datové struktury používané pro ukládání grafů vyvinuté pro teorii grafů. Takovými datovými strukturami jsou například matice sousedů, matice incidence nebo kolekce sousedů.

3.3.4.1 Výhody grafových databází

Grafové databáze se hodí pro nasazení kde, je potřebné rychlé zpracování dotazů, kde dochází k propojení velkého množství dat z různých entit. Příkladem můžou být sociální sítě, telekomunikační sítě nebo systémy pro vyhledávání dopravního spojení. Tyto databáze jsou také velice flexibilní při návrhu a změně struktury databáze, protože vlastně žádnou pevně danou strukturu nemají.

3.3.4.2 Nevýhody grafových databází

Grafové databáze nedokáží efektivně zpracovávat dotazy, které zasahují celou databázi, nebo části databáze jako celek. Příkladem může být změna jednoho atributu a všech entit určitého typu.

3.3.5 ROOT

ROOT[2][11] je systém pro zpracování dat vyvinutý v CERN za účelem ukládání, zpracování a analýzy dat produkovaných experimenty v oblasti částicové fyziky. ROOT je tedy optimalizovaný pro ukládání a zpracování velkého množství dat v dávkách.

Pro ukládání dat se v systému ROOT používají takzvané rootfile soubory. Tyto soubory obsahují komprimovaná binární data a také popis datové struktury dat v něm uložených, aby bylo možné data přečíst i když není k dispozici program, který je vytvořil, či dokumentace k daným datům. Data uložená v těchto souborech mohou být organizována jako n-tice, nebo jako stromy.

N-tice jsou v rootfile uložené v seznamech a vytvářejí tak strukturu podobnou tabulce. Každý seznam má své jméno a definici formátu n-tic, které jsou v něm uložené. N-tice se skládá z fixního počtu prvků kde každý prvek má svůj název a typ, kde typ musí být primitivního datového typu nebo pole, kde pole má buď fixní délku, nebo délku vázanou na jiný prvek. Pokud si tedy celý seznam n-tic představíme jako tabulku, tak prvky n-tice představují sloupce tabulky a samotné n-tice představují řádky tabulky. ROOT umožňuje data v těchto tabulkách ukládat po řádcích nebo po sloupcích. Výchozí nastavení ukládá tabulku po sloupcích, protože se vychází z předpokladu, že se bude často krát přistupovat pouze k malé podmnožině sloupců. Při ukládání

po sloupcích se tak pro každý sloupec vytvoří samostatný buffer a ROOT tak může přistupovat k jednotlivým sloupcům bez toho, aby musel číst data ze sloupců, které právě nepotřebuje. To umožňuje optimalizovat množství dat načítaných z disku a tím zvýšit výkon při dotazování se na data. Ukládání po sloupcích také umožňuje lépe komprimovat uložená data, protože se komprimuje každý buffer samostatně, přitom v každém bufferu se nachází pouze data stejného typu, díky čemu je možné je lépe komprimovat.

Stromy v rootfile poskytují stromovou strukturu pro organizaci a ukládání dat. Formát uložených dat je předepsán touto datovou strukturou, kde každý strom má svůj název a dále obsahuje větve. Každá větev má svůj název a svůj typ, kde typ může být další strom nebo objekt primitivního datového typu nebo pole, kde pole má buď fixní délku nebo délku vázanou na jiný prvek. Každá větev stromu je uložená ve vlastním bufferu, přičemž když větev obsahuje objekt nebo strom, je možné určit jestli celý objekt nebo strom uložit do jednoho bufferu, nebo udělat buffer pro každý atribut a každou větev. To umožňuje programátorovi nastavit rozdělení do bufferů, přesně podle potřeb daného systému. Obdobně jako u n-tic i u stromů rozdělení do bufferů umožňuje lepší komprimaci dat a vyšší výkon při dotazovaní se na data. Každý záznam v této datové struktuře pak kopíruje definovanou stromovou strukturu a v rootfile je uložený seznam těchto záznamů, kde seznam je rozdělen do datovou strukturou definovaných bufferů.

Díky flexibilitě při návrhu se v rootfile používají stromy častěji než n-tice. Aby ROOT nemusel pracovat s příliš velkými soubory, je maximální velkost jednoho rootfile daná na 2GB. Když je při ukládání dat tato velikost překročena, vytvoří se nový rootfile se stejnou datovou strukturou a pokračuje se dále v ukládání do tohoto souboru. K propojení stromů stejnou strukturou z jiných souborů se pak využívá zřetězení stromů a ty se pak v programu jeví jako jeden strom, i když jsou uloženy ve vícero souborech. Pokud by bylo potřeba strom rozšířit o nějaké větve, je možné strom rozšířit přímo, to však může vést ke ztrátě dat, pokud bychom při tom překročili maximální velikost rootfile. Z toho důvodu jsou v ROOT zavedené také spřátelené stromy, díky čemu můžeme simulovat rozšíření stromu pomocí nového spřáteleného stromu.

3.3.5.1 Výhody ROOT

Možnosti přesného nastavení datových bufferů dovolují ROOT optimalizovat rychlost přístupu k datům a také velmi dobrou komprimaci dat. Vlastnosti rootfile jako maximální velikost nebo to, že obsahují popis struktur v nich uložených dat, také dovolují jednoduchou přenositelnost dat mezi jednotlivými systémy. Možnost vytvářet spřátelené stromy taktéž umožňuje jednoduché přizpůsobení datového modelu. ROOT je tedy vhodný pro nasazení, kde se pracuje s velkým množstvím dat a je kladen důraz na analýzu dat a dotazy, které dávkově zpracovávají velké množství záznamů.

3.3.5.2 Nevýhody ROOT

Absence vestavěného indexování dat a pokročilého řízení přístupu k datům znamená, že ROOT není vhodný pro nasazení v situacích kde je potřebné pokročilého řízení přístupu, nebo kde jsou dotazy na data krátké.

3.4 Cluster analýza

Data získaná z přístroje SATRAM spolu s navigačními a stavovými údaji družice jsou po odeslání na Zem v surovém stavu. Pro další využití je tedy potřebné tyto data dále zpracovat, aby bylo možné tyto data použít k další analýze. Při zpracování těchto dat se využívá cluster analýza. Během cluster analýzy dochází k rozpoznávání vzorů[12] na jednotlivých snímcích. Při rozpoznávání vzorů se na snímcích vyhledávají clustery, to jsou skupiny pixelů, které spolu sousedí a současně zaznamenali signál při expozici snímku. Clustery nacházející se na snímku odpovídají částicím, které interagovali s detektorem během expozice snímku. Tvar, velkost a síla signálu zanechaného částicí je ovlivněn zejména typem, energií a směrem částice vzhledem k rovině senzoru. Příklad snímku můžete vidět na obrázku 3.1. Při vyhledávání clusterů nalezne pixel, ve kterém je hodnota signálu vyšší než stanovená hranice a následně ve všech směrech vyhledává, jestli daný cluster zasahuje taky do sousedních pixelů. Pomocí tohoto procesu se vytvoří binární maska, kde 0 označuje místo kde nejsou žádné clustery a 1 označuje místo kde se clustery nacházejí. Následně se v této masce identifikují clustery které se překrývají, a tyto clusteru jsou vyřazené z další analýzy. Maska tak poté obsahuje pouze nepřekrývající se clustery, kde každá skupina sousedících jedniček označuje jeden cluster. Následně dochází k analýze tvaru jednotlivých clusterů a celkové energie, která v nich byla zaznamenána a její rozložení. Na základe těchto parametrů jsou vypočteny vlastnosti jednotlivých clusterů a pak jsou clustery rozděleny do šesti předdefinovaných skupin. Tyto skupiny jsou: Clustery jsou pak rozděleny do těchto skupin:

dot foton < 20 keV

small blob foton $\sim$ 4 50 keV

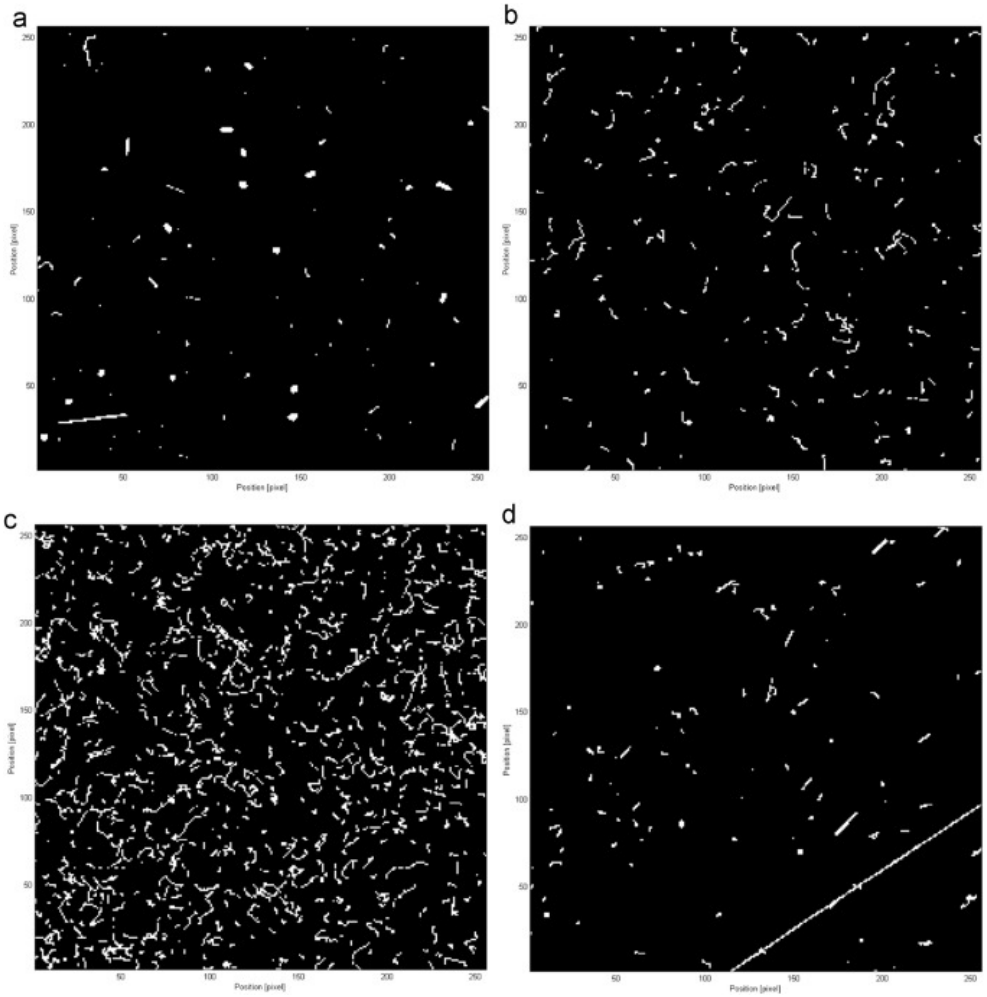
heavy blob alfa částice, těžké ionty, pomalé neutrony

straight thin track minimálně ionizující částice

curly track elektrony, elektrony produkované fotony s > 50 keV

heavy track protony > 1 MeV, neutrony > 1 MeV

K rozdělení dochází na základě geometrických vlastností jednotlivých clusterů jako konvexní obálka, plocha, objem, počet vnitřních a hraničních pixelů, délka



Obrázek 3.1: Příklad snímků z detektoru radiace Timepix. Jsou to data zachycená přístrojem SATRAM. Snímky byli získany (a) 11.11.2013 nad polostrovem Korea, a 23.4.2014 nad (b) jižním Indickým oceánem, (c) jižním pólem a (d) Jihoatlantickou anomálií (SAA)

ohraničení clusteru, maximální počet pixelů v přímé linii, průměr clusteru a podobně. Tyto vlastnosti jsou pak použity k výpočtu parametrů každého clusteru na základě kterých je cluster zařazen do některé z kategorií.

3.4.1 Pixelman

Pixelman[3] je multiplatformní software pro získávaní a zpracování dat z detektorů Medipix2, Timepix a Medipix3. Jedná se o balíček pluginů s GUI napsaným v programovacím jazyku Java, pro který je možné vytvářet pluginy pomocí programovacích jazyků Java nebo C++.

