**UNIVERZITA MATEJA BELA V BANSKEJ BYSTRICI**
**FAKULTA PRÍRODNÝCH VIED**

# ANALÝZA ÚDAJOV PRE MANAŽOVANIE ÚLOH VO VYSOKOVÝKONNOM POČÍTANÍ

**DIPLOMOVÁ PRÁCA**

**f67d3d9b-989b-4157-a41c-9c2042cd915d**

**2018** **Bc. Martin Trnik**

**UNIVERZITA MATEJA BELA V BANSKEJ BYSTRICI**
**FAKULTA PRÍRODNÝCH VIED**

# ANALÝZA ÚDAJOV PRE MANAŽOVANIE ÚLOH VO VYSOKOVÝKONNOM POČÍTANÍ

Diplomová práca

**f67d3d9b-989b-4157-a41c-9c2042cd915d**

Študijný program: Aplikovaná informatika
Študijný odbor: Aplikovaná informatika
Pracovisko: Katedra informatiky
Vedúci diplomovej práce: doc. Ing. Jarmila Škrinárová, PhD.

**Banská Bystrica 2018** **Bc. Martin Trnik**


Univerzita Mateja Bela v Banskej Bystrici
Fakulta prírodných vied

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Bc. Martin Trnik
**Študijný program:** aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
**Študijný odbor:** aplikovaná informatika
**Typ záverečnej práce:** Magisterská záverečná práca
**Jazyk záverečnej práce:** slovenský
**Sekundárny jazyk:** anglický

**Názov:** Analýza údajov pre manažovanie úloh vo vysokovýkonnom počítaní

**Anotácia:** Analyzujte výpočtové zdroje a úlohy vo vysokovýkonnom počítačovom systéme. Definujte optimalizačné kritériá pre rozvrhovanie úloh. Definujte základné ciele a požiadavky, v súvislosti so znižovaním spotreby systému. Analyzujte a zbierajte údaje pre manažovanie úloh. Navrhnite a implementujte modely analýzy údajov s cieľom znižovať spotrebu vysokovýkonného klastra. Navrhnite metodiku overovania, urobte testovanie, špecifikujte a porovnajte dosiahnuté výsledky.

**Vedúci:** doc. Ing. Jarmila Škrinárová, PhD.
**Katedra:** KIN FPV - Katedra informatiky
**Vedúci katedry:** RNDr. Miroslav Melicherčík, PhD.

**Dátum zadania:** 10.03.2017

**Dátum schválenia:** 15.03.2017

prof. RNDr. Roman Nedela, DrSc.
garant študijného programu

## Čestné vyhlásenie

Vyhlasujem, že som celú diplomovú prácu vypracoval samostatne pod vedením vedúcej práce doc. Ing. Jarmily Škrinárovej, PhD. s použitím uvedenej odbornej literatúry.

Banská Bystrica, 26. apríl 2018 .....

*vlastnoručný podpis*

## Poďakovanie

Ďakujem vedúcemu práce, doc. Ing. Jarmile Škrinárovej, PhD. za odbornú pomoc a prínosné konzultácie, ktoré mi poskytovala pri vypracovávaní tejto diplomovej práce.

Za sprístupnenie údajov o výpočtových prostriedkoch vysokovýkonného počítačového klastra Univerzity Mateja Bela v Banskej Bystrici a o vykonávaných úlohách na uvedenom klastri ďakujem jeho správcovi Mgr. Jozefovi Siláčimu.

# ABSTRAKT

TRNIK, Martin Bc.: Analýza údajov pre manažovanie úloh vo vysokovýkonnom počítaní. [Diplomová práca] / Martin Trnik. - Univerzita Mateja Bela v Banskej Bystrici. Fakulta prírodných vied; Katedra informatiky. - Vedúci: doc. Ing. Jarmila Škrinárová, PhD. - Stupeň odbornej kvalifikácie: Magister. - Banská Bystrica : Fakulta prírodných vied UMB, 2018. 77 s.

Predmetom skúmania diplomovej práce je analýza údajov pre manažovanie úloh vo vysokovýkonnom počítaní. Ciele práce sú zbierať a analyzovať údaje o výpočtových zdrojoch a úlohách vo vysokovýkonnom počítačovom systéme, definovať optimalizačné kritériá pre rozvrhovanie úloh, na základe analýzy údajov navrhnúť a implementovať model spotreby elektrickej energie a optimalizačný model rozvrhovania úloh na vysokovýkonnom počítačovom klastri Univerzity Mateja Bela, navrhnúť metodiku overovania modelov a porovnať dosiahnuté výsledky. V teoretickej časti práce sa nachádza analýza všeobecných charakteristík vysokovýkonných počítačových systémov so zameraním na výpočtové zdroje, záťaž výpočtového systému a kritériá rozvrhovania úloh na výpočtové zdroje. Ďalej sa v teoretickej časti práce vyskytuje charakteristika techník a postupov slúžiacich na znižovanie spotreby elektrickej energie výpočtových systémov. Teoretická časť práce obsahuje aj opis technológií použitých pri vykonávaní analýz údajov. V praktickej časti sa nachádzajú charakteristiky skúmaného vysokovýkonného počítačového klastra, podrobný opis údajov, ktoré boli v práci analyzované, návrh a implementácia aplikácie pre analýzu údajov a zhodnotenie výsledkov. V práci boli použité aktuálne nástroje pre riešenie problémov tohto typu - distribuovaná služba Google BigQuery, určená na interaktívne dátové analýzy nad veľkými údajmi, softvérová knižnica Google Optimization Tools, určená na optimalizáciu zložitých problémov a Business Intelligence nástroj Tableau. Použité údaje pochádzajú z monitorovacích a plánovacích nástrojov, ktoré sú využívané na vysokovýkonnom počítačovom klastri UMB. Kľúčovým prínosom práce je zistenie, aké množstvo elektrickej energie by sa dalo ušetriť upravením plánovacích algoritmov podľa princípov zeleného počítania.

**Kľúčové slová:** Rozvrhovanie úloh. Spotreba elektrickej energie. Optimalizácia. Google BigQuery. OR-Tools.

# ABSTRACT

TRNIK, Martin, Bc. Managing tasks data analysis for high performance computing. [Master thesis] / Martin Trnik. – Matej Bel University, Banská Bystrica. Faculty of Natural Sciences; Department of Informatics. – Supervisor: doc. Ing. Jarmila Škrinárová, PhD. – Qualification degree: Master (Mgr.) – Banská Bystrica: Faculty of Natural Sciences, MBU, 2018. 77 p.

The master thesis is focused on data analysis for task management in high-performance computing. The aim of the master thesis is to collect and analyse data about computational resources and jobs in high-performance computer system, to define optimization criteria for job scheduling, to design and implement a model of electric energy consumption and an optimization model of job scheduling on Matej Bel University's high-performance computing cluster, verify models and compare the results achieved. In the theoretical part of the thesis there is an analysis of general characteristics of high performance computer systems focusing on computational resources, computing system load and job scheduling criteria. Further, in the theoretical part, there is a description of techniques and procedures used to reduce the electrical energy consumption of computational systems. The theoretical part of the thesis also contains a description of the technologies used in the practical part of thesis. The practical part contains the characteristics of the high-performance computer cluster, a detailed description of the data analyzed in the thesis, the design and implementation of the application for data analysis and evaluation of the results. Current tools for resolving issues of this type are used - distributed service Google BigQuery for interactive data analysis over large data, the Google Optimization Tools software library used for optimization of complex issues, and Tableau Business Intelligence tool. The used data comes from the monitoring and scheduling tools used on the MBU high-performance computer cluster. The key benefit of the work is to find out how much electrical energy could be saved by modifying planning algorithms according to green computing principles.

**Keywords:** Job scheduling. Power consumption. Optimization. Google BigQuery. OR-Tools.

# PREDHOVOR

Vysokovýkonné počítačové systémy sa v posledných rokoch, hlavne vďaka cloudovým službám, začali dostávať aj do povedomia laickej verejnosti. Cloudové služby sa stávajú čoraz bežnejšou súčasťou používania personálnych počítačov. Každé používateľské konto vytvorené v najnovšej edícií operačného systému Windows má priradené cloudové úložisko. Každé vytvorené „Gmailové“ konto takisto.

Poskytovatelia cloudových služieb si s nárastom ich používania začali postupne uvedomovať aj jednu negatívnu stránku presúvania IT sveta do cloudov. Ide o vysoký nárast spotreby elektrickej energie. Ten je pre nich nepríjemný hlavne preto, lebo zvyšuje ich prevádzkové náklady. Sekundárne takisto preto, že prispieva k zhoršovaniu životného prostredia, keďže zatiaľ len pomerne malé percento elektrickej energie pochádza z obnoviteľných zdrojov.

Preto sa začala rozvíjať iniciatíva environmentálne udržateľných výpočtov „Zelené počítanie“. Cieľom tejto iniciatívy je presadzovať návrhy, ktorých úlohou je znižovanie spotreby elektrickej energie IT sektora. V súlade s iniciatívou zeleného počítania sme sa rozhodli analyzovať údaje pre manažovanie úloh vysokovýkonného počítačového klastra na Univerzite Mateja Bela v Banskej Bystrici a hľadať možnosti zníženia spotreby elektrickej energie zmenou rozvrhovania úloh na jeho výpočtové uzly. Cieľom práce je určiť vzťah medzi záťažou na výpočtových uzloch a spotrebou elektrickej energie a zistiť, či je možné ušetriť elektrickú energiu upravením rozvrhovacích algoritmov.

K téme práce ma motivovala vedúca práce doc. Ing. Jarmila Škrinárová, PhD., ktorej by som sa chcel poďakovať za podnetné pripomienky a dôležité otázky, na ktoré väčšinou nebolo jednoduché odpovedať, ale celú prácu posúvali na vyššiu úroveň. Takisto by som sa chcel poďakovať správcovi vysokovýkonného počítačového klastra UMB Mgr. Jozefovi Siláčimu za spoluprácu, ochotu pomôcť a odpovedať na množstvo otázok, ktoré sa pri tvorení práce vyskytli.

# OBSAH

# ZOZNAM TABULIEK



# ZOZNAM PSEUDOKÓDOV

# ZOZNAM ILUSTRÁCIÍ

# ZOZNAM SKRATIEK A ZNAČIEK

| Skratka | Celé pomenovanie | Význam |
|---|---|---|
| API | Application Programming Interface | Aplikačné programové rozhranie |
| AWS | Amazon Web Services | Cloudová platforma spoločnosti Amazon |
| csv | Comma Separated Value | Formát textových súborov, v ktorých sú hodnoty oddelené pomocou čiarky |
| CUDA | Compute Unified Device Architecture | Paralelná výpočtová architektúra pre grafické výpočtové akcelerátory |
| DIMS | Dynamic Infrastructure Management System | Model riadenia dynamickej architektúry |
| DVFS | Dynamic Voltage / Frequency Scaling | Dynamická zmena napätia alebo frekvencie |
| FK | Foreign Key | Cudzí kľúč |
| GB | Gigabyte | $10^9$ Bajtov |
| GCP | Google Cloud Platform | Cloudová platforma spoločnosti Google |
| GHz | Gigahertz | $10^9$ Hertzov |
| GPU | Graphics Processing Unit | Grafický výpočtový akcelerátor |
| HPC | High Performance Computing | Vysokovýkonné počítanie |
| HPCC UMB | High Performance Computing Centre UMB | Centrum pre vysokovýkonné počítanie Univerzity Mateja Bela |
| HT | Hyperthreading | Virtualizačná technológia, pri využití ktorej sa jeden procesor aplikáciám javí ako dva |
| HTML | HyperText Markup Language | Hypertextový značkovací jazyk |
| IaaS | Infrastructure as a Service | Infraštruktúra ako služba |
| IT | Information Technology | Informačné technológie |
| ITEP | IT Equipment Power | Spotrebovaná elektrická energia IT zariadenia |
| LTS | Long Time Support | Typ operačného systému Ubuntu s rozšírenou podporou na dlhšie časové obdobie |
| MIP | Mixed Integer Programming | Celočíselná optimalizácia |
| OR-Tools | Google Optimization Tools | Optimalizačné nástroje spoločnosti Google |
| P-value | Probability value | Pravdepodobnostná hodnota |
| PaaS | Platform as a Service | Platforma ako služba |
| PBS | Portable Batch System | Softvér slúžiaci na riadenie zdrojov |
| PK | Primary Key | Primárny kľúč |
| PUE | Power Usage Effectiveness | Účinnosť využitia elektrickej energie |
| RAM | Random Access Memory | Operačná pamäť |
| REST API | Representational State Transfer | Architektúra, ktorá umožňuje výmenu informácií pomocou protokolu HTTP |

| | | |
|---|---|---|
| SaaS | Software as a Service | Softvér ako služba |
| TB | Terabyte | $10^{12}$ Bajtov |
| TFLOP | Tera Floating Point Operations Per Second | $10^{12}$ operácií s pohyblivou desatinnou čiarkou za sekundu |
| TFP | Total Facility Power | Celková spotrebovaná elektrická energia zariadenia |
| TORQUE | Terascale Open-source Resource and QUEue Manager | Softvér slúžiaci na riadenie zdrojov |
| UMB | Matej Bel University | Univerzita Mateja Bela |
| WRMS | Workload and Resource Management System | Model riadenia pracovnej záťaže a zdrojov |

# ÚVOD

Analyzovanie údajov o vysokovýkonných počítačových systémoch v reálnom čase je vo svete IT bežnou praxou. Systémovým administrátorom ponúka podrobný prehľad o funkčnosti jednotlivých súčastí systému a o ich využití používateľskými a systémovými úlohami. Nahromadené údaje z monitorovacích systémov sa ale postupom času stávali zbytočnou príťažou, keďže ich bolo veľké množstvo a pre systémových administrátorov boli bezcenné, keďže už neboli aktuálne.

Rozvoj IT v oblasti spracovania nahromadených údajov takýmto údajom vrátil hodnotu. Dátoví analytici dokážu spracovať údaje nahromadené za dlhšie časové obdobie a nájsť v nich závislosti a trendy, ktoré nie sú na prvý pohľad viditeľné. Jedným z trendov, ktoré boli týmto spôsobom preukázané je, že vysokovýkonné počítačové systémy väčšinu času pracujú pri nízkej záťaži. Ukázalo sa, že táto skutočnosť znižuje energetickú efektívnosť týchto systémov, pretože väčšie množstvo slabo zaťažených výpočtových prostriedkov spravidla spotrebuje viac elektrickej energie ako menšie množstvo „normálne“ zaťažených výpočtových prostriedkov. Nižšia energetická efektívnosť vysokovýkonných počítačových systémov spôsobuje ich prevádzkovateľom zvýšené prevádzkové náklady. Vysoká spotreba elektrickej energie je aj environmentálny problém, keďže len relatívne malé percento elektrickej energie je momentálne vyrábané z obnoviteľných zdrojov energie.

Aktuálnu situáciu v oblasti spotreby elektrickej energie vysokovýkonných počítačových systémov najpodrobnejšie zachytáva správa „*Power Management Techniques for Data Centers: A Survey“,* ktorú v roku 2014 pre *Oak Ridge National Laboratory* vyhotovil *Sparsh Mittal*. [1] Uvádza v nej, že približne 6000 dátových centier, serverov a superpočítačov v Spojených štátoch Amerických v roku 2006 spotrebovalo zhruba 61 miliárd kilowatt-hodín elektrickej energie. Množstvo spotrebovanej elektrickej energie dátovými centrami, servermi a superpočítačmi v Spojených štátoch Amerických rastie každoročne o zhruba 12 %. Okamžitá spotreba elektrickej energie najvýkonnejších superpočítačov zo zoznamu TOP500 [2] sa v špičke pohybuje v desiatkach megawattov, čo zhruba zodpovedá mestu s 40 000 obyvateľmi.

Zo všetkých uvedených informácií vyplýva, že v oblasti energetickej efektívnosti vysokovýkonných počítačových systémov je možné dosiahnuť významné pokroky a preto je výskum v tejto oblasti veľmi dôležitý.

V tejto diplomovej práci sme sa pripojili k výskumu energetickej efektívnosti vysokovýkonných počítačových systémov tak, že sme sa rozhodli zbierať a analyzovať údaje o manažovaní úloh vysokovýkonného počítačového klastra Univerzity Mateja Bela. Cieľom teoretickej časti práce je definovať základné ciele a požiadavky pre znižovanie spotreby vysokovýkonného počítačového systému. Praktická časť má za cieľ analyzovať vzťah medzi záťažou výpočtových uzlov vysokovýkonného klastra UMB a ich spotrebou elektrickej energie. Ďalej navrhnúť a implementovať modely analýzy údajov s cieľom znižovať spotrebu vysokovýkonného klastra UMB. Následne navrhnúť metodiku overovania, vykonať porovnanie a zhodnotiť dosiahnuté výsledky.

Relevantným výskumom sa zaoberá aj práca „*Energy Efficiency for Large-Scale MapReduce Workloads with Significant Interactive Analysis*“, v ktorej pracovníci univerzity Berkeley, California v spolupráci s pracovníkmi spoločnosti Facebook opisujú, ako optimalizovaním rozvrhovania úloh s ohľadom na spotrebu elektrickej energie pri zachovaní dostupnosti a kvality služieb znížili energetickú náročnosť pri niektorých typoch úloh o 40 až 50 %.

Naša práca nadväzuje na práce „*Programovacia paradigma MapReduce pre prácu s Big Data*“, [3] „*Mnohorozmerná analýza rozvrhovania vo vysokovýkonných počítačových systémoch*“, [4] „*Analýza činiteľov ovplyvňujúcich spotrebu elektrickej energie klastra pre vysokovýkonné výpočty*“ [5] a „*Model optimalizácie rozvrhu úloh klastra pre HPCC s ohľadom na spotrebu elektrickej energie*“, [6] ktoré sme prezentovali na rôznych študentských vedeckých konferenciách na Slovensku a v Českej Republike počas štúdia.

Práca je štruktúrovaná do 7 hlavných kapitol. Prvá kapitola predstavuje problematiku vysokovýkonných počítačových systémov so zameraním na výpočtové zdroje, ich záťaž, vykonávané úlohy a kritériá rozvrhovania úloh. V druhej kapitole sa nachádza charakteristika rôznych stratégií, ktoré zlepšujú energetickú efektívnosť počítačových systémov. Tretia kapitola obsahuje opis technológií, ktoré využívame v praktickej časti práce. Vo štvrtej kapitole charakterizujeme skúmaný klaster a údaje, ktoré zbierame. Piata kapitola predstavuje návrh aplikácie pre analýzu spotreby elektrickej energie. V šiestej kapitole opisujeme implementáciu analýzy a v poslednej kapitole opisujeme dosiahnuté výsledky.

# 1 VYSOKOVÝKONNÝ POČÍTAČOVÝ SYSTÉM

Pojem vysokovýkonné počítanie (angl. *High Performance Computing*, HPC) sa používa na opis výpočtových prostredí, ktoré využívajú superpočítače a počítačové klastre na: [7]

- riešenie zložitých výpočtových problémov,
- podporu aplikácií s vysokými nárokmi na výpočtový čas,
- na spracovanie značného množstva údajov.

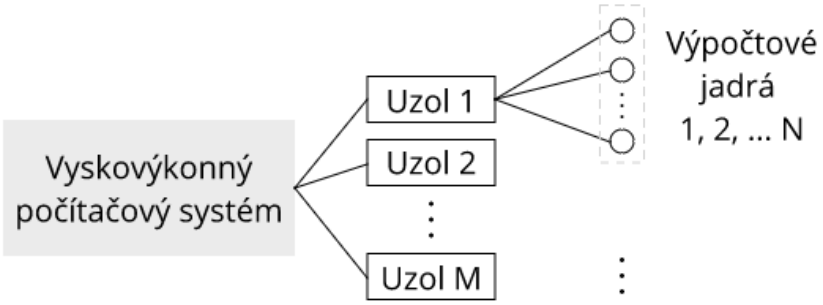
Vysokovýkonné počítačové systémy sa historicky považovali za nástroj na vedecké výpočty a výpočtovo náročné úlohy, ale postupom času sa stali vhodnými aj pre aplikácie s vysokým množstvom spracúvaných údajov. Využívajú vysoký stupeň vnútorného paralelizmu a špecializované multiprocesory s vlastnými pamäťovými architektúrami, ktoré sú optimalizované pre numerické výpočty. [8] Základom každého vysokovýkonného výpočtového systému sú zdroje, ktorými disponuje.

## 1.1 Zdroje vo vysokovýkonnom počítačovom systéme

Zdroj (angl. *resource*) je fyzický alebo virtuálny prvok počítačového systému s limitovanou veľkosťou úložného priestoru a obmedzeným množstvom procesorov. Podľa úlohy, ktorú zdroj v systéme vykonáva rozdeľujeme zdroje najmä na: [9]

- výpočtové zdroje (tiež sa používajú pomenovania uzly, prostriedky), ktoré slúžia na vykonanie výpočtovej úlohy,
- zdroje pre vstupno-výstupné operácie.

V práci sa zameriavame na výpočtové zdroje. Na obrázku 1.1 môžeme vidieť základnú schému výpočtových zdrojov vo vysokovýkonnom počítačovom systéme.




Obrázok 1.1 Schéma výpočtovej časti vysokovýkonného počítačového systému

Zdroje vo vysokovýkonnom počítačovom systéme je potrebné riadiť. Existujú dva základné modely prístupu k riadeniu zdrojov: [9]

- model riadenia pracovnej záťaže a zdrojov (angl. *Workload and Resource Management System*, WRMS),
- model riadenia dynamickej infraštruktúry (angl. *Dynamic Infrastructure Management System*, DIMS).

### 1.1.1 Model riadenia pracovnej záťaže a zdrojov

WRMS sa štandardne používa v gridovom počítaní a využíva fyzické zdroje. Používateľ zasiela úlohu na spracovanie v podobe opisu (angl. *job description*, pozri kapitolu 1.3). Systém naplánuje spracovanie úlohy tak, aby splnil požiadavky zadané v opise úlohy. Model WRMS zabezpečuje tri základné druhy činností: [9]

- správu výpočtových zdrojov,
- správu úloh,
- rozvrhovanie úloh na výpočtové zdroje.

