Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Vizualizace výpočetních úloh v systému DIRAC

2022

Daniel Čampiš

Vedoucí práce:
Mgr. Martin Trnečka, Ph.D.

Studijní program: Aplikovaná informatika,
kombinovaná forma

Bibliografické údaje

Autor:	Daniel Čampiš
Název práce:	Vizualizace výpočetních úloh v systému DIRAC
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2022
Studijní program:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	Mgr. Martin Trnečka, Ph.D.
Počet stran:	41
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Daniel Čampiš
Title:	Visualization of Computation Jobs in DIRAC System
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2022
Study program:	Applied Computer Science, combined form
Supervisor:	Mgr. Martin Trnečka, Ph.D.
Page count:	41
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Práce pojednává o vývoji informačního systému pro vizualizaci výpočetních úloh v systému DIRAC. Vizualizace byla vyvíjena jako webová aplikace založená na frameworku React.js. Skládá se z části pro import datové sady a části vizualizační, která je formou prezentace dat s možností jejich selekce dle přidružených parametrů. Bakalářská práce v první části čtenáře seznámí s nástroji a technologiemi použitých při realizaci a technickou stránkou aplikace. Dále v části uživatelské příručky je vysvětlena funkcionalita aplikace a způsob použití. Nakonec budou shrnuty směry, kterými se bude moci ubírat následný vývoj aplikace.

Synopsis

The thesis deals with the development of an information system for the visualization of computational tasks in the DIRAC system. The visualization was developed as a web application based on the React.js framework. It consists of a data set import part and a visualization part, which is a form of data presentation with the possibility of their selection according to associated parameters. In the first part of the bachelor's thesis, the reader will be introduced to the tools and technologies used in the implementation and the technical side of the application. Further, in the section of the user manual, the functionality of the application and the method of use are explained. Finally, the directions in which the subsequent development of the application can be taken will be summarized.

Klíčová slova: vizualizace dat, webová aplikace, JavaScript, React.js

Keywords: data visualization, web application, JavaScript, React.js

Tímto bych chtěl poděkovat vedoucímu bakalářské práce panu Mgr. Martinu Trnečkovi, Ph.D. za odborné vedení, jeho čas a trpělivost. Mé poděkování též směřuji Igoru Pelevanyukovi z SÚJV, který vedl projekt v době mé stáže v ústavu a dokázal mě vždy směřovat správným směrem.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Úvod

Ve vědeckém světě je často nutné řešit otázku, jakým způsobem vyvíjet softwarové nástroje, které umožňují nebo usnadňují práci pracovníkům. V mnoha případech není možné, aby sami vědečtí pracovníci věnovali svůj čas pro vývoj takových nástrojů. Na tomto místě je nezbytná spolupráce informatika s vědeckými skupinami, ke které mimo proces vývoje patří i pochopení odborného problému vývojářem. Tato bakalářská práce vychází z projektu, který mi byl přidělen na stáži v mezinárodní organizaci Spojeného ústavu jaderného výzkumu (SÚJV). Ústav se zajímá vědeckým výzkumem napříč mnoha obory, jakými jsou částicová a nukleární fyzika, materiálová fyzika, medicína, biologie, informatika aj. Po stránce výpočetní je ústav zastřešen pod Laboratoří informačních technologií, kde souběžně s teoretickým výzkumem je poskytována výpočetní podpora celému centru. Mimo další se zde nachází superpočítač třídy TIER-1 s výpočetní kapacitou 1.7 Pflops a databankou s úložištěm větším než 50 PB. Tento a přidružené superpočítače jsou spravovány systémem DIRAC The Interware, který řídí mj. vykonávání výpočetních úloh. Příkladem takových úloh mohou být simulace částicových interakcí nebo analýza naměřených dat. Systém má pod sebou mnoho heterogenních výpočetních částí fyzicky rozmístěných po mnoha zemích. Optimalizace systému se stala z jedním úkolů týmu, se kterým jsem spolupracoval. Význam projektu Vizualizace výpočetních úloh v systému DIRAC, dále jen aplikace, vzešel z potřeby analýzy chování zpracování výpočetních úloh na podmínkách fyzického stroje, na který systém DIRAC úlohu přidělil. Při opomenutí významu přenosu dat úlohy sítí, kterým jsem se zabýval v jiné práci, je rozhodující parametr výpočtu úlohy výkon procesoru. Ten je na každém stroji před zahájením výpočtu úlohy zjištěn v tzv. pilot skriptu, který mimo jiné spustí benchmark na daném stroji. Data získaná z tohoto benchmarku se staly ústřední částí projektu.

1 Technická dokumentace

V následující části budou popsány technologie a nástroje použité při vývoji. V další části bude pojednáno o struktuře projektu a v poslední řadě bude zmíněn postup řešení několika problému z vývoje aplikace.

1.1 Použité technologie

Použité technologie byly vybírány s ohledem pro multiplatformní podporu se zaměřením na architekturu x86-64 a Armv8. Při výběru jednotlivých komponent projektu byla zvýhodněna open-source povaha licence.

1.1.1 JavaScript

JavaScript je vysokoúrovňový multiparadigmatický just-in-time kompilovaný programovací jazyk, který vznikl pro účel vytváření skriptů webových prohlížečů. Podléhá standardu ECMA-262 spravované organizaci ECMA International . Mezi klíčové charakteristiky patří slabé dynamické typování, design založený na objektově orientovaném paradigmatu s prototypy. Funkce nesou vlastnost first-class objektu.
Jazyk patří mezi nejpopulárnější, což dokazuje průzkum webu stack overflow, ve kterém ho v roce 2021 uvedlo přes 68% profesionálních developerů za jazyk, který významně použili při své práci.
Nejhojněji rozšířeným jádrem (též nazývaný jako JavaScript engine), který má na starost vykonávání instrukcí je V8 vyvíjený skupinou The Chromium Project. Je součásti prohlížeče Google Chrome či běhových prostředí Node.js a Deno.
Tento jazyk byl vybrán na základě jeho nativní podpory ve webových prohlížečích. Dále ve formě implementace Node.js ho bylo možné použít pro tvoření skriptů na straně serveru. Tato implementace nabízí ve své standardní knihovně nad rámec standardu ECMA mnoho modulů, které jazyku dávají možnost univerzálního použití při řešení rozličných úloh.
Jak prohlížeče, tak běhové prostředí podporují běh na vícero architekturách. Např. binární spustitelný soubor Node.js je oficiálně ke stažení pro instrukční sadu x86, x86-64, ARMv7, ARMv8.
Z důvodu povahy projektu, která nevyžadovala podporu zpětné kompatibility prohlížečů byla použita poslední verze ECMAScript 2021.

1.1.2 Webpack

Webpack[**8**] slouží pro sestavování aplikace klienta do podoby vhodné pro šíření protokolem HTTP(S). Konkrétně tak dochází k sloučení, minifikaci a oddělení nepoužitého zdrojového kódu. Výsledkem je tak adresář, který je připraven k nahrání na webový server.
Též je možné specifikovat místa v zdrojovém kódu, kde dojde k rozdělení jinak jediného souboru s JavaScriptem. Takto je vhodné postupovat při rozsáhlých

webových aplikací. Samotný nástroj pak k takovému kroku doporučuje, jakmile velikost souboru převyšuje 244 KiB, z důvodu snížení očekávané odezvy při načítaní samotné aplikace.
Do výstupního adresáře je též vložen vstupní bod aplikace, který je v případě webu HTML soubor, jehož obsahem je hlavička s definicí na vstupní JavaScriptový soubor. Dále webpack zajišťuje vložení ostatních souborů, jakými jsou např.: css, obrázky, fonty a jiné.
Webpack navíc nabízí vývojový webový server s podporou hot module replacement, který při úpravě kódu cíleně v prohlížeči nahrazuje pouze změněné části.

1.1.3 Babel

Hlavní funkcionalitou nástroje Babel[**12**] je schopnost transformace JavaScriptového kódu. Důvody k ní mohou být zajištění zpětné kompatibility s starými prohlížeči, zpřístupnění nestandardních rošíření jazyka nebo odstranění odlišností v jednotlivých implementacích. Takovýto přepis zdrojového kódu se označuje jako transpilace.

1.1.4 Eslint

Eslint[**13**] je jedním z JavaScriptových linterů. Slouží ke statické analýze kódu za účelem upozornění na chyby programátora. Zejména se jedná o chyby typografického charakteru, které nemusí vést k ovlivnění funkcionality programu, avšak jsou podstatné pro produktivitu a přehlednost. Též se může jednat o upozornění v případě nalezení antipatternu, tedy struktury kódu, která zpravidla vede k chybám a není jí tak doporučeno využívat.
Je možné zvolit soubor pravidel, jimiž by se programátor měl držet. Často vycházejí z praxe jako „best practices.“ V projektu byl použit upravený soubor pravidel vydaný společností Airbnb.