Tento software má flexibilní modulární architekturu rozdělenou do těchto základních skupin pluginů:

Hardwarové knihovny zajišťují komunikaci s detektory Medipix, Timepix a jinými zařízeními s podporou různých rozhraní jako USB, MUROS, FITPix, Flatpanel a podobně. Každá hardwarová knihovna implementuje rozhraní pro komunikaci s *Ovládací knihovnou*, čímž je zajištěn přístup zařízením pomocí implementovaných rozhraní nezávisle na jejich implementaci.

Ovládací knihovny pro Medipix obsluhuje všechny připojené Medipix a Timepix zařízení pomocí *Hardwarových knihoven*. Poskytuje tak synchronizovaný přístup zařízením a řídí získávaní dat ze zařízení, konfigurování zařízení a obsluhuje data buffer.

Pixelman Manager obsluhuje všechny pluginy napsané v C++. Těmto pluginům poskytuje přístup k *Ovládací knihovně* a řídí synchronizaci a komunikaci mezi pluginy. *Pixelman Manager* také spravuje registrované funkce, události a řetězce filtrů.

C++ pluginy můžou kontrolovat specifický hardware a můžou zpracovávat data nebo můžou řídit celý experiment. Pluginy mají přístup k funkcím *Ovládací knihovny* a k ostatním pluginům. Pluginy můžou registrovat události nebo se můžou registrovat jako příjemce událostí.

Java Wraper načítá C++ jádro software Pixelman a vytváří rozhraní mezi C++ jádrem a částmi napsanými v jazyce Java. Tento nástroj tak umožňuje Java pluginům přístup ke stejným funkcím jako mají C++ pluginy.

Java manager podobně jako *Pixelman Manager* obsluhuje všechny pluginy napsané v jazyce Java a těmto pluginům poskytuje přístup k *Ovládací knihovně*, řídí synchronizaci a komunikaci mezi pluginy a spravuje registrované funkce, události a řetězce filtrů.

Java pluginy podobně jako *C++ pluginy* můžou kontrolovat specifický hardware a můžou zpracovávat data nebo můžou řídit celý experiment. Pluginy mají přístup k funkcím *Ovládací knihovny* a k ostatním pluginům. Pluginy můžou registrovat události nebo se můžou registrovat jako příjemce událostí.

Mimo jiných byli pro tento software vytvořené také tyto pluginy:

- plugin pro provádění cluster analýzy. Tento plugin byl napsán v jazyku C++
- plugin pro online vizualizaci dat z detektorů
- plugin pro kontrolu detektorů a sběr dat

Tyto pluginy jsou v současné době využívané pro práci s přístrojem SATRAM, vizualizaci a zpracování dat z tohoto přístroje.

3.4.1.1 Software pro cluster analýzu a korelaci s navigačními údaji

Software pro cluster analýzu a korelaci s navigačními údaji, je software aktuálně vyvíjený na ÚTEF ČVUT. Autoři tohoto softwaru jsou *Benedikt Bergmann, MSc.* a *Stefan Gohl, MSc.*. Tento software vzniká přepisem pluginu pro software Pixelman[12], který slouží pro cluster analýzu. Autoři tohoto pluginu jsou *Ing. Jan Jakůbek, Ph.D.*, *Ing. Tomáš Holý* a *Ing. Daniel Tureček*. V tomto softwaru se využívají algoritmy napsané v daném pluginu, k nim jsou přidané části pro načítání a ukládání dat. Pro ukládání dat se využívají *rootfile* popsané v podsekci 3.3.5. Důvodem vytváření tohoto software je zjednodušení automatizace zpracování dat z detektorů Timepix, které jsou hojně využívány na ÚTEF ČVUT, zejména v rámci SATRAM, nebo při experimentech na urychlovači částic LHC v CERN.

3.4.2 Výstupní data

V této podsekci si popíšeme výstupní data ze *Software pro cluster analýzu a korelaci s navigačními údaji*, protože tento nástroj bude používán pro zpracovaní dat z přístroje SATRAM a tedy tyto data budou spravována námi vytvořeným systémem. Výstupní data jsou ukládána do *rootfile*. Rootfile jsou datové soubory systému ROOT vyvinutého v CERN za účelem ukládání, zpracování a analýzy dat produkovaných experimenty v oblasti částicové fyziky. Data jsou v těchto výstupních souborech organizovány v rámci takzvaných stromů, kde je využita stromová struktura pro organizování struktury dat. Bližší popis sytému ROOT a souborů *rootfile* se nachází v podsekci 3.3.5.

Výstupní data v *rootfile* jsou organizována do dvou stromů a to dscData a clusterFile.

Strom dscData obsahuje zejména navigační a stavová data. Tyto data popisují podmínky za nichž vznikl daný snímek. Jeden záznam obsahuje data o jednom snímku. Tento strom obsahuje následující atributy:

Start_time je čas vytvoření snímku. Protože systém SATRAM je pouze jeden a obsahuje jenom jeden detektor, je tento čas jedinečný pro každý snímek a je tak možné ho použít jako jednoznačný identifikátor snímků. Pro použití s více detektory by bylo nutné přidat atribut ChipboardID, který by identifikoval detektor a jednoznačným identifikátorem snímku by byla kombinace atributů ChipboardID a Start_time. Jedná se o unix timestamp

Acq_time je délka expozice snímku v sekundách

analog_voltage1 je analogové napětí na detektoru Timepix úrovně 1. Jedná se o dvouprvkové pole kde první prvek je maximální a druhý minimální hodnota během expozice snímku. Hodnoty jsou ve Voltech

analog_voltage2 je analogové napětí na detektoru Timepix úrovně 2. Jedná se o dvouprvkové pole kde první prvek je maximální a druhý minimální hodnota během expozice snímku. Hodnoty jsou ve Voltech

BiasVoltage_min je minimální úroveň předpětí detektoru Timepix během expozice snímku. Hodnoty jsou ve Voltech

BiasVoltage_max je maximální úroveň předpětí detektoru Timepix během expozice snímku. Hodnoty jsou ve Voltech

DAC_voltage je úroveň napětí na DAC v Timepix. Jedná se o dvouprvkové pole kde první prvek je maximální a druhý minimální hodnota během expozice snímku. Hodnoty jsou ve Voltech

satelite_position je pozice družice při začátku expozice. Jedná se o tříprvkové pole kde první prvek je vzdálenost družice od středu Země, druhý je zeměpisná délka a třetí je zeměpisná šířka

satelite_attitude je rotace družice vzhledem k Zemi a Slunci při začátku expozice. Jedná se kvaternion tj. uspořádanou čtveřici čísel, která slouží pro zapisování rotace v 3D prostoru

B_field_strength intenzita magnetického pole

proba_V_powerStatus - informace o napájení z družice

proba_V_thermalStatus - informace o teplotě z družice

PSU_failures je počet chyb zdroje energie během expozice snímku

PSU_status signalizuje je-li zdroj energie funkční

internal_temperature_min je úroveň minimální vnitřní teploty v přístroji SATRAM během expozice snímku v °C

internal_temperature_max je úroveň maximální vnitřní teploty v přístroji SATRAM během expozice snímku v °C

Jeden záznam, v tomto stromě odpovídá jednomu snímku z přístroje SATRAM. Jedná se zejména o navigační data z družice a stavová data z přístroje SATRAM a družice.

Strom clusterFile obsahuje zejména data, která jsou výstupem z cluster analýzy. Navíc obsahuje data potřebná ke zpětné rekonstrukci snímku a data, pomocí kterých můžeme propojit tyto data s navigačními a stavovými daty. Jeden záznam obsahuje data o jednom clusteru. Tento strom obsahuje následující atributy:

CounterValue je jednoznačný identifikátor clusteru v rámci jednoho *rootfile*.

Start_time je čas vytvoření snímku, ve kterém se daný cluster nachází. Protože systém SATRAM je pouze jeden a obsahuje jenom jeden detektor, je tento čas jedinečný pro každý snímek a je tak možné tento atribut použít pro přiřazení clusterů k jednotlivým snímkům. Pro použití s více detektory by bylo nutné přidat atribut ChipboardID, který by identifikoval detektor a pro přiřazení clusterů k jednotlivým snímkům by byla použita kombinace atributů ChipboardID a Start_time. Jedná se o unix timestamp

Acq_time je délka expozice snímku v sekundách

clstrSize je počet pixelů, které tvoří cluster

cstrLength je průměr clusteru v pixelech

clstrRoudness určuje nakolik je daný cluster kulatý.

clstrRegion index regionu do kterého cluster patří

clstrLinearity určuje nakolik je daný cluster lineární.

clstrLET určuje lineární přenos energie tj. veličina popisující hustotu předávání energie prostředí ionizujícím zářením.

maxClstrHeight maximální výška clusteru

minClstrHeight minimální výška clusteru

clstrMeanX je x-ová souřadnice těžiště clusteru na snímku

clstrMeanY je y-ová souřadnice těžiště clusteru na snímku

clstrVolume součet hodnot počítadel všech clusterů

clstrType je vypočtený typ clusteru popsaný v sekci 3.4

PixX je pole x-ových souřadnic všech pixelů tvořících daný cluster. velikost pole je rovna atributu clstrSize

PixY je pole y-ových souřadnic všech pixelů tvořících daný cluster. velikost pole je rovna atributu clstrSize

Polar určuje úhel dopadu částice vzhledem k rovině detektoru. Hodnota je mezi 0° až 90°

Azimut určuje úhel dopadu částice vzhledem na y-ovou osu detektoru. Hodnota je mezi -90° až 90°

Jeden záznam, v tomto stromě odpovídá jednomu clusteru. Jedná se zejména o data vypočtená během cluster analýzy.

Tyto data jsou tedy takzvaně cluster orientovaná data, kde základní jednotkou, se kterou pracujeme je cluster. Pomocí dat z atributů PixX a PixY je možné zpětně zrekonstruovat původní data ze snímků po prvotním zpracování tj. odstranění signálů pod zvolenou hranicí a odstranění překrývajících se clusterů. Jeden *rootfile* obsahuje data za jeden den.

3.5 GUI

Z analýzy požadavků na systém vyplývá, že ke konfiguraci systému bude docházet pouze při nasazení systému a při přidávání funkčních celků do systému. Protože k těmto událostem nebude docházet často, pro tento typ konfigurace postačují strukturované konfigurační soubory.

Z analýzy požadavků dále vyplývá, že uživatelé systému nebudou do systému zasahovat, pouze ho využívat jako zdroj dat pomocí komunikačního API. K samotné hlubší analýze a prezentaci dat budou sloužit až systémy uživatelů.

Pro jednoduchou prezentaci dat může sloužit webové uživatelské rozhraní.

3.5.1 Webové rozhraní pro prezentaci dat

Prezentace dat je nejjednodušší způsob využití navrhovaného systému pro správu dat z přístroje SATRAM. Uživatelům je v takovémto případě nutné poskytnou přístup k vizualizovaným datům v podobě obrázků jednotlivých snímků po analýze s možností jednoduchého vyhledávaní a filtrování snímků, pohybu mezi vybranými snímky, filtrovaní clusterů zobrazených na jednotlivých snímcích a zobrazení detailů o jednotlivých snímcích.

Uživatelské rozhraní by také mělo poskytovat možnost stažení dat, která jsou v daném momentě prezentována, zdrojových dat, ze kterých byla prezentována data vyfiltrována a údajů potřebných pro získaní prezentovaných dat přímo z komunikačního API. Důvodem vytvoření přístupu k těmto technickým datům uživatelům je ulehčení učení se přístupu k API a tím zvýšení možnosti, že uživatelé vytvoří aplikace využívající navrhovaný systém jako zdroj dat, a tím zvýší vědecký přínos těchto dat nebo více popularizují vědu.

Případy užití[13] tohoto systému tedy budou:

- Výběr časového ohraničení požadovaných snímků
- Výběr parametrů požadovaných snímků
- Prohlížení časové osy
- Výběr konkrétního snímku z časové osy
- Prohlížení snímků

- Prohlížení dat o snímku
- Prohlížení mapy s vyznačenou polohou pořízení snímku
- Přepínaní na předchozí a následující snímek
- Výběr clusteru na snímku
- Prohlížení údajů o clusteru
- Výběr parametrů pro filtrování clusterů
- Stažení
  - dat pro vytvoření časové osy
  - dat o zobrazeném snímku
  - dat o vybraném clusteru
  - zdrojových dat
- Stažení údajů potřebných pro získaní prezentovaných dat přímo z komunikačního API
  - pro časovou osu
  - pro zobrazení snímku
  - pro zobrazení dat o konkrétním clusteru
- Přechod na stránky ÚTEFk ČVUT o projektu SATRAM

GUI bude navržené tak aby bylo schopno efektivně obsloužit tyto případy užití.