### 1.1.2 Model riadenia dynamickej infraštruktúry

DIMS sa typicky využíva v cloudovom počítaní a je založený na poskytovaní infraštruktúry ako služby (angl. *Infrastructure as a Service*, IaaS). Používateľovi je na základe požiadaviek, ktoré do systému zadá pridelený virtuálny stroj, s ktorým môže pracovať ako s izolovaným fyzickým strojom. [9]

## 1.2 Záťaž výpočtových zdrojov vo vysokovýkonných počítačových systémoch

„*Každý výpočtový zdroj pracuje určitou rýchlosťou. Rýchlosť zdroja určuje čas, ktorý je potrebný na vykonanie určitej úlohy na tomto zdroji. Záťaž zdroja reprezentuje jeho využitie v danom časovom intervale. Záťaž zdroja sa v priebehu jeho činnosti spravidla mení.*“ [9]

Záťaž výpočtového systému je z prostredia operačných systémov založených na jadre Linux známa ako parameter *load*. Ide o reálne číslo patriace do intervalu $\langle 0, \infty \rangle$. Na obrázku 1.2 vidíme, že štandardne sa uvádza ako trojčíslie priemerov (*load averages*).

```
~ $ uptime
23:05 up 14 days, 6:08, 7 users, load averages: 0.65 0.42 0.36
```

Obrázok 1.2 Ukážka reprezentácie záťaže výpočtového systému [10]

Prvé číslo z trojčíslia určuje priemernú záťaž výpočtového systému za ostatnú minútu, druhé číslo z trojčíslia priemernú záťaž za ostatných 5 minút a posledné číslo

priemernú záťaž za ostatných 15 minút prevádzky výpočtového systému. Vzorkovacia frekvencia určovania záťaže je štandardne 5 sekúnd, ale je možné ju konfigurovať. [10]

Vysvetliť výpočet tohto parametru nie je jednoduché. Preto si pomôžeme ľahšie predstaviteľnou analógiou z bežného života. Procesor s jedným výpočtovým jadrom môžeme chápať ako jednoprúdovú cestu. Predstavme si, že sme správca mosta. Potrebujeme zrozumiteľne vyjadriť aktuálnu dopravnú situáciu (záťaž) na moste. Zmysluplným číselným vyjadrením aktuálnej dopravnej záťaže je počet vozidiel prechádzajúcich mostom spolu s počtom vozidiel čakajúcich na vstup na most v aktuálnom okamihu v dopravnej zápche. Ak v dopravnej zápche nečakajú žiadne vozidlá, prichádzajúci vodiči vedia, že cez most môžu prejsť bez zdržania. Naopak, ak vozidlá stoja v dopravnej zápche, tak aj prichádzajúci vodiči vedia, že sa zdržia. Číselne by sme to mohli vyjadriť napríklad takto: [10]

- dopravná záťaž 0,00 znamená, že na moste, ani v dopravnej zápche nie sú žiadne vozidlá,
- dopravná záťaž 1,00 znamená, že kapacita mostu je vyčerpaná, ale v dopravnej zápche nestoja žiadne vozidlá,
- dopravná záťaž 2,00 znamená, že kapacita mostu je vyčerpaná a v dopravnej zápche čaká toľko vozidiel, koľko práve prechádza mostom.

Pre lepšiu ilustráciu uvádzame obrázky1.3, 1.4 a 1.5.

Obrázok 1.3 Ilustrácia dopravnej záťaže 1,00 [10]

Obrázok 1.4 Ilustrácia dopravnej záťaže 0,50 [10]

Obrázok 1.5 Ilustrácia dopravnej záťaže 1,70 [10]

Princíp výpočtu záťaže výpočtového systému s jedným výpočtovým jadrom je rovnaký. Vozidlá prechádzajúce most sú aktuálne vykonávané procesy používajúce procesor výpočtového systému. Vozidlá čakajúce v dopravnej zápche sú procesy,

ktoré čakajú na začiatok svojho vykonávania v rade plánovaných procesov. Počet procesov v týchto dvoch stavoch určuje aktuálnu záťaž výpočtového systému s jedným výpočtovým jadrom. [10]

Analógiu je možné rozšíriť aj na výpočtové systémy s viacerými výpočtovými jadrami tak, že si most predstavíme ako viacprúdový. Je potrebné si uvedomiť, že pri takto definovanom výpočte záťaže je záťaž 1,00 vo výpočtovom systéme s dvomi výpočtovými jadrami ekvivalentná záťaži 0,50 vo výpočtovom systéme s jedným výpočtovým jadrom. [10]

### 1.2.1 Ideálna záťaž výpočtového zdroja

V predchádzajúcej časti sme vysvetlili pojem záťaž výpočtového systému. Vo vysokovýkonných počítačových systémoch, ktoré obsahujú množstvo výpočtových zdrojov je možné sledovať záťaž nielen systému ako celku, ale osobitne záťaž každého výpočtového zdroja, ktorý obsahuje výpočtové jadrá (pozri kapitolu 1.1).

Určiť ideálne zaťaženie výpočtového zdroja je skoro filozofická otázka. Pre jednojadrový výpočtový zdroj je to v ideálnom prípade číslo, ktoré sa čo najviac blíži hodnote 1,00, ale neprekročí ju. Všeobecne pre *N*-jadrový výpočtový zdroj je to číslo, ktoré je čo najvyššie, ale zároveň nie je väčšie ako *N*, aby procesy nemuseli čakať. To je v dynamicky sa meniacom systéme (pozri kapitolu 1.4) prakticky nemožné, preto väčšina systémových administrátorov považuje za ideálnu záťaž pohybujúcu sa vo všeobecnosti okolo 0,7 * *N*. Pre výpočtový zdroj so 4 jadrami to predstavuje záťaž 2,80. [10]

## 1.3 Úlohy vo vysokovýkonnom počítačovom systéme

Úloha (angl. *job, task*) je základná jednotka, ktorá sa vykonáva v počítačovom systéme. Úloha sa do vysokovýkonného počítačového systému zadáva spolu s jej opisom, ktorého všeobecná podoba je uvedená vo vzťahu 1.1: [9]

$$Desc(j) = < Res_j, Exec_j, Env_j, D_j > \quad (1.1),$$

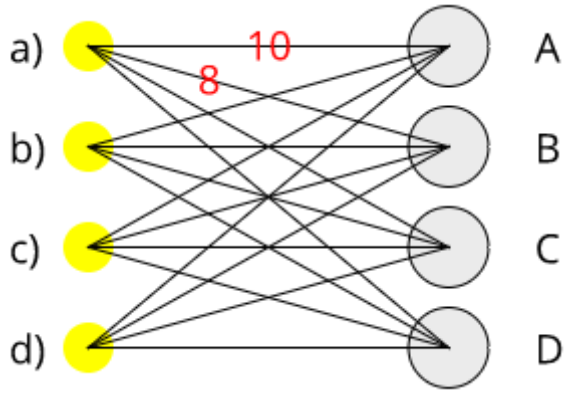
kde $Res_j$ predstavuje požiadavky na výpočtové zdroje, $Exec_j$ predstavuje aplikáciu, ktorú má systém vykonať, $Env_j$ sú požiadavky na softvérové vybavenie výpočtových zdrojov (najmä operačný systém, kompilátory, knižnice a aplikačný softvér) spolu so vstupnými a výstupnými údajmi a $D_j$ je odhad doby trvania úlohy. [9]

## 1.4 Rozvrhovanie úloh vo vysokovýkonnom počítačovom systéme

Rozvrhovanie úloh (angl. *job scheduling*) sa zaoberá problémom prideľovania výpočtových zdrojov pre konkrétne úlohy. Rozvrh (angl. *schedule*) vo vysokovýkonných počítačových systémoch je priradenie spracovania úloh výpočtovým zdrojom tak, aby kapacita výpočtového zdroja nebola prekročená a zároveň boli splnené hardvérové a softvérové požiadavky úlohy na výpočtové zdroje. Rozvrh určuje pre každý časový okamih množinu úloh, ktoré sa v tom okamihu majú vykonávať a množinu výpočtových zdrojov, na ktorých sa úlohy majú vykonávať. Rozvrhovanie úloh je predovšetkým špecifikované: [9]

- množinou výpočtových zdrojov,
- množinou úloh s ich požiadavkami,
- optimalizačným kritériom,
- dodatočnými obmedzujúcimi podmienkami.

Úlohu rozvrhovania úloh na výpočtové uzly je z pohľadu optimalizácie možné chápať ako problém dopravnej úlohy, pri ktorom je potrebné tovar (v našom prípade úlohy) rozdistribuovať do prevádzok (v našom prípade výpočtové uzly) za čo najnižšiu cenu určenú optimalizačným kritériom a účelovou funkciou.




Obrázok 1.6 Schéma rozvrhovania úloh

Na obrázku 1.6 môžeme vidieť základnú schému rozvrhovania úloh v počítačovom systéme. Rozvrhovanie úloh, podobne ako dopravná úloha sa zvyčajne vizualizuje ako graf. Na ľavej strane grafu sa nachádzajú úlohy *a)* až *d)*, na pravej strane grafu sa nachádzajú uzly *A* až *D*. Na hranách medzi úlohami a uzlami sú červenou farbou naznačené ceny vykonávania úlohy *a)* na uzle *A* a úlohy *a)* na uzle *B*. Úlohou optimalizačného algoritmu je nájsť také priradenie úloh na uzly, ktorého cena je minimálna. Na ilustračnom obrázku je počet úloh a počet uzlov rovnaký, čo pri reálnom rozvrhovaní spravidla väčšinu času nebýva.

Na základe toho, či je množina úloh a množina výpočtových zdrojov vopred daná a nemenná delíme rozvrhovacie systémy na: [9]

- statické (angl. *off-line scheduling*),
- dynamické (angl. *on-line scheduling*).

Pri statickom rozvrhovaní sú množina úloh a množina výpočtových zdrojov známe už pred začiatkom rozvrhovania. Rozvrh sa vytvorí na začiatku a už sa nemení. [9]

Pri dynamickom rozvrhovaní úlohy do systému prichádzajú dynamicky v ľubovoľnom čase. Rozvrh sa musí so vstupom každej úlohy do systému modifikovať. [9]

Vo vysokovýkonných počítačových systémoch sa štandardne používa dynamické rozvrhovanie úloh.

*„Hľadanie optimálneho rozvrhu patrí medzi NP-ťažké problémy, teda sa nedá vyriešiť v rozumnom čase, pokiaľ nejde o triviálnu úlohu. Cieľom je nájsť suboptimálne riešenie v polynomiálnom čase.“* [9]

## 1.5 Optimalizačné kritériá pre rozvrhovanie úloh

Optimalizačné kritérium pre rozvrhovanie úloh predstavuje jednu alebo viacero premenných, v závislosti na ktorých sa minimalizuje alebo maximalizuje hodnota účelovej funkcie. Medzi zvyčajne používané kritériá optimálnosti pri rozvrhovaní úloh patrí najmä: [9]

- čas dokončenia poslednej úlohy, označovaný tiež ako maximálny čas konca úloh (angl. *makespan*),
- suma časov dokončenia všetkých úloh,
- oneskorenie oproti plánovanému času dokončenia úlohy,
- celková doba obehu všetkých úloh v systéme.

# 2 INICIATÍVA ZELENÉHO POČÍTANIA

Zelené počítanie (angl. *Green Computing*, *Green IT*) je implementácia takých postupov, ktoré zlepšujú efektívnosť výpočtových zdrojov tak, že znižujú ich spotrebu elektrickej energie a znižujú dopad ich využitia na životné prostredie. [11]

Ukázalo sa, že moderné výpočtové zdroje skoro celý pracovný čas pracujú medzi 10 až 50 % maximálnej možnej záťaže, väčšinu času dokonca medzi 15 až 20 percentami. [12] Napriek skutočnosti, že priemerná záťaž výpočtového zdroja zostáva nízka, existujú krátke vysoké nárasty záťaže (angl. *peak*), kvôli ktorým sú operátori nútení prideľovať úlohám veľké množstvo výpočtových zdrojov, čo vedie k nízkej energetickej efektivite. Z týchto dôvodov sa návrh zelených riešení pre moderné vysokovýkonné počítačové systémy stal dôležitou témou výskumu. Bolo vytvorených viacero techník, ktoré majú za úlohu znižovať spotrebu vysokovýkonného počítačového systému. [1]

## 2.1 Techniky znižovania spotreby vysokovýkonného počítačového systému

Existujú rôzne techniky na znižovanie spotreby vysokovýkonných počítačových systémov. Vo všeobecnosti je možné tieto techniky klasifikovať do nasledujúcich stratégií: [1]

- stratégie založené na dynamickej zmene napätia alebo frekvencie (angl. *Dynamic Voltage / Frequency Scaling*, DVFS),
- stratégie založené na konsolidácii výpočtových zdrojov a alokovaní len požadovaných výpočtových prostriedkov (angl. *server-consolidation based approach*),
- stratégie založené na správe úloh a pracovného zaťaženia (angl. *job scheduling techniques*),
- stratégie založené na využití informácií o teplote (angl. *thermal-aware techniques*),
- iné stratégie.

Stratégie nie sú jednoznačne definované a oddelené, takže konkrétna technika môže využívať a často aj využíva princípy viacerých stratégií.

### 2.1.1 Dynamická zmena napätia alebo frekvencie

DVFS je bežne používaná stratégia riadenia výkonu, pri ktorej sa dynamicky upravuje taktovacia frekvencia procesora tak, aby umožňovala zníženie napájacieho napätia na dosiahnutie úspory energie. Princíp DVFS je, že frekvencia taktovania procesora vyžaduje pre stabilnú prevádzku určité napätie. Keď sa zníži taktovacia frekvencia, môže sa znížiť aj napájacie napätie, čo vedie k významnej úspore energie. Obmedzením DVFS je to, že zníženie taktovacej frekvencie znižuje výkon systému, takže DVFS musí byť aplikovaná tak, aby výkon systému zostal dostatočný pre vykonanie úlohy. [1]

### 2.1.2 Konsolidácia serverov

Konsolidácia serverov je stratégia zameraná na zabezpečenie efektívneho využívania zdrojov vysokovýkonného počítačového systému znižovaním celkového počtu využívaných výpočtových prostriedkov pri zachovaní priepustnosti systému. Pri tejto stratégii sa aplikácie zlučujú na menej výpočtových prostriedkov. Vďaka tomu môžu byť nevyužité výpočtové prostriedky prepnuté na stav s nízkou spotrebou alebo úplne vypnuté a ostatné výpočtové prostriedky môžu byť naplno využité. [1]

### 2.1.3 Správa úloh a využitie informácií o teplote

Súčasné vysokovýkonné výpočtové systémy majú zvyčajne vysoký počet výpočtových prostriedkov a rozhodnutie o umiestnení úlohy výrazne ovplyvňuje záťaž výpočtového prostriedku, spotrebu elektrickej energie a odvod tepla. [1]

## 2.2 Metriky zeleného počítania

V súčasnosti najpoužívanejšie metriky zeleného počítania (angl. *Green Computing Metrics*) súvisia s energiou. S cieľom pomôcť IT organizáciám zlepšiť efektívnosť vysokovýkonných počítačových systémov bolo navrhnutých niekoľko metrík posudzujúcich energetickú efektivitu. Medzi najpoužívanejšie patria predovšetkým: [13]

- celková spotrebovaná energia (angl. *Total Power Consumption*),
- účinnosť využitia energie (angl. *Power Usage Effectiveness*, PUE).

### 2.2.1 Celková spotrebovaná energia

Táto metrika je považovaná za celkovo najpopulárnejšiu. Základom hodnotenia je väčšinou cena energie a množstvo spotrebovaných kilowattov. Väčšinou sa využíva pri sledovaní spotreby energie podľa zariadenia, aplikácie alebo používateľa. [13]

### 2.2.2 Účinnosť využitia energie

Účinnosť využitia elektrickej energie je definovaná vzťahom (2.1): [14]

$$PUE = TFP / ITEP \quad (2.1),$$

kde *PUE* predstavuje účinnosť využitia elektrickej energie, *ITEP* elektrickú energiu spotrebovanú IT zariadením a *TFP* celkovú spotrebovanú elektrickú energiu zariadenia.

Spotrebovaná elektrická energia IT zariadenia (angl. *IT Equipment Power*, ITEP) je definovaná ako energia spotrebovaná záťažou súvisiacou s procesormi, úložiskami, sieťovým vybavením a periférnymi zariadeniami. [14]

Celková spotrebovaná elektrická energia zariadenia (angl. *Total Facility Power*, TFP) je spotrebovaná energia nameraná na meracom prístroji určenom pre vysokovýkonný počítačový systém. Okrem zariadení, ktoré sú priamo potrebné pre výpočty je najvýznamnejším odberateľom energie chladiaci systém vysokovýkonného počítačového systému. [14]

PUE vyjadruje pomer medzi energiou spotrebovanou celým vysokovýkonným počítačovým systémom a energiou spotrebovanou v priamej súvislosti s výpočtami na vysokovýkonnom počítačovom systéme. Ideálny stav by bol, keby PUE bola rovná 1, takže všetka energia by bola spotrebovaná len na výpočty, čo v skutočnom systéme nie je možné. Čím bližšie k hodnote 1 sa hodnota PUE dostane, tým efektívnejšie je energia vo vysokovýkonnom počítačovom systéme využívaná. [14]

## 2.3 Rozvrhovanie úloh s ohľadom na zelené počítanie

Štandardné optimalizačné kritériá pre rozvrhovanie úloh vo vysokovýkonných počítačových sme uviedli v kapitole 1.5. Vytvorené metriky zeleného počítania umožnili zadefinovanie optimalizačných kritérií pre rozvrhovanie úloh vo vysokovýkonných počítačových systémoch súvisiacich so spotrebou elektrickej energie, medzi ktoré predovšetkým patria:

- množstvo vykonaných výpočtov s právajúcou desatinnou čiarkou na jednotku spotrebovanej elektrickej energie [FLOPS/W], [15]
- množstvo elektrickej energie spotrebované na vykonanie úlohy.

# 3 VYUŽITÉ POMOCNÉ TECHNOLÓGIE

V nasledujúcej kapitole predstavíme niekoľko technológií, ktoré sme v práci využili. Žiadna z nich nie je primárnym zameraním tejto práce. Považujeme ich len za prostriedky, vďaka ktorým je možné jednoducho a efektívne vykonať výpočty potrebné pre navrhnutie modelu spotreby elektrickej energie a optimalizačného modelu rozvrhovania úloh s cieľom znižovania spotreby energie vysokovýkonného počítačového systému. Napriek tomu považujeme za potrebné poskytnúť aspoň základné informácie o týchto technológiách tak, aby kroky, algoritmy a výpočty, ktoré budeme používať boli ľahšie pochopiteľné.

## 3.1 Cloudové počítanie

Pojem cloudové počítanie (angl. *Cloud Computing*) označuje buď aplikácie dodávané ako služby cez internet alebo hardvér a softvér v datacentrách, ktorý poskytuje tieto služby. Dodávané služby sú označované pojmom Softvér ako služba (angl. *Software as a Service*, SaaS). Viacerí poskytovatelia cloudového počítania pre služby, ktoré poskytujú okrem pojmu SaaS používajú aj termíny Infraštruktúra ako služba (angl. *Infrastructure as a Service*, IaaS) a Platforma ako služba (angl. *Platform as a Service*, PaaS). Významnou technológiou, ktorá umožnila vznik a rozvoj cloudového počítania je technológia virtualizácie. [16]

Koncept cloudového počítania priniesol do oblasti IT niekoľko nových myšlienok, z ktorých najvýznamnejšie sú: [16]

- dojem neobmedzených výpočtových zdrojov dostatočne rýchlo dostupných na požiadanie tak, aby kopírovali nárast zaťaženia, čím sa eliminuje povinnosť používateľov plánovať dopredu kapacitu systému,
- odstránenie predbežných zmluvných záväzkov používateľov, čo umožňuje spoločnostiam začať s malými zdrojmi a postupne ich navyšovať, pokiaľ si to okolnosti vyžadujú.
- možnosť platiť za použitie výpočtových zdrojov podľa potreby v krátkodobom horizonte a možnosť ich uvoľnenia, keď už nie sú potrebné,
- nulové náklady na udržiavanie hardvérovej infraštruktúry pre používateľa cloudového systému.

Medzi najvýznamnejšie platformy cloudového počítania v súčasnosti patria najmä:

- Amazon Web Services (AWS), [17]
- Google Cloud Platform (GCP), [18]
- Microsoft Azure. [19]

Každá z týchto platforiem má svoje klady a zápory, prebieha medzi nimi intenzívny konkurenčný boj. V tabuľke 3.1 môžeme vidieť základné cloudové služby dostupné v týchto platformách.

Tabuľka 3.1 Základné služby dostupné v cloudových platformách [20] [21]

| **Kategória služby** | **Cloudová služba** | **GCP** | **AWS** | **Azure** |
|---|---|---|---|---|
| **Výpočty** | *IaaS* | Google Compute Engine | Amazon Elastic Compute Cloud (EC2) | Virtual Machines |
| **Výpočty** | *PaaS* | Google App Engine | AWS Elastic Beanstalk | App Service |
| **Výpočty** | *Serverless funkcie* | Google Cloud Functions | AWS Lambda | Functions |
| **Úložisko** | *Objektové úložisko* | Google Cloud Storage | Amazon Simple Storage Service (S3) | Azure Blob Storage |
| **Databázy** | *RDBMS* | Google Cloud SQL | Amazon Relational Database Service | SQL Database |
| **Databázy** | *NoSQL: Kľúč - Hodnota* | Google Cloud Datastore, Google Cloud Bigtable | Amazon DynamoDB | Table Storage |
| **Big Data & Analytics** | *Dávkové spracovanie údajov* | Google Cloud Dataproc, Google Cloud Dataflow | Amazon Elastic MapReduce | HDInsight |
| **Big Data & Analytics** | *Spracovanie prúdu údajov* | Google Cloud Dataflow | Amazon Kinesis | Stream Analytics |
| **Big Data & Analytics** | *Analýzy* | Google BigQuery | Amazon Redshift | Data Lake Analytics |

Po zvážení podstatných faktorov sme sa pre našu prácu rozhodli využiť cloudovú platformu ponúkanú spoločnosťou Google.