1.1.5 React

React[**9**] je knihovnou jazyka JavaScript vyvíjenou společností Meta určenou pro prohlížeče. Zároveň se dá označit za framework, protože uživatele vede k dodržování vlastního paradigmatu, v tomto případě deklarativního. Je určen pro dynamickou tvorbu grafického rozhraní webových stránek. Struktura stránky je tak prohlížečem vykreslována bez předlohy na základě značkovacího jazyka HTML, ale dynamicky generována z kódu v jazyce JavaScript. K zápisu HTML tagů zde slouží syntaxní rozšíření JSX, které nepatří ke standardu podporovaného prohlížeči, proto je tuto část nutné transformovat na standardní variantu za pomocí nástroje Babel. Mimo zmíněné reprezentace DOM struktury webové stránky lze též React využít pro správu událostí, či globálního stavu aplikace.
Na místo navigace mezi jednotlivými webovými stránkami formou hypertextových odkazu, které v sobě obsahují adresu nadcházející stránky, která je opět požadována od webového serveru, React nabízí SPA (sinle page application)

navigaci. V tomto případě je následující stránka překreslena do okna prohlížeče dle instrukcí z JavaScriptu bez nutnosti zaslání dotazu na webový server. Výjimku tvoří dynamicky importované čísti, které je je nutno ze serveru dotazovat, opět se však jedná o součást JavaScriptu, z jehož instrukcí je nová stránka vykreslena.

1.1.6 MobX

MobX[**14**] je další knihovnou jazyky JavaScript, která si klade za cíl správu globálního stavu aplikace. Je postavená na reaktivním paradigmatu, kterým se vhodně doplňuje s podstatou fungování React komponent, do kterých je změna stavu ihned propagována. Stavebním kamenem této funkcionality je zprostředkován pomocí Proxy objektu, který umožnil princip metaprogramování v JavaScriptu.
Knihovna využívá nestandardního jazykového rozšíření o dekorátory, kterými usnadňují správu stavových objektů.

1.1.7 Highcharts

V případě volby knihovny pro vykreslování grafů byl kladen důraz na nutnost optimalizace na rozsáhlé datové sady. K zobrazení více jak milionu bodu a jejich dynamickému zobrazování v reálném čase byla potřeba podpora grafické akcelerace, kterou velice snadno integrovala právě knihovna Highcharts[**10**].
Značnou výhodu též představovala výrazné možnost kustomizace prvků grafu. Též bylo uvažováno o nasazení knihovny D3js, která však při své komplexnosti nenabízela jiných výhod, a proto bylo tato varianta opuštěna.

1.1.8 Bootstrap

Součástí webového vývoje je vizuální stylování komponent za pomocí kaskádových stylů. V projektu byl využit soubor předpřipravených stylů z toolkitu Bootstrap[**15**].

1.1.9 GitLab

Pro verzování a správu zdrojového kódu byl zvolen nástroj git ve spojení s platformou GitLab. Samotný git zajišťuje ukládání záznamů změn mezi jednotlivými úpravami kódu, větvení,kolaboraci a distribuci vývoje.
Webová platforma GitLab[**16**] navíc nabízí možnost hostování aplikace s v rámci GitLab Pages.

1.2 Struktura projektu

Projekt aplikace je organizován jako NPM 8 balíček, který nabízí snadnou správu závislostí potřebných pro vývoj a sestavování aplikace. Git repozitář aplikace se nachází na platformě GitLab na adrese projektu: `https://gitlab.com/jinr-dc/performance-monitoring`.

1.2.1 Struktura adresáře projektu

V následující části bude popsána struktura adresáře projektu. Vývojový název projektu byl zvolen performance monitoring. Struktura adresáře se v průběhu vývoje značně měnila a v současnosti obsahuje části, které nejsou aktivně nasazeny v současné verzi. Počítá se však s jejich zapojením v následujícím vývoji.

1.2.1.1 Adresář: front Jedná se o hlavní adresář, který obsahuje zdrojový kód k aplikaci.

1.2.1.2 Adresář: front/dist Adresář dist (zkr. pro distribution) je výstupní adresář výsledku bundlingu neboli sestavení webové aplikace pomocí nástroje webpack. Nachází se zde kompletní webová aplikace určená pro přímé nasazení na webovém serveru. V nejjednodušší podobně k nasazení aplikace postačuje její překopírování do adresáře, který je na webovém serveru konfigurován jako domovský. Mezi těmito soubory se nachází vstupní HTML soubor index.html, který slouží klientskému prohlížeči jako vstupní bod celé aplikace. Dalšími je proměnlivý počet souborů s transpilovaným JavaScript kódem. Při bundlingu se původní kód pozmění několika způsoby.

1.2.1.2.1 Spojení původních JavaScript souborů Veškeré soubory s JavaScript kódem se spojí v jeden nebo více v závislosti na konfiguraci dynamického importu. Při použití protokolu HTTP/1 na webovém serveru je efektivnější upřednostňovat přenos menšího počtu souboru. Projekt je rozdělen současně do 53 JavaScript souborů, a proto je sloučení více než vhodné. Za předpokladu, že uživatel nebude pokaždé využívat veškerou funkcionalitu aplikace, je vhodné definovat moduly funkčnosti. Klient si vyžádá vždy jen takové moduly, které potřebuje a zbytek aplikace nemusí být přenesen. Např. při spuštění aplikace na úvodní stránce není přenášen zdrojový kód stránky konfigurace. Po navigaci na tuto stránku je příslušný funkční modul přenesen.

1.2.1.2.2 Minifikace V zdrojovém kódu se nachází mnoho částí, které nesou nezbytné pro její spuštění a jejich význam je důležitý především pro samotný vývoj a vývojáře. Mezi tyto části řadíme: zbytné prázdné znaky, komentáře v kódu, popisné jména proměnných a jiných výrazů. Jejich vypuštěním dojde k výraznému poklesu velikosti výstupních souborů, a tedy zkrácení doby přenosu aplikace na prohlížeč.
Další strategií zmenšení velikosti je tzv. tree-shaking. Za pomocí stromu importovaných JavaScriptových modulů je analyzováno, ve kterých souborech jsou tyto moduly uloženy. Ostatní souboru jsou dále vypuštěny.

1.2.1.2.3 Transpilace Transpilace neboli přepisu jednoho často vyššího programovacího jazyka do nižšího je při bundlingu využíváno při:

- přidání funkcionality zdrojovému kódu, např.: syntaxe JSX, @ dekorátory

- zajištění kompatibility mezi prohlížeči nebo jejich verzemi

1.2.1.2.4 Linting Pomocí lintingu je programátor upozorněn na chyby nebo varování v kódu. Jedná se o statickou analýzu a kód tedy není vykonáván. Kontrolované pravidla často patří mezi tzv. best practices, jejichž následování snižuje chybovost kódu.

1.2.1.2.5 Inline začlenění Součásti aplikace jako např. obrázky mohou být přeneseny jako samostatné soubory a nebo mohou být vloženy přímo do JavaScript souboru ve formě řetězce. Jsou-li vkládané data binární, jsou převedena do textové podoby např. algoritmem base64.

1.2.1.2.6 Kopírování nepozměněných souborů Zbytek souborů, které nejsou bundlerem zpracovány, jsou do výstupního adresáře překopírovaný. Patří mezi ně soubory typu: html, obrázky, fonty, aj.

1.2.1.3 Adresář: front/src Zde se nachází zdrojový kód aplikace v jeho vývojové formě. Mezi významné soubory patří:

- index.html – vstupní html soubor aplikace bez statického obsahu
- index.js – vstupní JavaScript soubor, určení kořene React komponentů v DOM stromě stránky, import globálních kaskádových stylů stránku ve formátu sass
- App.js – definice routeru aplikace, definice globálního stavu zpravovaného pomocí MobX knihovnou, definice cest pro částečné testování
- style.sass – soubor globálních kaskádových stylu ve formátu sass, import stylů Bootstrap

1.2.1.4 Adresář: front/src/chart/diracChart Tento adresář obsahuje zdrojový kód hlavní stránky vizualizace.

- Layout.js – rozvržení grafické podoby stránky, zobrazení navigace, routování mezi stránkou vizualizace a konfigurací
- DiracChart.js – komponenta hlavního grafu vizualizace
- DiracHistogramChart.js – komponenta grafu časové řady
- DiracChartToolbox.js – komponenta panelu nástrojů grafu
- StatusPanel.js – komponenta tabulky statistik grafu
- Sidebar.js – komponenta bočního panelu se selektory

- utils.js – knihovna pomocných funkcí vícekrát využívaných na stránce
- stateStore.js – třída globálního stavu stránky
- components/ - obecné komponenty stránky
- DataOptions/ - stránka konfigurace vizualizace
- influxApi/ - procedury pro získání, zpracování datové sady a vytvoření časové řady
- selectors.js – třídy pro data selektorů
- SelectorComponents – komponenty selektorů

1.2.1.5 Adresář: front/src/tests Adresář slouží pro sestavování náhledů na jednotlivé komponenty. Ty jsou vyvíjeny odděleně od celku a jsou testovány jejich případy užití. Z důvodu nesestavování celého projektu, ale jen jeho části, je urychlen proces bundlování a zvyšuje se tak komfort vývoje.