Kapitola 4

Návrh

4.1 Schéma systému

Systém bude fungovat jako zdroj dat, který bude poskytovat přístup k datům pomocí komunikačního rozhraní. Systém tak bude součástí většího celku, který bude sloužit na získávání a analyzování dat o radiačním poli v okolí Země.

4.1.1 Sběr dat

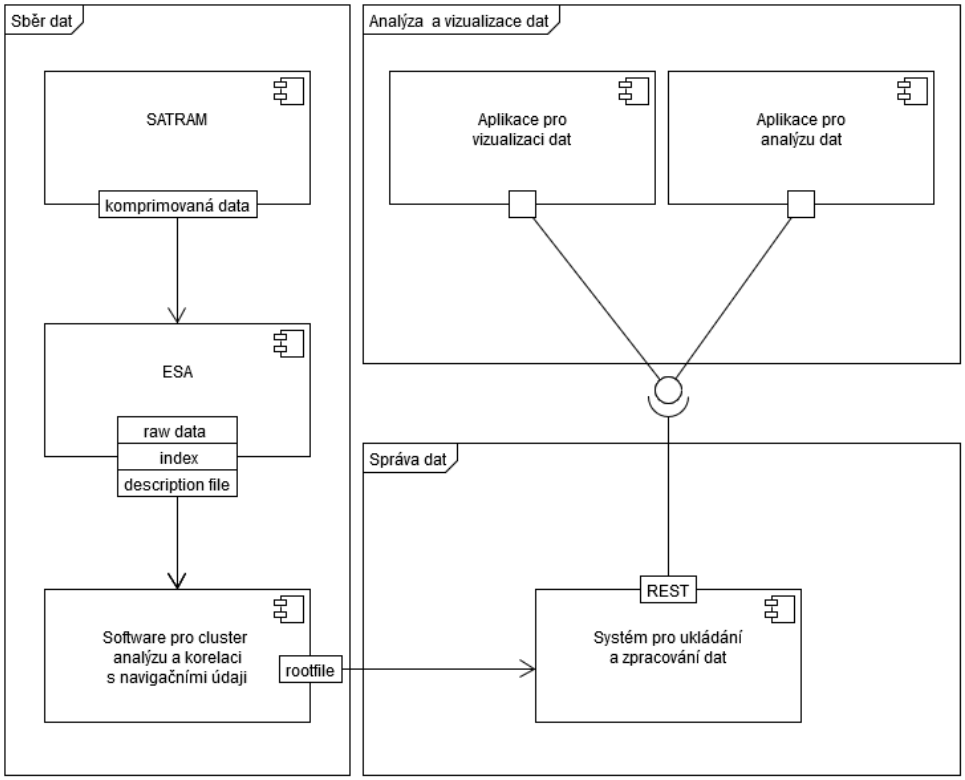
Sběr dat bude fungovat automaticky a bude probíhat jako postupná transformace dat, kde si jednotlivé komponenty budou předávat postupně transformovaná data. Při zpracování dat budou použity programy pro zpracovaní dat z přístroje SATRAM vyvinuté v rámci ÚTEF ČVUT.

Data ze přístroje SATRAM jsou na Zemi odesílána jednou 90 minut při každém oběhu družice. Po přijetí dat z družice v ESA budou data dekomprimována, rozdělená do třech souborů popsaných v podsekci 3.1.1 a odeslána na ÚTEF ČVUT. Tam jsou data rozdělena na jednodenní bloky aby byli připraveny na další zpracovaní. Zpracování dat bude probíhat pomocí *Software pro cluster analýzu a korelaci s navigačními údaji* popsaného v podsekci 3.4.1.1. Výsledkem analýzy bude *rootfile* obsahující všechna data za jeden den měření. Tyto soubory pak budou zpracovány námi navrhovaným systémem.

Soubory s daty budou nakonec organizovány v složkách na serveru po měsících. Data pro každý den tak budou mít své přesně definované místo.

4.1.2 Blokové schéma celkového systému

Námi navrhovaný systém bude částí většího celku, který bude sloužit pro získávaní, zpracování, ukládání, analýzu a prezentaci dat. Tento systém se bude starat o všechna data od jejich vzniku. Systém bude rozdělen do funkčních bloků, kde každý blok bude mít jinou funkci. Jednotlivé bloky budou pro-



Obrázek 4.1: Blokové schéma systému jehož součástí bude systém pro ukládání a zpracování dat

pojené zejména pomocí dat, ne API. Blokové schéma celkového systému je zobrazeno na obrázku 4.1.

Data tedy budou nejdříve sesbírána a zpracována pomocí již existujících aplikací. Pak budou tyto data předána námi navrhovanému systému ve formě *rootfile* souborů. Námi navrhovaný systém bude spravovat všechny vytvořené *roofile* soubory a bude poskytovat API pro přístup k těmto datům. Toto API budou využívat aplikace pro analýzu a vizualizaci dat.

4.1.3 Vnitřní struktura systému

Smyslem systému je poskytnou rychlý přístup k datům s možností tyto data filtrovat. Nejpodstatnější využití je pro analýzu dat. Při analýze dat se bude přistupovat zejména k datům z velkého časového rozsahu z kterých bude filtrována malá podmnožina údajů. Vnitřní architektura systému musí být optimalizována pro takovéto využití. Systém se tedy bude skládat ze dvou vrstev a to komunikační vrstvy a datové vrstvy.

Komunikační vrstva bude mít za úlohu příjímání požadavků skrze komu-

nikační API, parsování parametrů pro výběr a filtraci dat, řízení funkčnosti datové vrstvy a odesílaní odpovědí na požadavky pomocí komunikačního API. Komunikační vrstva tedy bude řídit celý systém.

Datová vrstva se bude starat o přístup k datům, filtraci dat, přidávaní dat a vytváření těla odpovědí na požadavky.

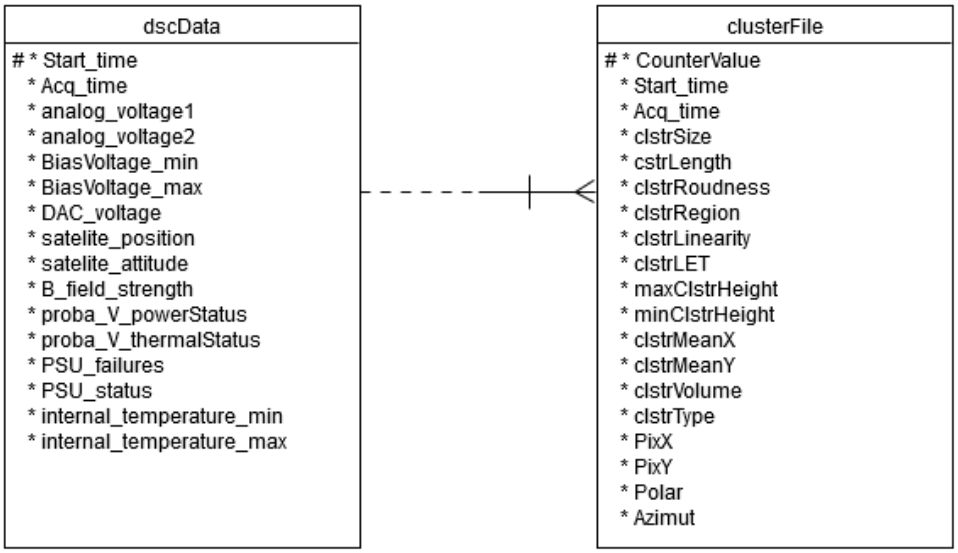
Důvodem pro použití tak jednoduché architektury je možnost optimalizace běhu systému pro rychlost na úkor přehlednosti kódu. Díky tomu že se charakter dat, se kterými se pracuje v rámci ÚTEF ČVUT, se mezi jednotlivými projekty příliš nemění, při rozšiřování systému, aby mohl pracovat pracovat s daty z jiných projektů než SATRAM, se budou moct části systému pouze zkopírovat. Těmto novým částem se pak pouze přidá jiná základní URL a provedou se v nich změny v definici dat. Části systému starající se o různé projekty tak budou navzájem zcela nezávislé.

4.2 Způsob uložení dat

Námi navrhovaný systém bude pracovat s daty z oblasti částicové fyziky, nebo s daty které svým charakterem odpovídají těmto datům datům. Protože systém ROOT, blíže podsekci 3.3.5, byl vyvinutý v CERN za účelem zpracování takových dat, rozhodli jsme se pro použití tohoto systému. Výhodou použití tohoto systému je také to, že bude možné použít přímo data produkovaná pomocí *Software pro cluster analýzu a korelaci s navigačními údaji* a nebude tak potřebné tyto data uchovávat zvlášť. Tím se také zjednoduší proces přidávání nových dat a také se zabrání zbytečné duplikaci dat. Systém tak bude pro uchovávání dat využívat *rootfile* soubory, kde jeden soubor bude obsahovat data pro jeden den.

4.2.1 Struktura dat

Data budou v *rootfile* souborech rozdělena do dvou stromů a to *dscData* a *clusterFile*. Data v stromě *dscData* budou data týkající se jednoho snímku jako celku. Data v stromě *clusterFile* budou data týkající se jednoho clusteru. Data ve stromě *clusterFile* tak budou závislá na datech ze stromu *dscData*, protože cluster musí patřit do právě jednoho snímku. Protože můžou být snímky, které neobsahují žádný cluster, nemusí být k datům o konkrétním snímku přiřazená žádná data o clusteru. Data tedy budou mít velmi jednoduchou hierarchii zobrazenou na Entitně relačním modelu 4.2. Pro materializaci vztahu mezi stromy bude použit parametr *Start_time*. Důvodem použití toho parametru je, že čas vytvoření je jedinečný pro každý snímek a současně každý cluster z daného snímku má tento parametr nastavení na shodnou hodnotu. Záznamy v stromě *clusterFile* jsou identifikačně závislé na stromě *dscData*. Důvodem neexistence univerzálního identifikátoru pro data v stromě *clusterFile* je, že při vytváření dat neví *Software pro cluster analýzu a korelaci s navigačními*



Obrázek 4.2: Entitně relační model dat

údaji nic o již existujících datech. Proto nemůže zajistit, že přidaný identifikátor by byl universální. Mohli by jsme takový identifikátor přidat dodatečně při vkládaní dat do systému, avšak přidaní tohoto parametru by jsme mohli způsobit nekompatibilitu dat s jinými systémy, proto jsme tuto možnost zavrhli. Podrobnější popis dat a jednotlivých atributů je v analýze v podsekci 3.4.2.

4.2.2 Indexace dat

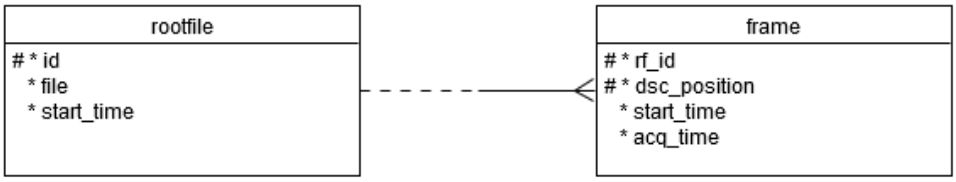
Přesto, že systém ROOT je téměř ideální pro potřeby námi navrhovaného systému, chybí tomuto systému možnost tvorby vestavěných indexů. Chybějící indexy nevadí při analýze dat, kdy se přistupuje k velkému množství dat současně. Problém nastává při vizualizaci dat, kdy se přistupuje často pouze k jednotlivým záznamům. Bez indexů by byl tento typ využití systému velice neefektivní.

Z výše uvedeného důvodu jsme se rozhodli přidat externí indexaci dat v jednotlivých *rootfile*. Indexace bude řešená za pomoci relační databáze, kde budou vytvořeny tabulky s daty pro indexy a pro přístup do *rootfile* souborů.