### 3.1.1 Charakteristika cloudovej platformy spoločnosti Google

Google Cloud Platform je sada cloudových výpočtových služieb, ktoré pracujú na rovnakej infraštruktúre ako je interná infraštruktúra spoločnosti Google pre svoje produkty, ako napríklad vyhľadávač Google alebo služba YouTube. Zahŕňa viaceré

služby, ako napríklad vzdialené výpočty, vzdialené ukladanie údajov, analýzy údajov a služby strojového učenia. Prvé vydanie sa stalo dostupným 6. októbra 2011, pričom sa sada dostupných produktov postupne doplňovala a skvalitňovala. Najpoužívanejšie služby spolu so základnou funkcionalitou, ktoré sú v súčasnosti dostupné, sú uvedené v tabuľke 3.1. [22]

GCP je spoplatnená platforma. Každá ponúkaná služba má svoj jedinečný cenník používania. Pri registrácii používateľa je potrebné zadať číslo kreditnej karty, z ktorej by v prípade potreby bolo možné stiahnuť spotrebované finančné prostriedky. Pre väčšinu najpoužívanejších služieb platí, že existujú denné a mesačné kvóty, ktoré pokiaľ používateľ neprekročí, tak je možné používať GCP bez finančných nákladov. Okrem toho je možné jednorazovo vytvoriť skúšobný účet, v rámci ktorého používateľ dostane kredit v hodnote 300$ na 12 mesiacov, z ktorého sa jeho poplatky uhrádzajú. [23] Pre lepšiu ilustráciu denných limitov uvádzame, že pri vytváraní praktickej časti tejto práce sme denný limit prekročili štyrikrát a z voľne dostupného 300$ kreditu sme minuli zhruba 2$.

## 3.2 Dopytovacia služba Dremel a jej implementácia v Google BigQuery

*Google BigQuery* je cloudová služba poskytovaná v rámci Google Cloud Platform, ktorá umožňuje vykonávanie interaktívnych dopytov na veľkých súboroch údajov. Táto služba je externá implementácia technológie *Dremel*, ktorú vyvinula v roku 2006 spoločnosť Google pre svoje interné potreby. [24]

Dremel je dopytovacia služba, ktorá umožňuje vykonávať dopyty podobné jazyku SQL na veľmi veľkých súboroch údajov a získať výsledky v rádoch sekúnd, pri zložitejších dopytoch desiatok sekúnd. Vývojári uvádzajú, že vďaka masívnej paralelizácii by Dremel mal byť schopný prehľadať 35 miliárd neindexovaných databázových záznamov, ktoré dohromady obsahujú zhruba 20 TB údajov za pár desiatok sekúnd. [25] Konceptuálne je Dremel postavený na dvoch princípoch:

- stĺpcovo orientovanej pamäti (angl. *Columnar Storage*),
- stromová architektúra (angl. *Tree Architecture*).

Google BigQuery poskytuje základný súbor funkcií dostupných v technológii Dremel prostredníctvom:

- webového rozhrania,
- rozhrania príkazového riadku,
- rozhrania REST API.

## 3.3 Porovnanie dopytovacej služby Dremel a programovacej paradigmy MapReduce

Pôvodne sme začali aplikáciu vyvíjať s použitím technológie na paralelné spracovanie údajov *MapReduce*. [26] Zložitejšie časti aplikácie sme za použitia tejto paradigmy ale nedokázali implementovať. Okrem toho sme zistili, že pri probléme, ktorý sme riešili existuje efektívnejší nástroj, ktorým je služba Google BigQuery. V tejto časti sme si dali za cieľ oboznámiť čitateľa s principiálnymi rozdielmi medzi týmito dvomi technológiami.

Hlavný rozdiel je v samotnom koncepte týchto technológií:

- Dremel je navrhnutý ako nástroj pre interaktívne dátové analýzy veľkých sád údajov,
- MapReduce je navrhnutý ako programovací framework určený na dávkové spracovanie veľkých sád údajov.

Dremel je navrhnutý, aby väčšinu dopytov dokončil rádovo počas sekúnd alebo desiatok sekúnd. MapReduce úloha už pri najľahšom zadaní potrebuje aspoň minúty, pri zložitejších zadaniach hodiny, či dokonca dni na dokončenie svojej činnosti. MapReduce nie je horšou technológiou ako Dremel, ale vyniká v iných oblastiach, ako napríklad v spracovaní neštruktúrovaných údajov, s čím si Dremel nedokáže poradiť. V našom prípade pracujeme so štruktúrovanými údajmi, na ktorých potrebujeme vykonať komplexnú analýzu, na ktorú je vhodnejší systém Dremel.

## 3.4 Optimalizačné nástroje spoločnosti Google

Optimalizačné nástroje spoločnosti Google (angl. *Google Optimization Tools*, často aj ako *OR-Tools*) sú rýchlou a prenosnou softvérovou knižnicou určenou pre riešenie optimalizačných problémov. Knižnica obsahuje najmä nástroje pre: [27]

- optimalizáciu s obmedzujúcimi hraničnými podmienkami,
- jednoduché a jednotné programové rozhranie pre riešenie problémov lineárnej a celočíselnej optimalizácie (angl. *Mixed Integer Programming*, MIP),
- grafové algoritmy (najkratšie cesty, minimálny tok, maximálny tok),
- algoritmus problému obchodného cestujúceho, algoritmus problému smerovania vozidiel,
- mnohé iné algoritmy.

Google optimalizačné nástroje boli naprogramované v jazyku C++. Vďaka nástroju pre vývoj softvéru SWIG [28], ktorý prepája programy napísané v jazykoch C a C++ s viacerými vysokoúrovňovými programovacími jazykmi (angl. *high-level programming language*) je možné OR-Tools využívať aj v jazykoch Python, C# a Java. [27]

OR-Tools sme sa rozhodli použiť z viacerých dôvodov: [27]

- je to slobodný softvér (angl. *open source*) distribuovaný pod licenciou Apache 2.0, [29]
- je aktívne udržiavaný a vylepšovaný,
- je dobre zdokumentovaný,
- je prenosný na všetky v súčasnosti najpoužívanejšie operačné systémy (Windows, Linux, Mac OS X),
- je efektívny, využívaný priamo v nástrojoch spoločnosti Google,
- je spoľahlivo otestovaný.

## 3.5 Javascript na strane servera

Ďalšou technológiou, ktorú sme v práci využili je *Node.js*, platforma postavená na výpočtovom jadre prehliadača Google Chrome, ktorá slúži na vytváranie rýchlych a škálovateľných aplikácií. Node.js využíva udalosťami riadený (angl. *event-driven*) asynchrónny programovací model, ktorý je vhodný hlavne pre aplikácie náročné na množstvo spracúvaných údajov. Jazyk Javascript sa historicky využíval hlavne na skriptovanie na strane klienta prostredníctvom skriptov vložených v HTML dokumentoch, ktoré sú vykonávane pomocou výpočtových jadier moderných webových prehliadačov. Node.js umožňuje vývojárom vytvárať skripty na strane servera pre vytvorenie dynamických webových stránok predtým, ako je obsah odoslaný do webového prehliadača používateľa. [30]

# 4 ANALÝZA MANAŽOVANIA ÚLOH VO VYSOKOVÝKONNOM POČÍTANÍ

Informácie uvedené v predchádzajúcich kapitolách využívame v praktickej časti práce, v ktorej sa zameriavame na analyzovanie spotreby elektrickej energie klastra pre vysokovýkonné výpočty Univerzity Mateja Bela (UMB). Spotrebu elektrickej energie skúmame a analyzujeme v období od polovice novembra 2017 do konca marca 2018. Cieľom tejto kapitoly je predstaviť ciele praktickej časti práce, opísať vysokovýkonný počítačový klaster Univerzity Mateja Bela so zameraním na výpočtové zdroje a charakterizovať údaje, ktoré máme k dispozícii a analyzujeme ich. Predpokladáme, že vďaka analýze údajov je možné vytvoriť optimalizačný model rozvrhovania úloh s cieľom znižovania spotreby elektrickej energie.

## 4.1 Ciele praktickej časti práce

Globálnym cieľom praktickej časti práce je zbierať a analyzovať údaje o výpočtových prostriedkoch a vykonávaných úlohách na vysokovýkonnom počítačovom klastri UMB v súvislosti so znižovaním spotreby elektrickej energie. Ďalším cieľom je implementovať optimalizačný model rozvrhovania úloh s cieľom znižovania spotreby elektrickej energie vysokovýkonného počítačového klastra UMB. Každý z týchto cieľov obsahuje niekoľko čiastkových cieľov, na dosiahnutie ktorých je potrebné postupne vykonať viacero krokov a vyriešiť viacero problémov.

## 4.2 Charakteristiky vysokovýkonného počítačového klastra UMB

V nasledujúcej kapitole uvedieme základné charakteristiky vysokovýkonného počítačového klastra Univerzity Mateja Bela, ktoré majú vplyv na návrh analýzy. Vysokovýkonný počítačový klaster Univerzity Mateja Bela je súčasťou Centra pre vysokovýkonné počítanie na Univerzite Mateja Bela (HPCC UMB), ktoré využíva infraštruktúru projektu Slovenskej infraštruktúry pre vysokovýkonné počítanie. [31]

### 4.2.1 Výpočtové prostriedky vysokovýkonného počítačového klastra UMB

Hierarchia výpočtových prostriedkov vysokovýkonného počítačového klastra UMB zodpovedá všeobecnej schéme výpočtových prostriedkov vysokovýkonných počítačových systémov uvedenej na obrázku 1.1. Výpočtové prostriedky vysokovýkonného počítačového klastra UMB boli zaobstarávané v dvoch etapách.

V tabuľke číslo 4.1 je možné vidieť stav zdrojov klastra po ukončení prvej etapy budovania v roku 2012.

Tabuľka 4.1 Stav zdrojov klastra UMB po ukončení prvej etapy budovania [32]

| **Atribút** | **Stav** |
|---|---|
| Počet výpočtových uzlov | 24 |
| Počet výpočtových jadier | 288 |
| Výpočtový výkon | viac ako 3 TFLOP |
| Kapacita RAM pre jeden uzol | 48 GB |
| Kapacita dátového úložiska | 96 TB |

V prvej etape budovania vysokovýkonného počítačového klastra UMB bolo zapojených 24 výpočtových uzlov, pričom každý obsahuje 12 výpočtových jadier. Dva výpočtové uzly okrem toho obsahujú aj grafické výpočtové akcelerátory (angl. *Graphics Processing Unit*, GPU). V tabuľke 4.2 sa nachádzajú zdroje, ktoré boli doplnené počas druhej etapy zaobstarávania hardvéru počas roku 2014.

Tabuľka 4.2 Zdroje pridané v rámci druhej etapy budovania klastra UMB [32]

| **Atribút** | **Stav** |
|---|---|
| Nové výpočtové uzly so 128 GB RAM pre jeden uzol | 14 |
| Nové výpočtové uzly so 64 GB RAM a dvomi grafickými kartami pre jeden uzol | 3 |
| Počet nových výpočtových jadier | 272 |
| Dodaný výpočtový výkon | 2,8 TFLOP |

V rámci druhej etapy budovania vysokovýkonného počítačového klastra UMB bolo do systému pridaných 17 výpočtových uzlov, pričom každý obsahoval 16 výpočtových jadier. Tri výpočtové uzly okrem toho obsahovali aj po dva grafické výpočtové akcelerátory. Nové výpočtové uzly obsahujú aj podporu technológie *Hyperthreading* (HT), vďaka ktorej sa z jedného fyzického výpočtového jadra stávajú dve virtuálne jadrá. Procesor sa voči aplikačnému softvéru správa tak, ako keby bol v počítači dvakrát. Tabuľka 4.3 sumarizuje výpočtový výkon klastra.

Tabuľka 4.3 Výpočtový výkon klastra UMB [32]

| **Atribút** | **Stav** |
|---|---|
| Celkový počet výpočtových uzlov | 41 |
| Celkový počet výpočtových jadier | 560 (virtuálne sa rozširuje na 740) |
| Celkový výpočtový výkon | viac ako 5,8 TFLOP |
| Grafické akcelerátory s 448 CUDA jadrami | 2 |
| Grafické akcelerátory s 2496 CUDA jadrami | 6 |

Pre lepšie pochopenie komplexnosti a usporiadania klastra uvádzame tabuľku 4.4, ktorá ukazuje podrobnosti o jednotlivých výpočtových uzloch. Každý výpočtový uzol je tvorený jedným samostatným výpočtovým serverom.

Tabuľka 4.4 Podrobnosti o výpočtových uzloch klastra UMB [32]

| **Názov výpočtového uzla** | **Použitý server** | **Počet jadier** | **RAM** | **V prevádzke** | **Ďalšie informácie** |
|---|---|---|---|---|---|
| comp01 | dx360M3 | 12 | 48 GB | Áno | |
| comp02 | dx360M3 | 12 | 48 GB | Áno | |
| comp03 | dx360M3 | 12 | 48 GB | Áno | |
| comp04 | dx360M3 | 12 | 48 GB | Nie | |
| comp05 | dx360M3 | 12 | 48 GB | Nie | |
| comp06 | dx360M3 | 12 | 48 GB | Áno | |
| comp07 | dx360M3 | 12 | 48 GB | Áno | |
| comp08 | dx360M3 | 12 | 48 GB | Nie | |
| comp09 | dx360M3 | 12 | 48 GB | Nie | |
| comp10 | dx360M3 | 12 | 48 GB | Áno | |
| comp11 | dx360M3 | 12 | 48 GB | Áno | |
| comp12 | dx360M3 | 12 | 48 GB | Áno | |
| comp13 | dx360M3 | 12 | 48 GB | Áno | |
| comp14 | dx360M3 | 12 | 48 GB | Áno | |
| comp15 | dx360M3 | 12 | 48 GB | Áno | |
| comp16 | dx360M3 | 12 | 48 GB | Áno | |
| comp17 | dx360M3 | 12 | 48 GB | Áno | |
| comp18 | dx360M3 | 12 | 48 GB | Áno | |
| comp19 | dx360M3 | 12 | 48 GB | Áno | |
| comp20 | dx360M3 | 12 | 48 GB | Áno | |
| comp21 | dx360M3 | 12 | 48 GB | Áno | |
| comp22 | dx360M3 | 12 | 48 GB | Áno | |
| comp23 | dx360M3 | 12 | 48 GB | Nie | 1x GPU (448 CUDA) |
| comp24 | dx360M3 | 12 | 48 GB | Nie | 1x GPU (448 CUDA) |
| comp25 | dx360M4 | 16 | 128 GB | Áno | |
| comp26 | dx360M4 | 16 | 128 GB | Áno | |
| comp27 | dx360M4 | 16 | 128 GB | Áno | |
| comp28 | dx360M4 | 16 | 128 GB | Áno | |
| comp29 | dx360M4 | 16 | 128 GB | Áno | |
| comp30 | dx360M4 | 16 | 128 GB | Áno | |
| comp31 | dx360M4 | 16 | 128 GB | Áno | |
| comp32 | dx360M4 | 16 | 128 GB | Áno | |
| comp33 | dx360M4 | 16 | 128 GB | Áno | |
| comp34 | dx360M4 | 16 | 128 GB | Áno | |
| comp35 | dx360M4 | 16 | 128 GB | Áno | |
| comp36 | dx360M4 | 16 | 128 GB | Áno | |
| comp37 | dx360M4 | 16 | 128 GB | Áno | |
| comp38 | dx360M4 | 16 | 128 GB | Áno | |
| comp39 | dx360M4 | 16 | 64 GB | Áno | 2x GPU s 2496 CUDA |
| comp40 | dx360M4 | 16 | 64 GB | Áno | 2x GPU s 2496 CUDA |

| | | | | | |
|---|---|---|---|---|---|
| comp41 | dx360M4 | 16 | 64 GB | Áno | 2x GPU s 2496 CUDA |

Z uvedených informácií vidíme, že ide o heterogénny klaster. Niektoré výpočtové uzly počas skúmaného obdobia neboli v prevádzke, nemáme z nich žiadne údaje a preto sme ich pri analyzovaní a navrhovaní modelov nebrali do úvahy.

### 4.2.2 Riadenie zdrojov vo vysokovýkonnom počítačovom klastri UMB

V kapitole 1.1 sme uviedli a opísali dva štandardne využívané modely prístupu k riadeniu zdrojov. Vo vysokovýkonnom počítačovom klastri UMB sa využíva prvý uvedený, model riadenia pracovnej záťaže a zdrojov (WRMS). Konkrétnu implementáciu modelu WRMS na vysokovýkonnom počítačovom klastri UMB predstavuje systém *Terascale Open-source Resource and QUEue Manager,* známy ako *TORQUE*. [33] *Torque Resource Manager* je pokročilý slobodný softvér založený na projekte *Portable Batch System* (*PBS*). Komunita vývojárov systému Torque rozšírila pôvodný PBS, aby zlepšila škálovateľnosť, odolnosť voči chybám a ponúkla novú funkcionalitu. Torque vykonáva riadenie úloh a distribuovaných výpočtových prostriedkov. Záznamy o svojej činnosti ukladá do logovacích súborov, z ktorých je možné spätne zrekonštruovať všetky kroky, ktoré vykonal.

### 4.2.3 Monitorovacie nástroje vysokovýkonného počítačového klastra UMB

Zdroje vysokovýkonných počítačových klastrov je potrebné monitorovať z viacerých dôvodov, najmä aby mohli byť odstránené prípadné poruchy a tak bola zabezpečená neustála dostupnosť poskytovaných služieb. Existuje viacero platených aj neplatených nástrojov, ktoré je možné na monitorovanie vysokovýkonného počítačového systému využiť. Vysokovýkonný počítačový klaster UMB je monitorovaný systémom *Zabbix*. [34]

Zabbix je monitorovací slobodný softvér určený pre veľké spoločnosti. Slúži na monitorovanie a sledovanie sieťových služieb, serverov a aplikácií. Na ukladanie zaznamenaných údajov využíva relačné databázové systémy. Backendová časť systému Zabbix, ktorá slúži na monitorovanie systému a vykonávanie meraní je napísaná v jazyku C a jeho webová frontendová časť, ktorá slúži na zobrazovanie grafov a informácií o systéme pre systémových administrátorov je napísaná v jazyku PHP. [34]

## 4.3 Charakteristika údajov potrebných k vykonaniu analýzy

V praktickej časti práce používame údaje, ktoré sme zbierali od polovice novembra 2017 do konca marca 2018. Sú to podrobné údaje o:

- úlohách, ktoré boli na klastri UMB pre vysokovýkonné výpočty vykonávané,
- výpočtových prostriedkoch klastra UMB.

### 4.3.1 Údaje o úlohách

Údaje o vykonávaných úlohách pochádzajú z plánovacieho systému TORQUE, ktorý zabezpečuje riadenie zdrojov vo vysokovýkonnom počítačovom klastri UMB a ukladá všetky údaje o svojej činnosti do logovacích súborov priamo na klastri (pozri kapitolu 4.2.2). Z logovacieho súboru sú údaje o rozvrhovaní vyťahované do relačnej databázy, ktorú sme mali pri analýze k dispozícii. Ukážku údajov o vykonávaných úlohách je možné vidieť na obrázku 4.1.

| JobId | JobStarted | JobFinished | JobHostsFullText | Slots | Nodes |
|---|---|---|---|---|---|
| 124545 | 2017-11-29 14:11:04.000 UTC | 2017-11-29 14:11:41.000 UTC | comp06-0/4+comp06-0/5+comp06-0/6+comp06-0/7 | 4 | 1 |
| 124447 | 2017-11-28 08:02:35.000 UTC | 2017-11-28 08:02:35.000 UTC | comp22-0/0+comp22-0/1+comp22-0/2+comp22-0/3 | 4 | 1 |
| 124448 | 2017-11-28 08:02:38.000 UTC | 2017-11-28 08:02:38.000 UTC | comp22-0/0+comp22-0/1+comp22-0/2+comp22-0/3 | 4 | 1 |
| 124438 | 2017-11-28 06:37:01.000 UTC | 2017-12-01 21:10:10.000 UTC | comp01-0/8+comp01-0/9+comp01-0/10+comp01-0/11 | 4 | 1 |
| 124439 | 2017-11-28 06:37:04.000 UTC | 2017-11-28 18:58:07.000 UTC | comp02-0/8+comp02-0/9+comp02-0/10+comp02-0/11 | 4 | 1 |
| 124440 | 2017-11-28 06:37:07.000 UTC | 2017-11-28 18:48:08.000 UTC | comp03-0/8+comp03-0/9+comp03-0/10+comp03-0/11 | 4 | 1 |
| 124445 | 2017-11-28 07:57:41.000 UTC | 2017-12-01 22:12:59.000 UTC | comp06-0/8+comp06-0/9+comp06-0/10+comp06-0/11 | 4 | 1 |
| 124411 | 2017-11-27 16:31:14.000 UTC | 2017-11-27 17:08:42.000 UTC | comp06-0/8+comp06-0/9+comp06-0/10+comp06-0/11 | 4 | 1 |
| 124412 | 2017-11-27 16:31:33.000 UTC | 2017-11-27 16:31:59.000 UTC | comp07-0/8+comp07-0/9+comp07-0/10+comp07-0/11 | 4 | 1 |
| 124446 | 2017-11-28 08:02:28.000 UTC | 2017-11-28 08:19:33.000 UTC | comp07-0/8+comp07-0/9+comp07-0/10+comp07-0/11 | 4 | 1 |
| 124212 | 2017-11-20 01:40:07.000 UTC | 2017-11-20 02:34:13.000 UTC | comp01-0/0+comp01-0/1+comp01-0/2+comp01-0/3+comp01-0/4+comp01-0/5 | 6 | 1 |

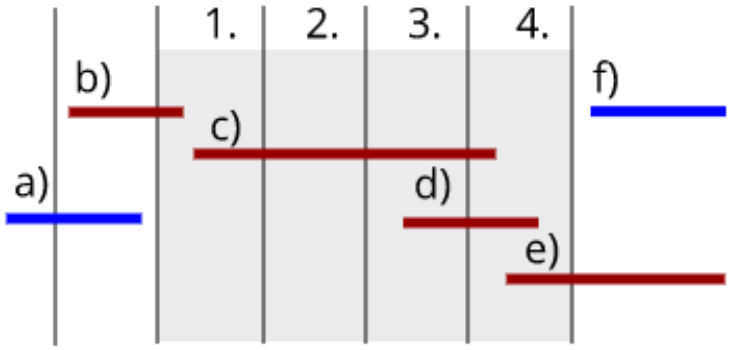
Obrázok 4.1 Ukážka údajov o vykonávaných úlohách

Z dostupných údajov sme sa zamerali hlavne na údaje:

- *JobId* – identifikačné číslo úlohy,
- *JobStarted* – čas začiatku vykonávania úlohy,
- *JobFinished* – čas ukončenia vykonávania úlohy,
- *JobHostsFullText* – zoznam výpočtových uzlov, na ktorých bola úloha vykonávaná,
- *Slots* – počet výpočtových jadier, ktoré úloha na svoje vykonávanie potrebovala,
- *Nodes* – počet výpočtových uzlov, ktoré úloha na svoje vykonávanie potrebovala.