1.2.1.6 Adresář: node modules Projekt je spravován za pomocí NPM správce balíčků ekosystému Node.js. Adresář node module obsahuje závislé balíčky vyžadované projektem a jejich rekurzívních závislostí.

1.2.1.7 Soubor: .eslintrs.cjs Soubor obsahuje konfiguraci linteru ESLint. Verze podporovaného JavaScriptu je určena ES2021. Mezi další podporovanou syntax patří: api prohlížeče, api Node.js, framework testovaní Jest, React syntaxe zápisu virtual DOM stromu JSX. Seznam definovaných pravidel je přejatý ze známé Airbnb konfigurace s vlastními úpravami.

1.2.1.8 Soubor: .gitignore Soubor je standardní konfigurací GIT nástroje. Obsahem je výčet jmen složek a souborů, které nebudou GITem sledovány.

1.2.1.9 Soubor: .gitlab-ci.yml Jedno z míst, kde je aplikace hostovaná je platforma GitLab. Tento soubor je konfigurací pro nasazení aplikace na dané platformě. Je v něm obsažen postup, jakým způsobem virtuální stroj platformy aplikaci instaluje. Tím se aplikace stává dostupná na síti Internetu.

1.2.1.10 Soubor: package.json Jedná se o hlavní konfigurační soubor pro NPM správce balíčků ekosystému Node.js. Mezi definované části patří název projektu, verze, druh systému modulů. V sekci scripts jsou definovány jména pro rychlé volání skriptů, jako jsou např. skripty pro bundlování, testování, spuštění vývojového webového serveru, nasazení na vzdálený server. Poslední součástí je seznam závislých balíčků je jejich verzí definovaných semver formátem.

1.2.1.11 Soubor: webpack.*.js

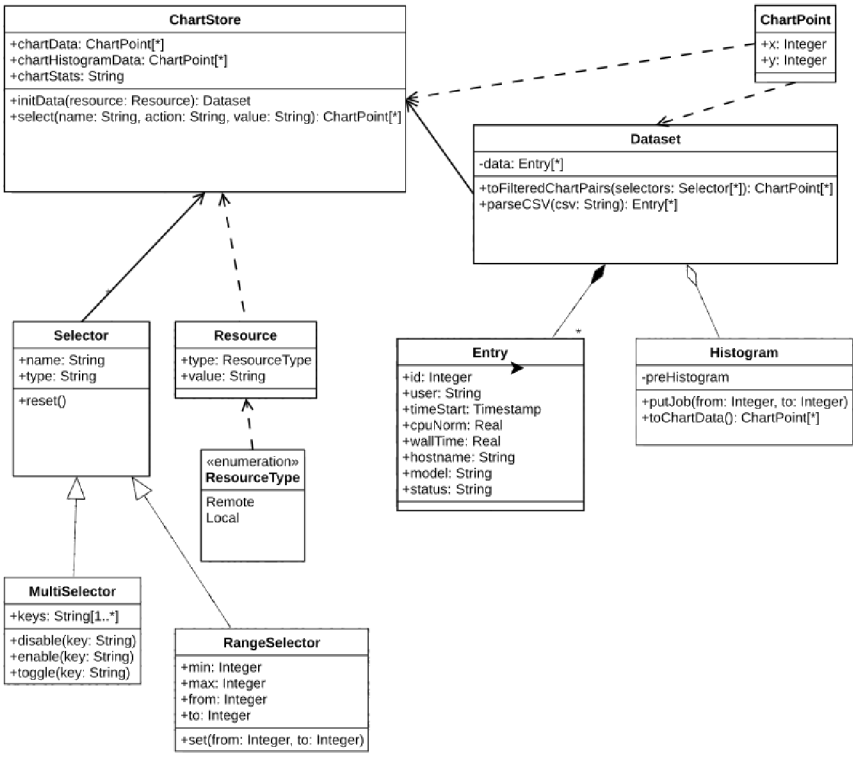
Tyto soubory konfigurují chování bundleru Webpack podle cíle. Základní konfigurace je obsažena v webpack.conf.js, ke kterému jsou připojovány dodatky pro produkční cíl, vývojový cíl a cíl určený pro nasazení v platformě Gitlab. V rámci vývojového cíle je definováno spuštení vývojového webového serveru a aktivace source-map. Source-map umožňují zobrazení původního zdrojového kódu v prohlížeči i poté, co je upraven a transpilován do změněné podoby. Mezi praktickou funkci patří tzv. hot-reloading, za pomocí něho lze po úpravě zdrojového kódu a jeho uložení pozorovat okamžitou změnu v okně prohlížeče bez nutnosti dalšího načtení stránku a bez nutnosti bundlování celé aplikace. Dalším podstatným parametrem je *publicPath*, který určuje cestu na které je aplikace hostována za webovém serveru.

1.3 Architektura aplikace

Architektura aplikace byla založena na dvou hlavních knihovnách React.js, část prezenční, a MobX, část stavová a řídící.
Po načtení komponenty Layout hlavní stránky aplikace je inicializován stavový objektu knihovny MobX, který je instancí třídy ChartStore. Posléze je volána metoda initData, která získá datovou sadu a zpracuje ji pro potřeby vizualizace do instance třídy DiracDataset. Voláním interní metody update stavového objektu ChartStore jsou přepočítány výstupní data pro hlavní graf vizualizace, časové řady a tabulky statistik a jsou uloženy do příslušných pozorovatelných vlastností stavového objektu. Z principu reaktivního paradigmatu používaného knihovnou MobX jsou komponenty knihovny React.js obeznámeny s změnou hodnoty vlastností stavového objektu a tím jsou nové hodnoty předány příslušným komponentám. Komponenty pak vykreslí změnu do DOM stromu webové stránky. Tímto se centrem logiky aplikace stává stavový objekt.
V případě změny stavu selektorů je propagace změny totožná s tím rozdílem, že iniciátor změny dat ve stavovém objektu je jeho metoda volaná komponentou selektoru.
Diagram tříd znázorněný na obrázku 1. zachycuje zjednodušený model logické části aplikace.

1.4 Zabezpečení

Aplikace neobsahuje části, které by vyžadovali omezení přístupu nebo autorizaci uživatele. Jako nástroj pro zobrazování parametrů dat je volně přístupná na platformě Gitlab. Dalším místem nasazení aplikace je server instituce bez veřejné ip adresy. Zde dochází k shromažďování dat a jejich zpřístupnění v lokální síti.
Komunikace mezi webovým serverem a klientem podporuje standardní šifrovaný přenos za pomocí TLS protokolu.



Obrázek 1: Zjednodušený diagram tříd

1.5 Popis řešení

V této části bude popsáno několik výzev, které vznikly vývoji aplikace.

1.5.1 Získání a zpracování dat

Zdroje datové sady je vybrán z přednastaveného seznamu URL adres známých zdrojů. Postupuje se od první adresy. V případě zaslaná úspěšné odpovědi na HTTP dotaz metody HEAD serverem, kde se datová sada nachází, je tato adresa použita pro získání datové sady. V případě neúspěšného spojení je vybrána následující adresa nebo je uživatel upozorněn na chybu nedostupného zdroje dat.Podporované formáty datové sady jsou v současné chvíli obecné CSV a anotované CSV. Bližší definice struktury viz. v příslušné kapitole uživatelské dokumentace.
Po přenosu datové sady z vzdáleného zdroje nebo z místního adresáře je určen typ formátu dat jeho analýzou a následně jsou data předána příslušnému parseru. Parser ověří neprázdnost, validitu datové sady a další parametry např. typ znaku/ů reprezentující nový řádek.
Postupuje se po řádcích. První řádek reprezentující hlavičku je převeden na mapu názvů sloupů a jejich pořadí. Každý následující řádek, který již obsahuje data je převeden na pole, kde každá položka odpovídá sloupci z CSV tabulky. Dle známého schématu datové sady jsou jednotlivé hodnoty převedeny na příslušné

datové typy.
Výsledek parsování je pole polí, kde vnější pole reprezentuje položky datové sady a vnitřní její sloupce. Poli je přidána vlastnost headers, jejíž hodnota je mapa názvů sloupců na jejich pořadí.
V případě úspěšného parsování je přistoupeno k přidruženým úkonům jakými jsou:

- randomizace hodnot sloupce `cpu_norm` (Je vyžadována z důvodu rozptýlení hodnot, které budou vizualizovány na ose X. V opačném případě by při diskrétním charakteru této veličiny nebylo možno pozorovat kýžené klastrování. Jednotlivé body by byly vykresleny hustě přes sebe na přímkách kolmo protínající osu X s odstupem jedné desetiny. Randomizace se blíží uniformnímu rozdělení a hodnoty jsou unikátní. Unikátnosti je využíváno při dohledávání vizualizovaného bodu v grafu v poli záznamů.)
- seskupení hodnot dle zvoleného sloupce (Uživatel má možnost seskupovat záznamu dle předdefinovaných sloupců v grafu pomocí rozdílných barev bodů.)
- indexování záznamů (Z důvodu optimalizace vyhledávání záznamu dle dané hodnoty v sloupci jsou záznamy indexované v mapě touto hodnotou na index v poli záznamů.)
- vytvoření datové struktury, která umožňuje vizualizaci záznamů v knihovně grafů Highcharts
- výpočet časové řady

1.5.2 Generování časové řady

Generování časové řady se sestává z dvou kroků. V prvním kroku je při procházení záznamů u každého předána doba zahájení a jeho trvání instanci časové řady. Ta zvýší hodnotu v každé položce interního volného pole o jedna v případě, že na daném indexu reprezentujícím časový interval aktivity záznamu byl záznam obsazen po celou dobu intervalu. V případě začátku a konce, který z pravidla nemusí být totožný z začátkem a koncem intervalu (v aplikaci je volen interval o délce 1 hodiny), je přidán pouhý poměr doby aktivity záznamu vůči délce intervalu.
V druhém kroku je volné pole převedeno na pole polí, které je vizualizovatelné knihovnou Highcharts.

1.5.3 Filtrování datové sady

Na základe aktivních selektorů vybraných uživatelem je datová sada filtrována pro následnou vizualizaci. Filtrace je optimalizována několika způsoby:

- je zjišťováno, zda je aktivní (není v defaultním stavu) některý ze selektorů

- jsou porovnávány pouze aktivní selektory
- funkce filtrující záznamy je dynamicky generována při každí změně selektoru (minimalizace kroků funkce)
- filtrované data jsou ukládána v cache paměti frameworku React

2 Uživatelská dokumentace

V této části je vysvětlena obsluha aplikace ve formě manuálu. Aplikaci tvoří dvě hlavní části. První je domovská stránka samotné vizualizace a druhá slouží pro účely konfigurace a určení zdrojů dat.

2.1 Stránka vizualizace

Stránka vizualizace je hlavní a úvodní stránkou aplikace. Je dělena do šesti sekcí:

- **navigace**
- **hlavní graf vizualizace**
- **graf časová řada**
- **tabulka četnosti skupin**
- **selektory**
- **panel nástrojů**

2.1.1 Navigace

Lišta navigace zachycená na obrázku číslo 2 v horní části stránky slouží k přepínání mezi hlavní stranou vizualizace a její konfigurací týkající se zdrojů datových sad určených pro vizualizaci.

Obrázek 2: Navigace

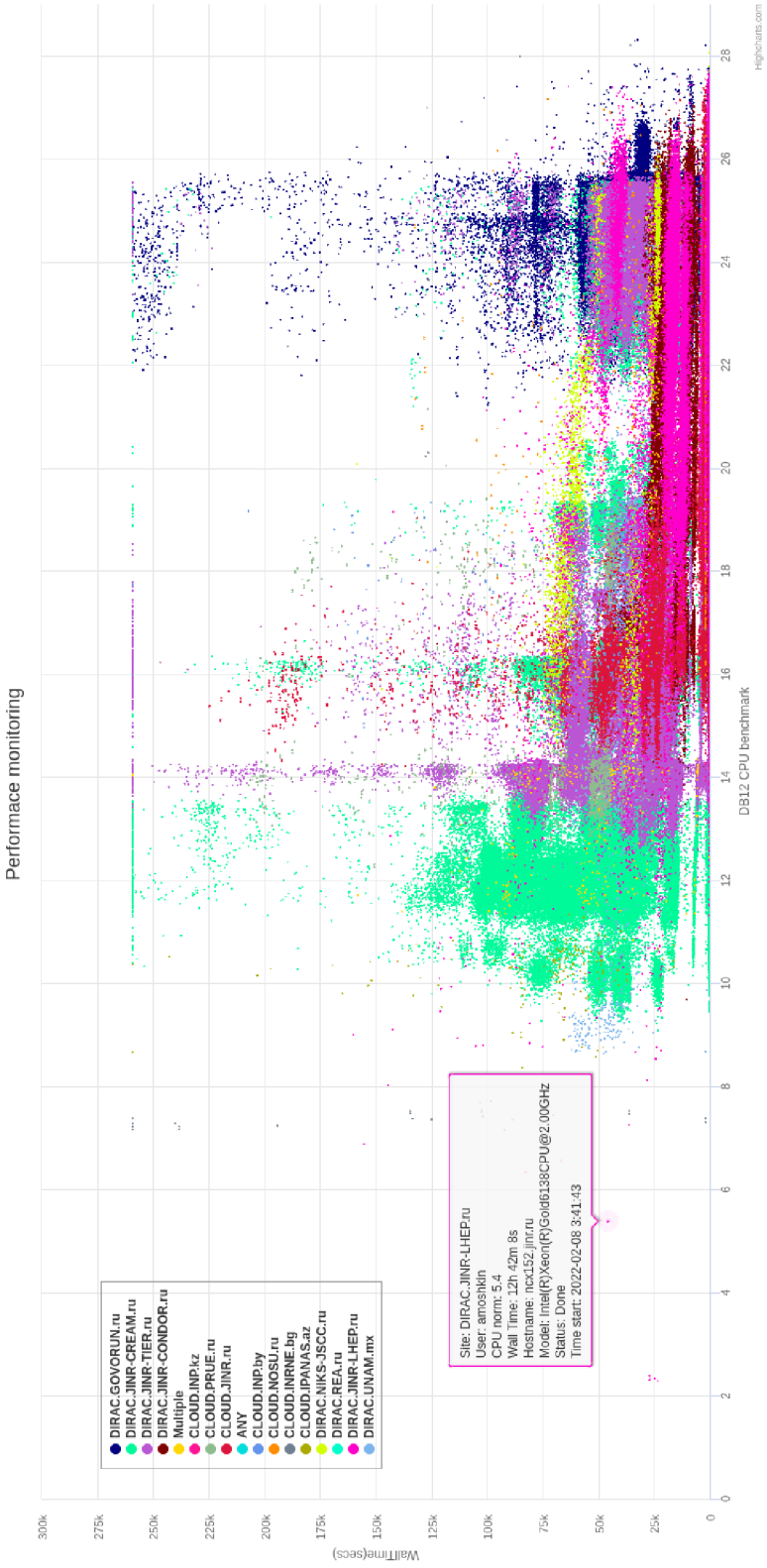
2.1.2 Hlavní graf vizualizace

Tato sekce tvoří ústřední součást aplikace. Bodový graf viz. obrázek číslo 3 vizualizuje závislost délky doby zpracovávané úlohy na indexu benchmarku daného stroje v daném prostředí, na kterém úloha byla vypočítávána. Jednotky na ose X benchmarku stroje reprezentují bezrozměrnou hodnotu, získanou před zahájením vykonávání každé úlohy. Na ose Y je doba vynaložená na vykonání úlohy v sekundách. Záznam o každé jednotlivé úloze je po načtení a zpracování datové sady zobrazen jako bod s přidělenou barvou. Ta znázorňuje seskupení bodu dle dalších klíčů v záznamu. Přednastavený údaj seskupování je vybrán dle doménového jména místa, ze kterého záznam pochází. Mezi další možné klíče pro seskupení patří: mode procesoru a hostname stroje, jméno uživatele pod

kterým úloha byla vytvořena, status průběhu úlohy. Výčet hodnot aktuálního klíče seskupení je zobrazen v legendě s možností skrytí.
Graf je interaktivní v zobrazení podrobných informací o úloze a možnost přibližování částí grafu.
Po přesunutí kurzoru myši nad bod je v jeho blízkosti zobrazena tabulka shrnující údaje daného úlohy. Mezi ně patří tyto vlastnosti:

- User: Jméno uživatele, který vytvořil úkolu.
- Site: Doménové jméno místa, kde úloha byla prováděna. Koreluje s místy organizací zapojených do projektu.
- Status: Status výpočtu úlohy. V případě úspěšného dokončení úlohy – *Done*, neúspěšného – *Failed*. Úlohy, které v době zobrazení jsou v průběhu zpracování (*Pending*), nejsou do vizualizace zahrnuty.
- CPU Norm: Index benchmarku daného stroje získaný před zahájením výpočtu úlohy.
- Wall Time: Doba, po kterou trvat výpočet úlohy a síťového přenosu vstupních a následně výstupních dat.
- Hostname: Hostname daného stroje, na kterém byl výpočet prováděn.
- Model: Model procesoru, na kterém byl výpočet prováděn.
- Time Start: Datum a čas, kdy začal výpočet úlohy (včetně síťového přenosu vstupních dat)

Možnost přiblížení grafu je uživateli zpřístupněna označením části grafu za pomocí stisku levého tlačítka na počátku výběru a uvolnění tohoto tlačítka na konci. K přechodu k původnímu zvětšení slouží tlačítko *Reset zoom* v horní pravém rohu grafu, které se po přiblížení automaticky zobrazí.