Pro SATRAM budou vytvořeny dvě tabulky a to tabulka *rootfile* a tabulka *frame*. Tabulka *rootfile* bude sloužit pro identifikaci souboru, ze kterého chceme data číst. Tabulka bude obsahovat tyto atributy:

id bude jedinečný identifikátor daného *rootfile* souboru

file bude relativní cesta k danému *rootfile* souboru



Obrázek 4.3: Entitně relační model indexů

start_time bude hodnota atributu *Start_time* z prvního záznamu v stromě *dscData* v daném *rootfile* souboru

Tabulka *frame* bude sloužit pro identifikaci konkrétního snímku v rámci *rootfile* souboru. Bude tedy obsahovat indexy pro strom *dscData*. Tabulka bude obsahovat tyto atributy:

rf_id bude cizí klíč do tabulky *rootfile*

dsc_position bude pořadí reprezentovaného záznamu v stromě *dscData*. Spolu s *rf_id* bude tvořit jedinečný identifikátor

start_time bude hodnota atributu *Start_time* z reprezentovaného záznamu ve stromě *dscData*. Tento parametr slouží jako index pro atribut *Start_time*.

acq_time bude hodnota atributu *Acq_time* z reprezentovaného záznamu ve stromě *dscData*. Tento parametr slouží jako index pro atribut *Acq_time*.

Indexy budou tedy tvořit hierarchii zobrazenou na Entitně relačním modelu 4.3. Využíván bude zejména index pro atribut *Start_time*. Indexace by tak měla výrazným způsobem urychlovat práci systému při vizualizaci dat, kdy je k snímkům přistupováno zejména na základě času jejich vytvoření. V případě potřeby bude možné doplnit další indexy a to buď přidáním nového sloupce do tabulky *frame* pro indexaci snímků nebo přidáním další tabulky pro indexaci dat o jednotlivých clusterech.

Kombinace systému ROOT a relační databáze přinese do námi navrhovaného systému výhody obou systému pro správu dat a efektivní zpracování velkého množství záznamů a současně rychlí přístup k jednotlivým záznamům.

4.3 Komunikační API

Pro komunikaci s aplikacemi, které chtějí přistupovat k datům, které námi navrhovaný systém spravuje bude využito REST API. Protože ostatní aplikace budou mít možnost k datům pouze přistupovat, budou v rámci REST API využity pouze GET požadavky.

4.3.1 Návrh API

Systém bude poskytovat jedenáct základních cest, pro přístup k datům. Výsledek dotazu bude možné ovlivnit pomocí JSON, který bude součástí těla požadavku. Pro použití JSON v těle požadavku jsme se rozhodli z několika důvodů. Hlavními důvody jsou snížení složitosti samotného API a také možnost jednoduše toto API přizpůsobit pro využití s daty z jiných projektů než je SATRAM.

Pro projekt SATRAM bude API obsahovat tyto základní cesty:

/satram/frame/{Start_time} v rámci této cesty se posílají údaje o jednotlivých snímcích. Jedná se o přístup k datům uložených v stromě *dscData* kde {Start_time} je jednoznačný identifikátor a je to timestamp vytvoření snímku. Požadavky na tuto cestu je nutné dále specifikovat pomocí těla dotazu. Specifikace slouží k výběru větve stromu *dscData*, ze kterých budou data vybrána. Filtrování pomocí specifikací v těle dotazu není pro tuto cestu dostupné.

Informace budou v odpovědi odeslány ve formě JSON, který bude tvořen objektem *frames*, který je polem objektů, které reprezentují všechny vybrané snímky. Vybraný bude pouze požadovaný snímek. Objekt bude obsahovat atributy odpovídající vybraným větvím stromu *dscData*

/satram/frame/{Start_time}/asrootfile v rámci této cesty se posílají údaje o jednotlivých snímcích. Jedna se o přístup k datům uložených ve stromě *dscData* kde {Start_time} je jednoznačný identifikátor a je to timestamp vytvoření snímku. Požadavky na tuto cestu je nutné dále specifikovat pomocí těla dotazu. Specifikace slouží k výběru větve stromu *dscData*, ze kterých budou data vybrána. Filtrování pomocí specifikací v těle dotazu není pro tuto cestu dostupné.

Informace budou v odpovědi odeslány ve formě *rootfile* souboru, který bude obsahovat strom *dscData*. Tento strom bude odpovídat stromu *dscData* z původních souborů s tím rozdílem, že se v něm budou nacházet pouze vybrané větve a data z požadovaného snímku

/satram/frame/{from}/to/{to} v rámci této cesty se posílají údaje o snímcích v rámci specifikovaného časového rámce. Jedná se o přístup k datům uložených v stromě *dscData* kde {Start_time} je jednoznačný identifikátor a je to timestamp vytvoření snímku. V odpovědi budou informace o všech snímcích jejichž *Start_time* je mezi hodnotami {from} a {to} včetně. Požadavky na tuto cestu je nutné dále specifikovat pomocí těla dotazu. Specifikace slouží k výběru větve stromu *dscData*, ze kterých budou data vybrána.

Informace budou v odpovědi odeslány ve formě JSON, který bude tvořen objektem *frames*, který je polem objektů, které reprezentují všechny

vybrané snímky. Každý objekt bude obsahovat atributy odpovídající vybraným větvím stromu *dscData*

/satram/frame/{from}/to/{to}/asrootfile v rámci této cesty se posílají údaje o snímcích v rámci specifikovaného časového rámce. Jedná se o přístup k datům uložených v stromě *dscData* kde {Start_time} je jednoznačný identifikátor a je to timestamp vytvoření snímku. V odpovědi budou informace o všech snímcích jejichž *Start_time* je mezi hodnotami {from} a {to} včetně. Požadavky na tuto cestu je nutné dále specifikovat pomocí těla dotazu. Specifikace slouží k výběru větve stromu *dscData*, ze kterých budou data vybrána.

Informace budou v odpovědi odeslány ve formě *rootfile* souboru, který bude obsahovat strom *dscData*. Tento strom bude odpovídat stromu *dscData* z původních souborů s tím rozdílem, že se v něm budou nacházet pouze vybrané větve a data z vybraných snímků

/satram/cluster/{Start_time}/index/{CounterValue} v rámci této cesty se posílají údaje o jednotlivých clustrech. Jedná se o přístup k datům uložených v stromech *dscData* a *clusterFile* kde *Start_time* je jednoznačný identifikátor a je to timestamp vytvoření snímku, a dvojice {Start_time} a {CounterValue} jednoznačný identifikátor clusteru. Požadavky na tuto cestu je nutné dále specifikovat pomocí těla dotazu. Specifikace slouží k výběru větví stromů *dscData* a *clusterFile*, ze kterých budou data vybrána. Filtrování pomocí specifikací v těle dotazu není pro tuto cestu dostupné.

Informace budou v odpovědi odeslány ve formě JSON, který bude tvořen objektem *frames*, který je polem objektů, které reprezentují všechny vybrané snímky. Vybraný bude pouze snímek, ve kterém se nachází požadovaný cluster. Objekt bude obsahovat atributy odpovídající vybraným větvím stromu *dscData*. Objekt bude dále obsahovat atribut *clusters*, který je polem objektů reprezentujících všechny vybrané clustery obsažené v daném snímku. Vybraný bude pouze požadovaný cluster. Objekt bude obsahovat atributy odpovídající vybraným větvím stromu *clusterFile*.

/satram/cluster/{Start_time}/index/{CounterValue}/asrootfile v rámci této cesty se posílají údaje o jednotlivých clustrech. Jedná se o přístup k datům uložených v stromech *dscData* a *clusterFile* kde *Start_time* je jednoznačný identifikátor a je to timestamp vytvoření snímku, a dvojice {Start_time} a {CounterValue} jednoznačný identifikátor clusteru. Požadavky na tuto cestu je nutné dále specifikovat pomocí těla dotazu. Specifikace slouží k výběru větví stromů *dscData* a *clusterFile*, ze kterých budou data vybrána. Filtrování pomocí specifikací v těle dotazu není pro tuto cestu dostupné.

Informace budou v odpovědi odeslány ve formě *rootfile* souboru, který bude obsahovat stromy *dscData* a *clusterFile*. Tyto stromy budou odpovídat stromům z původních souborů s tím rozdílem, že se v nich budou nacházet pouze vybrané větve a data z vybraného snímků a clusteru

/satram/cluster/{from}/to/{to} v rámci této cesty se posílají údaje o clusterech v rámci specifikovaného časového rámce. Jedná se o přístup k datům uložených v stromech *dscData* a *clusterFile* kde *Start_time* je jednoznačný identifikátor a je to timestamp vytvoření snímku. V odpovědi budou informace o všech snímcích jejichž *Start_time* je mezi hodnotami {from} a {to} včetně a o všech clusterech, které se v těchto snímcích nacházejí. Požadavky na tuto cestu je nutné dále specifikovat pomocí těla dotazu. Specifikace slouží k výběru větví stromů *dscData* a *clusterFile*, ze kterých budou data vybrána a také k dodatečné filtraci vybraných snímků a clusterů.

Informace budou v odpovědi odeslány ve formě JSON, který bude tvořen objektem *frames*, který je polem objektů, které reprezentují všechny vybrané snímky. Každý objekt bude obsahovat atributy odpovídající vybraným větvím stromu *dscData*. Každý objekt bude dále obsahovat atribut *clusters*, který je polem objektů reprezentujících všechny vybrané clustery obsažené v daném snímku. Každý objekt bude obsahovat atributy odpovídající vybraným větvím stromu *clusterFile*

/satram/cluster/{from}/to/{to}/asrootfile v rámci této cesty se posílají údaje o clusterech v rámci specifikovaného časového rámce. Jedná se o přístup k datům uložených v stromech *dscData* a *clusterFile* kde *Start_time* je jednoznačný identifikátor a je to timestamp vytvoření snímku. V odpovědi budou informace o všech snímcích jejichž *Start_time* je mezi hodnotami {from} a {to} včetně a o všech clusterech, které se v těchto snímcích nacházejí. Požadavky na tuto cestu je nutné dále specifikovat pomocí těla dotazu. Specifikace slouží k výběru větví stromů *dscData* a *clusterFile*, ze kterých budou data vybrána a také k dodatečné filtraci vybraných snímků a clusterů.

Informace budou v odpovědi odeslány ve formě *rootfile* souboru, který bude obsahovat stromy *dscData* a *clusterFile*. Tyto stromy budou odpovídat stromům z původních souborů s tím rozdílem, že se v nich budou nacházet pouze vybrané větve a data z vybraných snímků a clusterů

/satram/rootfile/{from}/to/{to} v rámci této cesty se informace o dostupných *rootfile* souborech v časovém rozsahu od {from} do {to} v timestamp. Požadavky na tuto cestu se nedají dále specifikovat pomocí těla dotazu.

Informace budou v odpovědi odeslány ve formě JSON, který bude tvořen polem objektů s názvem *rootfiles*. Každý z objektů v tomto poli bude

odpovídat jednomu *rootfile* souboru a bude obsahovat atributy *path*, který specifikuje relativní cestu k danému souboru v rámci serveru, dále *start_time*, který specifikuje čas vytvoření prvního snímku v daném souboru ve formě timestamp a nakonec *start_time_formated*, který specifikuje čas vytvoření prvního snímku v daném souboru ve formě čitelného data

/satram/rootfile/{from}/to/{to}/asrootfile v rámci této cesty se informace o dostupných *rootfile* souborech v časovém rozsahu od {from} do {to} v timestamp. Požadavky na tuto cestu se nedají dále specifikovat pomocí těla dotazu.

Informace budou v odpovědi odeslány ve formě *rootfile* souboru, který bude obsahovat strom *AT*. Tento strom bude obsahovat záznamy, z nichž každý bude odpovídat jednomu *rootfile* souboru a bude obsahovat větev *path*, která specifikuje relativní cestu k danému souboru v rámci serveru, dále *start_time*, která specifikuje čas vytvoření prvního snímku v daném souboru ve formě timestamp a nakonec *start_time_formated*, která specifikuje čas vytvoření prvního snímku v daném souboru ve formě čitelného data

/satram/rootfile/{date} v rámci této cesty se budou posílat přímo zdrojové soubory odpovídající datu {date} v timestamp. Požadavky na tuto cestu se nedají dále specifikovat pomocí těla dotazu.

API tak bude umožňovat požadavky na jeden specifický cluster, jeden specifický snímek, clustery z určitého časového rozsahu, snímky z určitého časového rozsahu, informace o *rootfile* souborech, ve kterých jsou uložená data, z určitého časového rozsahu. Odpovědi na všechny tyto požadavky budou dostupné jako JSON nebo *rootfile*. API bude umožňovat také stažení zdrojových souborů.