Za skúmané časové obdobie sú k dispozícii údaje o 15 433 výpočtových úlohách, ktoré boli v systéme vykonávané. Mnohé z vykonávaných úloh pozostávajú z ďalších podúloh, ktoré v rámci svojho vykonávania volajú, ale v praktickej časti našej práce úlohu chápeme a analyzujeme ako nemenný celok, pretože z dostupných údajov nemáme možnosť sledovať čo sa deje v jej vnútri.

Vysokovýkonný počítačový klaster UMB je dynamický systém, do ktorého v ľubovoľnom čase vstupujú nové úlohy, pričom vykonávanie pôvodných môže stále pokračovať. Pri návrhu analýzy je potrebné brať do úvahy skutočnosť, že v čase začiatku analýzy už je klaster v určitom stave. Na viacerých výpočtových uzloch môžu byť vykonávané úlohy, o ktorých analýza musí vedieť napriek tomu, že ich začiatok je mimo jej pracovného okna. Analýza je navrhnutá tak, aby brala do úvahy úlohy, ktorých vykonávanie začalo pred prvým analyzovaným dňom a zasiahlo aspoň do prvého analyzovaného dňa alebo ich vykonávanie začalo počas určitého analyzovaného dňa. Pre lepšie pochopenie uvádzame obrázok 4.2.




Obrázok 4.2 Ilustrácia akceptovania úloh, ktoré vstupujú do aplikácie

Dni sú znázornené vertikálnymi čiarami, vykonávané úlohy horizontálnymi. Analyzujeme dátumový interval medzi dňami 1 až 4, čo je na obrázku znázornené sivým podfarbením. Vidíme, že úloha *a)* začne aj skončí svoje vykonávanie ešte pred začiatkom analyzovaného intervalu, takže táto úloha systém na začiatku analýzy už neovplyvňuje. Naopak, úloha *b)* začne svoje vykonávanie pred začiatkom analýzy, ale trvá až do dňa 1, takže je potrebné brať do úvahy jej vplyv na systém v prvom dni analýzy. Úlohy *c)* a *d)* začnú aj skončia svoje vykonávanie počas analyzovaného intervalu, takže je potrebné brať do úvahy celé ich vykonávanie. Úloha *e)* začne svoje vykonávanie počas analyzovaného intervalu, takže ju v rámci analýzy musíme brať do úvahy napriek tomu, že svoje vykonávania skončí až po ukončení analyzovaného intervalu. Vykonávanie úlohy *f)* začne až po skončení analyzovaného intervalu, takže

je potrebné zabezpečiť, aby táto úloha analýzu neovplyvnila. Úlohy, ktoré v rámci analýzy berieme do úvahy sú znázornené hnedou farbou a úlohy, ktorú v rámci analýzy neberieme do úvahy sú znázornené modrou farbou.

### 4.3.2 Údaje o výpočtových prostriedkoch klastra

Údaje o jednotlivých výpočtových prostriedkoch vysokovýkonného počítačového klastra UMB pochádzajú z monitorovacieho systému Zabbix (pozri kapitolu 4.2.3), ktorý monitoruje skúmaný vysokovýkonný počítačový klaster v pravidelných intervaloch a namerané údaje ukladá do tabuliek v MySQL databáze.

Každý skúmaný parameter je uložený v samostatnej databázovej tabuľke. Ukážku údajov o záťaži na výpočtových uzloch je možné vidieť na obrázku 4.3. Podstatné atribúty pre analýzu sú:

- *loadValue* – zistená záťaž na výpočtovom uzle,
- *loadDateTime* – čas merania,
- *loadComp* – výpočtový uzol, na ktorom bola záťaž zistená.

| Row | loadItemId | loadClock | loadValue | loadNS | loadName | loadComp | loadDateTime |
|---|---|---|---|---|---|---|---|
| 1 | 38454 | 2017-11-07 00:14:09.000 UTC | 0.09 | 980870045 | system.cpu.load[,avg1] | comp01 | 2017-11-07T01:14:09 |
| 2 | 38454 | 2017-11-07 01:14:49.000 UTC | 1.01 | 145347760 | system.cpu.load[,avg1] | comp01 | 2017-11-07T02:14:49 |
| 3 | 38454 | 2017-11-07 01:19:04.000 UTC | 1.09 | 728587527 | system.cpu.load[,avg1] | comp01 | 2017-11-07T02:19:04 |
| 4 | 38454 | 2017-11-07 07:00:49.000 UTC | 9.5 | 779425278 | system.cpu.load[,avg1] | comp01 | 2017-11-07T08:00:49 |
| 5 | 38454 | 2017-11-07 07:09:04.000 UTC | 7.57 | 724581696 | system.cpu.load[,avg1] | comp01 | 2017-11-07T08:09:04 |
| 6 | 38454 | 2017-11-07 07:15:04.000 UTC | 1.57 | 791606842 | system.cpu.load[,avg1] | comp01 | 2017-11-07T08:15:04 |
| 7 | 38454 | 2017-11-07 08:07:49.000 UTC | 8.94 | 518837977 | system.cpu.load[,avg1] | comp01 | 2017-11-07T09:07:49 |
| 8 | 38454 | 2017-11-07 08:31:34.000 UTC | 10.93 | 880056349 | system.cpu.load[,avg1] | comp01 | 2017-11-07T09:31:34 |
| 9 | 38454 | 2017-11-07 08:47:24.000 UTC | 6.16 | 993722411 | system.cpu.load[,avg1] | comp01 | 2017-11-07T09:47:24 |
| 10 | 38454 | 2017-11-07 09:09:00.000 UTC | 9.24 | 30883962 | system.cpu.load[,avg1] | comp01 | 2017-11-07T10:09:00 |
| 11 | 38454 | 2017-11-07 09:10:30.000 UTC | 10.53 | 42126645 | system.cpu.load[,avg1] | comp01 | 2017-11-07T10:10:30 |
| 12 | 38454 | 2017-11-07 09:15:49.000 UTC | 9.46 | 769069468 | system.cpu.load[,avg1] | comp01 | 2017-11-07T10:15:49 |
| 13 | 38454 | 2017-11-07 09:21:44.000 UTC | 6.53 | 720200564 | system.cpu.load[,avg1] | comp01 | 2017-11-07T10:21:44 |
| 14 | 38454 | 2017-11-07 10:00:49.000 UTC | 10.07 | 971091620 | system.cpu.load[,avg1] | comp01 | 2017-11-07T11:00:49 |
| 15 | 38454 | 2017-11-07 10:06:14.000 UTC | 7.95 | 251180036 | system.cpu.load[,avg1] | comp01 | 2017-11-07T11:06:14 |
| 16 | 38454 | 2017-11-07 11:03:09.000 UTC | 9.55 | 160895065 | system.cpu.load[,avg1] | comp01 | 2017-11-07T12:03:09 |

Table JSON

First < Prev Rows 1 - 16 of 67041731 Next > Last

Obrázok 4.3 Ukážka údajov o záťaži výpočtových uzlov

Ukážku údajov o spotrebe výpočtových uzlov je možné vidieť na obrázku 4.4. Podstatné atribúty pre analýzu sú:

- *powerReading* – zistená spotreba elektrickej energie,
- *powerTimestamp* – čas merania,
- *powerHostname* – výpočtový uzol, pre ktorý je spotreba uvádzaná.

| Row | powerHostname | powerTimestamp | powerID | powerSDRType | powerType | powerSNum | powerName | powerStatus | powerReading |
|---|---|---|---|---|---|---|---|---|---|
| 33 | comp01 | 2017-11-22T23:59:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 34 | comp01 | 2017-11-23T06:24:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 35 | comp01 | 2017-11-25T01:33:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 36 | comp01 | 2017-11-25T09:48:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 37 | comp01 | 2017-11-26T01:31:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 38 | comp01 | 2017-11-26T15:00:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 39 | comp01 | 2017-11-26T16:25:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 40 | comp01 | 2017-11-27T12:00:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 41 | comp01 | 2017-11-27T13:06:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 236.00 W |
| 42 | comp01 | 2017-11-28T07:13:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 336.00 W |
| 43 | comp01 | 2017-11-28T08:04:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 336.00 W |
| 44 | comp01 | 2017-11-28T09:39:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 344.00 W |
| 45 | comp01 | 2017-11-28T11:23:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 360.00 W |
| 46 | comp01 | 2017-11-28T12:33:04 | 000b | Full | Current | 2f | AC Avg Power | OK | 352.00 W |
| 47 | comp01 | 2017-11-28T13:12:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 348.00 W |
| 48 | comp01 | 2017-11-28T13:13:05 | 000b | Full | Current | 2f | AC Avg Power | OK | 340.00 W |

Table JSON

First < Prev Rows 33 - 48 of 19383186 Next > Last

Obrázok 4.4 Ukážka údajov o spotrebe výpočtových uzlov

# 5 NÁVRH APLIKÁCIE PRE ANALÝZU SPOTREBY ELEKTRICKEJ ENERGIE KLASTRA

Cieľom tejto kapitoly je navrhnúť algoritmy a postupy, ktoré je možné použiť pri implementácii praktickej časti práce. Máme k dispozícii niekoľko typov údajov, ktoré sa síce všetky týkajú klastra UMB, ale nie sú nijako prepojené a preto v nich nie je možné vidieť akékoľvek súvislosti. Sú to údaje o záťaži na výpočtových uzloch, o spotrebe elektrickej energie výpočtových uzlov a údaje o vykonávaných úlohách na výpočtových uzloch. Domnievame sa, že tieto údaje spolu úzko súvisia, pretože spotreba elektrickej energie závisí od záťaže výpočtového uzla, ktorá je spôsobovaná najmä vykonávanými úlohami. Preto v záujme dosiahnutia cieľov praktickej časti práce uvedených v kapitole 4.1, medzi ktoré patrí najmä analýza údajov o spotrebe elektrickej energie klastra UMB a návrh optimalizačného modelu rozvrhovania úloh, navrhujeme:

1. Zlúčiť údaje o zistenej záťaži výpočtových uzlov klastra UMB s údajmi o ich spotrebe.
2. Hľadať vzťah medzi záťažou výpočtových uzlov klastra UMB a ich spotrebou elektrickej energie. Nájdený vzťah predstavuje model spotreby elektrickej energie klastra UMB.
3. Navrhnúť optimalizačný model rozvrhovania úloh na klastri UMB na základe modelu spotreby elektrickej energie klastra UMB s cieľom ušetriť elektrickú energiu.
4. Implementovať navrhnutý optimalizačný model v konkrétnom programovacom jazyku.
5. Implementovaný optimalizačný model overiť a zhodnotiť zistené výsledky.

## 5.1 Návrh zlúčenia údajov o záťaži a údajov o spotrebe výpočtového uzla

Ako sme už uviedli v kapitole 4.3.2 údaje o výpočtových prostriedkoch klastra sa nachádzajú v rôznych databázových tabuľkách. Údaje z týchto databázových tabuliek máme sprístupnené v podobe exportovaných súborov vo formáte *.csv*. Pre vykonávanie analýzy je potrebné nájsť spôsob ako tieto tabuľky prepojiť alebo zlúčiť do jednej. V klasických relačných databázových systémoch sa na prepojenie dvoch tabuliek štandardne používa príkaz *JOIN*, ktorého všeobecný tvar je:

tabuľka_1 JOIN tabuľka_2 ON tabuľka_1.FK = tabuľka_2.PK ,

kde *FK* predstavuje stĺpec s cudzím kľúčom (angl. *Foreign Key*) v tabuľke 1 a *PK* predstavuje stĺpec s primárnym kľúčom (angl. *Primary Key*) v tabuľke 2. V časti príkazu nasledujúcej za kľúčovým slovom *ON* je možné použiť aj viac podmienok spojených logickými spojkami.

V spracúvaných údajoch o výpočtových uzloch klastra sa ale nenachádzajú žiadne primárne, ani cudzie kľúče. Aby analýza poskytovala zmysluplné výsledky, musia sa zlúčené údaje viazať na rovnaký výpočtový uzol v rovnakom časovom okamihu. Výpočtové uzly je možné priamo použiť v podmienke príkazu *JOIN*, keďže sú ich hodnoty jednoznačné. Keby sme sa takto pokúsili použiť aj časové atribúty, tak by sme nedostali korektné výsledky, pretože zisťovanie záťaže a zisťovanie spotreby sa síce vykonáva v pravidelných časových intervaloch, ale nie v zhodnom čase, takže prienik by bol buď prázdna množina, alebo len zopár nereprezentatívnych spárovaných záznamov. Časové údaje zo vstupných súborov je najskôr potrebné zaokrúhliť. Navrhujeme zaokrúhliť časové údaje na minúty nadol (príkaz *ROUND_MIN* v pseudokóde) a následné vykonať zlúčenie (pozri pseudokód 5.1).

```
údaje_o_záťaží JOIN údaje_o_spotrebe
               ON
údaje_o_záťaži.uzol = údaje_o_spotrebe.uzol
               AND
ROUND_MIN(údaje_o_záťaži.čas) = ROUND_MIN(údaje_o_spotrebe.čas)
```

Pseudokód 5.1 Návrh zlúčenia údajov o záťaži a údajov o spotrebe

## 5.2 Návrh na vytvorenie modelu spotreby elektrickej energie klastra

Na vytvorenie modelu spotreby elektrickej energie vysokovýkonného počítačového klastra zo vzťahu medzi záťažou výpočtových uzlov klastra UMB a ich spotrebou elektrickej energie navrhujeme použiť rôzne typy metód regresnej analýzy údajov a porovnať ich výsledky. Ako vstupný parameter pre metódy regresnej analýzy navrhujeme použiť záťaž výpočtového uzla (nezávislá premenná) a ako výstupný parameter spotrebu elektrickej energie výpočtového uzla vysokovýkonného počítačového systému UMB (závislá premenná).

## 5.3 Návrh overenia správnosti modelu spotreby elektrickej energie klastra

Navrhnuté modely spotreby elektrickej energie navrhujeme overiť a porovnať pomocou metódy kvadratickej chyby (angl. *least squares*, často aj ako metóda najmenších štvorcov). Výpočet, ktorý navrhujeme použiť je uvedený vo vzťahu 5.1:

$$e_i = (y_i - s_i)^2 \tag{5.1},$$

kde $e_i$ predstavuje kvadratickú chybu modelu pri $i$-tej vzorke, $y_i$ predstavuje spotrebu elektrickej energie odhadnutú modelom pri $i$-tej vzorke a $s_i$ predstavuje skutočnú zistenú spotrebu elektrickej energie pri $i$-tej vzorke merania.

Ďalej pre každý model navrhujeme zistiť absolútnu a relatívnu priemernú odchýlku od priemernej skutočnej hodnoty spotrebovanej elektrickej energie pri rôznych záťažiach, ktoré vo vysokovýkonnom počítačovom systéme nastávajú.

Pre určenie absolútnej a relatívnej priemernej odchýlky potrebujeme najskôr určiť celkovú chybu modelu, ktorú navrhujeme určiť ako sumu všetkých parciálnych chýb (pozri vzťah 5.2).

$$E = \sum_{i=1}^{N} e_i \tag{5.2},$$

kde $E$ predstavuje celkovú kvadratickú chybu modelu, $e_i$ parciálnu chybu modelu pri $i$-tej vzorke a $N$ predstavuje počet vzoriek.

Priemernú absolútnu chybu modelu navrhujeme určiť ako odmocninu z podielu celkovej chyby modelu a počtu vzoriek (pozri vzťah 5.3).

$$\bar{E} = \sqrt{\frac{E}{N}} \tag{5.3},$$

kde $\bar{E}$ predstavuje priemernú absolútnu chybu modelu, $E$ predstavuje celkovú kvadratickú chybu modelu a $N$ predstavuje počet vzoriek.

Priemernú relatívnu chybu modelu navrhujeme určiť ako podiel priemernej absolútnej chyby modelu a priemernej hodnoty skutočne spotrebovanej elektrickej energie (pozri vzťah 5.4).

$$\tilde{E} = \frac{\bar{E}}{s} \tag{5.4},$$

kde $\tilde{E}$ predstavuje priemernú relatívnu chybu modelu, $\bar{E}$ predstavuje priemernú absolútnu chybu modelu a $s$ predstavuje priemernú hodnotu skutočne spotrebovanej elektrickej energie.

Na základe priemernej chyby modelu navrhujeme vybrať model spotreby elektrickej energie, ktorý najlepšie opisuje závislosť medzi záťažou výpočtového uzla a jeho spotrebou elektrickej energie.

## 5.4 Návrh optimalizačného modelu rozvrhovania úloh

Cieľom optimalizácie je nájsť najlepšie riešenie problému z veľkej množiny prípustných riešení. Každý optimalizačný problém je charakterizovaný: [35]

- optimalizačným kritériom, ktoré je potrebné optimalizovať,
- účelovou funkciou (angl. *objective function*), ktorá vypočíta hodnotu cieľovej premennej pre akékoľvek prípustný vstup,
- hraničnými podmienkami (angl. *constraints*), čo sú obmedzenia množiny možných riešení na základe špecifických požiadaviek problému. Prípustné riešenie musí spĺňať všetky definované hraničné podmienky problému.

Optimálne riešenie je také, pri ktorom je hodnota účelovej funkcie najlepšia, čo môže byť maximálna alebo minimálna v závislosti na konkrétnom type problému. [35] V praxi pri zložitých problémoch nie je vždy možné zistiť optimálne riešenie, ale často je postačujúce pseudooptimálne riešenie.

V rámci aplikácie pre analýzu spotreby klastra UMB ako optimalizačné kritérium pre rozvrhovanie úloh používame množstvo elektrickej energie spotrebované na vykonanie úlohy (pozri kapitolu 2.3). Na dosiahnutie minimalizácie spotreby elektrickej energie klastra UMB na základe tohto optimalizačného kritéria navrhujeme zostaviť nasledujúci optimalizačný model:

1. Spätne zložiť pôvodný rozvrh vykonávaných úloh z údajov o vykonávaných úlohách.
2. Zlúčiť údaje o výpočtových uzloch a vykonávaných úlohách na základe času merania a výpočtového uzla záznamu.
3. Pre každú vykonávanú úlohu z údajov o záťaži na výpočtových uzloch klastra UMB spätne určiť záťaž, ktorú úloha na výpočtovom uzle spôsobila (pozri kapitolu 5.4.1).
4. Ako maticu cien optimalizačného modelu použiť maticu záťaží úloh podľa bodu 3 tohto návrhu.
5. Ako účelovú funkciu použiť model spotreby elektrickej energie navrhnutý v kapitole 5.2.
6. Ako hraničné podmienky použiť:

6.1. Každá vykonávaná úloha musí mať pridelených toľko výpočtových jadier, koľko požaduje vo svojom opise.

6.2. Kapacita žiadneho výpočtového uzla vyjadrená počtom používaných výpočtových jadier nemôže byť prekročená.

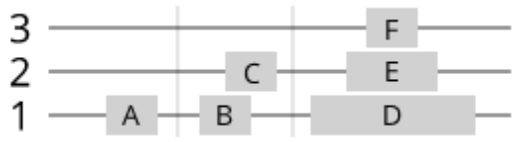
7. Ako optimalizačnú metódu použiť metódu dopravnej úlohy a modifikovať ju na prácu s dynamickým rozvrhovaním (pozri kapitolu 5.4.2).

### 5.4.1 Návrh na určenie záťaže úlohy

Z údajov o výpočtových uzloch (pozri kapitolu 4.3.2) poznáme údaje o zaťažení každého výpočtového uzla. Zo spätne zloženého pôvodného rozvrhu úloh vieme, v ktorých časoch, a na ktorých uzloch boli úlohy vykonávané. Ako vstup do optimalizačného algoritmu potrebujeme informáciu o tom, ktorá úloha akou mierou záťaž výpočtového uzla spôsobuje. Vzhľadom na skutočnosť, že jednotlivé úlohy sa na výpočtových uzloch prelínajú, bolo potrebné záťaž pre konkrétnu úlohu zo záťaže o výpočtových uzloch zisťovať v algoritme po krokoch:

1. V prvom kroku bolo potrebné nájsť úlohy, ktoré aspoň istú časť svojho vykonávania v časovom okne boli na výpočtovom uzle vykonávané samostatne. V rámci časového intervalu, počas ktorého boli vykonávané samostatne sme vypočítali priemer záťaže z údajov o výpočtovom uzle. Priemer predstavuje záťaž, ktorú sme úlohe priradili.
2. V druhom kroku sme určili záťaž úloh, ktoré počas svojho vykonávania v časovom okne nikdy neboli na uzle vykonávané samostatne, ale aspoň istú časť svojho vykonávania boli vykonávané spolu s úlohou, ktorej záťaž sa podarilo určiť v prvom kroku. Záťaž sme rovnako ako v prvom kroku vypočítali ako priemer záťaže z údajov o uzle, ale navyše sme odpočítali záťaž súbežne vykonávanej úlohy z kroku 1.
3. V každom ďalšom kroku sme postupovali iteratívne presne tak isto, až kým sme neurčili záťaž všetkých úloh.

Pre lepšie pochopenie uvádzame obrázok 5.1, ktorý ilustruje vykonávanie úloh na jednom z výpočtových uzlov.




Obrázok 5.1 Návrh výpočtu záťaže úlohy

Na výpočtovom uzle, ktorý obsahuje 3 jadrá sú vykonávané úlohy A až F. Úloha A je vykonávaná samostatne, takže celú záťaž uzla v danej chvíli spôsobuje jedine táto úloha. Preto môžeme ako záťaž úlohy A určiť priemernú hodnotu záťaže uzla počas vykonávania tejto úlohy.

Úlohy B a C sa čiastočne prekrývajú, ale obe sa na nejaký čas v systéme ocitnú samostatne. Záťaž úlohy B je možné určiť ako priemer záťaží uzla pred začatím vykonávania úlohy C. Záťaž úlohy C je možné určiť ako priemer záťaží uzla po ukončení vykonávania úlohy B.

Pre úlohy A až C stačilo použiť krok 1) z vyššie uvedeného algoritmu. Takisto záťaž úlohy D je možné vypočítať ako priemer záťaží uzla pred začatím úlohy E a po skončení jej vykonávania.

Úlohy E a F sa v systéme nikdy samostatne neocitnú, ich záťaž sa dá ale určiť podľa algoritmu v krokoch 2 a 3. Počas vykonávania úlohy E je záťaž na uzle spôsobená súbežne vykonávanými úlohami D a E. Záťaž, ktorú spôsobuje úloha D sme už zistili. Záťaž úlohy E teda vieme určiť ako priemer záťaží uzla počas vykonávania úlohy E mínus záťaž úlohy D. Rovnako záťaž úlohy F vieme určiť ako priemer záťaží uzla počas jej vykonávania mínus záťaže úloh D a E.