Obrázek 3: Hlavní graf vizualizace

2.1.3 Graf časové řady

Graf časové řady viz. obrázek číslo 4 reprezentuje počet úloh vykonávaných v daném časovém intervalu. Časový interval grafu je jedna hodina. V případě úlohy, která nebyla vykovávána v délce celého intervalu, je započítán její podíl trvání na délce intervalu.
Např. jsou-li i intervalu jedné hodiny vykonány tři úlohy, kdy doba trvání každé je deset minut, hodnota daného intervalu na ose y bude součet podílů těchto úloh ku délce intervalu, tedy 0.5.
Graf je interaktivní při přesunu kurzoru nad jeho plochou zobrazením přesných hodnot doby intervalu a jeho hodnoty. Tažením kurzoru se stlačeným levým tlačítkem dojde k výběru časového intervalu, který určí rozsah zobrazených úloh.

Obrázek 4: Časová řada

2.1.4 Tabulka počtu úloh ve skupinách

Řádek tabulky viz. obrázek číslo 5 reprezentuje četnost úloh zařazených do dané skupiny. Při selekci úloh jsou četnosti přepočítány. Mimo jednotlivé skupiny se zde nachází údaj o celkovém počtu záznamů s a bez aktivních selektorů.

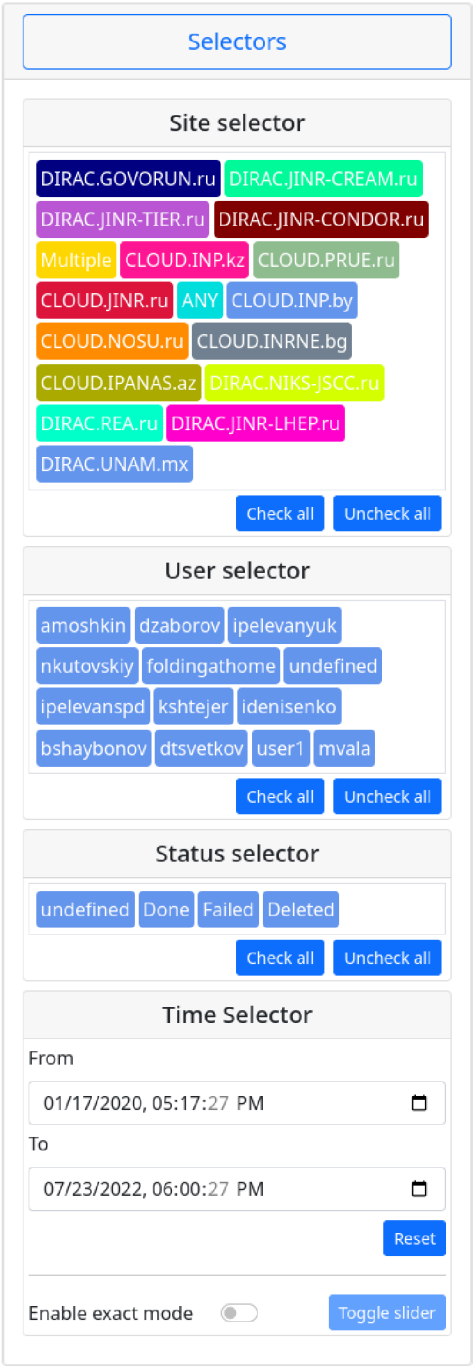
	Number of records
DIRAC.GOVORUN.ru group	268221
DIRAC.JINR-CREAM.ru group	257157
DIRAC.JINR-TIER.ru group	340866
DIRAC.JINR-CONDOR.ru group	73529
Multiple group	944
CLOUD.INP.kz group	104
CLOUD.PRUE.ru group	2870
CLOUD.JINR.ru group	60960
ANY group	7
CLOUD.INP.by group	501
CLOUD.NOSU.ru group	843
CLOUD.INRNE.bg group	90
CLOUD.IPANAS.az group	178
DIRAC.NIKS-JSCC.ru group	3411
DIRAC.REA.ru group	10
DIRAC.JINR-LHEP.ru group	29155
DIRAC.UNAM.mx group	913
All with filters	1039759
All	1039759

Obrázek 5: Tabulka statistik

2.1.5 Selektory

V sloupci nalevo od hlavního grafu se nachází panel selektorů viz. obrázek číslo 6. Ty jsou určeny pro jemný výběr jednotlivých úloh určených pro vizualizaci dle předem určených klíčů. Změna selektorů není potvrzována, ale je obratem vykreslena do grafů a tabulky četnosti. Selektory mohou být dvou typů.
Selektor mnohonásobného výběru umožňuje výběr diskrétních hodnot, kterých klíč může nabývat. Výběr je uživateli umožněn kliknutím na název hodnoty. Barevné pozadí hodnoty reprezentuje vybraný stav, šedé pozadí stav nevybraný. V případě, že selektorem je zároveň klíč seskupování v hlavním grafu, jsou pozadí jednotlivých hodnot vykresleny stejnou barvou jaká náleží dané skupině v grafu. Pro zrychlený výběr nebo jeho zrušení všech hodnot jsou k dispozici tlačítka *Check all* a *Uncheck all*. V aplikaci je využívám selektor mnohonásobného výběru pro výběr hodnot klíčů: Site, User a Status.
Druhým typem selektoru je určen pro výběr rozsahu hodnot mezi nimi je možné uvažovat uspořádání. V aplikaci je využit pro výběr časového rozsahu zobrazovaných úloh výběrem data a času od a do. Zvláštní mód jménem *exact time* umožňuje výběr bodu v čase, který určuje takové úlohy, které v tomto bodě byly prováděny. Po aktivaci tohoto módu se zpřístupní tlačítko zobrazení jezdce výběru. Po zobrazení se na spodní části obrazovky objeví jezdec, kterým je vybrán požadovaný bod v čase. Popřípadě je možné zadat tento časový bod

přesně přímo v samotném selektoru. Během posunu jezdce je možné interaktivně pozorovat vykreslování úloh na hlavním grafu a znázornění časového bodu na grafu časové řady.

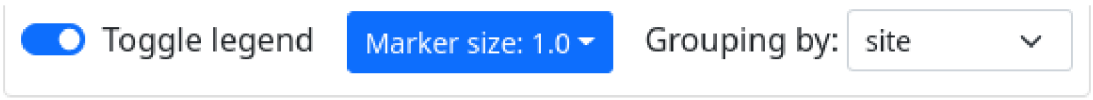


Obrázek 6: Panel selektorů

2.1.6 Panel nástrojů

Panel nástrojů viz. obrázek číslo 7 umístěný v pravém horním rohu stránky slouží nastavení parametrů hlavního grafu. Patří mezi ně:

- *Toggle legend*: pro zobrazení/skrytí legendy v poli grafu
- *Marker size*: pro výběr průměru bodů reprezentující v grafu jednotlivé úlohy
- *Grouping by*: pro výběr klíče seskupování záznamů do barvou odlišitelných skupin



Obrázek 7: Panel nástrojů hlavního grafu

2.2 Stránka konfigurace

Na této straně se nacházejí informace o zdroji dat, ze kterého jsou data pro následovnou vizualizaci získávána, dále sekce pro umožnění přidání nového zdroj dat. V poslední části je zde umožněno nastavení parametrů dat, jako je např. jejich filtrace před samotným zpracováním.

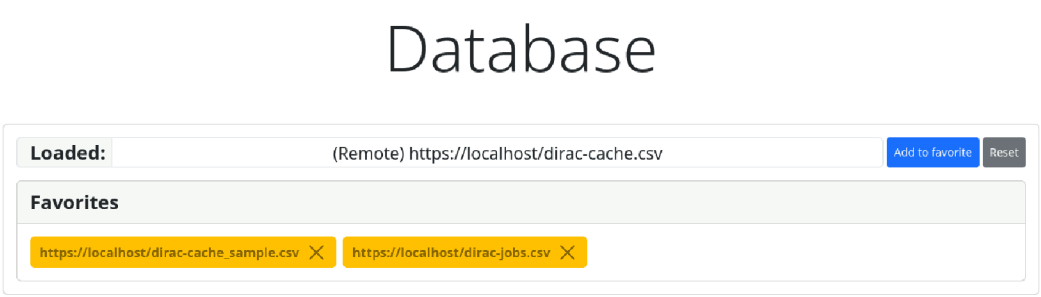
2.2.1 Status zdroje dat

2.2.1.1 Stavový řádek V úvodní sekci stránky konfigurace se nachází stavový řádek viz. obrázek číslo 8 s adresou zdroje dat. Současně je podporován vzdálený a místní zdroj. Vzdálený zdroj je zprostředkován pomocí http popř. https protokolu. Podmínka pro navázání spojení klienta webového prohlížeče se serverem hostujícím datovou sadu podléhá plnění opatření CORS (Cross-Origin Resource Sharing). V praxi to znamená, aby vzdálený webový server v rámci odpovědi na dotaz adresy datové sady v hlavičce též udával tyto záznamy:

- access-control-allow-headers: *
- access-control-allow-methods: GET, POST, OPTIONS
- access-control-allow-origin: *

Parametry těchto hlavičkových záznamů mohou být určeny restriktivněji tak, aby reflektovali bezpečnostní opatření daného serveru.
V případě absence výše zmíněných záznamů v hlavičce je odpověď klientem považovaná za neoprávněnou a je odmítnuta.
Místním zdrojem je soubor datové sady uložený v adresáři na daném počítači.

V případě, že aktuální zdroj datové sady je vzdálený, je možné adresu takového zdroje uložit mezi oblíbené pomocí tlačítka *Add to favorite*.
Tlačítkem *Reset* je navrácen zdroj datové sady na přednastavený
Ukázka stavového řádku i s vloženými oblíbenými zdroji je vidět na obrázku níže.

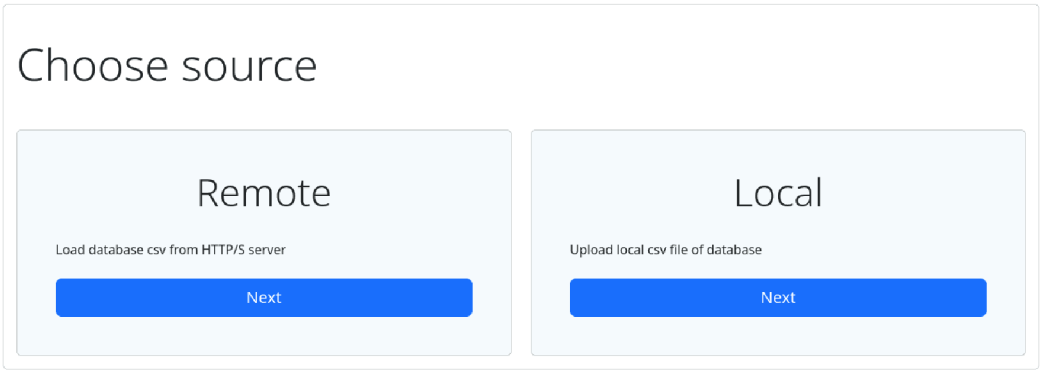


Obrázek 8: Řádek stavu datové sady a sekce oblíbených vzdálených zdrojů

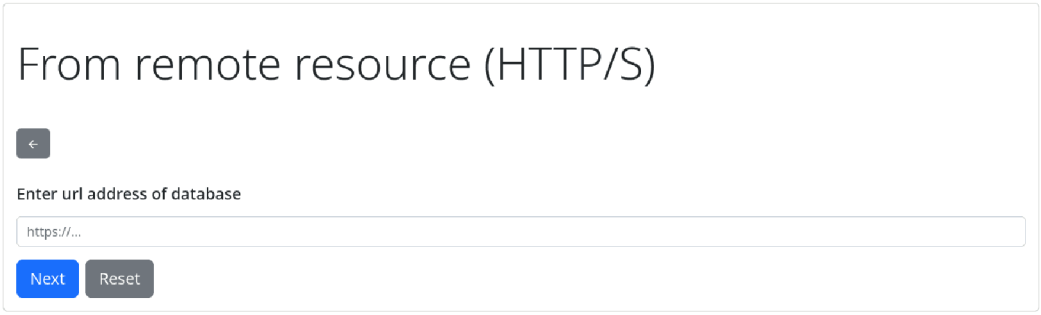
2.2.1.2 Oblíbené zdroje V druhé části této sekce se nachází lišta s přednastavenými či uživatelem uloženými adresami vzdálených serverů. V případě, že jedna položka ze seznamu je aktuálně aktivní, uživatel je upozorněn změnou zabarvení pozadí z žluté na zelenou barvu. Výběr mezi těmito obílenými zdroji je umožněn kliknutím. K odstranění slouží symbol křížku na pravé straně odkazu.
Záznamy oblíbených položek jsou uloženy v paměti prohlížeče, ve které perzistující mezi opětovným načtením stránky.

2.2.2 Výběr zdroje datové sady

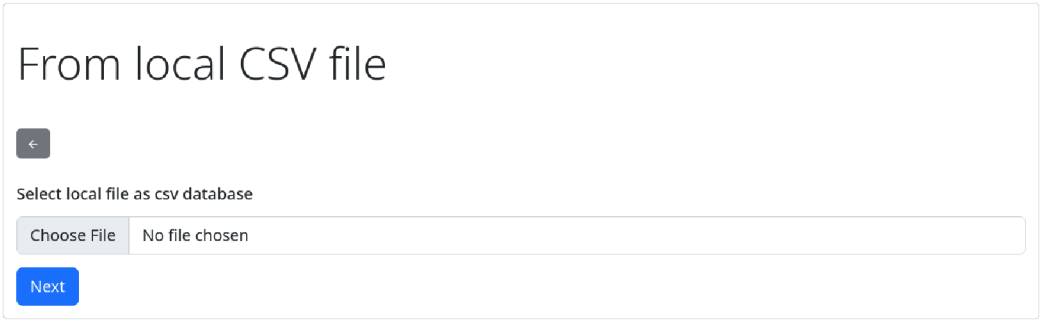
Tato sekce zobrazená na obrázku číslo 9 je určena pro rychlý výběr nového zdroje datové sady. Lze volit z možnosti vzdáleného a místním zdrojem. Datová sada je soubor formátu CSV (Comma-separated values).
Výběrem vzdáleného zdroje viz. obrázek číslo 10 je uživatel dialogem v dané sekci dotázán na zadání adresy tohoto zdroje. Jak bylo přesněji popsáno výše, je podporován protokol http případně https za podmínek CORS. Příkladem takovéto adresy je: „`https://jinr.dccd.cf:4446/dirac-jobs.csv`“.
Adresa je po kliknutí tlačítka *Next* validována a je ověřena její dostupnost.
V případě výběru místního zdroje viz. obrázek číslo 11 je uživateli umožněn výběr csv souboru uloženého v místním adresáři.
Bez vyvstanutí problému v předcházejícím kroku je následně datová sada získána ze vzdáleného či místního zdroje a uživatel je přeměřován na hlavní stránku vizualizace. Zde dojde ke zpracování a zobrazení dat.
Na obrázcích níže se nachází dialog pro výber zdroje datové sady ve třech svých krocích.



Obrázek 9: Dialog přidání nového zdroje dat



Obrázek 10: Přidání vzdáleného zdroje



Obrázek 11: Přidání místního zdroje dat

2.2.3 Natavení datové sady

V současné době tato sekce umožňuje nastavení omezení nejstaršího záznamu, který bude předán pro následné zpracování a zobrazení. Důvodem této prvotní filtrace je nárůst objemu datové sady a tady umožnění zkrácení doby následného zpracování a zobrazování.

2.3 Specifikace datové sady

Aplikace podporuje dva formáty datové sady, oba vycházející ze standardu CSV (RFC 4180). Jedná se o textový způsob ukládání dat. Následně bude popsána struktura těchto formátů.

2.3.1 Standardní CSV

První řádek udává hlavičku tabulky. Hlavička se skládá z po sobě seřazených názvů sloupců oddělených oddělovacím znakem. Následující řádky reprezentují jednotlivé záznamy, kde pořadí hodnot je určeno pořadím názvů těchto hodnot v hlavičce. Odděleny jsou totožným způsobem.
Tento formát odpovídá struktuře CSV souboru s drobnými rozdíly, jakými jsou:

- použití tabulátoru na místo čárek pro oddělení sloupců
- vynechání uzávorkování jednotlivých položek v řádku

2.3.2 Anotované CSV

Výskyt tohoto formátu vychází z historického použití databázového systému InfluxDB. Aplikace umožňuje přímé dotazování pomocí HTTP API na tuto databázi. S přechodem na verzi InfluxDB v2.0 bylo rozhraní značně pozměněno a na tomto základě se přešlo na databázi s formát standardního CSV.
Dle interní logiky databáze InfluxDB jsou záznamy při dotazu klienta seskupovány do skupin s rozdílnými hlavičkami. Tyto skupiny jsou oddělené dvojicí nového řádku. Jako oddělovací znak je použit středník. Hodnoty nejsou uzávorkovány. Samotná skupina připomíná standardní CSV s těmito odlišnostmi.

- Je uvozena až třemi druhy anotací. V aplikaci je používána pouze anotace datového typu. Odlišení řádku s anotací je provedeno uvození řádku křížkem (number sign). Po něm následuje jméno anotace – datatype a dále pořadí datových typů oddělených středníkem. Mezi databází užívané datové typy patří: string, long, double, a dateTime: RFC3339

Po řádcích s anotacemi následuje hlavička skupiny se jmény sloupců a tělo shodující se se standardním CSV formátem.
Je možná konfigurace parametrů tohoto formát přímo v dotazu klienta.
Autentizace klienta je zprostředkována obsažením tokenu do hlavičky dotazu.

2.3.3 Předdefinovaná struktura CSV tabulky

Pro úspěšnou vizualizaci je nutné nejen, aby datová sada odpovídala podporovanému formátu, ale též obsahovala minimální počet sloupců.
Mezi nutně obsažené sloupce (včetně příklady hodnoty) patří:

- id: 1494507 (unikátní identifikátor záznamu; typ: integer)

- `start_time`: 1642168336 (čas započetí vykonávání úlohy, typ: UNIX timestamp)
- `cpu_norm`: 23.4 (benchmark stroje, typ: float)
- `wall_time`: 550.919 (délka vykonávání úlohy, typ: float)

Mezi další volitelné sloupce patří např.:

- user: ipelevanyuk (uživatelské jméno zakladatele úlohy, typ: string)
- userGroup: `mpd_user` (projektová skupina k níž uživatel náleží, typ: string)
- site: DIRAC.GOVORUN.ru (doménové jméno klastru, kam byla úloha přiřazena, typ: string)
- hostname: n02p039.gvr.local (hostname stroje, na kterém byla úloha vypočtena, typ: string)
- model: Intel(R)Xeon(R)Platinum8268CPU@2.90GHz (model procesoru stroje, typ: string)
- memory: 1099332 (paměťové nároky výpočtu, typ: integer)
- `end_time`: 1642168898 (čas dokončení úlohy, typ: UNIX timestamp)
- status: Done (status dokončení úlohy, typ: string)
- `cpu_time`: 12890.592 (doba strávená procesorem nad výpočtem, typ: float)
- `cpu_freq`: 3271 (průměrná frekvence procesoru při výpočtu úlohy, typ: integer)

Hodnoty z sloupců přímo nevyžadované při tvorbě vizualizace jsou uživateli k dispozici po zobrazení detailu úlohy na hlavním grafu.

3 Plán vývoje

V této chvíli aplikace obsahuje základní funkcionalitu, tak jak byla navržena na počátku vývoje. Naskýtá se však mnoho příležitostí ke zdokonalení, kterým bude věnována tato část.

3.1 Sběr a ukládání dat

Způsob ukládání dat se v průběhu vývoje projektu několikrát měnil. Na počátku byl aplikací dotazován webový server, skrz který po získání odpovědi z interní SQL databáze systému DIRAC vygeneroval dynamickou odpověď ve formátu JSON. Z důvodu omezené doby, po kterou jsou úlohy uloženy v interní databázi, se přistoupilo k vytvoření samostatné databáze na samém serveru, kde byla hostovaná webová aplikace. Pro tuto databázi byl vybrán systém InfluxDB, jehož výhody zahrnovaly již zabudované rest http api, které umožňovalo datovou sadu přenášet bez potřeby druhotného backendu přímo na dotaz klienta webové aplikace. Pro zvýšení stability byly dotazy klienta na databázi InfluxDB přesměrovány na zavedeném http serveru Lighttpd, na kterém je hostovaná samotná webová aplikace. Z důvodu masivní změny http api InluxDB s verzí 2.0 a odstranění podpory formátu json bylo rozhodnuto o opuštění tohoto databázového systému.
Řešením se ukázalo ukládání dat do formátu CSV, který v sobě zahrnoval několik výhod. Jeho textová forma je snadno čitelná v surové podobně primitivními nástroji pro zobrazení textu. Mezi další patří snadná přenositelnost mezi systémy, snadná serializace a deserializace, porozumění struktuře dat jiným neškoleným pracovníkem, široké spektrum nástrojů pro analýzu dat tohoto formátu. V komprimované podobě je tento formát vhodný pro sítový přenos.
Nevýhodou je datová repetice v případě těch sloupců jehož hodnoty by bylo možné reprezentovat enumerací. Např. název modelu procesoru jakým je „Intel(R)Xeon(R)Platinum8268CPU@2.90GHz“ by bylo úspornější reprezentovat odkazem na tabulku známých modelů procesoru oproti řetězci. Toto však CSV formát neumožňuje.
Nyní je sběr dat zabezpečen pomocí skriptu, který dotazuje interní databázi systémy DIRAC. Po načtení unikátních identifikátorů dosud známých úloh ze souboru CSV jsou procházeny záznamy z interní databáze a vybrány takové, u kterých jejich unikátní identifikátor není znám. Taková to množina záznamů je převedena na řetězec ve formátu CSV a připsána na konec původního CSV souboru databáze. Data jsou aktualizována s denní četností. CSV soubor databáze je následně dostupný na známé adrese přes http(s) protokol.
Je plánováno testovat nový binární formát dat, jehož požadovanou výhodou by mělo být co možná největší omezení objemu datového přenosu na síti. V současnosti, kdy velikost databáze ve formátu CSV o jednotkách miliónu záznamů činí stovky megabytů a po následné komprimaci desítky megabytů, problém není natolik palčivý. Druhou možností možností by mohlo být rozdělení datové sady dle časových intervalů (např. měsíců) a následnou možnost

konfigurace klienta, který by žádal pouze nezbytný počet záznamů. Toto by však vyžadovalo vybudování backend api pro průzkum adresáře s dělenou databází, protože protokol http sám o sobě průzkum statického adresáře nepodporuje. Výhodou současného stavu bez použití backendu je možnost použití jakéhokoliv generického webového serveru pro statické hostování souboru databáze.

3.2 Využití webassembly při zpracování dat a jejich filtrování

Časově náročnou operací je zpracování dat ve formátu CSV do interní reprezentace aplikace a následné filtrování, které je ústřední funkcionalitou následné vizualizace. Při současných jednotkách miliónu záznamů je po získání ze vzdáleného serveru doba zpracování obvykle do několika sekund. Palčivější je pak délka odezvy při filtrování, která je v tuto chvíli stále uživatelsky únosná. Např. při kontinuálním filtrování pomocí času je graf vykreslován několikrát za sekundu. I přes mnoho provedených optimalizací ekosystém JavaScriptu neposkytuje další možné urychlení a tím dosažení lepší uživatelské ovladatelnosti bez ohledu na předpoklad přibývajícího počtu záznamů.
Jedním z navrhovaných řešení by mohlo být začlenění webassembly při procesu filtrování. Pro uskutečnění je nutné překonat několik překážek. Jestliže by data byla uložená v proměnné v prostředí javascriptu, je nutné celou interní reprezentaci databáze kopírovat do prostředí webassembly, zde provést filtraci a následně data v paměti opět zkopírovat do prostředí JavaScriptu. Doba vynaložená na toto kopírování smaže časovou úsporu rychlejšího kompilovaného kódu webassembly. Jediná možnost, jak předejít nutnosti kopírování dat v paměti je její sdílení, avšak datový typ, kterému je to v prostředí JavaScrptu umožněno je pouze SharedArrayBuffer, tedy druh pole, kde každý jeho prvek je jedním z povolených primitivních datových typů. Řešením by pak mohlo být použití binární reprezentace databáze zmiňované v předchozí kapitole o sběru a ukládání dat. Tato binární databáze by nemusela být po získání se vzdáleného serveru deserializována, jako je tomu právě u formátu CSV nebo JSON, ale přímo by byla nahraná do paměti v prostředí webassembly. Grafické rozhraní aplikace by následně volalo webassembly funkci a výsledek filtrace by byl kopírován a transformován pouze jednou, tak aby byl zpracován grafovou knihovnou Highcharts.
Avšak vývoj vlastního binárního formátu databáze se ukázal komplexním a časově náročným, a proto bylo pro zatím od této strategie upuštěno. Dále slibovaná časová úspora při filtrování nebyla s ohledem na úsilí při přechodu na tuto strategii spolehlivě předpovězena.

3.3 Práce v offline režimu

Jedna z praktických, ale projektem nezbytně nevyžadovaných vlastností se mohla stát možnost instalace aplikace v prohlížeči a její fungování bez připojení k Internetu. Za využití již rozšířeného prohlížečového api ServiceWorker je možné

otevřít aplikaci na adrese pod podmínkou předchozího nahrátí do cache paměti prohlížeče jejím navšívením. ServiceWorker v tomto případě funguje jako proxy pro síťové spojení se serverem hostující aplikaci a zajistí uložení nebytné části. Ve spojení s možností nahrání dat přímo z místního adresáře odpadá po instalaci nutnost internetového připojení.

3.4 Načítání odlišných datových sad

V případě pokusu o načteni datové sady, ve které nejsou obsaženy všechny nutné sloupce totožně pojmenované, dojde k chybě. Není tak možné pracovat s jinými typy datových sad než definované ve specifikaci. Současně jediným zdrojem dat pro vizualizaci je skript, který plně dodržuje tuto specifikaci. Pro usnadnění použitelnosti pro datové sady generované jinými způsoby, je navrženo doplnit dialog přidání nového vzdáleného zdroje dat o část, v níž je uživateli nabídnuta možnost přidělení významu jednotlivým sloupcům, se kterými pak vizualizace dokáže nakládat.

Závěr

Práce byla dokončena v požadovaném rozsahu dle specifikace projektu. Aplikace se osvědčila při svém nasazení v praxi a dovedla zefektivnit analýzu dat o zpracování výpočetních úloh v systému DIRAC. Aplikace se v provozu prokázala intuitivním použití, očekávanou rychlostí při filtraci a robustností. Analýza systémy zprostředkovaná touto aplikací objevila podněty pro možnou optimalizaci zasílání a výpočtu úloh, které jsou v současné době v šetření.
Nesmírnou měrou si vážím možnosti spolupráce s institucí SŮJV, která mi nabídla unikátní pohled na práci programátora ve vědeckém prostředí a způsobila přelom mého odborného směřování. V prostředí Laboratoře informatických technologií ústavu jsem byl seznámen s fungováním superpočítače. Při práci na tomto projektu jsem nabyl neocenitelné zkušenosti od počátečního návrhu softwarového řešení, přes vývoj a testování, po nasazení aplikace do provozu, které mi při mé budoucí práci budou užitečné.

Conclusions

The work was completed to the required extent according to the project specification. The application has proven itself in practice and has been able to streamline the analysis of data on the processing of computational tasks in the DIRAC system. In operation, the application has proven to be intuitive to use, the expected filtering speed and robustness. The analysis of the systems mediated by this application discovered the stimuli for the possible optimization of the sending and calculation of tasks, which are currently under investigation. I greatly value the possibility of cooperation with the JINR institution, which offered me a unique view of the work of a programmer in a scientific environment and caused a turning point in my professional direction. In the environment of the Institute's Information Technology Laboratory, I was introduced to the operation of a supercomputer. While working on this project, I gained invaluable experience from the initial design of a software solution, through development and testing, to deployment of the application, which will be useful for my future work.

A Instalace a spuštění serveru

Produkční verze aplikace je nasazena na dvou místech. Prvním z nich je platforma Gitlab. Za využití nástrojů platformy CI/CD (continuous integration/continuous delivery) a Giltab Pages je proces nasazení aplikace automaticky spuštěný po každého příkazu git push a tedy nahrátí příslušného commitu do git repozitáře projektu. Aplikace je dostupná na adrese:
`https://jinr-dc.gitlab.io/performance-monitoring/`
Druhé nasazení aplikace je na virtuálním serveru s operačním systémem CentOS 7 v místní síti ústavu SÚJV. V současnosti tomuto serveru není přidělena veřejná adresa a veškerá komunikace je mimo toto pracoviště uskutečňována za pomocí SSH port forwardingu a tedy nepřístupná pro neautentizovaného uživatele.
Popis instalace bude popsán pro případ druhý, tedy v případě virtuálního nebo fyzického serveru. Instalace nezahrnuje část shromažďující záznamy výpočetních úloh.

A.1 Požadavky

Požadavky pro hostování webové aplikace

- Operační systém založený na UNIX (vyvíjeno paralelně na CentOS 7 a Arch Linux)
- Lighttpd webový server (vyvíjeno na verzi 1.4.54)

A.2 Instalace

Instalace se skládá z kroků:

- instalace webového serveru Lighttpd
- konfigurace webového serveru Lighttpd
  - určení kořenového adresáře hostované aplikace
  - aktivace komprese http odpovědí serveru
  - aktivace šifrování tls a nahrání příslušného certifikátu ve formátu pem včetně privátního klíče
  - přesměrování každého dotazu na neexistující hostovaný soubor na vstupní html soubor aplikace dle pravidel SPA React.js
  - definování CORS hlaviček pro možnost cross-site dotazů klienta prihlížeče
  - nastavení doménových a subdoménových jmen serveru
- konfigurace u poskytovatele domény, směřování doménového jména na ip adresu serveru

- klonování zdrojového kódu aplikace z git repozitáře příkazem `$ git clone https://gitlab.com/jinr-dc/performance-monitoring.git`
- přechod do adresáře projektu `$ cd performance-monitoring`
- instalace závislostí projektu `$ npm install`
- sestavení aplikace `$ npm run build`
- přesunutí obsahu adresáře `./front/dist/` do adresáře nakonfigurovaného na webovém serveru pro hostování aplikace

A.3 Instalace

Po spuštění služby webového serveru `$ systemctl start lighttpd` je aplikace dostupna z ip adresy serveru. Pro okamžité spuštění služby po spuštění serveru je doporučeno ji aktivovat `$ systemctl enable lighttpd` v případě, kdy tento krok není zahrnut při samotné instalaci webového serveru.

B Spuštění aplikace

Aplikace byla testována na prohlížeči Google Chrome v93.0 a Firefox v93 a je těmito prohlížeči plně podporována. V případě ostatních prohlížečů není zaručena správná funkcionalita.
Prohlížeč je nainstalován dle postupu v příslušném manuálu. Pro spuštění aplikace není zapotřebí jiného zásahu do nově nainstalovaného prohlížeče. K spuštění aplikace dojde po zadání její adresy do adresního řádku.

C Obsah přiloženého datového média

`performance-monitoring/`
: Složka obsahuje git repozitář projektu, ze kterého je webová aplikace sestavitelná dle instrukcí v souboru readme.txt

`doc/`
: Obsahuje text práce ve formátu PDF a soubory nutné pro jeho vygenerování.

`readme.txt`
: Soubor obsahuje informace pro zprovoznění serveru, aplikace a také přístupové údaje do administrace.

Navíc médium obsahuje:

`data/`
: Ukázková a testovací data použitá v práci a pro potřeby testování práce při tvorbě posudků a obhajoby práce.

Bibliografie

[1] OSMANI, Addy. Learning JavaScript design patterns: [a JavaScript and jQuery developer's guide]. Sebastopol: O'Reilly, 2012. ISBN 978-144-9331-818.

[2] BUGL, Daniel. Learn React Hooks: build and refactor Modern React.js applications using Hooks. Birmingham: Packt, 2019. ISBN 978-183-8641-443.

[3] ZAKAS, Nicholas C. JavaScript pro webové vývojáře: build and refactor Modern React.js applications using Hooks. Brno: Computer Press, 2009. Programujeme profesionálně. ISBN 978-802-5125-090.

[4] MASTERS, Jon a Richard BLUM. Linux profesionálně: programování aplikací. Brno: Zoner Press, 2008. Encyklopedie Zoner Press. ISBN 978-80-86815-71-8.

[5] Documentation. Documentation - DIRAC Documentation [online]. [cit. 2022-08-03]. Dostupné z: https://dirac.readthedocs.io/

[6] Dirac The Interware [online]. [cit. 2022-08-03]. Dostupné z: http://mardirac.in2p3.fr/

[7] The modern JavaScript tutorial. The Modern JavaScript Tutorial [online]. [cit. 2022-08-03]. Dostupné z: https://javascript.info/

[8] Webpack [online]. [cit. 2022-08-03]. Dostupné z: https://webpack.js.org/

[9] Getting started. React. [online]. [cit. 2022-08-03]. Dostupné z: https://reactjs.org/docs/

[10] Highcharts documentation: Highcharts. Interactive javascript charts library. [online]. [cit. 2022-08-03]. Dostupné z: https://www.highcharts.com/docs/index

[11] Babel [online. [cit. 2022-08-03]. Dostupné z: https://babeljs.io/

[12] Eslint [online]. [cit. 2022-08-03]. Dostupné z: https://eslint.org/

[13] MobX [online]. [cit. 2022-08-03]. Dostupné z: https://mobx.js.org/README.html

[14] Bootstrap [online]. [cit. 2022-08-03]. Dostupné z: https://getbootstrap.com/

[15] Gitlab [online]. [cit. 2022-08-03]. Dostupné z: https://gitlab.com/