Při dotazech na clustery budou dostupná také data o snímcích, ve kterých se dané clustery nacházejí. Důvodem je, že clustery jsou identifikačně závislé na snímcích proto jsou tyto data vždy snadno a jednoznačně dohledatelné a není důvod neumožnit dostupnost taky těchto dat v rámci daného dotazu.

4.3.2 Filtrování dat

Filtrování dat je velmi důležitou funkcí systému. Filtrováním se nejen omezí množina dat na data, o které má uživatel zájem, ale také se výrazně zmenší množství přenášených dat. Filtrovaní bude probíhat na dvou úrovních.

Filtrování na úrovni indexů, kdy se k filtrování dat budou využívat data z relační databáze a nad nimi postavené indexy. Toto filtrování bude sloužit k prvotnímu hrubému zmenšení počtu záznamů k nimž budeme přistupovat během druhé úrovně filtrování. Nastavení těchto filtrů bude probíhat pomocí parametrů získaných v rámci cesty požadavku na API.

Filtrování na úrovni *rootfile* souborů, kdy se bude sekvenčně procházet množina všech záznamů vybraných na první úrovni filtrování. Tyto záznamy budou porovnávaný s nastavenými filtry a pokud budou vyhovovat daným filtrům, budou dále použity pro generování odpovědi. Nastavení těchto filtrů bude docházet pomocí JSON obsaženého v těle dotazu. Důvodem použití JSON obsaženého v těle dotazu, namísto většího množství složitelnějších cest, je velké množství atributů, pomocí kterých je možné data filtrovat a taky možnost vyžít kombinace těchto filtrů. K filtrování bude možné využít všechny větve všech stromů v zdrojových souborech.

JSON v těle dotazu bude poskytovat robustní datovou strukturu pro definování filtrů a jejich kombinací. JSON může obsahovat objekt s názvem *filters*, který bude polem objektů. Každý z těchto objektů bude definovat jeden filtr. Výsledek filtrování bude konjunkce těchto filtrů. Každý z filtrů bude obsahovat tyto atributy pro definování filtrovaných dat:

attribute je povinný atribut jehož hodnota bude definovat název větve stromu, pomocí které se bude filtrovat. Pokud větev s daným názvem neexistuje, bude daný filtr vyhodnocen jako splněný

position je atribut, který se bude v objektu nacházet právě tehdy když větev stromu, pomocí které daný filtr filtruje je pole. Hodnota tohoto parametru definuje pozici v daném poli, pomocí které se bude filtrovat.

Objekt bude dále obsahovat kombinaci těchto atributů pro nastavení samotného filtru:

> splněn pokud je hodnota dat větší než hodnota daného parametru

>= splněn pokud je hodnota dat větší nebo rovna hodnotě daného parametru.

== splněn pokud je hodnota dat rovna hodnotě daného parametru.

<= splněn pokud je hodnota dat menší nebo rovna hodnotě daného parametru.

< splněn pokud je hodnota dat menší než hodnota daného parametru.

!= splněn pokud je hodnota dat není rovna hodnotě daného parametru.

Výsledkem filtru bude disjunkce vyhodnocení těchto parametrů.

JSON v těle dotazu bude také obsahovat objekt s názvem *info*, který bude polem názvů větví. V odpovědi se budou nacházet pouze data z větví definovaných v tomto poli. Pokud větev s daným názvem neexistuje bude daný název ignorován. Množina větví, ze kterých se budou čerpat data pro odpověď a množina větví, které se budou používat pro filtrování nemusí obsahovat žádné společné prvky.

4.4 Nové části systému pomocí konfigurace

Spolu s výše navrženým rozhraním a funkčností bude námi navrhovaný systém obsahovat také API, které bude řízené konfiguračním souborem. Toto API má sloužit pro jednoduché přidávání nové funkcionality do systému. Bude tedy možné pomocí konfiguračního souboru přidat nový zdroj dat do systému a tím jej přizpůsobit pro použití s jinými projekty, bez nutnosti programovat novou funkcionalitu. Při implementaci pak nakonfigurujeme API pro zpracování dat ze systému SATRAM a otestujeme jej proti ručně napsanému API. Tak ověříme jestli koncept vytvářené API pomocí konfigurace má smysl v rámci zprávy dat s podobným charakterem jako mají data ze systému SATRAM.

Definice konfigurace API bude součástí konfiguračního souboru pro celý systém. Tento soubor bude ve formátu XML. API vzniklé pomocí konfigurace bude mít definovanou vlastní základní cestu, vlastní relační databázi pro indexy a vlastní složku pro uchovávaní souborů. Systém bude moci obsahovat více API vzniklých konfigurací současně. V konfiguračním souboru se bude nacházet tag *generate*, ve kterém budou definovány všechny tyto API. Definice jednoho API bude ohraničená tagem *server* s atributem *name*, kterého hodnota definuje název API a také základní cestu tohoto API. Uvnitř tagu *server* se dále budou nacházet tagy definující další vlastnosti API. Tag *dataConnection* bude obsahovat definici připojení k relační databázi, tag *rootfile_folder* bude obsahovat cestu ke složce kde budou uloženy datové soubory, tag *new_rootfile_folder* bude obsahovat cestu ke složce odkud budou vybírány nové datové soubory a tag *constants* bude obsahovat definice konstant pro dané API. Jedna konstanta bude definována jedním tagem *constant* s atributy *name* pro název konstanty, *type* pro definici datového typu konstanty a *value* pro definici hodnoty dané konstanty.

4.4.1 Popis rootfile

Dále bude následovat popis struktury *rootfile* souborů, se kterými bude dané API pracovat. Nejdříve budou definovány stromy, které se budou v daných souborech nacházet. Definice těchto stromů se bude nacházet v těle tagu *trees*, kde definice jednoho stromu bude ohraničená tagem *tree* s atributem *name*, kterého hodnota definuje název stromu. V těle tohoto tagu budou definovány jednotlivé větve stromu pomocí tagu *branch* s těmito atributy:

name je název větve

type je datový typ větve

size pokud větev neobsahuje pole je hodnota tohoto atributu 0, pokud obsahuje je hodnota tohoto atributu buď konstanta, pokud má dané pole konstantní délku, nebo název větve, která definuje délku daného pole.

max_size pokud atribut *size* má hodnotu 0, tento atribut není přítomný, pokud ne pak tento atribut definuje maximální velkost pole.

Po definici stromů následuje definice hierarchie těchto stromů, která se bude nacházet v těle tagu *hierarchy*. Vztah dvou stromů navzájem bude definován tagem *connection*, který bude obsahovat atributy:

master je název stromu na vyšší úrovni

slave je název stromu na nižší úrovni

V těle tohoto tagu budou pomocí tagů *connect_by* definovány větve, pomocí kterých budou stromy propojené. Tento tag bude obsahovat atributy:

master je název větve v stromě s vyšší úrovní

slave je název větve v stromě s nižší úrovní

Takto bude systém vědět s jakými daty pracuje dané API.

4.4.2 Popis indexů

Následovat bude popis indexů, se kterými daném API bude pracovat. Definice indexů se bude nacházet v těle tagu *indexies*, ve kterém se budou nacházet tagy *index*, z nich každý definuje jednu tabulku v relační databázi. Tyto tagy budou obsahovat atributy:

name je název tabulky

base_index určuje jestli je daná tabulka základní. Konfigurace každého API musí obsahovat právě jednu základní tabulku. Všechny ostatní tabulky musí být tranzitivně závislé na základní tabulce. Základní tabulka definuje zejména cesty k jednotlivým datovým souborům, musí tedy obsahovat sloupec, který určuje cestu k reprezentovanému datovému souboru.

map_tree je název stromu, ze kterého se budou čerpat data.

Tělo tagu *index* obsahuje tagy *columns* a *constraints*. V těle tagu *columns* jsou definovány sloupce tabulky. Každému sloupci odpovídá jeden tag *column* s těmito atributy:

name je název sloupce

type je datový typ sloupce v databázi

server_type je datový typ sloupce v systému

map_branch je název větve stromu, která se na daný sloupec namapuje. Atribut není přítomný pokud je přítomný atribut *special*

special definuje, že daný sloupec obsahuje speciální data. Atribut special může mít tyto hodnoty:

autoincrement znamená, že daný sloupec bude automaticky vyplňovat vlastní hodnotu tak aby byla jedinečná

foreign_key znamená, že daný sloupec bude cizím klíčem do jiné tabulky

position znamená, že do daného sloupce se vyplní pozice záznamu v rámci datového souboru, který byl použit jako zdroj dat pro vytvoření daného řádku tabulky

rootfile znamená, že daný sloupec obsahuje cestu k datovému souboru. V celé konfiguraci každého API se nachází právě jeden sloupec s touto hodnotou atributu *special*

Atribut je přítomný pokud není přítomný atribut *map_branch*

V těle tagu *constraints* jsou definovány primární klíč a cizí klíče tabulky. Primární klíč je definován v tagu *primary_key*. V těle tohoto tagu se nachází tagy *key* s atributem *name*, který obsahuje název sloupce, který je součástí primárního klíče tabulky. Pokud tabulka obsahuje cizí klíče, jsou tyto klíče definovány v těle tagu *foreign_keys*. Jeden cizí klíč je pak definovaný v jednom tagu *foreign_key* s atributem *name*, který definuje název cizího klíče. V těle toho tagu se nachází tagy *key* s těmito atributy:

name je název sloupe, který je částí cizího klíče

reference je název tabulky, na kterou daný sloupec odkazuje

reference_key je název sloupce, na který daný sloupec odkazuje

where slouží pro definici propojení při přidávání nových dat. Atribut může mýt tyto hodnoty:

last znamená, že při přidávání se použije hodnota posledně přidaného záznamu z tabulky, do které se odkazujeme

rootfile znamená, že při přidávání se použije hodnota záznamu, který má hodnotu sloupce, který odkazuje na umístnění datového souboru stejnou jako je cesta k aktuálně přidávanému datovému souboru. Tuto hodnotu lze použít pouze pro cizí klíče do základní tabulky

same znamená, že při přidávání se použije hodnota z větve stromu, která má stejný název jako atribut *reference_key* tohoto cizího klíče

Takto systém bude vědět vytvořit indexy pro dané API, aby mohl pracovat s datovými soubory.

4.4.3 Popis rozhraní

Nakonec bude popsáno samotné komunikační API, které do systému tímto způsobem vkládáme. Definice rozhraní se bude nacházet v těle tagu *routes*, ve kterém se budou nacházet tagy *route*, z nich každý definuje jednu cestu v rámci REST rozhraní. Tyto tagy budou obsahovat atributy:

route je definice samotné cesty.

action je definice akce, kterou bude systém na dané cestě vykonávat. Atribut může mýt tyto hodnoty:

- **filter** je akce kdy dochází k filtrování dat z datových souborů, ze kterých se pak vytvoří odpověď na požadavek
- **info** je akce kdy dochází k výběru dat z relační databáze, ze kterých se pak vytvoří odpověď na požadavek
- **file** je akce kdy se vybírá zdrojový soubor, který pak bude odeslán jako odpověď na požadavek
- **update** je akce kdy je systém informován, že by měl zkontrolovat jestli nepřibily nové zdrojové soubory

result_as definuje formát odpovědi na požadavek na tuto cestu. Atribut může mýt tyto hodnoty:

- **JSON** výstup bude ve formě JSON
- **rootfile** výstup bude ve formě *rootfile* souboru

Jestli cesta obsahuje parametry, pomocí kterých se nastavuje její funkce, je význam těchto parametrů popsán v těle tagu *route* pomocí tagů *data_source*, *index* a *specific_rows*.