### 5.4.2 Návrh na modifikáciu algoritmu na výpočet dopravnej úlohy

Na optimalizáciu rozvrhu úloh sme navrhli využiť algoritmus na výpočet dopravnej úlohy. Štandardný algoritmus na výpočet dopravnej úlohy [36] má niekoľko obmedzení, ktoré znemožňujú jeho použitie pri úlohe dynamického rozvrhovania úloh a preto sme ich museli odstrániť. Medzi tieto obmedzenia patrí:

- množina úloh je vopred známa a nemenná,
- matica cien je vopred známa a nemenná,
- algoritmus začína v nulovom stave, žiadna úloha ešte nie je vykonávaná,
- algoritmus končí priradením všetkých úloh na výpočtové uzly, pričom cena priradenia je pre danú množinu výpočtových uzlov, úloh a pre danú maticu cien najnižšia.

Vo všeobecnosti je možné povedať, že štandardný algoritmus na výpočet dopravnej úlohy by bol ideálnym riešením pri statickom rozvrhovaní úloh (pozri kapitolu 1.4). Ako sme už uviedli, klaster UMB využíva dynamické rozvrhovanie, preto sme navrhli a vykonali modifikáciu algoritmu na výpočet dopravnej úlohy tak, aby bol použiteľný na dynamicky sa meniaci systém.

Navrhovaná modifikácia spočíva v tom, že optimalizačný algoritmus sa nevolá jednorazovo, ako pri štandardnom algoritme na výpočet dopravnej úlohy, ale volá sa v cykle, pričom pri každej iterácii dostane na vstupe úlohu, ktorá do systému práve dynamicky vstúpila. Zároveň je potrebné uchovávať aktuálne dostupné kapacity výpočtového systému v poli dostupných výpočtových jadier. V rámci každej iterácie modifikovaný algoritmus vykoná nasledujúce kroky:

1. V prvom kroku algoritmus hľadá úlohy, ktoré od predošlej iterácie skončili.
2. Pre každú úlohu, ktorá od predchádzajúcej iterácie cyklu skončila algoritmus odstráni jej záťaž z matice cien a zároveň vráti naspäť do poľa dostupných výpočtových jadier všetky výpočtové jadrá, ktoré úloha využívala.
3. Na základe záťaže úlohy, ktorá do systému dynamicky vstúpila algoritmus prepočíta maticu cien na aktuálne hodnoty.
4. V tomto bode sa zavolá štandardný algoritmus na výpočet dopravnej úlohy, ktorý priradí úlohu na výpočtový uzol, s ohľadom na najnižšie množstvo spotrebovanej elektrickej energie.
5. Modifikovaný algoritmus z poľa dostupných výpočtových jadier odráta výpočtové jadrá, ktoré pre úlohu alokoval.

Pre lepšiu ilustráciu tohto postupu uvádzame pseudokód 5.2:

```
1   for novaUloha in poleVsetkychUloh:
2     poleUlohNaOdstranenie = []
3     for vykonavanaUloha in poleVykonavanychUloh:
4       if vykonavanaUloha->koniec < novaUloha->zaciatok:
5         ulohyNaOdstranenie.append(vykonavanaUloha)
6     for ulohaNaOdstranenie in poleUlohNaOdstranenie:
7       poleVykonavanychUloh.pop(ulohaNaOdstranenie)
8       vratJadra(poleDostupnychJadier, ulohaNaOdstranenie)
9       prepocitajMaticuCien(maticaCien, ulohaNaOdstranenie)
10    prepocitajMaticuCien(maticaCien, novaUloha)
11    umiestniUlohu(novaUloha)
12    odoberJadra(poleDostupnychJadier, novaUloha)
```

Pseudokód 5.2 Návrh ma modifikáciu algoritmu na výpočet dopravnej úlohy

## 5.5 Návrh implementácie optimalizačného modelu rozvrhovania úloh

Pre implementáciu optimalizačného modelu rozvrhovania úloh navrhnutého v kapitole 5.4 navrhujeme využiť modifikovaný algoritmus dopravnej úlohy v spolupráci s optimalizačnými nástrojmi spoločnosti Google, ktoré sú opísané

v kapitole 3.4. Optimalizačný model rozvrhovania úloh navrhujeme implementovať v jazyku Python na lokálnom počítači s operačným systémom Ubuntu 16.04 LTS, dvomi procesormi Intel Core I5 s taktovacou frekvenciou 2,5 GHz a operačnou pamäťou 6 GB. Navrhujeme ako vstup použiť predspracované údaje o výpočtových uzloch a vykonávaných úlohách vo formáte .csv, dostupné na lokálnom pevnom disku. Výstup z optimalizačného modelu rozvrhovania úloh navrhujeme naformátovať ako .csv súbor.

# 6 IMPLEMENTÁCIA APLIKÁCIE PRE ANALÝZU SPOTREBY ELEKTRICKEJ ENERGIE KLASTRA

Cieľom tejto kapitoly je podrobne priblížiť proces vytvárania aplikácie pre analýzu údajov vysokovýkonného počítačového klastra UMB. Pri implementácii aplikácie sa využívajú technológie opísané v kapitole 3 a vstupné údaje podrobne opísané v kapitole 4.3. Implementačný proces vychádza z návrhu aplikácie uvedeného v 5. kapitole a podrobnejšie rozpracovaného v jej podkapitolách. Kapitola je štruktúrovaná nasledovne:

- kapitola 6.1 obsahuje informácie o predspracovaní vstupných údajov o výpočtových prostriedkoch klastra a o zlúčení údajov o záťaži výpočtových uzlov a spotrebe elektrickej energie,
- v kapitolách 6.2 a 6.3 sa nachádzajú informácie o vytváraní a overovaní modelu spotreby elektrickej energie,
- kapitola 6.4 obsahuje informácie o predspracovaní údajov o úlohách a zlúčení údajov o úlohách s údajmi o výpočtových uzloch,
- v kapitole 6.5 opisujeme vytvorenie optimalizačného modelu rozvrhovania úloh,
- v kapitole 6.6 sa nachádzajú informácie o implementácií optimalizačného modelu rozvrhovania úloh.

## 6.1 Zlúčenie údajov o záťaži a údajov o spotrebe

Z monitorovacieho nástroju Zabbix máme k dispozícii veľké množstvo údajov, pričom nie všetky zaznamenávané údaje sú pre nás potrebné. Navyše, tieto údaje nie sú vo formáte, ktorý je vhodný pre analýzu. Na to, aby sme vstupné údaje dostali do formátu, v ktorom ich vieme analyzovať je potrebné vykonať dva kroky:

1. v prvom kroku je potrebné vstupné údaje predspracovať,
2. v druhom kroku je potrebné vykonať zlúčenie údajov z dvoch vstupných súborov.

### 6.1.1 Predspracovanie vstupných údajov o výpočtových prostriedkoch klastra

Cieľom predspracovania je upraviť vstupné údaje do formátu, v ktorom je možné vykonať ich zlúčenie. Predspracovanie bolo vykonané osobitne na údajoch o:

- záťaži na výpočtových uzloch,

- spotrebe elektrickej energie výpočtových uzlov.

Na predspracovanie sme vytvorili viacero Google BigQuery dopytov, ktoré sú v ďalších častiach kapitoly opísané vo forme pseudokódov.

#### *6.1.1.1 Predspracovanie údajov o záťaži na výpočtových uzloch*

V pseudokóde 6.1 uvádzame dopyt, ktorý sme vytvorili na predspracovanie údajov o záťaži výpočtových uzlov klastra UMB pre vysokovýkonné výpočty.

```
SELECT
  FORMAT_DATETIME("%Y-%m-%d %H:%M", loadDateTime) AS loadTime,
  loadComp,
  ROUND(AVG(loadValue), 2) AS loadValueRound
FROM load
GROUP BY loadTime, loadComp
ORDER BY loadTime, loadComp
```

Pseudokód 6.1 Skript na predspracovanie údajov o záťaži výpočtových uzlov

Zo všetkých údajov o záťaži, ktorých ukážku je možné vidieť na obrázku 4.3 vyberáme údaje o:

- čase merania (parameter *loadTime*), ktorý pomocou funkcie *FORMAT_DATETIME* formátujeme tak, aby sme vynechali údaje o sekundách, napríklad 2018-01-01 08:00,
- výpočtovom uzle (parameter *loadComp*),
- zaokrúhlenom aritmetickom priemere záťaží na výpočtovom uzle (parameter *loadValueRound*), ktorý je zoskupený podľa zvyšných uvedených parametrov.

Týmto postupom získame priemernú hodnotu nameranej záťaže pre každú minútu na každom skúmanom výpočtovom uzle. Ukážku výstupných údajov po vykonaní predspracovania uvedeného v pseudokóde 6.1 je možné vidieť na obrázku 6.1.

| loadTime | loadComp | loadValueRound |
|---|---|---|
| 2017-11-07 00:26 | comp16 | 0.04 |
| 2017-11-07 00:26 | comp17 | 0.0 |
| 2017-11-07 00:26 | comp18 | 0.02 |
| 2017-11-07 00:26 | comp19 | 0.02 |
| 2017-11-07 00:26 | comp20 | 0.0 |
| 2017-11-07 00:26 | comp21 | 0.0 |
| 2017-11-07 00:26 | comp22 | 0.0 |
| 2017-11-07 00:26 | comp25 | 15.51 |
| 2017-11-07 00:27 | comp01 | 0.0 |
| 2017-11-07 00:27 | comp02 | 0.1 |
| 2017-11-07 00:27 | comp03 | 0.0 |
| 2017-11-07 00:27 | comp04 | 0.0 |
| 2017-11-07 00:27 | comp06 | 0.0 |
| 2017-11-07 00:27 | comp07 | 0.02 |

Obrázok 6.1 Ukážka údajov o záťaži výpočtových uzlov po ich predspracovaní

Predspracovanie údajov o záťaži výpočtových uzlov výrazne znížilo ich objem. Z takmer 6 GB údajov o záťaži výpočtových uzlov, ktoré boli umiestnené v takmer 86 miliónoch riadkov databázových záznamov po predspracovaní zostalo len 232 MB údajov v 7 150 000 databázových záznamoch. Výstupné údaje sme vyexportovali z Google BigQuery vo formáte *.csv*, aby sme s nimi mohli ďalej pracovať mimo tohto nástroja.

#### *6.1.1.2 Predspracovanie údajov o spotrebe elektrickej energie výpočtových uzlov*

Pri predspracovaní údajov o spotrebe elektrickej energie sme najskôr pridali k vstupným údajom atribút, ktorý obsahuje spotrebu elektrickej energie ako číselnú hodnotu, pretože vo vstupných údajoch sa zistená spotreba elektrickej energie uvádzala ako textový reťazec aj so značkou jednotky spotreby elektrickej energie W (pozri obrázok 4.4). Pre túto časť predspracovania sme vytvorili skript, ktorý uvádzame v pseudokóde 6.2.

```
UPDATE power
SET powerValue =
  CAST(SUBSTR(powerReading, 1, CHAR_LENGTH(powerReading)-2) AS FLOAT)
WHERE 1=1
```

Pseudokód 6.2 Skript pre orezanie a pretypovanie textu s hodnotou spotreby elektrickej energie na desatinné číslo

V pseudokóde 6.3 uvádzame dopyt, ktorý sme vytvorili na vyselektovanie údajov o:

- čase merania (parameter *powerTime*), ktorý pomocou funkcie *FORMAT_DATETIME* formátujeme tak, aby sme vynechali údaje o sekundách, napríklad 2018-01-01 08:00,
- výpočtovom uzle (parameter *powerHostname*),
- zaokrúhlenom aritmetickom priemere spotreby elektrickej energie na výpočtovom uzle (parameter *powerValueRound*), ktorý je zoskupený podľa zvyšných uvedených parametrov.

```
SELECT
  FORMAT_DATETIME("%Y-%m-%d %H:%M", powerTimestamp) AS powerTime,
  powerHostname,
  ROUND(AVG(powerValue), 2) AS powerValueRound
FROM power
GROUP BY powerTime, powerHostname
ORDER BY powerTime, powerHostname
```

Pseudokód 6.3 Skript na predspracovanie údajov o spotrebe elektrickej energie

Týmto postupom získame priemernú hodnotu nameranej spotreby elektrickej energie pre každú minútu na každom skúmanom výpočtovom uzle. Ukážku výstupných údajov po vykonaní predspracovania uvedeného v pseudokóde 6.3 je možné vidieť na obrázku 6.2.

| powerTime | powerHostname | powerValueRound |
|---|---|---|
| 2017-11-18 15:58 | comp27 | 195.0 |
| 2017-11-18 15:58 | comp28 | 140.0 |
| 2017-11-18 15:58 | comp29 | 165.0 |
| 2017-11-18 15:58 | comp30 | 215.0 |
| 2017-11-18 15:58 | comp31 | 265.0 |
| 2017-11-18 15:58 | comp32 | 200.0 |
| 2017-11-18 15:58 | comp33 | 210.0 |
| 2017-11-18 15:58 | comp34 | 230.0 |
| 2017-11-18 15:58 | comp35 | 195.0 |
| 2017-11-18 15:58 | comp36 | 155.0 |
| 2017-11-18 15:58 | comp37 | 170.0 |
| 2017-11-18 15:58 | comp38 | 175.0 |
| 2017-11-18 15:58 | comp39 | 235.0 |
| 2017-11-18 15:58 | comp40 | 225.0 |
| 2017-11-18 15:58 | comp41 | 90.0 |

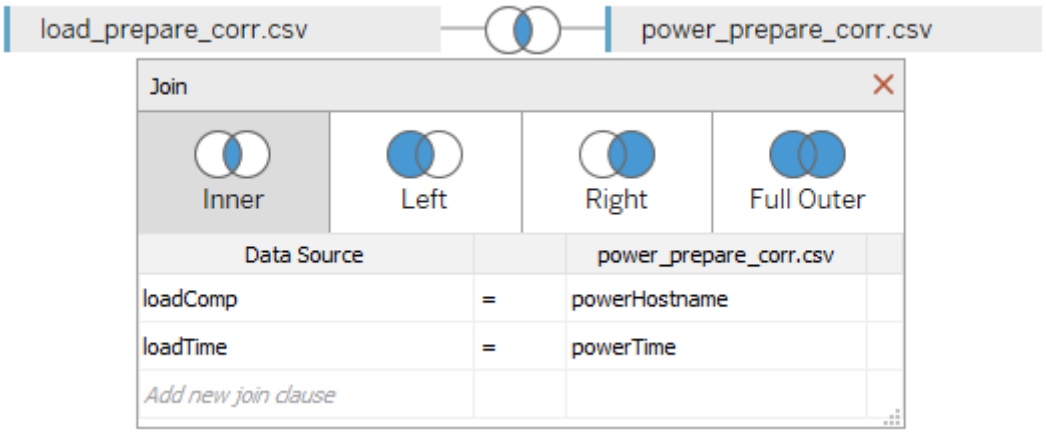
Obrázok 6.2 Ukážka údajov o spotrebe elektrickej energie po ich predspracovaní

Predspracovanie údajov o spotrebe elektrickej energie výpočtových uzlov znížilo ich objem, hoci nie tak výrazne ako pri predspracovaní údajov o záťaži výpočtových uzlov. Vzorkovacia frekvencia meraní záťaže je vyššia ako pri meraní spotreby elektrickej energie a teda záznamov za minútu, ktoré sa pri údajoch o záťaži výpočtových uzlov spriemerujú, je viac.

Z 518 MB údajov o spotrebe elektrickej energie výpočtových uzlov, ktoré boli umiestnené v takmer 6,8 miliónoch riadkov databázových záznamov po predspracovaní zostalo 210 MB údajov v takmer 6 500 000 databázových záznamoch. Výstupné údaje sme vyexportovali z Google BigQuery vo formáte *.csv*, aby sme s nimi mohli ďalej pracovať mimo tohto nástroja.

### 6.1.2 Zlúčenie údajov o výpočtových prostriedkoch klastra

Po predspracovaní údajov v cloudovej služby Google BigQuery a ich následnom exportovaní do dvoch súborov s údajmi o záťaži a o spotrebe elektrickej energie výpočtových uzlov môžeme vykonať zlúčenie údajov z týchto .csv súborov. Vďaka zlúčeniu údajov do jedného súboru je možné vytvoriť modely spotreby elektrickej energie. Na zlúčenie údajov sme použili príkaz *INNER JOIN*, keďže nás zaujímajú len tie údaje, pri ktorých sú známe hodnoty oboch parametrov. Parametre príkazu INNER JOIN pri zlúčení údajov o záťaži a spotrebe elektrickej energie výpočtových uzlov môžeme vidieť na obrázku 6.3. Zlúčenie údajov sme vykonali priamo v analytickom nástroji Tableau, ktorý sme využili aj neskôr na vytvorenie modelu spotreby elektrickej energie.




Obrázok 6.3 Zlúčenie súborov s údajmi o výpočtových prostriedkoch klastra

Ukážku zlúčených údajov je možné vidieť na obrázku 6.4.

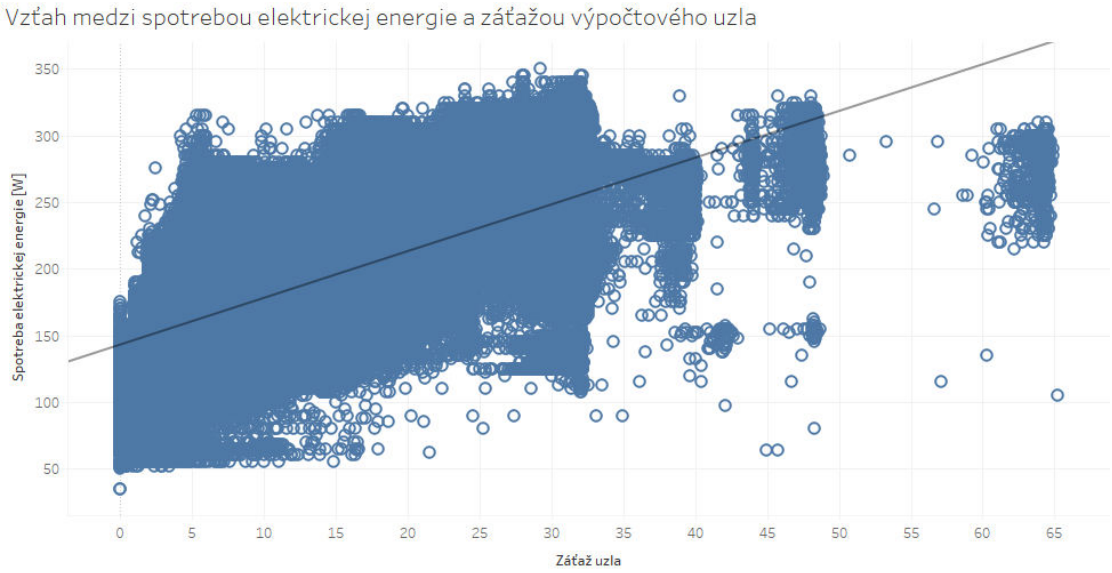
| loadTime | loadComp | loadValueRound | powerTime | powerHostname | powerValueRound |
|---|---|---|---|---|---|
| 18.11.2017 15:58:00 | comp01 | 8,0900 | 18.11.2017 15:58:00 | comp01 | 124,000 |
| 18.11.2017 15:58:00 | comp02 | 8,4700 | 18.11.2017 15:58:00 | comp02 | 232,000 |
| 18.11.2017 15:58:00 | comp03 | 9,5600 | 18.11.2017 15:58:00 | comp03 | 232,000 |
| 18.11.2017 15:58:00 | comp06 | 10,0200 | 18.11.2017 15:58:00 | comp06 | 204,000 |
| 18.11.2017 15:58:00 | comp07 | 8,3000 | 18.11.2017 15:58:00 | comp07 | 232,000 |
| 18.11.2017 15:58:00 | comp10 | 2,0000 | 18.11.2017 15:58:00 | comp10 | 116,000 |
| 18.11.2017 15:58:00 | comp11 | 9,7500 | 18.11.2017 15:58:00 | comp11 | 120,000 |
| 18.11.2017 15:58:00 | comp12 | 10,2800 | 18.11.2017 15:58:00 | comp12 | 236,000 |
| 18.11.2017 15:58:00 | comp13 | 0,0000 | 18.11.2017 15:58:00 | comp13 | 68,000 |
| 18.11.2017 15:58:00 | comp14 | 0,0200 | 18.11.2017 15:58:00 | comp14 | 64,000 |
| 18.11.2017 15:58:00 | comp15 | 0,0000 | 18.11.2017 15:58:00 | comp15 | 64,000 |
| 18.11.2017 15:58:00 | comp16 | 0,0000 | 18.11.2017 15:58:00 | comp16 | 68,000 |

Obrázok 6.4 Ukážka zlúčených údajov o výpočtových prostriedkoch klastra

Veľkosť údajov po zlúčení je vyše 400 MB, ktoré sú obsiahnuté v takmer 6 500 000 zlúčených záznamoch.

## 6.2 Vytvorenie modelu spotreby elektrickej energie klastra

Na zlúčených údajoch sme hľadali vzťah medzi záťažou na uzle a spotrebou elektrickej energie. Vytvorili sme viacero regresných modelov, ktorých výsledky je možné vidieť na obrázkoch 6.5 až 6.8. V každom z nich sa na osi *X* nachádza nameraná záťaž uzla a na osi *Y* nameraná spotreba elektrickej energie vo wattoch. Modré krúžky predstavujú záznamy zo zlúčených údajov.