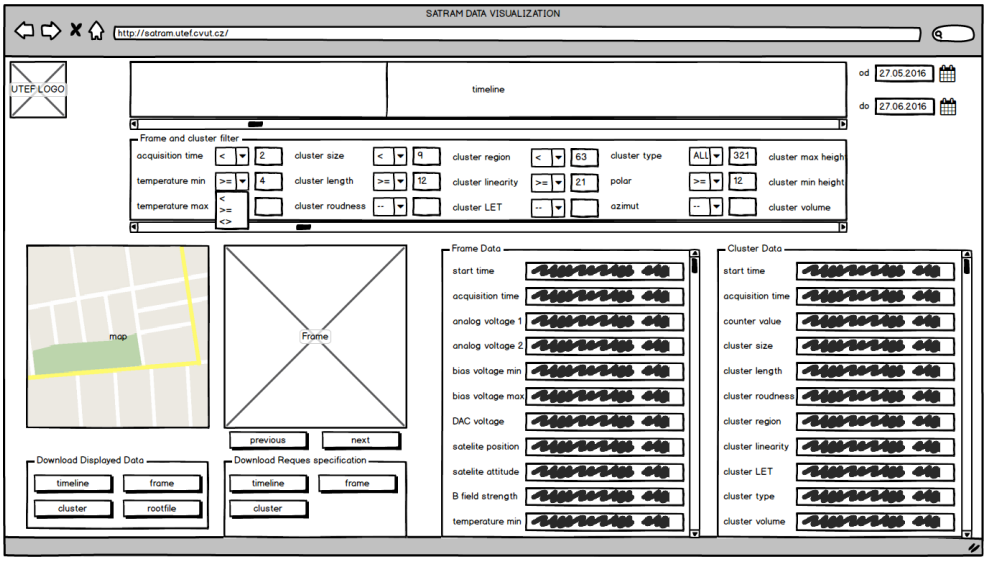
Tag *data_source* obsahuje atribut *name*, který definuje název stromu, ze kterého se budou čerpat data, spolu s definovaným stromem se budou data čerpat z jemu nadřazeného stromu. Tohle pravidlo platí rekurzivně.

Tag *index* definuje jak a k jaké tabulce v relační databázi se bude přistupovat pomocí parametrů v cestě. Tento tag obsahuje atribut *name*, který definuje název tabulky. Tělo tagu bude obsahovat tagy *attribute*, které pro parametry definují jejich konkrétní funkci. Tyto tagy budou obsahovat atributy:

position definuje, který parametr v pořadí daný tag popisuje

name definuje název sloupce v tabulce, se kterým se bude hodnota parametru porovnávat

comparison definuje funkci porovnání. Přípustné jsou hodnoty >, >=, =, <=, < a <>



Obrázek 4.4: Návrh webového rozhraní pro prezentaci dat

Tag *specific_rows* se požije jestli některý z parametrů definuje přesné pořadí požadovaného záznamu v některém ze stromů. Tento tag bude obsahovat tagy *specific_row*. Tyto tagy budou obsahovat atributy:

position definuje, který parametr v pořadí daný tag popisuje

data_source definuje název stromu, ve kterém se daný záznam nachází

Takto bude systém vědět jaké komunikační API má vytvořit.

4.5 GUI

Při návrhu GUI[13] budeme vycházet zejména z případů užití, které jsme zjistili při analýze GUI 3.5. V této sekci tak ukážeme návrh GUI a poskytneme k němu jednoduchý popis. Návrh můžete vidět na obrázku 4.4.

Námi navržené webového rozhraní obsahuje ovládací prvky tak aby bylo možné pomocí tohoto rozhraní vykonávat všechny případy užití, které jsme zjistili při analýze. Rozhodli jsme se všechnu funkcionalitu zařadit do jediné stránky bez přechodů, aby jsme uživatele nemátli různými přechody a tím jestli se data z předchozí stránky ztratila, nebo budou přítomná, až se vrátí zpět.

V levém horním rohu se nachází logo ÚTEF ČVUT, které slouží jako odkaz na na webové stránky ÚTEF ČVUT o projektu SATRAM. Vpravo od něj se nachází časová osa, která slouží pro výběr snímků ze zvoleného časového rozsahu. Uživatel může pomocí myši na časové ose zvolit konkrétní snímek, který bude zobrazen níže. Vpravo od časové osy se nachází dva ovládací prvky

pro výběr časového rozsahu zobrazeného na časové osy. Pod časovou osu se nachází blok pro filtrování snímků a clusterů. Blok obsahuje všechny atributy, pomocí kterých lze filtrovat. Při každém atributu se nastaví hodnota, se kterou se bude porovnávat, a jak se bude porovnávat.

Pod blokem pro nastavení filtrování se nachází ovládací prvky týkající se jednoho snímku a jeho clusterů. Vlevo se nachází mapa, na které je zobrazena poloha vytvoření aktuálně zobrazeného snímku. Vpravo od ní se nachází samotný snímek. Jedná se o snímek, který je vybrán na časové ose. Tento snímek je interaktivní a je možné na něm zvolit některý z clusterů. Pod snímkem se nachází tlačítka pro přepnutí na předchozí, či následující snímek. Vpravo od snímku se pak nachází bloky pro zobrazení dat ze snímku a z clusteru. V bloku pro snímek se zobrazují všechna data o zobrazeném snímku. V bloku pro cluster se zobrazují všechna data o zvoleném clusteru.

Nakonec v levém dolním rohu se nachází bloky pro stažení zobrazených dat a pro stažení dat o požadavcích na API, kterých výsledky byli zobrazená data.

Námi navržené GUI může sloužit jako podklad pro vytvoření webového rozhraní pro prezentaci dat, z námi navrhovaného systému, v budoucích projektech.

Kapitola 5

Realizace

Systém je napsán v jazyku C++ a ke svému chodu využívá PostgreSQL databázi a systém ROOT.

PostgreSQL[14] slouží jako relační databáze pro uchovávání indexů nad daty uloženými v *rootfile* souborech. Pro komukaci s touto databází systém využívá knihovnu *libpqxx*[15], která je standardní knihovnou pro propojení *PostgreSQL* s C++ aplikací.

Systém ROOT slouží pro práci s *rootfile* a to zejména čtení souborů, které slouží pro ukládaní dat a vytváření dočasných souborů, pokud je od systému požadovaná odpověď ve formě *rootfile* souboru. Náš systém tedy nepřistupuje k datům v souborech přímo, ale využívá API pro práci *rootfile* poskytované systémem ROOT. Systém ROOT se pak samostatně stará o čtení a zápis dat do souborů, o paralelní přístup k datům a o všechnu s tím spojenou režii jako například správu paměti.

Pro vytvoření jádra našeho systému je použitá kolekce C++ HTTP knihoven zvaná *Proxygen*[16]. Tato kolekce knihoven poskytuje komplexní funkcionalitu pro vytvoření HTTP serveru schopného paralelního zpracování požadavků.

Dále náš systém využívá knihovnu *RapidJSON*[17] pro práci s daty ve formátu JSON a také knihovnu *TinyXML-2*[18] pro práci s daty ve formátu XML.

5.1 Fungování systému

Po startu systému se načte konfigurace z konfiguračního souboru configuration.xml umístěného ve stejné složce jako spustitelný soubor samotného systému. Následně systém vytvoří požadovaný počet vláken, které se budou starat o obsluhu požadavků. V těchto vláknech je řízení předáno instanci třídy *DataServerHandlerFactory*.

Třída *DataServerHandlerFactory* se stará zejména o vytváření správných *RequestHandler* objektů na základě cesty požadavku na server. Při vytváření

instance třídy *DataServerHandlerFactory* se vytvoří podle konfigurace dosud neexistující tabulky pro indexy v PostgreSQL a zkontroluje složky sloužící pro přidávání nových dat, jestli se v nich nenacházejí soubory, které je potřebné přidat. Když systém obdrží požadavek, je instancí třídy *DataServerHandlerFactory* zavolána metoda *RequestHandler* DataServerHandlerFactory::onRequest(RequestHandler* handler, HTTPMessage* message)*. V této metodě je pak na základě cesty dotazu vybrána odpovídající podtřída třídy *RequestHandler* a je vytvořená nová instance této podtřídy, které jsou předány parametry z požadavku. Tato třída je pak předána jednomu z vláken na zpracování.

Podtřídy třídy *RequestHandler* se starají o obsluhu vyhodnocení dotazu na systém. Uvnitř těchto tříd dochází k parsování parametrů obsažení v těle dotazu, výběru datových souborů, které je nutné zpracovat a k filtrování dat pomocí indexů. Dále tyto třídy řídí třídy, které zpracovávají údaje z datových souborů a kombinují jejich výsledky do konečné podoby.

Třídy, které zpracovávají údaje z datových souborů pro přímo implementované API jsou napsané speciálně pro potřeby konkrétního stromu z datového souboru, který je v nich zpracováván. Pro zpracování údajů z datových souborů pro API, které je definováno v konfiguračních souborech, je použitá třída *Tree*, která poskytuje obecnou funkcionalitu, která je nastavená podle údajů z konfiguračního souboru.

5.2 Přístup k datům v rootfile

V našem systému přistupujeme pouze k datům uloženým v *rootfile* ve formě stromů. K těmto datům budeme přistupovat pomocí API, které nám poskytuje systém ROOT.

Nejdříve je vytvořená instance třídy *TFile*, která reprezentuje soubor otevřený v systému ROOT. Následně můžeme z toho souboru vybrat námi požadované stromy pomocí metody *virtual TObject * Get (const char *namecycle)* kde *namecycle* představuje název námi požadovaného stromu. Vrácený objekt je pak nutné přetypovat na *TTree*

TTree je objekt reprezentující datový strom používaný v systému ROOT. Záznamy v tomto stromu jsou rozděleny do větví. Záznamy z těchto se nevybírají přímo, ale je nutné pro každou větev, ze které chceme získat data, vytvořit buffer, do kterého se budou ukládat hodnoty záznamu. Pro načítaní dat ze stromu v systému využíváme tyto metody třídy *TTree*:

Int_t SetBranchAddress (const char *bname, T *add, TBranch **ptr=0) slouží k přiřazení bufferu ke konkrétní větvi stromu. Parametr *bname* je název větve, ke které chceme buffer přiřadit a parametr *ptr* je ukazatel na daný buffer

virtual void SetBranchStatus (const char *bname, Bool_t status=1, UInt_t *found=0) slouží ke specifikováni, jestli se z dané větve mají načítat data do bufferu. Parametr *bname* je název větve, pro kterou chceme specifikovat, jestli se z ní mají načítat data do bufferů a parametr *status* nastavuje jestli budou data z dané větve načtena do bufferu nebo ne

virtual Int_t GetEntry (Long64_t entry=0, Int_t getall=0) slouží pro načtení dat z konkrétního záznamu do bufferů. Parametr *entry* je pořadí záznamu, ze kterého chceme získat data

virtual Long64_t GetEntries () const vrací počet záznamů v daném stromě

5.3 Přidávání dat

Přidávání dat do systému probíhá buď při startu systému, nebo může být vyvoláno požadavkem na speciální cestu v API, která má na starosti spouštění přidávání dat. API má pro každý projekt vlastní cestu pro vyvolání přidávání dat. Přidávání dat má dvě fáze.

5.3.1 Fáze 1, převzetí dat

Když je v některém z API vyvolán proces přidávání dat, je zkontrolovaná složka pro přidávání dat daného API. Pokud se v této složce nacházejí soubory, nebo složky, které obsahují soubory, jsou tyto soubory přeneseny do složky pro uchovávání datových souborů daného API, přičemž je zachován jejich název a relativní cesta vzhledem na složku pro přidávaní a složku pro ukládání dat. Složky pro přidávání a ukládání jsou definováni pro každé API v konfiguračním souboru.

5.3.2 Fáze 2, přidávání dat do indexů

Když jsou data přenesena, je potřeba přidat data o těchto souborech do tabulek v relační databázi, které představují indexy, aby s nimi mohl systém správně pracovat. Nejdříve se pro každý soubor zkontroluje, jestli již není v primární tabulce záznam o souboru se stejnou relativní cestou a názvem. Pokud ano víme, že stejný soubor jsme už jednou přidávali, co znamená, že indexy již obsahují data potřebná pro práci s tímto souborem.

Pokud v primární tabulce záznam o souboru se stejnou relativní cestou a názvem není, začne systém postupně vyplňovat všechny indexy. Způsob vyplňování indexů se liší v závislosti, jestli se jedná o API, které bylo přímo napsané, nebo o API popsané v konfiguračním souboru.

5.3.2.1 Přímo napsané API

V přímo popsaném API se o způsob vyplňování indexů stará programátor daného API. V systému je zatím napsáno pouze API pro projekt SATRAM. O přidávání dat do tohoto API se stará třída *SatramUpdateRequestHandler*. V něm je nejdříve přidán záznam to tabulky *rootfile*, která je primární tabulkou. Následně jsou přidávány záznamy do tabulky *frames* kde jsou uloženy údaje o každém snímku v daném datovém souboru.

5.3.2.2 API popsané v konfiguračním souboru

V API, které je popsané v konfiguračním souboru se indexy vyplňují v pořadí, v jakém jsou definovány v konfiguračním souboru. Každý z indexů je navázán na hodnoty z jednoho stromu. U indexů reprezentovaných tabulkou, která není primární, pak jeden záznam ve stromě odpovídá jednomu záznamu v indexu. Pokud je daný index reprezentován primární tabulkou, víme, že pro jeden soubor je v takové tabulce právě jeden záznam, a tedy budeme přidávat právě jeden záznam. Pokud tabulka není primární, může být potřebné do ní přidat více záznamů, podle toho kolik záznamů je v přiřazeném stromu.

Přidání jednoho záznamu je řízené definicí daného indexu v konfiguračním souboru. Pro přidávání záznamů jsou využívány *insert* dotazy, které se postupně budují. Nejdříve jsou ze stromu načítána data ze záznamu, která chceme přidat do indexu. Jsou vybrané pouze větve, které jsou specifikované v *map_branch* atributech tagů *column* daného indexu, které nemají nastavená atribut *special*. Následně jsou do *insert* dotazu postupně přidávány data pro každý sloupec tabulky reprezentující index. To jaké data jsou vyplněny závisí od hodnoty atributu *special* tag *column*, který popisuje daný sloupec. Hodnoty atributu *special* můžou být:

atribut není nastaven v tomto případe se vyplní data, která byla načítána ze stromu z větve, která je specifikována v atributu *map_branch* daného tagu

autoincrement v tomto případě je vyplněná hodnota *"DEFAULT"*. Relační databáze pak automaticky vyplní data v daném sloupci

position v tomto případě se vyplní pořadí daného záznamu v stromě

rootfile v tomto případě se vyplní relativní cesta k aktuálně přidávanému datovému souboru

foreign_key v tomto případě se jedná o cizí klíč. Hodnota, která se vyplní, tak dále závisí na specifikaci daného cizího klíče a to zejména atributu *where*, který slouží pro definici propojení při přidávání nových dat. Atribut *where* může mýt tyto hodnoty:

last znamená, že při přidávání se použije hodnota ze sloupce s názvem definovaným v atributu *reference_key* daného cizího klíče, z posledně přidaného záznamu z tabulky, do které se odkazujeme

rootfile znamená, že při přidávání se použije hodnota ze sloupce s názvem definovaným v atributu *reference_key* daného cizího klíče, ze záznamu z tabulky, do které se odkazujeme. Tento záznam má stejnou hodnotou sloupce, který obsahuje relativní cestu k datovému souboru, jako je relativní cesta k aktuálně přidávanému datovému souboru.

same znamená, že při přidávání vyplní data, která byla načítána ze stromu z větve, která je specifikována v atributu *name* daného cizího klíče.

5.4 Filtrace a čtení dat

Při zpracování dat vždy využíváme co nejméně větví, aby jsme minimalizovali množství dat načítaných ze souboru. Rozhodli jsme se, že filtrování dat a čtení požadovaných dat oddělíme, protože větve, pomocí kterých se při daném požadavku filtruje, častokrát nemusí odpovídat větvím, ze kterých jsou požadována výslední data a současně lze předpokládat, že počet záznamů, které bude nutné filtrovat je mnohem vyšší než počet záznamů po filtraci. Vyhodnocování požadavku tak bude mít čtyři fáze. Rozdělením vyhodnocování dotazů na čtyři fáze minimalizujeme čas potřebný pro zpracování údajů.

5.4.1 Fáze 1, filtrování pomocí indexů

V této fázi je pomocí dat z relační databáze vybraná podmnožina záznamů takových, které splňují podmínky filtrace zadané v rámci cesty požadavku a jejich parametrů a všechny datové soubory, ve kterých se tyto sloupce nacházejí.

K filtrování jsou využívaný *select* dotazy. Tyto dotazy jsou pro naprogramované API předem definovány. Vybere se pouze konkrétní verze dotazu na základe cesty požadavku a vyplní se parametry dotazu z parametrů obsažených v cestě požadavku.

Pro API definované v konfiguračním souboru je situace složitější. Na základě cesty požadavku se nejdříve rozhodne, o kterou z cest se v konfiguračním souboru jedná. Definice této cesty pak obsahuje návod jak vygenerovat *select*, pomocí kterého vyfiltrujeme požadované data.

Struktura toho návodu je popsána v podsekci 4.4.3 v popisu tagu *index*, který je součástí těla tagu *route*. Tento návod tak definuje:

- tabulku, ze které se bude filtrovat
- sloupce, pomocí kterých se bude filtrovat

- přiřazení parametrů v cestě k sloupcům, ze kterých se bude filtrovat
- způsob porovnání

Za pomoci těchto informací je vygenerován *select* dotaz. Součástí tohoto dotazu je projekce na sloupec z této tabulky, který je označený jako specifický sloupec typu *position*. Výsledkem tohoto dotazu jsou pozice v davovém souboru, které prošli daným filtrem.

Pokud tabulka, pomocí které filtrujeme, není primární tabulkou, je vytvořen další dotaz se stejnými filtry, ale projekcí na cizí klíč. Protože všechny tabulky musí být tranzitivně závislé na primární tabulce, takový cizí klíč vždy existuje. Systém následně rekurzivně prochází cizí klíče dokud se nedostane k základní tabulce, kde vybere cestu k potřebnému datovému souboru.

Fáze 2 a 3 jsou spouštěný nad vybranými datovými soubory postupně.

5.4.2 Fáze 2, filtrování v rootfile

V této fází se postupně přejdou všechny záznamy v daném datovém souboru, které byli vybrány ve fázi 1. V souboru jsou vybrané pouze větve, které jsou potřebné pro filtrování. Podmínky tohoto filtrování jsou nastaveny pomocí definice filtrů v těle požadavku. Při filtrování je vybraná podmnožina záznamů takových, které splňují podmínky této filtrace.

Pokud je strom, ze kterého filtrujeme závislý na jiném stromě, jsou filtrovány taky záznamu z tohoto stromu. To platí rekurzivně dokud jsou stromy závislé na jiném stromu. Pokud záznam, v některém ze stromu neprojde filtrací, jsou vyřazeny také všechny záznamy z jeho podstromu, které jsou na tomto záznamu závislé.

5.4.3 Fáze 3, načtení požadovaných dat

V této fázi se postupně přejdou všechny záznamy v daném datovém souboru, které byli vybrány ve fázi 2. V souboru jsou vybrané pouze větve, ze kterých jsou požadována výsledná data. Definice větví, ze kterých jsou požadována výsledná data je součástí těla dotazu. Při procházení dat jsou výsledná data uložena do příslušné datové struktury a to buď JSON nebo TTree v závislosti na požadovaném formátu výstupních dat. Datová struktura TTree je využita pro výstupní data tehdy, pokud má být odpověď na dotaz ve formě *rootfile* souborů.

Pokud je strom, ze kterého čteme data závislý na jiném stromě, jsou data čtená také z tohoto stromu. To platí rekurzivně dokud jsou stromy závislé na jiném stromu. Při čtení pak postupně procházíme vybrané záznamy ze stromu, který není závislý na žádném jiném stromě. Po načtení dat z jednoho záznamu, pak začneme postupně procházet vybrané záznamy, z na něm závislého stromu, které jsou provázány s tímto záznamem. Takto postupujeme rekurzivně, dokud se nedostaneme k stromu, ze kterého jsme chtěli původně filtrovat data. Takto

procházíme záznamy v jakési abstraktní stromové struktuře, kde k záznamům z nezávislého stromu postupně přidáváme záznamy z na něm závislých stromů.

Pokud jsou výstupní data ukládána do JSON, jsou v něm data uložená v takové formě, která odpovídá výše popsanému abstraktnímu stromu, a tedy vzájemná poloha dat ve struktuře definuje jejich vztah. Pokud jsou výstupní data ukládána do TTree, je pro každý typ stromu z datových souborů, ze kterého jsme čerpali data vytvořen také vlastní TTree, s odpovídajícími větvemi, do kterého se uloží odpovídající vybrané záznamy. Vztah mezi jednotlivými záznamy je pak vyjádřen pouze hodnotami v záznamech.

Pokud jsou některé z vybraných větví závislé na hodnotě z jiné větve, je tato větev taky přidána do výsledků nezávisle na tom jestli byla mezi vybranými větvemi. Pro API definované v konfiguračním souboru se takovéto větve poznají pomocí hodnoty atributu *size* tagu *branch* popsaného v podsekci 4.4.1.

5.4.4 Fáze 4, spojení dat a odeslání odpovědi

V této fázi jsou spojená všechna výsledná data ze všech vybraných datových souborů do jednoho celku a odeslána v těle odpovědi na požadavek. Pokud byl jako formát odpovědi požadován JSON, jsou výslední data rovnou odeslána. Pokud byl jako formát odpovědi požadován rootfile, jsou data nejprve uložena do dočasného souboru, který je pak odeslán v těle dotazu. Po odeslání dočasného souboru, je jeho lokální kopie vymazána.

Kapitola 6

Testování

Při testování se zaměříme na měření závislosti doby vyřízení požadavku vzhledem na různé parametry dat nebo požadavků. Vždy se budeme měnit jednu proměnná a bude sledovat jaký to má efekt. Každé měření bylo opakované 25 krát a následné byla použita průměrná hodnota z těchto měření.

6.1 Množství vstupních záznamů

V této sekci se zaměříme na to, jak vplývá množství dat na dobu vyřízení požadavku.

Nejdříve jsme měnili množství datových souborů v databázi. Každý datový soubor obsahuje data pro jeden den. Požadavky jsme posílali na cestu

```
/satram/cluster/1425254400/to/1460000000
```

a tělo požadavků bylo

```
{ "info":["Start_time","clstrSize","Acq_time"
,"clstrLET"], "filters":[ { "attribute":"Start_time"
, "==":1425254550} ] }
```

V tomto nastavení se vždy projdou všechny záznamy, ze všech dostupných datových souborů. Ve výsledku budou vždy informace o právě jednom snímku. Výsledky měření jsou v tabulce 6.1

počet souborů	1	2	5	10	15	20	25
čas (ms)	49	75	160	283	405	521	639

Tabulka 6.1: Vliv množství souborů na dobu vyřízení na čas

Doba vyřízení požadavku, dle měření, stoupá lineárně s množstvím dat, které je potřebné filtrovat bez pomoci indexů.

Pak jsme jsme se začali měnit počet dní pomocí parametrů v cestě. Požadavky jsme posílali na cestu

```
/satram/cluster/1425254400/to/X
```

kde X je parametr který jsme měnili. Tělo požadavků bylo

```
{ "info":["Start_time","clstrSize","Acq_time"
,"clstrLET"], "filters":[ { "attribute":"Start_time"
, "==":1425254550} ] }
```

V tomto nastavení se vždy projdou všechny záznamy, ze všech dostupných datových souborů v časovém rozsahu daném parametry cesty. Ve výsledku budou vždy informace o právě jednom snímku. Výsledky měření jsou v tabulce 6.1

počet dní	1	2	5	10	15	20	25
čas (ms)	77	106	176	297	443	526	621

Tabulka 6.2: Vliv množství dat na dobu vyřízení na čas

Doba vyřízení požadavku, dle měření, stoupá lineárně s množstvím dat, které je potřebné filtrovat bez pomoci indexů. Oproti datům v tabulce 6.1, jsou v tabulce 6.1 o trocha delší časy. Důvodem bude zřejmé nutnost data nejdříve filtrovat v pomocí indexů.

6.2 Množství výstupních záznamů

V této sekci se zaměříme na to, jak vplývá množství výstupních záznamů na dobu vyřízení požadavku. Pro měření jsme požadavky posílali na cestu

```
http://127.0.0.1:8002/satram/cluster/1425340800/to/X
```

kde X je parametr který jsme měnili. Tělo požadavků bylo

```
{ "info":["Start_time","clstrSize"
,"Acq_time","clstrLET"], "filters":
[ { "attribute":"Acq_time", ">=":20} ] }
```

V tomto nastavení se vždy projdou všechny záznamy, ze všech dostupných datových souborů v časovém rozsahu daném parametry cesty. Ve výsledku budou bude množství záznamů lineárně závislé na délce časového rozsahu, za předpokladu, že jsou snímky rovnoměrně rozděleny v čase. Výsledky měření jsou v tabulce 6.2

počet hodin	1	2	5	10	15	20	25
čas (ms)	141	176	288	555	1642	2298	2714

Tabulka 6.3: Vliv množství výstupních záznamů na čas

Doba vyřízení požadavku, dle měření, stoupá zhruba dvakrát rychleji než by měl stoupat počet výstupních záznamů. Tento fakt je zřejmě spojený s vysokou režií pří vytváření uzlů v JSON, který je výstupem.