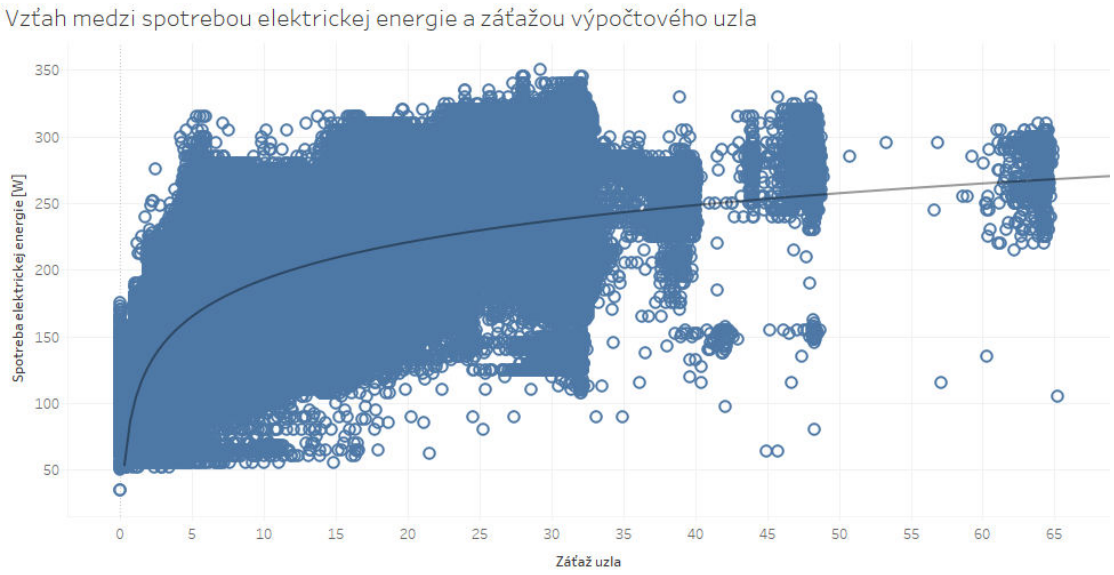



Obrázok 6.5 Model spotreby elektrickej energie pri využití lineárnej regresie

Prvý regresný model, ktorý sme vytvorili využíva na modelovanie vzťahu medzi záťažou výpočtového uzla a jeho spotrebou elektrickej energie regresnú priamku. Koeficienty vytvorenej regresnej priamky sú uvedené vo vzťahu 6.1:

$$y = 3{,}5075x + 142{,}966 \qquad (6.1),$$

kde $x$ predstavuje záťaž výpočtového uzla a $y$ predstavuje spotrebu elektrickej energie.

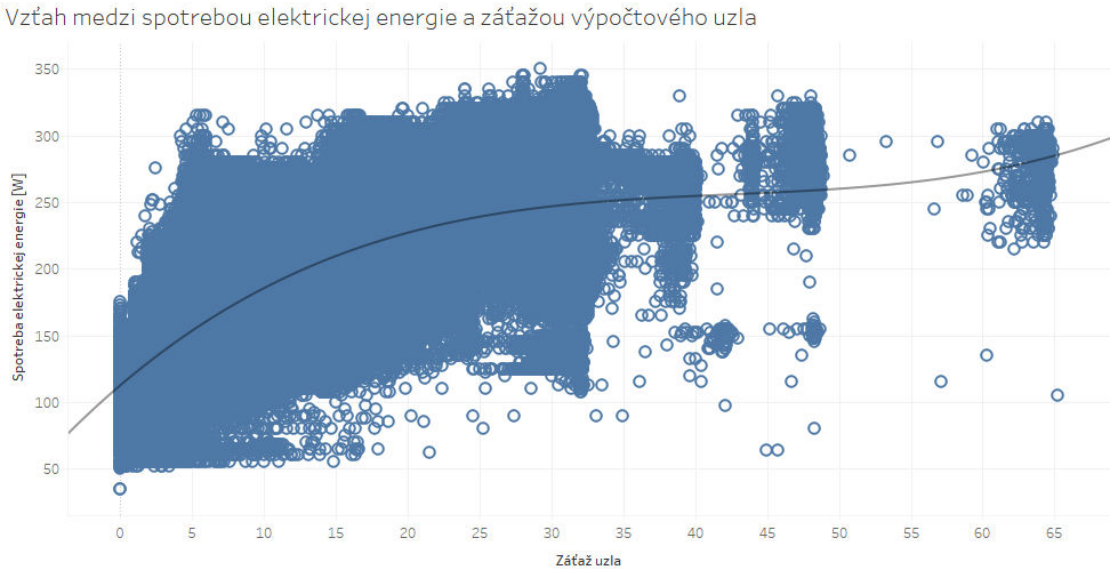



Obrázok 6.6 Model spotreby elektrickej energie pri využití nelineárnej regresie pomocou logaritmickej funkcie

Druhý regresný model, ktorý sme vytvorili opisuje vzťah medzi záťažou a spotrebou elektrickej energie výpočtového uzla logaritmickou funkciou. Koeficienty logaritmickej krivky sa nachádzajú vo vzťahu 6.2:

$$y = 40{,}2216 * \ln x + 100{,}015 \qquad (6.2),$$

kde $x$ predstavuje záťaž výpočtového uzla a $y$ predstavuje spotrebu elektrickej energie.

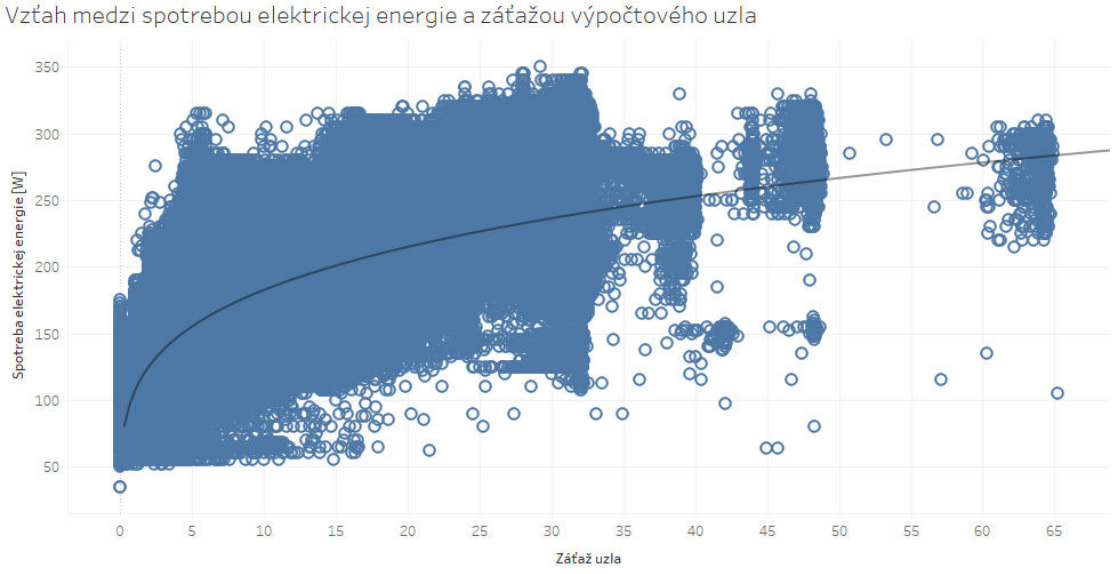



Obrázok 6.7 Model spotreby elektrickej energie pri využití polynomiálnej regresie

V poradí tretí regresný model, ktorý sme vytvorili využíva na opísanie vzťahu medzi záťažou a spotrebou elektrickej energie výpočtového uzla polynóm tretieho stupňa. Koeficienty polynómu môžeme vidieť vo vzťahu 6.3:

$$y = 0{,}0016 * x^3 - 0{,}2056 * x^2 + 9{,}2016x + 112{,}125 \quad (6.3),$$

kde $x$ predstavuje záťaž výpočtového uzla a $y$ predstavuje spotrebu elektrickej energie.




Obrázok 6.8 Model spotreby elektrickej energie pri využití nelineárnej regresie pomocou mocninovej funkcie

Posledný vytvorený regresný model využíva na opísanie vzťahu medzi záťažou a spotrebou elektrickej energie výpočtového uzla mocninovú funkciu. Koeficienty mocninovej funkcie sú uvedené vo vzťahu 6.4:

$$y = 106{,}106 * x^{0{,}2354} \quad (6.4),$$

## 6.3 Overenie a porovnanie vytvorených modelov spotreby elektrickej energie

Dôležitým pojmom pri štatistickom spracovaní údajov je P-hodnota (angl. *p-value*). P-hodnota vyjadruje pravdepodobnosť, že vzťah zistený pri tvorbe štatistického modelu je náhodný. Čím je P-hodnota menšia, tým je prepojenie medzi hodnotami závislej a nezávislej premennej (nazývaná tiež prediktor) výraznejšie, hodnota závislej premennej viac závisí od hodnoty nezávislej premennej. P-hodnota menšia alebo rovná 0,05 je v praxi často považovaná za dostatočnú, vytvára 95 % štatistický interval spoľahlivosti (nazývaný tiež aj konfidenčný interval). [37]

Pri všetkých modeloch spotreby elektrickej energie vytvorených v kapitole 6.2 je P-hodnota menšia ako 0,0001, takže závislosť medzi záťažou výpočtového uzla a jeho spotrebou elektrickej energie je štatisticky významná.

Porovnanie vytvorených modelov vykonávame v tabuľkovom procesore Microsoft Excel na základe vzťahov uvedených v návrhu overenia správnosti modelu spotreby energie klastra v kapitole 5.3. Priemernú absolútnu a priemernú relatívnu chybu regresných modelov uvádzame v tabuľke 6.1.

Tabuľka 6.1 Porovnanie vytvorených modelov spotreby elektrickej energie

| **Model** | Vzťah (6.1) | Vzťah (6.2) | Vzťah (6.3) | Vzťah (6.4) |
|---|---|---|---|---|
| Použitá funkcia | Lineárna funkcia | Logaritmická funkcia | Polynomiálna funkcia | Mocninová funkcia |
| Priemerná absolútna chyba modelu | 34,81 W | 22,7 W | 21,08 W | 23,56 W |
| Priemerná relatívna chyba modelu | 15,55 % | 10,14 % | 9,42 % | 10,53 % |

Z porovnania vidíme, že najlepšie spotrebu elektrickej energie modeluje vzťah 6.3, ktorý využíva polynomiálnu funkciu.

## 6.4 Zlúčenie údajov o vykonávaných úlohách a výpočtových uzloch klastra

Na zlúčenie údajov o vykonávaných úlohách, ktorých náhľad sa nachádza na obrázku 4.1 a výpočtových uzloch klastra (pozri obrázok 6.4) potrebujeme údaje

o vykonávaných úlohách najskôr predspracovať do požadovaného formátu, rovnako ako sme predspracovali aj údaje o výpočtových uzloch.

### 6.4.1 Predspracovanie údajov o úlohách

Je potrebné predspracovať hodnoty atribútu *JobHostsFullText*, v ktorom sa nachádza zoznam výpočtových uzlov a jadier, na ktorých bola úloha vykonávaná vo formáte:

comp06-0/4+comp06-0/5+ comp07-0/1+comp07-0/2,

z ktorého vidíme, že ilustračná úloha bola vykonávaná na:

- výpočtových jadrách číslo 4 a 5 uzla *comp06*,
- výpočtových jadrách číslo 1 a 2 uzla *comp07*.

Úlohu, ktorá má v atribúte *JobHostsFullText* viac ako jeden rôzny výpočtový uzol potrebujeme rozdeliť na viacero databázových záznamov, z ktorých každý má priradený len jeden unikátny výpočtový uzol, aby sme ju v rámci optimalizačného modelu rozvrhovania úloh vedeli správne rozvrhnúť. Konkrétne výpočtové jadrá pri analýze nie sú dôležité, dôležitý je len ich počet. Na rozdelenie sme vytvorili funkciu v node.js, ktorú uvádzame v pseudokóde 6.4.

```
1  function get_hosts(val, writer) {
2   job_hosts = val.hosts.split("+");
3   unique_hosts = [];
4   for (j = 0; j < job_hosts.length; j++) {
5      host_name = job_hosts[j].substr(0, 6);
6      if (unique_hosts.indexOf(host_name) === -1) {
7        unique_hosts.push(host_name);
8      }
9   }
10  for (j = 0; j < unique_hosts.length; j++) {
11    writer.write(
12      {
13        jobId:             val.jobId,
14        jobStarted:       val.jobStarted.toISOString().replace('T', ' ').substr(0, 19),
15        jobFinished:      val.jobFinished.toISOString().replace('T', ' ').substr(0, 19),
16        nodeCount:        val.nodeCount,
17        host:              unique_hosts[j],
18        executionSlots: val.executionSlots / val.nodeCount,
19      });
20  }
21 }
```

Pseudokód 6.4 Funkcia na vytváranie záznamov s jedinečným výpočtovým uzlom pre každú úlohu

Funkcia *get_hosts()* prijíma dva vstupné parametre:

- *val* je objekt obsahujúci databázový záznam (jeden riadok z obrázku 4.1),
- *writer* je objekt, ktorý umožňuje zapisovanie do výstupného .csv súboru.

Na riadku 2 uvedeného pseudokódu do premennej *job_hosts* priraďujeme pole reťazcov, ktoré vznikne rozdelením jedného reťazca *val.hosts* (premenná obsahujúca reťazec *JobHostsFullText*) v mieste výskytu znaku „+“. Riadok 3 obsahuje vytvorenie prázdneho poľa v premennej *unique_hosts*. Na riadkoch 4 až 9 sa nachádza cyklus, v ktorom iterujeme cez pole reťazcov *job_hosts*¸ do premennej *host_name* vkladáme podreťazec obsahujúci názov výpočtového uzla a ak sa tento výpočtový uzol ešte nevyskytuje v poli výpočtových uzlov *unique_hosts*, tak ho tam vložíme. Na zvyšných riadkoch pseudokódu je uvedený ďalší cyklus, v ktorom prechádzame cez všetky výpočtové uzly z poľa *unique_hosts* a pre každý jedinečný výpočtový uzol zapisujeme do výstupného .csv súboru údaje:

- parameter *jobId,* ktorý obsahuje identifikačné číslo úlohy,
- parameter *jobStarted,* ktorý obsahuje čas začiatku vykonávania úlohy,
- parameter *jobFinished,* ktorý obsahuje čas ukončenia vykonávania úlohy,
- parameter *nodeCount,* ktorý obsahuje počet rôznych výpočtových uzlov, na ktorých bola úloha vykonávaná,
- parameter *host,* ktorý obsahuje jedinečný výpočtový uzol,
- parameter *executionSlots,* ktorý obsahuje počet výpočtových jadier na jedinečnom výpočtovom uzle.

Predspracovaním údajov o vykonávaných úlohách sa zvýšil počet záznamov o úlohách z 15 433 na 15 552. Predspracované údaje o vykonávaných úlohách sú odosielané do Google BigQuery cez REST API vo forme .csv súboru. Ukážku predspracovaných údajov je možné vidieť na obrázku 6.9.

| jobId | jobStarted | jobFinished | nodeCount | host | executionSlots |
|---|---|---|---|---|---|
| 138807 | 2018-04-05T14:10:50 | 2018-04-05T14:44:51 | 1 | comp01 | 1 |
| 136421 | 2018-02-11T13:28:09 | 2018-02-12T00:31:41 | 1 | comp01 | 1 |
| 137181 | 2018-02-19T22:11:13 | 2018-02-21T23:23:20 | 1 | comp01 | 1 |
| 137235 | 2018-02-21T22:57:56 | 2018-02-22T05:41:59 | 1 | comp01 | 1 |
| 136425 | 2018-02-11T13:28:26 | 2018-02-12T03:45:10 | 1 | comp01 | 1 |
| 124723 | 2017-12-01T13:39:48 | 2017-12-02T13:52:48 | 1 | comp01 | 1 |
| 129766 | 2017-12-11T14:48:48 | 2017-12-11T15:01:40 | 1 | comp01 | 1 |

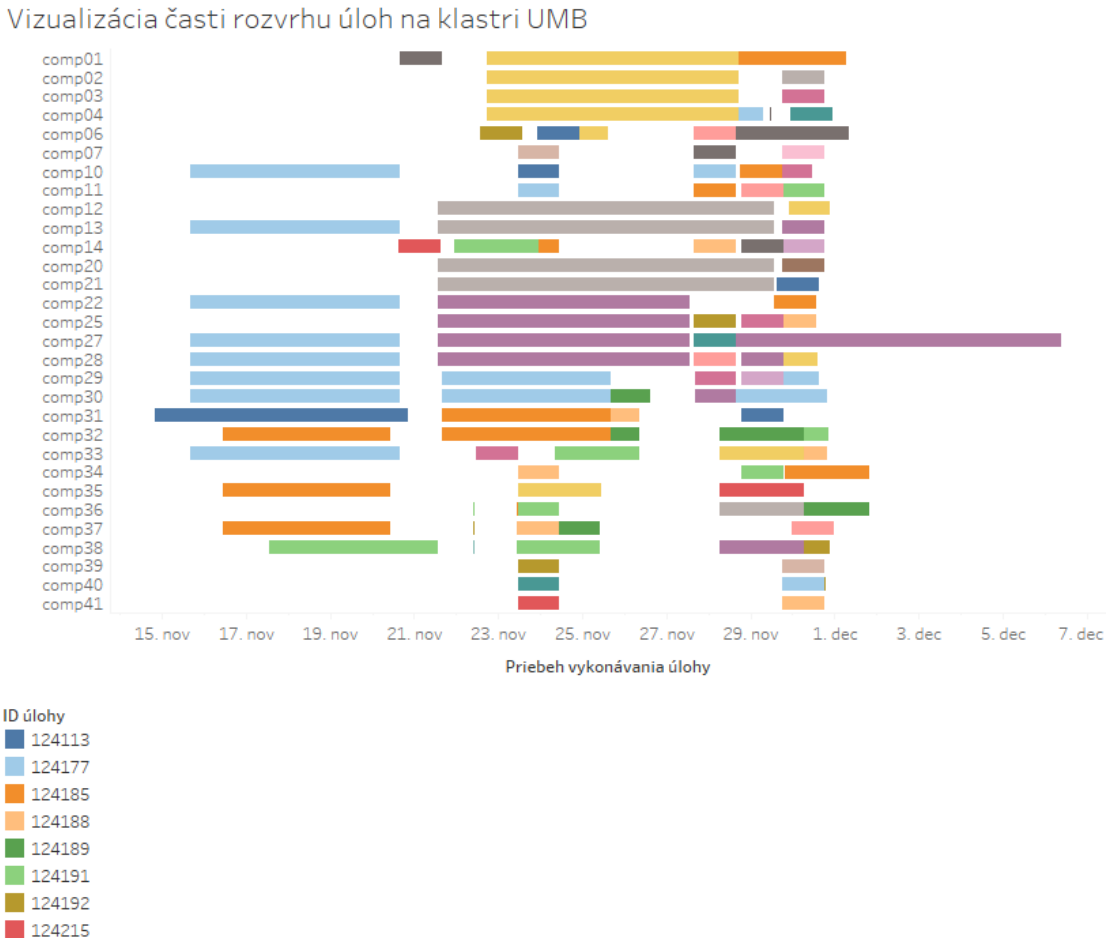
Obrázok 6.9 Ukážka výstupu z predspracovania údajov o úlohách

## 6.5 Vytvorenie optimalizačného modelu rozvrhovania úloh

Vďaka modelu spotreby elektrickej energie vytvorenému v kapitole 6.2 je možné odhadovať, koľko elektrickej energie sa pri určitom zaťažení výpočtového uzla spotrebuje. Zaťaženie výpočtového uzla spôsobujú najmä vykonávané úlohy. Na vytvorenie optimalizačného modelu rozvrhovania úloh potrebujeme pre každú úlohu zistiť, akú veľkú záťaž na výpočtovom uzle spôsobila. Na to potrebujeme vedieť, ako vyzeral pôvodný rozvrh úloh, preto ho musíme spätne zložiť.

### 6.5.1 Spätné skladanie pôvodného rozvrhu úloh

Na spätné skladanie pôvodného rozvrhu využívame údaje o úlohách vykonávaných na klastri UMB. Časť spätne zloženého rozvrhu uvádzame na vizualizácii v podobe Ganttovho diagramu na obrázku 6.10.




Obrázok 6.10 Vizualizácia časti spätne zloženého pôvodného rozvrhu

Na *Y*-ovej osi sa nachádza zoznam uzlov klastra UMB, ktoré boli v období od polovice novembra do začiatku decembra v prevádzke a zároveň na nich boli vykonávané úlohy. Ide len o časť celého skúmaného časového obdobia, pretože

vizualizovať celé 3 a pol mesačné obdobie by bolo ťažké a výsledná vizualizácia by bola neprehľadná. Na *X*-ovej osi sa nachádzajú dátumy. Diagram znázorňuje priebeh vykonávania úloh na rôznych výpočtových uzloch. Rôzne vykonávané úlohy sú v princípe označené rôznymi farbami, ale farebná paleta nie je dostatočná, pretože úloh je veľké množstvo, takže sa niektoré farby opakujú. V legende sa takisto nachádza len 8 vysvetliviek napriek tomu, že úloh v druhej polovici novembra je niekoľko desiatok, pretože zmyslom vizualizácie je poukázať na priebeh obsadenia systému, nie na podrobnosti o tom kedy, a na ktorom výpočtovom uzle bola konkrétna úloha vykonávaná. Na diagrame nie sú viditeľné všetky vykonávané úlohy, pretože niektoré úlohy sú vykonávané súbežne na jednom výpočtovom uzle alebo sú príliš krátke na to, aby boli na diagrame viditeľné. Môžeme takisto vidieť, že niektoré úlohy sú vykonávané na viacerých výpočtových uzloch súbežne. Túto skutočnosť je potrebné brať do úvahy pri implementácii optimalizačného modelu rozvrhovania úloh v kapitole 6.6.

### 6.5.2 Zlúčenie údajov o vykonávaných úlohách a výpočtových uzloch klastra

Zlúčenie údajov o vykonávaných úlohách a výpočtových uzloch klastra je súčasťou Google BigQuery skriptu, ktorý slúži na spätné určenie záťaže pre vykonávané úlohy (pozri riadky 6 až 9 pseudokódov 6.5 a 6.6).

### 6.5.3 Spätné určenie záťaže pre vykonávané úlohy

Spätné určenie záťaže, ktorú vykonávaná úloha na výpočtovom uzle spôsobila potrebujeme na to, aby sme pri rozvrhovaní úloh mohli úlohu umiestniť na taký výpočtový uzol, na ktorom by spotrebovala najmenej elektrickej energie. Záťaž, ktorú úloha na výpočtovom uzle spôsobila je potrebné určiť vo viacerých krokoch:

1. V prvom kroku je potrebné určiť priemernú záťaž na výpočtovom uzle počas vykonávania každej úlohy zoskupenú podľa počtu súbežne vykonávaných úloh.
2. V druhom kroku je potrebné pre každú úlohu určiť priemernú záťaž pri najmenšom počte súbežne vykonávaných úloh.
3. V poslednom kroku je potrebné odčítať od seba záťaže súbežných úloh iteratívnym algoritmom, ktorý sme navrhli v kapitole 5.4.1.

#### 6.5.3.1 *Určenie priemernej záťaže na výpočtovom uzle počas vykonávania každej úlohy*

V prvom kroku bolo potrebné určiť priemernú záťaž počas vykonávania každej úlohy na klastri UMB a zároveň počet súbežne vykonávaných úloh. Vzhľadom na skutočnosť, že Google BigQuery skript, ktorý sme vytvorili je pomerne rozsiahly a komplexný, tak sme ho rozdelili na dva pseudokódy. V pseudokóde 6.5 sa nachádza vnútorný (vnorený) dopyt a v pseudokóde 6.6 sa nachádza hlavný (vonkajší) dopyt.

```
1  SELECT
2    load2.loadTime AS subquery_time,
3    jobs2.host AS subquery_host,
4    COUNT(DISTINCT jobs2.jobId) AS subquery_value
5  FROM jobs jobs2
6  INNER JOIN load  load2
7    ON load2.loadComp = jobs2.host AND
8      PARSE_DATETIME("%Y-%m-%d %H:%M", load2.loadTime)
9        BETWEEN jobs2.JobStarted AND jobs2.JobFinished
10 WHERE PARSE_DATETIME("%Y-%m-%d %H:%M", load2.loadTime)
11   BETWEEN jobs2.JobStarted AND jobs2.JobFinished
12 GROUP BY subquery_time, subquery_host
```

Pseudokód 6.5 Skript na určenie počtu súbežne vykonávaných úloh na výpočtovom uzle

Cieľom vnoreného dopytu uvedeného v pseudokóde 6.5 je pre každý časový okamih na každom uzle určiť počet súbežne vykonávaných úloh. V dopyte sú využívané zlúčené údaje o výpočtových uzloch klastra UMB a o vykonávaných úlohách (riadky 6 až 9 pseudokódu). Na riadkoch 10 a 11 pseudokódu sa nachádza podmienka, ktorá vyselektuje len tie úlohy, ktoré sú v danom časovom okamihu vykonávané. Riadok 4 pseudokódu spočíta všetky jedinečné identifikačné čísla úloh, ktoré sa v danom okamihu vykonávajú a sprístupní ich počet vonkajšiemu dopytu v podobe premennej *subquery_value*.

```
1  SELECT
2    jobs.jobId, jobs.jobStarted, jobs.jobFinished, jobs.host, jobs.executionSlots,
3    subquery_value AS SimultaneousJobs,
4    ROUND(AVG(load.loadValueRound), 2) AS avg_load
5  FROM jobs
6  INNER JOIN load
7    ON load.loadComp = jobs.host AND
8    PARSE_DATETIME("%Y-%m-%d %H:%M", load.loadTime)
9      BETWEEN jobs.JobStarted AND jobs.JobFinished
10 INNER JOIN subquery (pozri pseudokód 6.5)
11   ON load.loadTime = subquery_time AND jobs.host = subquery_host
12 GROUP BY
```

```
13   jobs.jobId, jobs.jobStarted, jobs.jobFinished, jobs.host, jobs.executionSlots,
14   subquery.value
15 ORDER BY jobs.JobId
```

Pseudokód 6.6 Skript na určenie priemernej záťaže na výpočtovom uzle počas vykonávania každej úlohy

Vonkajší dopyt využíva hodnoty premennej *subquery_value* z vnútorného dopytu, volanie ktorého sa nachádza na riadkoch 10 a 11. V riadkoch 6 až 9 je opäť možné vidieť zlúčenie údajov o vykonávaných úlohách a výpočtových uzloch klastra UMB. Hodnota *subquery_value* sa používa v príkaze *GROUP BY*, vďaka čomu získame viacero záznamov o priemernej záťaži pre jednu vykonávanú úlohu (vypočítava sa na riadku 4), podľa toho, ako sa menil počet vykonávaných úloh na výpočtovom uzle. Výstup z tohto dopytu využívame v dopyte na určenie priemernej záťaže pri najmenšom počte súbežne vykonávaných úloh (pozri kapitolu 6.5.3.2). Ukážku výstupných údajov je možné vidieť na obrázku 6.11.

| jobId | jobStarted | jobFinished | host | CountOfSimultaneousJobs | executionSlots | avg_load |
|---|---|---|---|---|---|---|
| 123471 | 2017-11-06T21:32:04 | 2017-11-07T20:40:16 | comp25 | 1 | 32 | 9.64 |
| 123472 | 2017-11-07T00:40:07 | 2017-11-07T01:34:08 | comp01 | 1 | 6 | 0.0 |
| 123473 | 2017-11-07T06:58:45 | 2017-11-07T07:03:52 | comp01 | 1 | 12 | 0.0 |
| 123474 | 2017-11-07T06:58:45 | 2017-11-07T07:04:01 | comp02 | 1 | 12 | 0.04 |
| 123475 | 2017-11-07T06:58:45 | 2017-11-07T07:03:48 | comp03 | 1 | 12 | 0.0 |
| 123476 | 2017-11-07T06:58:45 | 2017-11-07T07:04:03 | comp04 | 1 | 12 | 0.0 |
| 123477 | 2017-11-07T06:58:45 | 2017-11-07T07:03:47 | comp06 | 1 | 12 | 0.0 |
| 123478 | 2017-11-07T06:58:45 | 2017-11-07T07:03:52 | comp07 | 1 | 12 | 0.03 |

Obrázok 6.11 Ukážka výstupných údajov s priemernou hodnotou záťaže a počtami súbežne vykonávaných úloh

#### *6.5.3.2 Určenie priemernej záťaže pri najmenšom počte súbežne vykonávaných úloh*

V predchádzajúcom kroku sme vytvorili zoznam všetkých vykonávaných úloh spolu s počtami súbežne vykonávaných úloh a priemernými záťažami počas ich vykonávania. Cieľom tohto kroku je vytvoriť tabuľku, v ktorej pre každú vykonávanú úlohu určíme priemernú záťaž pri najmenšom počte súbežne vykonávaných úloh, ideálne pri vykonávaní úlohy samotnej. Vo všetkých ostatných poliach namiesto priemernej záťaže uvádzame zástupnú hodnotu, v našom prípade „*x*“ (pozri pseudokód 6.7).

```
1  SELECT
2    jobs.JobId, jobs.host, jobs.JobStarted, jobs.JobFinished, jobs.executionSlots,
3    IF(MIN(jobs.SimultaneousJobs) = 1,   jobs.avg_load, 'x'),
4    IF(MIN(jobs.SimultaneousJobs) = 2,   jobs.avg_load, 'x'),
```

```
   ...
5   IF(MIN(jobs.SimultaneousJobs) = 14,   jobs.avg_load, 'x'),
6   IF(MIN(jobs.SimultaneousJobs) = 15,   jobs.avg_load, 'x')
7  FROM jobs
8  GROUP BY
9   jobs.JobId, jobs.host, jobs.JobStarted, jobs.JobFinished, jobs.executionSlots
10 ORDER BY jobs.JobId, jobs.host;
```

Pseudokód 6.7 Skript na určenie priemernej záťaže pri najmenšom počte súbežne vykonávaných úloh

Základom pseudokódu 6.7 sú príkazy *IF* uvedené na riadkoch 3 až 6, pričom tieto príkazy sa aplikujú pre všetky prirodzené čísla $\langle 1, MAX(SimultaneousJobs)\rangle$. Pri vstupných údajoch, ktoré sme použili je $MAX(SimultaneousJobs) = 15$, teda najväčší počet súbežne vykonávaných úloh na jednom výpočtovom uzle bol 15. Vďaka tomuto skriptu pre každú vykonávanú úlohu určíme jednoznačne najmenší počet súbežne vykonávaných úloh a takisto priemernú záťaž pri tomto stave. Výstupné údaje využívame v dopyte na konečné určenie záťaže pre každú úlohu, ktorý sa nachádza v kapitole 6.5.3.3. Ukážku výstupných údajov je možné vidieť na obrázku 6.12.

| JobId | host | JobStarted | JobFinished | executionSlots | f0_ | f1_ | f2_ | f3_ | f4_ | f5_ | f6_ | f7_ | f8_ | f9_ | f10_ | f11_ | f12_ | f13_ | f14_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 123907 | comp25 | 2017-11-11T18:02:41 | 2017-11-12T16:13:05 | 12 | x | 17.79 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 123908 | comp25 | 2017-11-11T18:02:42 | 2017-11-12T16:13:05 | 12 | x | 17.79 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 123909 | comp27 | 2017-11-11T18:02:42 | 2017-11-12T16:13:05 | 12 | x | 17.4 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 123910 | comp27 | 2017-11-11T18:02:42 | 2017-11-12T16:13:05 | 12 | x | 17.4 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 123911 | comp28 | 2017-11-11T18:02:42 | 2017-11-12T16:13:05 | 12 | x | 18.7 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 123912 | comp28 | 2017-11-11T18:02:42 | 2017-11-12T16:13:05 | 12 | x | 18.7 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 123913 | comp29 | 2017-11-11T18:02:42 | 2017-11-12T16:13:05 | 12 | x | 17.63 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 123914 | comp29 | 2017-11-11T18:02:42 | 2017-11-12T16:13:05 | 12 | x | 17.63 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 123915 | comp30 | 2017-11-11T18:02:42 | 2017-11-12T10:17:58 | 12 | x | 17.92 | x | x | x | x | x | x | x | x | x | x | x | x | x |

Obrázok 6.12 Ukážka výstupných údajov s priemernou záťažou pri najmenšom počte súbežne vykonávaných úloh pre každú úlohu

#### *6.5.3.3 Konečné určenie záťaže pre každú úlohu*

Vstupom do posledného kroku zisťovania údajov o záťaži vykonávaných úloh je výstupná tabuľka z predchádzajúceho kroku, v ktorej pre každú úlohu vieme jednoznačne určiť najmenší počet súbežne vykonávaných úloh a priemernú záťaž pri tomto stave. V poslednom kroku zisťovania údajov o záťaži vykonávaných úloh z nej potrebujeme určiť záťaž pre každú úlohu, ktorú by na výpočtovom uzle spôsobila, keby sa na ňom nachádzala samostatne, bez ostatných úloh. V tomto kroku postupujeme podľa návrhu na určenie záťaže úlohy uvedeného v kapitole 5.4.1.

Najjednoduchší prípad, ktorý môže nastať je vtedy, keď je najmenší počet súbežne vykonávaných úloh rovný 1, pretože vtedy sa úloha na výpočtovom uzle

vykonáva samostatne. Záťaž všetkých úloh, ktoré patria do tejto kategórie je možné určiť dopytom podľa pseudokódu 6.8.

```
1  SELECT
2    jobs.JobId, jobs.host, jobs.JobStarted, jobs.JobFinished, jobs.executionSlots,
3    jobs.f0
4  FROM jobs
5  WHERE jobs.f0 != 'x'
6  ORDER BY jobs.JobId, jobs.host
```

Pseudokód 6.8 Skript na určenie záťaže pre úlohy, ktoré sa na výpočtovom uzle vykonávajú samostatne

Podmienka uvedená na riadku 5 pseudokódu vyselektuje len tie úlohy, ktoré aspoň v nejakom časovom okamihu boli na výpočtovom uzle vykonávané samostatne.

Zložitejší prípad je taký, keď sa úloha na výpočtovom uzle nikdy nevyskytuje samostatne. Vtedy je potrebné použiť iteratívny algoritmus, ktorý postupne určí záťaže úloh. Najskôr pre tie, ktoré sa na výpočtovom uzle vykonávajú spolu s jednou inou úlohou, potom pre tie, ktoré sa vykonávajú s dvomi, až postupne pre všetky úlohy. Dopyt, ktorý sa volá iteratívne uvádzame v pseudokódoch 6.9 a 6.10, keďže opäť ide o dopyt zložený z vonkajšieho a vnoreného dopytu.

```
1  SELECT
2    jobs2.host AS subquery_host,
3    jobs2.JobStarted AS subquery_started,
4    jobs2.JobFinished AS subquery_finished,
5    jobs2.f0 AS subquery_load
6  FROM jobs jobs2
7  WHERE jobs2.f0 != 'x'
```

Pseudokód 6.9 Skript na vrátenie hodnoty záťaže úlohy, ktorá už je známa

Úlohou vnoreného dopytu uvedeného v pseudokóde 6.9 je vrátiť hodnotu záťaže úlohy ***A***, ktorá sa vykonáva počas vykonávania úlohy ***B***, ale na rozdiel od záťaže úlohy ***B*** je záťaž úlohy ***A*** známa, pretože už bola zistená počas vykonávania dopytu podľa pseudokódu 6.8.

```
1  SELECT
2    jobs.JobId, jobs.host, jobs.JobStarted, jobs.JobFinished, jobs.executionSlots,
3    IF (jobs.f1 - subquery_load) > 0,
4      ROUND(jobs.f1 - subquery_load, 2),
5      0
6    ) AS load
7  FROM jobs
8  INNER JOIN subquery (pozri pseudokód 6.9)
9    ON jobs.host = subquery_host AND
10     jobs.JobStarted BETWEEN subquery_started AND subquery_finished
11 WHERE jobs.f1 != 'x'
```

| 12 | **ORDER BY** jobs.JobId, jobs.host |
|---|---|

Pseudokód 6.10 Skript na konečné určenie záťaže pre každú úlohu

Úlohou dopytu uvedeného v pseudokóde 6.10 je určiť záťaž pre každú úlohu, ktorá na výpočtovom uzle nebola vykonávaná samostatne. Tento dopyt sa vykonáva iteratívne s postupne narastajúcim parametrom $f_i$, takže v jednotlivých iteráciách sa vyskytuje ako *f0*, *f1*, .. až *f14*. Od hodnoty záťaže aktuálnej úlohy sa odpočíta hodnota záťaže úlohy, ktorá bola v tom istom čase vykonávaná na rovnakom výpočtovom uzle vrátená v parametre *subquery_load* z vnoreného dopytu. Ukážku výstupného súboru je možné vidieť na obrázku 6.13. Záťaže jednotlivých úloh uvedené vo výstupnom súbore sa používajú v optimalizačnom algoritme ako matica cien.

| JobId | host | JobStarted | JobFinished | executionSlots | f0_ |
|---|---|---|---|---|---|
| 123471 | comp25 | 2017-11-06T21:32:04 | 2017-11-07T20:40:16 | 32 | 9.64 |
| 123472 | comp01 | 2017-11-07T00:40:07 | 2017-11-07T01:34:08 | 6 | 0 |
| 123473 | comp01 | 2017-11-07T06:58:45 | 2017-11-07T07:03:52 | 12 | 0 |
| 123474 | comp02 | 2017-11-07T06:58:45 | 2017-11-07T07:04:01 | 12 | 0.04 |
| 123475 | comp03 | 2017-11-07T06:58:45 | 2017-11-07T07:03:48 | 12 | 0 |
| 123476 | comp04 | 2017-11-07T06:58:45 | 2017-11-07T07:04:03 | 12 | 0 |
| 123477 | comp06 | 2017-11-07T06:58:45 | 2017-11-07T07:03:47 | 12 | 0 |
| 123478 | comp07 | 2017-11-07T06:58:45 | 2017-11-07T07:03:52 | 12 | 0.03 |
| 123479 | comp10 | 2017-11-07T06:58:45 | 2017-11-07T07:04:10 | 12 | 0 |
| 123480 | comp11 | 2017-11-07T06:58:45 | 2017-11-07T07:03:56 | 12 | 0.03 |

Obrázok 6.13 Ukážka výstupného súboru so zistenými záťažami každej vykonávanej úlohy

## 6.6 Implementácia optimalizačného modelu rozvrhovania úloh

Implementáciu optimalizačného modelu rozvrhovania úloh sme vykonali v programovacom jazyku Python s využitím optimalizačných nástrojov spoločnosti Google, ktoré sú bližšie opísané v kapitole 3.4. Základom optimalizačného modelu je modifikovaná metóda dopravnej úlohy navrhnutá a opísaná v kapitole 5.4.2. Implementáciu sme vykonali na notebooku značky Samsung, ktorý používa operačný systém Ubuntu 16.04 LTS, 6 GB operačnej pamäte a dvojjadrový Intel Core I5 procesor s taktovacou frekvenciou 2,5 GHz. Cieľom tejto kapitoly je vysvetliť najdôležitejšie časti implementácie optimalizačného modelu, ktorý sme vytvorili.

V pseudokóde 6.11 môžeme vidieť importovanie balíčku *pywralp* z knižnice *ortools*. Balíček pywraplp obsahuje sadu optimalizačných funkcií OR-Tools určenú pre jazyk Python.

```
from ortools.linear_solver import pywraplp
```

Pseudokód 6.11 Importovanie knižnice ortools

Pseudokód 6.12 obsahuje vytvorenie inštancie objektu *pywraplp.Solver,* s parametrami *SolveTransportationProblem*, ktorý určuje, že ide o riešenie dopravnej úlohy a *pywraplp.Solver.CBC_MIXED_INTEGER_PROGRAMMING*, ktorý určuje, že ide o celočíselnú optimalizáciu.

```
solver =
 pywraplp.Solver(
  'SolveTransportationProblem',
  pywraplp.Solver.CBC_MIXED_INTEGER_PROGRAMMING
 )
```

Pseudokód 6.12 Vytvorenie objektu pre riešenie optimalizačných problémov

Pseudokódy 6.13 a 6.14 obsahujú definície hraničných podmienok pre optimalizačný problém rozvrhovania úloh na klastri UMB.

```
for i in range(pocetUzlov):
  solver.Add(
    solver.Sum(
      pocetJadierVyuzivanychUlohovNaUzle[i] * x[i, j] for j in range(pocetUloh)]
    ) <= kapacitaUzla[i]
  )
```

Pseudokód 6.13 Definovanie hraničnej podmienky, ktorá zabezpečuje, že kapacita žiadneho výpočtového uzla nebude prekročená

V pseudokóde 6.13 definujeme hraničnú podmienku, ktorá zabezpečuje, že kapacita žiadneho výpočtového uzla vyjadrená počtom výpočtových jadier nebude prekročená. Matica *x[i, j]* je matica pravdivostných hodnôt, ktorá vyjadruje, či je úloha *j* priradená na vykonávanie výpočtovému uzlu *i.*

```
for j in range(pocetUloh):
  solver.Add(
    solver.Sum(
      [x[i, j] for i in range(pocetUzlov)]
    ) >= 1
  )
```

Pseudokód 6.14 Definovanie hraničnej podmienky, ktorá zabezpečuje, že každá úloha bude priradená na vykonávanie aspoň jednému výpočtovému uzlu

V pseudokóde 6.14 definujeme hraničnú podmienku, ktorá zabezpečuje, že každá úloha bude priradená na vykonávanie aspoň jednému výpočtovému uzlu. Rovnako ako v prípade pseudokódu 6.13 využívame maticu pravdivostných hodnôt *x[i, j].*

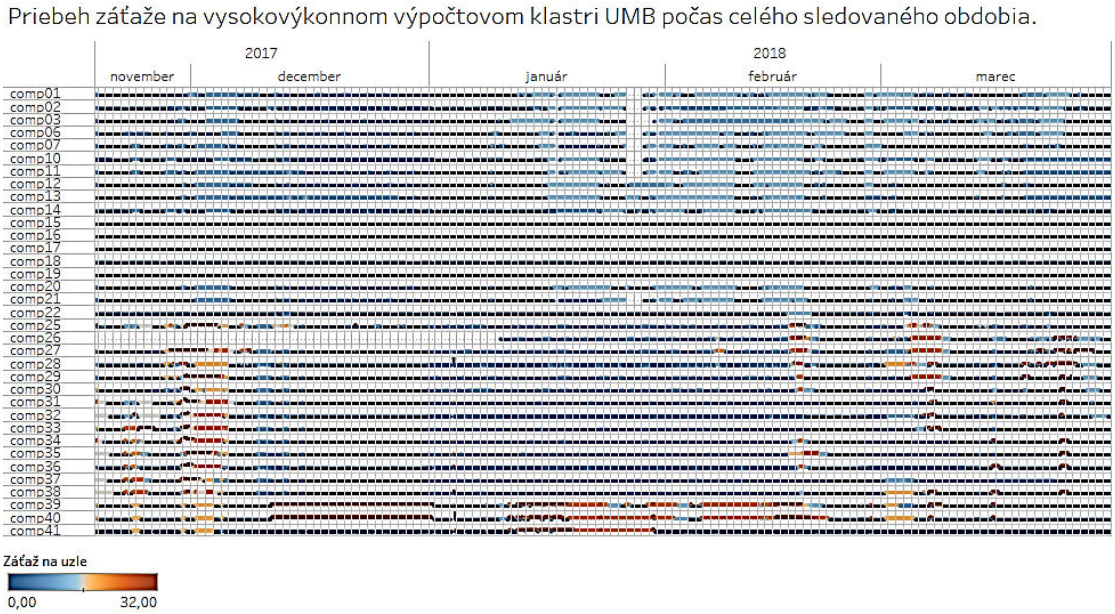
```
solver.Minimize(
   solver.Sum(
      [x[i, j] * vratSpotrebuZoZataze(maticaCien[i][j])
         for i in range(pocetUzlov) for j in range(pocetUloh)
      ]
   )
)
sol = solver.Solve()
```

Pseudokód 6.15 Vykonanie optimalizácie

Pseudokód 6.15 vykonáva optimalizáciu pomocou metódy modifikovanej dopravnej úlohy na základe modelu spotreby elektrickej energie s využitím polynomiálnej funkcie vytvoreného v kapitole 6.2.

# 7 DOSIAHNUTÉ VÝSLEDKY A DISKUSIA

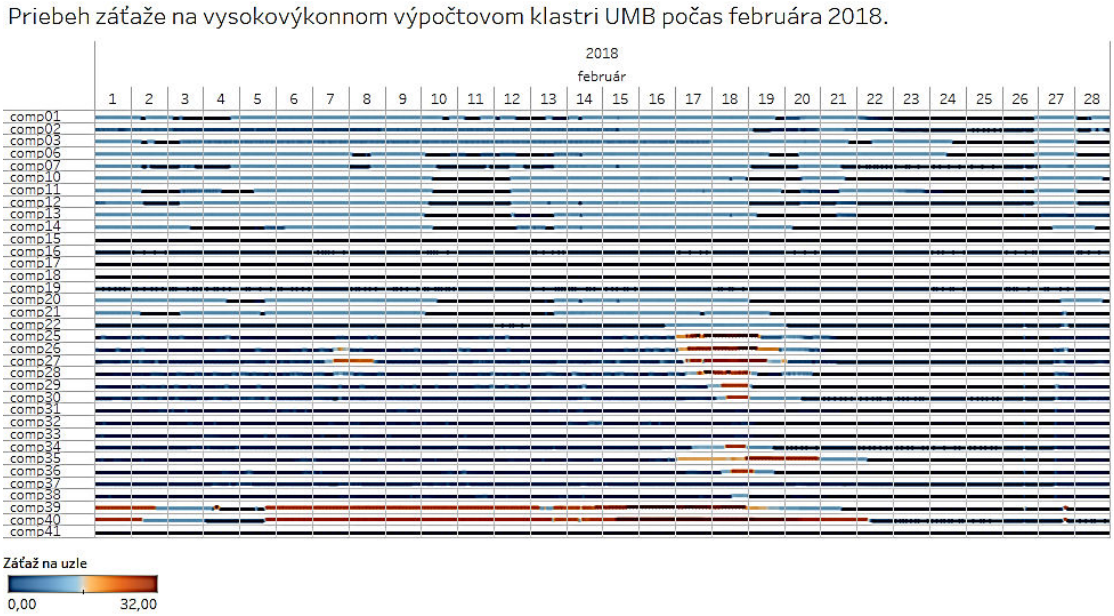
V praktickej časti práce sme vykonali analýzu údajov pre manažovanie úloh vo vysokovýkonnom počítaní. Na obrázkoch 7.1 až 7.3 je možné vidieť priebeh záťaže na výpočtových uzloch, o ktorých máme údaje. Na *Y*-ovej osi sa nachádza zoznam všetkých výpočtových uzlov, ktoré boli počas sledovaného obdobia v prevádzke a na osi *X* sa nachádza čas rozdelený na rok, mesiace, pri detailnejších vizualizáciách dni a hodiny. Na obrázkoch sa nachádza aj farebná legenda, na ktorej je uvedené, že silno modrá farba znamená záťaž výpočtového uzla blízku hodnote 0. Čím slabší je odtieň modrej farby, tým vyššiu hodnotu záťaže výpočtového uzla predstavuje, až po hodnotu 16. Od hodnoty 16 sa začína slabo oranžová farba a čím silnejšia oranžová, tým vyššia hodnota záťaže až po hodnotu 32. Počas skúmaného časového obdobia sa občasne vyskytovali aj vyššie hodnoty záťaže výpočtových uzlov ako 32, ale tie pre prehľadnosť vizualizácie špeciálne nezvýrazňujeme. Sú znázornené rovnakou oranžovou farbou aká znázorňuje hodnotu záťaže 32. Obrázky 7.1 až 7.3 sa vo väčšom rozlíšení nachádzajú v prílohe A.




Obrázok 7.1 Vizualizácia priebehu záťaže výpočtových uzlov počas celého sledovaného obdobia

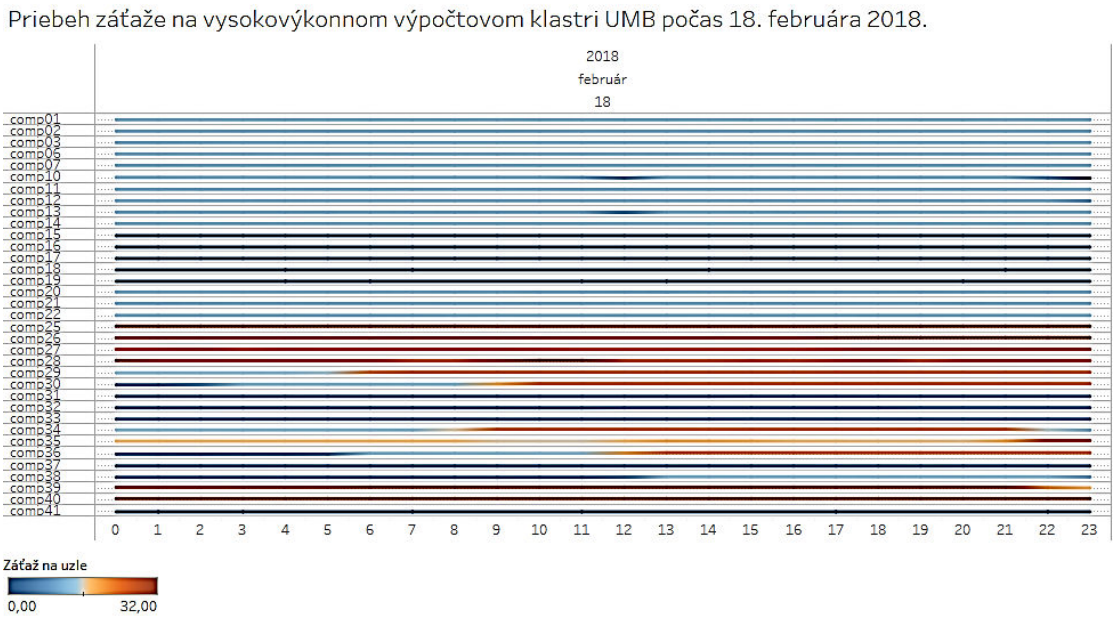
Obrázok 7.1 znázorňuje vizualizáciu priebehu záťaže na všetkých výpočtových uzloch, ktoré boli v prevádzke za celé sledované obdobie, teda od polovice novembra 2017 do konca marca 2018. Z vizualizácie je možné vidieť, že údaje o niektorých výpočtových uzloch, napríklad *comp01* až *comp11* alebo *comp26* nie sú kompletné, objavili sa ich výpadky. Ďalej môžeme vidieť, že vyššie hodnoty

záťaže sa vyskytujú na výpočtových uzloch *comp25* až *comp41*, ktoré boli zaobstarávané v rámci druhej etapy budovania klastra UMB (pozri kapitolu 4.2.1), čo je očakávateľné, pretože majú viac výpočtových jadier a teda väčšiu výpočtovú kapacitu.




Obrázok 7.2 Vizualizácia priebehu záťaže výpočtových uzlov vo februári 2018

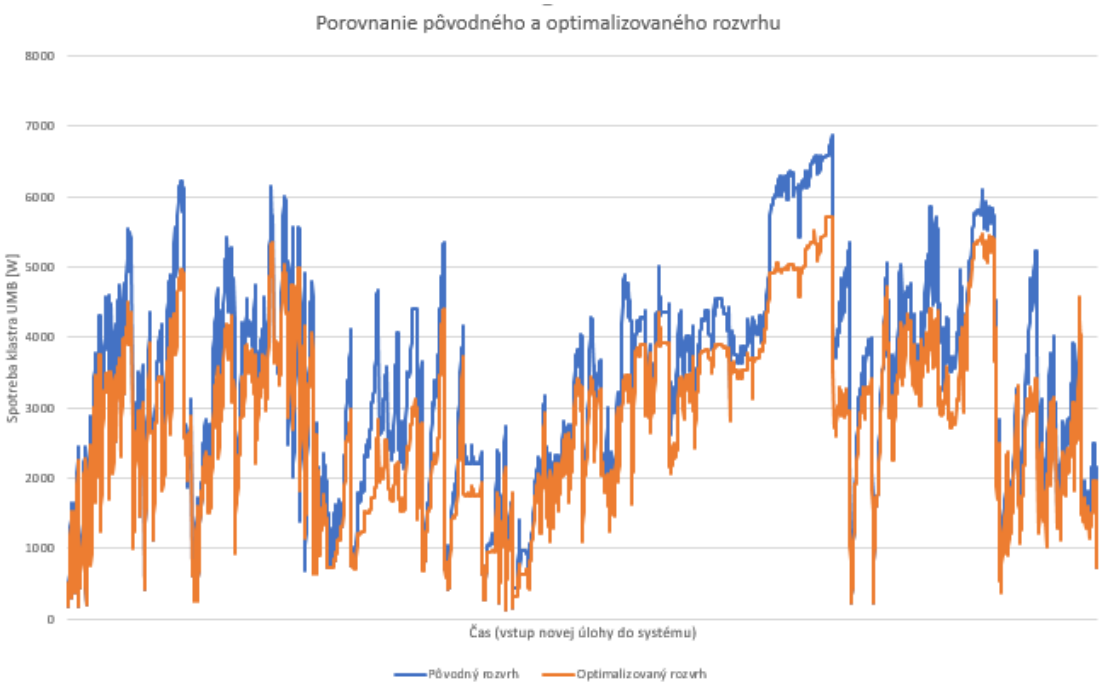
Podrobnejší pohľad na výpočtový systém ponúka obrázok 7.2, v ktorom analyzujeme priebeh záťaže na výpočtových uzloch v mesiaci február 2018. Aj na tomto obrázku je možné vidieť, že vyššie hodnoty záťaže sú dosahované na výpočtových uzloch zaobstarávaných v druhej etape budovania klastra UMB. Na tomto obrázku je navyše zreteľne vidieť, že dlhodobo najvyššiu záťaž dosahujú uzly *comp39* a *comp40*, ktoré obsahujú grafické výpočtové akcelerátory. Ďalšia dôležitá skutočnosť, ktorú je možné vidieť na uvedenej vizualizácii je, že záťaž výpočtových uzlov sa síce mení skokovo so začiatkom vykonávania výpočtovo náročnej úlohy na výpočtovom uzle, ale po náhlej zmene zostáva stabilná dlhšie časové obdobie. To je spôsobené tým, že vykonávanie väčšiny výpočtovo náročných úloh trvá rádovo niekoľko desiatok minút alebo hodín.




Obrázok 7.3 Priebeh záťaže počas 18. februára 2018

Ešte podrobnejší pohľad na výpočtový systém ponúka obrázok 7.3, v ktorom sa zameriavame na analýzu priebehu počas jediného dňa, 18. februára 2018. Aj na tomto obrázku je možné vidieť, že záťaž na väčšine výpočtových uzlov je počas dňa stabilná, napríklad výpočtové uzly *comp01* až *comp22*. Takisto je možné vidieť, že záťaž na niektorých výpočtových uzloch, ako napríklad *comp29* a *comp30* počas 18. februára rastie. Zaujímavý priebeh má aj záťaž výpočtového uzla *comp34*, kde môžeme vidieť, že záťaž na tomto výpočtovom uzle zhruba na 12 hodín stúpla na dvojnásobok a potom sa vrátila na pôvodnú. Je pravdepodobné, že počas týchto 12 hodín bola na uzle vykonávaná výpočtovo náročná úloha.

Vďaka optimalizačnému modelu rozvrhovania úloh sme vytvorili alternatívny rozvrh s cieľom zníženia spotreby elektrickej energie. Výsledky priebehu spotreby elektrickej energie podľa toho, ako úlohy prichádzali do systému je možné vidieť na obrázku 7.4.




Obrázok 7.4 Porovnanie pôvodnej a optimalizovanej spotreby

Graf obsahuje dve krivky – modrá krivka predstavuje spotrebu systému na základe pôvodne reálne vykonaného rozvrhu, oranžová krivka predstavuje spotrebu dosiahnutú vďaka rozvrhovaniu úloh na uzly s cieľom dosiahnutia najnižšej možnej spotreby. Každý záznam na *X*-ovej osi predstavuje vstup novej úlohy do systému, *Y*-ová os predstavuje okamžitú spotrebu systému vo wattoch v čase začiatku vykonávania novej úlohy.

Na grafe môžeme vidieť, že pri nízkych spotrebách elektrickej energie sa krivky príliš nelíšia, často sa dokonca prekrývajú. To je spôsobené tým, že systém je málo zaťažený a optimalizačný model rozvrhovania úloh nemá čo zlepšiť. Význam optimalizačného modelu rozvrhovania úloh sa prejaví až keď sa v systéme vykonáva viac úloh, pretože vtedy je možné ich vhodným umiestnením správne rozložiť záťaž a tak ušetriť elektrickú energiu.

Na niekoľkých miestach môžeme v grafe vidieť, že modrá (pôvodná) krivka sa dostane pod oranžovú (optimalizovanú) krivku. Keďže ide o dynamické rozvrhovanie, optimalizačný model nevie predikovať, čo sa stane v nasledujúcom kroku. Situácia na klastri sa môže vyvinúť tak, že pôvodný rozvrh úloh má v istom momente nižšiu spotrebu elektrickej energie ako ten optimalizovaný, ale väčšinou je to len na relatívne krátky čas, kým nevstúpia do systému nové úlohy. Preto rozvrh

zostavený optimalizačným modelom považujeme len za pseudooptimálny, nie optimálny.

Vypočítaním priemernej hodnoty spotreby elektrickej energie pre každú z kriviek sme zistili, že priemerná okamžitá modelovaná spotreba elektrickej energie pri modrej (pôvodnej) krivke je 3325,26 W. Pri oranžovej (optimalizovanej) krivke je priemerná modelovaná hodnota okamžitej spotreby elektrickej energie 2809,04 W. Upraveným rozvrhovaním úloh na výpočtové uzly s ohľadom na spotrebu elektrickej energie by v ideálnom prípade bolo možné ušetriť zhruba 15% spotreby elektrickej energie. O ideálnom prípade hovoríme preto, lebo v reálnej praxi je ťažké dopredu určiť, akú záťaž na výpočtovom uzle úloha spôsobí a teda ju priradiť na najvhodnejší výpočtový uzol.

# ZÁVER

V diplomovej práci sme sa zaoberali problematikou analyzovania údajov pre manažovanie úloh vo vysokovýkonnom počítaní. Analyzovali sme výpočtové zdroje a úlohy vo vysokovýkonnom počítačovom klastri Univerzity Mateja Bela. Definovali sme štandardné optimalizačné kritériá pre rozvrhovanie úloh. Opísali sme problematiku súvisiacu s energetickou efektívnosťou výpočtových systémov. Zbierali a analyzovali sme údaje o výpočtových uzloch a vykonávaných úlohách na klastri UMB. Navrhli a vytvorili sme skript na predspracovanie údajov o záťaži na výpočtových uzloch, skript pre orezanie a pretypovanie textu s hodnotou spotreby elektrickej energie na desatinné číslo, skript na predspracovanie údajov o spotrebe elektrickej energie, skript na určenie počtu súbežne vykonávaných úloh na výpočtovom uzle a niekoľko skriptov na určenie záťaže, ktorú vykonávaná úloha na výpočtovom uzle spôsobila. Tieto skripty sme implementovali na cloudovej platforme spoločnosti Google pomocou služby Google BigQuery. Ďalej sme navrhli a vytvorili funkciu na vytváranie záznamov s jedinečným výpočtovým uzlom pre každú úlohu, ktorú sme implementovali v jazyku node.js. Navrhli, vytvorili a porovnali sme 4 regresné modely na modelovanie spotreby elektrickej energie zo záťaže výpočtového uzla. Navrhli a implementovali sme modifikáciu štandardného algoritmu pre výpočet dopravnej úlohy, ktorý sme použili v optimalizačnom modeli rozvrhovania úloh. Ten sme implementovali v jazyku Python s využitím optimalizačnej knižnice OR-Tools.

Zistili sme, že z vytvorených regresných modelov spotreby elektrickej energie je najlepší ten, ktorý využíva na opis vzťahu polynomiálnu funkciu, s priemernou absolútnou chybou zhruba 21 W, čo pri priemernej spotrebe jedného výpočtového uzla zhruba 224 W je odchýlka 9,42%.

Zistili sme, že pri pôvodnom rozvrhu úloh na výpočtové uzly bola priemerná modelovaná okamžitá spotreba zhruba 3325 W. Pri optimalizovanom rozvrhu bola priemerná modelovaná spotreba elektrickej energie 2810 W, čo je úspora zhruba 15 % v ideálnom prípade.

Uvedomujeme si, že v reálnom systéme by sa zrejme nepodarilo znížiť spotrebu elektrickej energie až o 15 %, pretože do toho vstupujú ďalšie faktory a navyše je problematické dopredu odhadnúť záťaž, ktorú úloha na výpočtovom uzle spôsobí. Predikovanie záťaže úlohy a správania systému je oblasť, ktorej sme sa v práci nevenovali, ale určite je to vhodná oblasť na ďalšie pokračovanie výskumu v tejto oblasti.

# ZOZNAM POUŽITEJ LITERATÚRY


[1] MITTAL, S. *„Power Management Techniques for Data Centers: A Survey.“* Oak Ridge National Laboratory, 2014. [Online]. [Citované 24.4.2018]. Dostupné na: *https://www.academia.edu/6982393/Power_Management_Techniques_for_Data_Centers_A_Survey.*

[2] *„Top500 Supercomputer Sites.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://www.top500.org/*

[3] TRNIK, M. *„Programovacia paradigma MapReduce pre prácu s Big Data.“* In: Prírodovedec 2015, Banská Bystrica, 2015

[4] TRNIK, M. *„Mnohorozmerná analýza rozvrhovania vo vysokovýkonných počítačových systémoch.“* In: Prírodovedec 2016, Banská Bystrica, 2016

[5] TRNIK, M. *„Analýza činiteľov ovplyvňujúcich spotrebu elektrickej energie klastra pre vysokovýkonné výpočty.“* In: Prírodovedec 2017, Banská Bystrica, 2017

[6] TRNIK, M. *„Model optimalizácie rozvrhu úloh klastra pre HPCC s ohľadom na spotrebu elektrickej energie.“* In: Prírodovedec 2018, Banská Bystrica, 2018

[7] MIDDLETON, A.M. *„HPCC Systems: Introduction to HPCC.“* [Online]. Posledné úpravy: december 2015. [Citované 24.4.2018]. Dostupné na: http://cdn.hpccsystems.com/whitepapers/wp_introduction_HPCC.pdf

[8] SEVERANCE, C. – DOWD, K. *„High Performance Computing.“* Houston, Texas: Rice University, 2012, 294 s.

[9] ŠKRINÁROVÁ, J. *„Elastický klaster.“* Banská Bystrica: Belianum, 2017, 108 s.

[10] *„Understanding Linux CPU Load.“* [Online]. Posledné úpravy: 31.7.2009. [Citované 24.4.2018]. Dostupné na: *http://blog.scoutapp.com/articles/2009/07/31/understanding-load-averages*

[11] KANSAL, N.J. – CHANA, I. *„Cloud Load Balancing Techniques: A Step Toward Green Computing.“* In: IJCSI International Journal of Computer Science Issues. 2012, zv. 9, číslo 1, s. 238-246. [Online]. [Citované 24.4.2018]. Dostupné na: *https://pdfs.semanticscholar.org/4dc4/30f2439cc690eea612d5ef3e24c161938c80.pdf*

[12] VOGELS, W. *„Beyond Server Consolidation.“* In: acmqueue, 2008, zv. 6, číslo 1, s. 20-26. [Online]. [Citované 24.4.2018]. Dostupné na: *https://queue.acm.org/detail.cfm?id=1348590*

[13] HARMON, R. R. – AUSEKLIS, N. *„Sustainable IT Services: Assessing the Impact Of Green Computing Practises.“* In: Management of Engineering & Technology, 2009, PICMET 2009, s. 1707-1717. [Online]. [Citované 24.4.2018]. Dostupné na: *https://www.researchgate.net/profile/Robert_Harmon/publication/224595549_Sustainable_IT_services_Assessing_the_impact_of_green_computing_practices/links/0f3175387595da98c9000000.pdf*

[14] *„TGG Data Center Power Efficiency Metrics PUE and DCiE.“* [Online]. [Citované 24.4.2018]. Dostupné na: *http://www.premiersolutionsco.com/wp-content/uploads/TGG_Data_Center_Power_Efficiency_Metrics_PUE_and_DCiE.pdf*

[15] *„Green500“.* [Online]. [Citované 24.4.2018]. Dostupné na:

*https://www.top500.org/green500/*
[16] ARMBRUST, M. et al. *„A View of Cloud Computing.“* In: Communications of the ACM, 2010, zv. 53, číslo 4, s. 50-58. [Online]. [Citované 24.4.2018]. Dostupné na: *https://dl.acm.org/citation.cfm?id=1721672*
[17] *„Amazon Web Services (AWS).“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://aws.amazon.com*
[18] *„Google Cloud Computing.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://cloud.google.com*
[19] *„Microsoft Azure Cloud Computing Platform.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://azure.microsoft.com/*
[20] *„Map AWS services to Google Cloud Plarform products.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://cloud.google.com/free/docs/map-aws-google-cloud-platform*
[21] *„Map Microsoft Azure services to Google Cloud Plarform products.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://cloud.google.com/free/docs/map-azure-google-cloud-platform*
[22] *„Business Solutions for Enterpise.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://cloud.google.com/why-google-cloud*
[23] *„GCP Free Tier.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://cloud.google.com/free/*
[24] MELNIK, S. et al. *„Dremel: Interactive Analysis of Web-Scale Datasets.“* In: Proceedings of the VLDB Endowment, 2010, zv. 3, číslo 1. [Online]. [Citované 24.4.2018]. Dostupné na: *https://research.google.com/pubs/pub36632.html*
[25] SATO, K. *„An Inside Look at Google BigQuery.“* 2012. [Online]. [Citované 24.4.2018]. Dostupné na: *https://cloud.google.com/files/BigQueryTechnicalWP.pdf*
[26] DEAN, J. – GHEMAWAT, S. *„MapReduce: Simplified Data Processsing on Large Clusters.“* In: OSDI'04 Proceedings of the 6th conference on Symposium on Operating System Design & Implementation. [Online]. 2003, zv. 37, číslo 5, s. 29-43. [Citované 24.4.2018]. Dostupné na: *https://research.google.com/archive/mapreduce.html*
[27] *„Google Optimization Tools.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://developers.google.com/optimization/*
[28] *„Simplified Wrapper and Interface Generator.“* [Online]. [Citované 24.4.2018]. Dostupné na: *http://www.swig.org/*
[29] *„Apache License, Version 2.0.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://www.apache.org/licenses/LICENSE-2.0*
[30] LEI, K. – MA, Y. – TAN, Z. *„Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js.“* In: Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference, 2014, s. 661-668.
[31] *„SIVVP.“* [Online]. [Citované 24.4.2018]. Dostupné na: *http://www.sivvp.sk/*
[32] *„HPCC UMB.“* [Online]. [Citované 24.4.2018]. Dostupné na: *http://hpcc.umb.sk/*
[33] STAPPLES, G. *„TORQUE resource manager.“* In: SC '06 Proceedings of the 2006 ACM/IEEE conference on Supercomputing, 2006. ISBN: 0-7695-2700-0
[34] *„Zabbix features overview.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://www.zabbix.com/features*
[35] *„Using OR-Tools.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://developers.google.com/optimization/introduction/using*

[36] MUNKRES, J. *„Algorithms for the Assignment and Transportation Problems.“* In: Journal of the Society for Industrial and Applied Mathematics, 1957, zv. 5, číslo 1, s. 32-38.
[37] *„Trendlines significance.“* [Online]. [Citované 24.4.2018]. Dostupné na: *https://onlinehelp.tableau.com/current/pro/desktop/en-us/trendlines_significance.html*

# PRÍLOHY

Príloha A: Detail vizualizácií záťaže výpočtových uzlov

Príloha B: Zdrojové kódy aplikácie

Príloha C: Systémová dokumentácia

Príloha D: Používateľská príručka

Príloha E: Sprievodné CD so zdrojovými kódmi a ukážkou vstupných a výstupných súborov