6.3 Množství filtrovaných větví

V této sekci se zaměříme na to, jak vplývá množství filtrovaných větví tj. množství větví, pomocí kterých filtrujeme záznamy, na dobu vyřízení požadavku. Pro měření jsme požadavky posílali na cestu

```
http://127.0.0.1:8002/satram/cluster/1425340800/to/1425344400
```

tělo požadavků bylo

```
{ "info":["Start_time","clstrSize",
"Acq_time","clstrLET"], "filters":[ X ] }
```

Kde za X jsme dosazovaly postupně takové filtry, které nevyfiltrují žádné záznamy.

V tomto nastavení se vždy projdou všechny záznamy, ze všech dostupných datových souborů v časovém rozsahu daném parametry cesty. Ve výsledku bude množství záznamů vždy stejné. Výsledky měření jsou v tabulce 6.3

počet filtrů	1	2	3	4	5
čas (ms)	141	160	180	195	202

Tabulka 6.4: Vliv množství filtrovaných větví na čas

Doba vyřízení požadavku, dle měření, stoupá pomaleji než množství větví, pomocí kterých filtrujeme záznamy. To znamená, že když omylem přidáme nějaké filtry navýše, které nebudou účinné, výrazně tím nezhoršíme výkon systému.

6.4 Množství výstupních větví

V této sekci se zaměříme na to, jak vplývá množství výstupních větví tj. množství větví, ze kterých požadujeme data, na dobu vyřízení požadavku. Pro měření jsme požadavky posílali na cestu

```
http://127.0.0.1:8002/satram/cluster/1425340800/to/1425344400
```

tělo požadavků bylo

```
{ "info":[ X ], "filters":
[ { "attribute":"Acq_time", ">=":20}   ] }
```

Kde za X jsme dosazovaly postupně více a více větví.

V tomto nastavení se vždy projdou všechny záznamy, ze všech dostupných datových souborů v časovém rozsahu daném parametry cesty. Ve výsledku bude množství záznamů vždy stejné. Výsledky měření jsou v tabulce 6.4

počet větvý	1	2	3	4	5
čas (ms)	89	114	149	168	179

Tabulka 6.5: Vliv množství výstupních větví na čas

Doba vyřízení požadavku, dle měření, stoupá zhruba výrazné pomaleji než množství větví, ze kterých požadujeme data. To znamená, že když omylem přidáme nějaké větve navýše, které nepotřebujeme, téměř tím nezhoršíme výkon systému.

6.5 Formát výstupu

V této sekci se zaměříme na to, jak vplývá formát výstupu na dobu vyřízení požadavku. Porovnáme výstupy v JSON a v rootfile. Pro měření jsme požadavky posílali na cesty

```
http://127.0.0.1:8002/satram/cluster/1425340800/to/X
```

a

```
http://127.0.0.1:8002/satram/cluster/1425340800/to/X/asrootfile
```

kde X je parametr který jsme měnili. Tělo požadavků bylo

```
{ "info":["Start_time","clstrSize"
,"Acq_time","clstrLET"], "filters":
[ { "attribute":"Acq_time", ">=":20} ] }
```

V tomto nastavení se vždy projdou všechny záznamy, ze všech dostupných datových souborů v časovém rozsahu daném parametry cesty. Ve výsledku budou bude množství záznamů lineárně závislé na délce časového rozsahu, za předpokladu, že jsou snímky rovnoměrně rozděleny v čase. Výsledky měření jsou v tabulce 6.5

počet hodin	1	2	5	10	15	20	25
čas JSON (ms)	141	176	288	555	1642	2298	2714
čas rootfile (ms)	178	264	313	610	840	1053	1329

Tabulka 6.6: Vliv formátu výstupu na čas

Doba vyřízení požadavku, dle měření, je u výstupů ve formátu rootfile nejprve vyšší než u JSON, ale následně stoupá mnohem pomaleji. Z toho vyplývá, že výstup ve formě JSON je více vhodný pro vizualizaci dat a výstup ve formě rootfile je více vhodný pro analýzu dat.

6.6 API vzniklé konfigurací

V této sekci porovnáme výkon přímo napsaného API s API, které vzniklo pomocí konfigurace.

Pak jsme jsme se začali měnit počet dní pomocí parametrů v cestě. Pro měření jsme požadavky posílali na cesty

```
/satram/cluster/1425254400/to/X
```

a

```
/satram_generate/cluster/1425254400/to/X
```

kde X je parametr, který jsme měnili. Tělo požadavků bylo

```
{ "info":["Start_time","clstrSize","Acq_time"
,"clstrLET"], "filters":[ { "attribute":"Start_time"
, "==":1425254550} ] }
```

V tomto nastavení se vždy projdou všechny záznamy, ze všech dostupných datových souborů v časovém rozsahu daném parametry cesty. Ve výsledku budou vždy informace o právě jednom snímku. Výsledky měření jsou v tabulce 6.6

počet dní	1	2	5	10	15	20	25
čas přímé (ms)	77	106	176	297	443	526	621
čas z konfigurace (ms)	1239	2258	4820	8819	12620	16774	19995

Tabulka 6.7: Porovnání výkonu přímo napsaného API a API, které vzniklo pomocí konfigurace

Doba vyřízení požadavku, dle měření, je u API, které vzniklo pomocí konfigurace výrazně další. Důvodem jsou lepší možnosti optimalizace běhu v přímo napsaném API. API, které vzniklo pomocí konfigurace se tedy hodí pouze jako dočasné řešení, které bude nahrazené přímo napsaným API.

Závěr

V této práci se nám podařilo analyzovat, navrhnout a implementovat informační systém pro zpracování a ukládání dat z vědeckého přístroje SATRAM. Během toho smě také splnili všechny dílčí cíle, a to:

- Analyzovat vědecké a navigační údaje pocházející z přístroje SATRAM
- Analyzovat možnosti ukládaní dat, zejména prostudovat systém ROOT vyvinutý v CERN a vybrat vhodný způsob ukládaní dat
- Analyzovat a navrhnou GUI
- Prostudovat software Pixelman[3]
- Navrhnout architekturu informačního systému pro zpracování a ukládání dat z vědeckého přístroje SATRAM
- Navrhnout databázový systém pro ukládání dat
- Implementovat databázový systém pro ukládání dat
- Otestovat databázový systém pro ukládání dat
- Vytvořit dokumentaci

Největším zklamáním byl výkon API, které vzniklo pomocí konfigurace. Takto vytvořené API je je nevýkonné a nedá se v praxi používat jinak jako nouzové řešení.

Systém je funkční, avšak stále existuje mnoho částí, které by bylo dobré optimalizovat nebo vylepšit. Do budoucna by bylo vhodné vytvořit přívětivější rozhraní pro vytváření nových modulů, vzhledem k neúspěchu konceptu API, které vznikne pomocí konfigurace. Dále by bylo vhodné lépe optimalizovat proces dotazování na data a to zejména přidáním dalších indexů.

Literatura

[1] Granja, C.; Polansky, S.; Vykydal, Z.; aj.: The {SATRAM} Timepix spacecraft payload in open space on board the Proba-V satellite for wide range radiation monitoring in {LEO} orbit. *Planetary and Space Science*, ročník 125, 2016: s. 114 – 129, ISSN 0032-0633, doi:http://dx.doi.org/10.1016/j.pss.2016.03.009. Dostupné z: `http://www.sciencedirect.com/science/article/pii/S0032063316300216`

[2] CERN: *ROOT [online]*. [cit. 2016-01-14]. Dostupné z: `https://root.cern.ch/`

[3] Turecek, D.; Holy, T.; Jakubek, J.; aj.: Pixelman: a multi-platform data acquisition and processing software package for Medipix2, Timepix and Medipix3 detectors. *Journal of Instrumentation*, ročník 6, č. 01, 2011: str. C01046. Dostupné z: `http://stacks.iop.org/1748-0221/6/i=01/a=C01046`

[4] Valenta, M.: Pokročilé databázové systémy. 2016. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-PDB/`

[5] solidIT consulting & software development gmbh: *DB-Engines Ranking [online]*. [cit. 2016-01-23]. Dostupné z: `http://db-engines.com/en/ranking`

[6] Valenta, M.: Databázové systémy. 2016. Dostupné z: `https://edux.fit.cvut.cz/courses/BI-DBS/`

[7] Barry & Associates, Inc.: *Object-Oriented Database Management System (OODBMS) Definition [online]*. [cit. 2016-02-11]. Dostupné z: `http://www.service-architecture.com/articles/object-oriented-databases/object-oriented_database_oodbms_definition.html`

[8] Abadi, D. J.; Madden, S. R.; Hachem, N.: Column-stores vs. Row-stores: How Different Are They Really? In *Proceedings of the 2008 ACM SIG-*

MOD International Conference on Management of Data, SIGMOD '08, New York, NY, USA: ACM, 2008, ISBN 978-1-60558-102-6, s. 967–980, doi:10.1145/1376616.1376712. Dostupné z: http://doi.acm.org/10.1145/1376616.1376712

[9] Abadi, D.; Boncz, P.; Harizopoulos, S.; aj.: The Design and Implementation of Modern Column-Oriented Database Systems. *Foundations and Trends® in Databases*, ročník 5, č. 3, 2013: s. 197–280, ISSN 1931-7883, doi:10.1561/1900000024. Dostupné z: http://dx.doi.org/10.1561/1900000024

[10] Robinson, I.; Webber, J.; Eifrem, E.: *Graph Databases, 2nd Edition*. O'Reilly Media, 2015.

[11] Kumar, R.; Tripathi, A.: ROOT: A Data Analysis and Data Mining Tool from CERN [online]. 2008, [cit. 2016-02-12]. Dostupné z: https://www.casact.org/pubs/forum/08wforum/kumar_tripathi.pdf

[12] Holy, T.; Heijne, E.; Jakubek, J.; aj.: Pattern recognition of tracks induced by individual quanta of ionizing radiation in Medipix2 silicon detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, ročník 591, č. 1, 2008: s. 287 – 290, ISSN 0168-9002, doi:http://dx.doi.org/10.1016/j.nima.2008.03.074, radiation Imaging Detectors 2007Proceedings of the 9th International Workshop on Radiation Imaging Detectors. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0168900208004592

[13] Žikovský, P.: Návrh uživatelského rozhraní. 2016. Dostupné z: https://edux.fit.cvut.cz/courses/MI-NUR/

[14] The PostgreSQL Global Development Group: *PostgreSQL [online]*. [cit. 2016-02-13]. Dostupné z: https://www.postgresql.org/

[15] The PostgreSQL Global Development Group: *libpqxx [online]*. [cit. 2016-02-13]. Dostupné z: http://pqxx.org/development/libpqxx/

[16] Facebook: *Proxygen [online]*. [cit. 2016-02-13]. Dostupné z: https://code.facebook.com/posts/1503205539947302/

[17] Tencent: *RapidJSON [online]*. [cit. 2016-02-15]. Dostupné z: http://rapidjson.org/

[18] *TinyXML-2 [online]*. [cit. 2016-04-15]. Dostupné z: http://www.grinninglizard.com/tinyxml2/

Příloha **A**

Seznam použitých zkratek

API Application programming interface

CERN Evropská organizace pro jaderný výzkum

DAC Digital to analog converter

GPS Globální poziční systém

GUI Graphical user interface

HTTP Hyper text transfer protocol

JSON Javascript object notation

LHC Large hadron collider

OQL Object query language

SATRAM Space application of timepix based radiation monitor

SQL Structured query language

URL Uniform resource locator

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `data` ..... adresář se vzorovými daty
  - `rawdata` ..... vzorové vstupní data
  - `rootfile` ..... vzorové rootfile soubory
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `API-description.pdf` ... popis API pro projekt SATRAM ve formátu PDF
  - `configuration.xml` .... vzorový konfigurační soubor ve formátu XML
  - `instalation-guide.pdf` ..... instalační příručka ve formátu PDF
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS