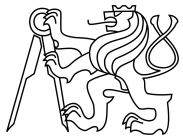
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Grafické rozhraní simulátoru disků DiskSim
Student:	Kamil Jakubovič
Vedoucí:	Ing. Jiří Kašpar
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Seznamte se s funkčností simulátoru DiskSim. Pro DiskSim navrhněte a implementujte GUI nadstavbu pro:
- editaci diskové konfigurace,
- volbu scénáře a parametrů simulace,
- spouštění běhu simulátoru a vyhodnocení výstupů simulátoru,
- grafické zobrazení výstupů simulace.

Funkčnost GUI nadstavby podrobte testům použitelnosti podle pokynů vedoucího práce.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 20. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Grafické rozhraní simulátoru disků DiskSim

Kamil Jakubovič

Vedoucí práce: Ing. Jiří Kašpar

16. května 2017

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Kašparovi za odborné vedení a podnětné rady.

Prohlášení

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

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

V Praze dne 16. května 2017 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Jakubovič, Kamil. *Grafické rozhraní simulátoru disků DiskSim*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017. Dostupný také z WWW: ⟨https://gitlab.fit.cvut.cz/jakubkam/disksimui⟩.

Abstrakt

Náplní této bakalářské práce je návrh a realizace grafického uživatelského rozhraní simulátoru disků DiskSim. Účelem rozhraní je snadná a přehledná práce se simulátorem s důrazem na grafickou interpretaci jeho výsledků. Klíčovými požadavky na rozhraní jsou snadná instalace, platformní přenositelnost a budoucí rozšiřitelnost. Ty spolu s poznatky z analýzy existujících řešení vedou k návrhu síťové architektury klient-server. Výsledkem praktické části práce je samostatný webový server spuštěný pod minimalistickým aplikačním rámcem Bottle sloužící jako spojovník mezi webovým prohlížečem uživatele a vlastním simulátorem. Použitím rozhraní REST lze navíc simulátor ovládat i vzdáleně a neinteraktivně. Snímky obrazovky prohlížeče uvedené v implementační části ozřejmí snadnost použití grafického rozhraní a názornou reprezentaci výsledků zadaných simulací.

Klíčová slova grafické rozhraní simulátoru disků, návrh, přenositelnost, simulátor DiskSim, aplikační rámec Bottle, rozhraní REST.

Abstract

This bachelor thesis deals with a design and implementation of a graphic user interface to the disk simulator DiskSim. The main target of the thesis is an easy and clear work with the simulator with an emphasis on a graphic representation of its results. Easy installation, platform independence, and future extensibility are the key features of the interface. These keys together with a result of already existing solutions lead in a network architecture client-server. The outcome of the practical part is a standalone web server run under a minimalist framework Bottle joining a user web browser and the simulator. The simulator can also be controlled remotely and noninteractively thanks to the used REST API. The web browser screenshots listed in the implementation chapter shows the easiness of the interface and the illustrative representation of simulator results.

Keywords graphic interface to disk simulator, design, platform independent, simulator DiskSim, framework Bottle, REST API.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Simulátor disků DiskSim [1] vyvinutý Carnegie Mellon univerzitou v Pittsburgu je výkonným a přesným simulátorem úložných zařízení používaných ve výpočetní technice. Přestože jeho vznik spadá ještě do předchozího milénia, je pro svou kvalitu simulace a bezprecedentní přesnost dodnes hojně využíván nejen v oblasti výzkumu a vzdělání. Uživatelským rozhraním simulátoru je příkazová řádka, která je sice mocná a efektivní, ovšem pracuji s ní v duchu tradičního černého pozadí prakticky poslepu. Její textová povaha velmi limituje názornost a přehlednost výsledku příkazů a programů v ní spuštěných. Proto akademické prostředí vzneslo požadavek na nové grafické rozhraní k tomuto programu.

Možností řešení zadání je nemálo, ovšem vyhovění požadavkům a postupnou analýzou vhodných technologií jsem dospěl k síťové architektuře klient-server realizované snadno přenositelným minimalistickým aplikačním rámcem webového serveru ovládající vlastní simulátor a prohlížeče, který plnohodnotně splňuje nastolená očekávání moderního uživatelského rozhraní.

V rámci návrhu řešení jsem uplatnil takové postupy a technologie, které zároveň umožní využití aplikace nad rámec zadání. Použitá architektura mi dovoluje oddělit uživatele od simulátoru a mohu tak např. provozovat jen jediný silnější stroj se simulátorem pro plnou učebnu tenkých klientů. Pečlivá platformní nezávislost a deklarativní konfigurace rozhraní simulátoru i uživatelského mi usnadňuje pozdější rozšíření o jiné další simulátory. Implementované rozhraní REST přímo vybízí i k neinteraktivnímu použití různými skripty např. pro automatické či pravidelné úlohy.

Vzhledem k popsaným vlastnostem řešené aplikace očekávám její použití i za branami akademického světa a proto mi bylo i toto její možné praktické široké využití významnou motivací k její realizaci.

Cílem práce jsou následující body:

- seznámení se simulátorem DiskSim,
- spouštění simulací přes grafické rozhraní,
- grafická prezentace výsledků simulací.

Práci jsem rozdělil na tři hlavní kapitoly. V Analýze se zabývám rešerší existujících a podobných řešení, popisuji simulátor DiskSim a sepisuji požadavky zadavatele. V Návrhu řešení předkládám základní koncepci, z ní vzešlou serverovou a klientskou část a k tomu popis aktivit. Poslední velká kapitola popisuje implementaci serverové a klientské části.

Kapitola 1

Analýza

Účelem této kapitoly je seznámení s problematikou, deklarace požadavků zadavatele, rozbor již dostupných nebo podobných řešení.

1.1 Rešerše existujících a podobných řešení

Užitečným krokem mi je vyhledání a rozbor již existujících řešení, na které lze buď navázat nebo se poučit jimi zakusenými úskalími.

- **FlashSIM, PFSsim, Structual Simulation Toolkit**

  Program DiskSim byl často použit s malými úpravami pro ověření speciálních úložných systémů např. FlashSIM [2], nebo jako součást komplexních simulačních systémů, např. PFSsim [3] či Structual Simulation Toolkit [4]. Ty ovšem nepokrývají požadavky práce, neboť nejsou vůbec nebo primárně grafickým rozhraním k programu DiskSim.

- **DiskSim Web**

  Jedinou takovou skutečnou alternativou je projekt DiskSim Web [5]. Oproti požadavkům ovšem není přenositelný (používá např. UNIXové cesty a bash skripty), pokrývá prakticky jediný scénář, grafické výstupy jsou generovány až na klientovi, aplikace je jednoúlohová, vyžaduje kompilaci programu DiskSim, není distribuovaná v balíčcích pro jednoduchou instalaci, vyžaduje přístup na Internet pro instalaci závisejících komponent, funkcionalita je na úrovni kódu a ne deklarativně v konfiguraci.

- **RAID Interactive Tutorial**

  Mezi zajímavé nástroje diskových polí patří například RAID Interactive Tutorial [6] od společnosti Intel. Ten ve spartánské terminálové grafice nabízí konfiguraci obvyklých diskových polí s možností změny pár parametrů diskového adaptéru, čili řadiče. Ačkoli jde o nástroj ukázkové

konfigurace firemních produktů, postrádá požadované možnosti simulace provozu a zpracování jejich výsledků.

- **RAID Simulator**

  Pro orientační náhled na práci běžných diskových polí lze vyzkoušet jednoduchou webovou stránku RAID Simulator [7], kde lze 4 diskům nastavit pravděpodobnost chyb a získat velmi strohou statistiku. Názorná jednoduchost ovšem zároveň dramaticky omezuje možnosti nastavení i získaných výstupů.

- **RAID Calculator**

  Dobrou představu o využitelnosti místa na různě uspořádaných diskových polích poskytuje webový RAID Calculator [8] pomocí interaktivního grafického konfigurátoru bezprostředně zobrazující poměry využitelného a volného místa. Zde se ovšem žádných simulací již nedočkáme.

1.2 Seznámení se simulátorem DiskSim

DiskSim je jednovláknový program pro simulaci magnetických pevných disků, SSD disků a MEMS úložišť [9], včetně ovladače zařízení, diskových řadičů, přenosových sběrnic a jednoúrovňových diskových polí. Je napsán v programovacím jazyce C a od operačního systému, pod kterým je spouštěn, vyžaduje pouze POSIX rozhraní.

Jeho poslední verze 4.0 byla vydaná v červnu roku 2008. Jsou k němu dostupná množství dalších rozšíření a úprav formou patch souborů. Překladem zdrojového kódu se vyrobí hlavní program s CLI rozhraním `disksim` a několik dalších, např. pro zjišťování a měření parametrů skutečných zařízení nebo pro zaznamenávání diskových operací nad nimi prováděnými, oboje dále použitelné v následných simulacích.

Princip simulace spočívá ve zpracování diskových požadavků jednotlivými komponentami simulovaného úložného systému vyúsťující v konkrétní diskové operace nad simulovanými disky. Během toho se důkladně zaznamenává provedená činnost a její měřitelné vlastnosti. Nakonec se vypočítají statistické přehledy. Výsledky simulace mají běžně textový formát.

1.2.1 Topologie úložného systému

Uspořádání úložného systému v simulátoru pracuje se čtyřmi základními typy komponent:

- právě 1 ovladač zařízení (*driver*, `iodriver`),
- diskový řadič (*controller*, `ctlr`) za ovladačem nebo i za prvním řadičem, s nejvíce čtyřmi následujícími sběrnicemi,
- diskové zařízení (*device*) jedině za řadičem, podporující tyto modely: konvenční (`disk`), zjednodušený s konstantní přístupovou dobou (`simpledisk`), MEMS úložiště (`mems`) a SSD disky (`ssd`) [10].
- přenosová sběrnice (`bus`) vždy povinně propojující předchozí tři komponenty, každá jich však nejvíce 15.

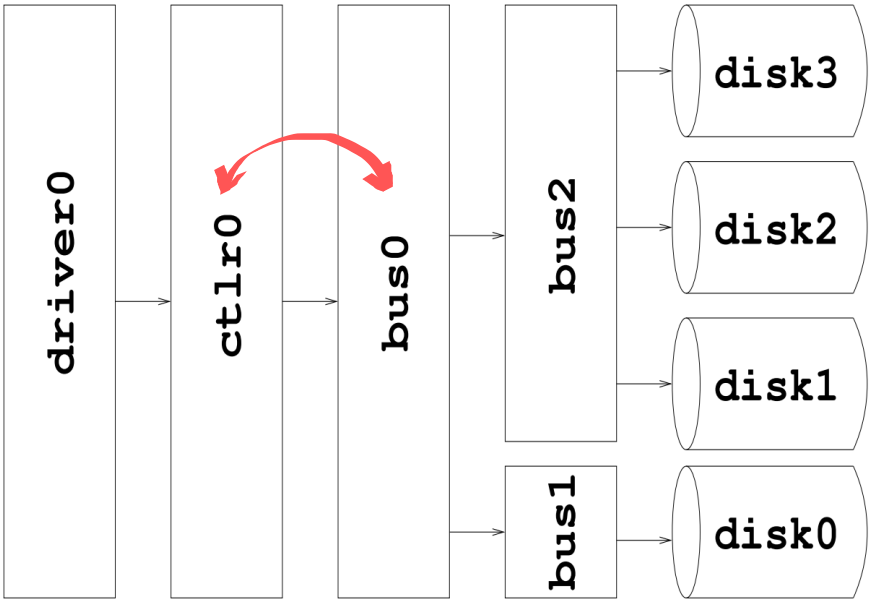
Ovladači, řadičům a diskům lze nastavovat i fronty s plánovačem.

Disková pole nejsou implementována na úrovni řadiče, ale pouze jako logická organizace (`logorg`) diskových zařízení [11, str. 35]. Kvůli tomu DiskSim podporuje jen tyto jednoúrovňové RAID typy: 0, 1, 3, 4, 5 a JBOD. Pole typu RAID 2 není podporováno, neboť DiskSim neumožňuje nakonfigurovat rozdělování dat přes disky v poli po bitech se zabezpečením Hammingovým kódem. Pole typu RAID 6 vyžaduje dva paritní disky, ovšem hodnota `Parity rotated` parametru `Redundancy scheme` podporuje pouze jeden takový disk [11, str. 35].

Příklad části konfigurace – topologie úložného systému [11, str. 34]:

```
topology disksim_iodriver driver0 [
  disksim_bus bus0 [
    disksim_ctlr ctlr0 [
      disksim_bus bus1 [
        disksim_disk disk0 []
      ], # end of bus1
      disksim_bus bus2 [
        disksim_disk disk1 [],
        disksim_disk disk2 [],
        disksim_disk disk3 []
      ] # end of bus2
    ] # end of ctlr0
  ] # end of bus0
] # end of system topology
```

Tomu odpovídá přehlednější zobrazení uvedené tamtéž [11, str. 34], ovšem s chybou, jež pro názornost ne zcela přesné dokumentace simulátoru ponechávám s barevným zvýrazněním:



Obrázek 1.1: Příklad topologie simulovaného úložného systému

1.2.2 Spuštění simulátoru

DiskSim potřebuje pro spuštění simulace soubor s parametry popisující celý simulovaný diskový systém i chování vlastního simulátoru. Dalším jeho nezbytným vstupem jsou diskové operace simulovaně nad systémem prováděné a to ve formě předem jinde zaznamenaných na nějakém reálném zařízení nebo synteticky generovaných podle zadaných kritérií. Výsledkem je textový protokol s detailními průběžnými i souhrnnými informacemi o proběhlé simulaci.

Jedna simulace se tedy spustí následujícím příkazem:

```
disksim   <parfile>    <outfile>   <tracetype> \
          <tracefile>  <synthgen> \
        [ <komponenta> <parameter> <hodnota> ... ]
```

kde argumenty jsou:

- `parfile` ... jméno souboru s parametry,
- `outfile` ... jméno výstupního souboru, přičemž hodnota `stdout` ponechává výstup programu na standardním výstupu,
- `tracetype` ... formát souboru `tracefile`,
- `tracefile` ... jméno souboru s předem zaznamenanými diskovými operacemi, přičemž hodnota `stdin` je čte ze standardního vstupu,
- `synthgen` ... příznak použití generovaných diskových operací místo předem zaznamenaných, pokud je hodnota jiná než `0`,

a kde volitelné parametry dané komponenty doplňují nebo nahrazují ty v souboru `parfile` zadanou novou hodnotou.

Formát souboru s parametry je popsán v referenční příručce [11], ovšem s tou nezdokumentovanou skutečností, že většina volitelných parametrů musí být uvedena, i když se v daném kontextu nepoužívá. Např. `Number of copies` má být při nastavení `Redundancy scheme = Noredun` ignorováno [11, str. 36], místo toho ale DiskSim vypisuje chybovou hlášku
`missing required parameter`.

Konfigurační direktiva `source` a několik málo parametrů (například `Stat definition file`) se odkazují na další soubory a tak je pro spuštění simulace typicky potřeba více než jen jeden konfigurační soubor. Zdrojový soubor simulátoru DiskSim `libparam/libparam.lex` ovšem obsahuje chybu při zpracování relativní cesty direktivy `source`, kdy cestu jinou než do aktuálního adresáře zdvojuje a tak je nezbytné mít všechny konfigurační soubory v jednom adresáři nebo používat absolutní cesty.

1.2.3 Výstupy simulátoru

Výstup simulátoru tvoří nejčastěji nestrukturalizovaný text s obsahem vcelku jemně konfigurovatelným parametry v bloku `stats` [11, str. 8]. Ukázka jeho rozmanitosti:

```
OVERALL I/O SYSTEM STATISTICS
-----------------------------
Overall I/O System Total Requests handled:      9999
Overall I/O System Response time average:       10.937387
Overall I/O System Response time std.dev.:      7.836442
Overall I/O System Response time maximum:       71.567332
Overall I/O System Response time distribution
   <  5    < 10    < 20    < 40    < 60    < 90    <120
       <150    <200    200+
   2338    2622    3961    1003      67       8       0
      0       0       0

IOdriver #0 device #2 Physical access time distribution
4.124         57          0.005701         0.289429
4.166          5          0.000500         0.289929
```

Výstupní textový soubor daný parametrem `Output file for trace of I/O requests simulated` obsahuje následující informace o každém příchozím diskovém požadavku: čas příchodu požadavku v milisekundách, číslo zařízení, číslo bloku, velikost diskových dat v blocích a doplňující příznaky (např. jde-li o čtení nebo zápis). Příklad řádky tohoto souboru:

```
7686.009993 0 14908464 8 40000002
```

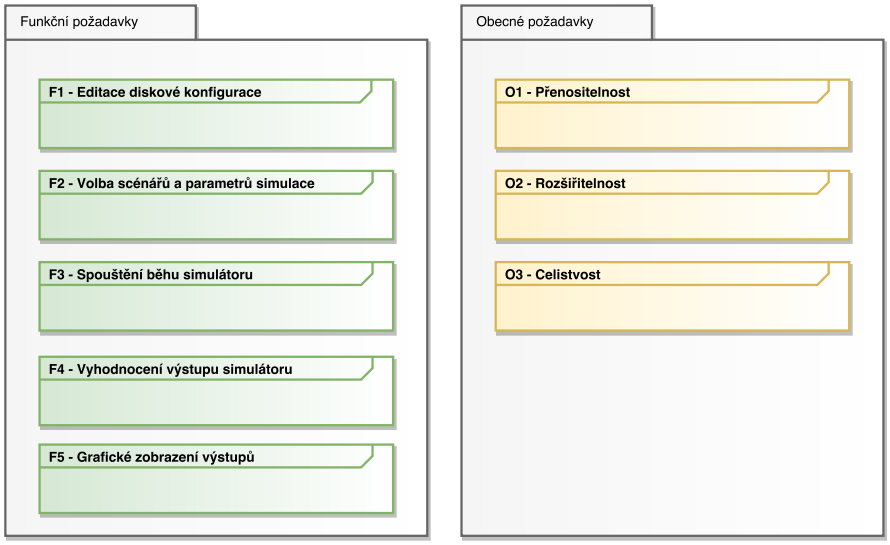
Dalším výstupem simulátoru je textový soubor daný parametrem `Detailed execution trace`, obsahující záznamy o každé provedené diskové operaci, typicky s řádky podobnými této:

```
Request completion: simtime 14203.169602, devno 2, blkno
    868240, time 0.000000
```

1.3 Shrnutí požadavků

Nezbytnou kapitolou pro splnění cíle je jasná a kompletní specifikace požadavků zadavatele. Správné porozumění často zprvu nevyslovených představ je naprosto klíčové jak pro analytickou část, tak též pro úspěšné předání díla.

Níže uvedený přehled požadavků je dále detailně rozebrán:



Obrázek 1.2: Přehled požadavků

1.3.1 Funkční požadavky

Tyto požadavky popisují co je a co není od aplikace očekáváno.

1.3.1.1 F1 – Editace diskové konfigurace

Uživateli se nabízí sada připravených konfigurací disků, sběrnic, řadičů, ovladačů, jejich různá kombinace, pole disků JBOD, RAID 0, 1, 5 v ustáleném stavu. Navíc může vytvářet vlastní zcela novou nebo kopírovat nějakou existující, upravovat je i mazat. Takovouto připravenou konfiguraci si může stáhnout jako konfigurační soubor pro DiskSim.

1.3.1.2 F2 – Volba scénářů a parametrů simulace

Uživateli se nabízí sada připravených scénářů a parametrů simulace, tedy popis diskových požadavků s možností kombinace až tří zdrojů zátěže definovaných ve vlastním konfiguračním souboru, s těmito typy zátěží: sekvenční/náhodný přístup, daný poměr čtení/zápis, velikost IO – konstantní/náhodná, konstantní/zvyšující se počet vláken. Dále lze nastavovat detaily výstupů a další zbylé parametry.

Lze volit z následujících režimů činnosti:

- saturační analýza – vliv zvyšující se zátěže na výstupní parametry – až do saturace,
- what-if analýza – vliv různých dalších změn na výstupní parametry:
  - porovnání různých zátěží pro stejnou konfiguraci,
  - porovnání různých konfigurací pro stejnou zátěž nebo porovnání saturační analýzy pro různé konfigurace,
  - vliv různých parametrů zátěže – velikost I/O, poměr čtení/zápis.

1.3.1.3 F3 – Spouštění běhu simulátoru

Výběrem diskové konfigurace, scénáře a parametrů simulace uživatel spustí simulaci. Tu server spustí na pozadí rovnou nebo ji zařadí do fronty kde čeká na volné prostředky. Uživatel mezitím může s rozhraním dál pracovat. Může prohlížet, prohazovat i mazat frontu čekajících simulací. Může nahlížet nebo přerušit právě běžící simulace. Dokončená simulace uživatele upozorní a nabídne mu další akce nad jejími výsledky, tedy zobrazení výstupů nebo popis chyb. Server průběžně kontroluje frontu požadavků na simulace a podle jejich priorit a volných prostředků je spouští na pozadí.

1.3.1.4 F4 – Vyhodnocení výstupu simulátoru

Z úspěšně dokončených simulací může uživatel získávat:

- detailní informace – propustnost (v IO/s a kB/s), latence,
- souhrnné statistiky formou předdefinovaných dotazů,
- textovou variantu výše uvedených výstupů ve formátu CSV,
- použitou konfiguraci simulátoru.

1.3.1.5 F5 – Grafické zobrazení výstupů

Nad úspěšně dokončenými simulacemi může uživatel vykreslovat grafy závislostí, vizualizovat konfigurace a zátěže a všechny tyto grafické výstupy snadno exportovat v různých formátech. Zadavatel si výslovně žádá použití programu `gnuplot` na straně serveru.

1.3.2 Obecné požadavky

Tyto požadavky popisují obecné vlastnosti aplikace.

1.3.2.1 O1 – Přenositelnost

Protože simulátor DiskSim požaduje od operačního systému hojně rozšířené POSIX rozhraní, není žádoucí omezovat použitelnost jeho grafického rozhraní zbytečnými závislostmi. Proto by požadovanou nadstavbu mělo jít spustit pod operačními systémy Linux, MS Windows i např. VMS. Platformní nezávislost navíc umožní snadné zprovoznění i na případných jiných systémech. Důležitým aspektem aplikace přinášejícím zásadní užitek je její rychlá a bezproblémová instalace na co nejširší možnou škálu typů systémů a jejích programových vybavení.

1.3.2.2 O2 – Rozšiřitelnost

Aplikace by měla být snadno upravitelná o:

- další scénáře – simulace zotavení z výpadku disku a jeho následná rekonstrukce,
- případné nové podporované diskové konfigurace simulátoru DiskSim – RAID 1+0 a RAID 6,
- parametrizaci výkonu řadiče – write-through/write-back, velikost cache, výkon CPU,
- parametrizaci vlastností disků,
- možnost volby SSD disku.

Navíc by ji mělo jít v budoucnu snadno přizpůsobit i pro případné jiné podobné použití (např. pro nástroj IOzone).

1.3.2.3 O3 – Celistvost

Aplikace nechť je jednoduchá, snadno čitelná a lehce upravitelná. Musí být nezávislá na konkrétním simulátoru a být oddělená i od nástroje na výrobu grafických výstupů. Tím se navíc ještě podpoří ostatní obecné požadavky. Namísto použití nějaké databáze pro ukládání a vnitřní výměnu dat si aplikace zcela vystačí se souborovým systémem. Primárním použitím aplikace je totiž samostatné simulační prostředí pro každého uživatele na jeho vlastním počítači.

1.4 Analýza požadavků

Zde popisuji jak splním jednotlivé funkční požadavky.

1.4.1 F1 – Editace diskové konfigurace

Uživatel má možnost kopírovat, upravovat i mazat jednotlivé části konfigurace v rámci úloh buď přímo anebo použít zjednodušenou konfiguraci v rámci scénářů. Každá úloha obsahuje jednotlivé části konfigurace, její nezbytné vstupní soubory a po spuštění simulací v ní výslednou poskládanou úplnou konfiguraci, výstupní soubory a z nich zpracované statistiky. Zjednodušená konfigurace nabízí jen typ diskového pole, typ použitého disku, počet disků a velikost chunku.

parametr	JBOD	RAID 0	RAID 1	RAID 4	RAID 5
`Addressing mode`	`Parts`	`Array`	`Parts`	`Array`	`Array`
`Distribution scheme`	`Asis`	`Stripped`	`Asis`	`Stripped`	`Stripped`
`Redundancy scheme`	`Noredun`	`Noredun`	`Shadowed`	`Parity_disk`	`Parity_rotated`
`Number of copies`	`0`	`0`	= disků	`0`	`0`
`Parity stripe unit`	`1`	`1`	`1`	= chunk	= chunk

Tabulka 1.1: Vliv typu diskového pole na konfigurační blok `logorg`

1.4.2 F2 – Volba scénářů a parametrů simulace

Uživatel si scénářem zadává různé zjednodušené konfigurace, které chce mezi sebou porovnávat, k nim různé zjednodušené zátěže. Nebo si uživatel může zcela svobodně a naprosto bez omezení poskládat svoji vlastní konfiguraci v dané úloze. Zjednodušený scénář nabízí volbu zátěže s počtem požadavků, volbou náhodného nebo sekvenčního přístupu, pravděpodobností operací čtení nad zápisy a volbu variability velikosti požadavků.

parametr	hodnota
`Number of I/O requests to generate`	počet požadavků

Tabulka 1.2: Vliv parametrů zátěže na konfigurační blok `disksim_synthio`

parametr	hodnota
`Probability of read access`	pravděpodobnost operací čtení nad zápisy
`Probability of sequential access`	náhodný nebo sekvenční přístup
`Sizes`	variabilita velikosti požadavků

Tabulka 1.3: Vliv parametrů zátěže na konfigurační bloky `disksim_synthgen`

1.4.3 F3 – Spouštění běhu simulátoru

Uživatel spouští zadaný scénář nebo vlastní konfiguraci v úloze, přitom však může s aplikací dále pracovat. Při spuštění může volitelně nastavit jeden proměnný parametr konfigurace nebo zátěže v případě scénáře nebo zcela libovolný parametr v případě vlastní konfigurace. Úlohy spouští serverová část, její výsledky ukládá do samostatných nezávislých adresářů a rovnou je i zpracuje do strukturovaných statistik. Pokud uživatel zadal volitelný parametr, připraví se tolik simulací dané úlohy, kolik hodnot parametrů bylo požadováno tak, že se použije stále ta samá konfigurace a programu DiskSim se na konec argumentů příkazové řádky přidá pozměňující komponenta, parametr a konkrétní hodnota.

1.4.4 F4 – Vyhodnocení výstupu simulátoru

Z úspěšně dokončených simulací si uživatel nechává vypisovat její výstupy formou detailních statistik, které serverová část vyrobila bezprostředně po skončení simulací a které jsou tak rychle a snadno k dispozici klientovi. Výstupních statistik je mnoho a tak jsou seskupeny do logických celků, přičemž zřejmě nejpoužívanější bude `Overall I/O System`. Tím se uživateli zpřehledňuje práce s výstupy simulátoru.

1.4.5 F5 – Grafické zobrazení výstupů

Z úspěšně dokončených simulací si uživatel může i nechat vykreslit grafy závislostí nebo jen sloupcové porovnání v případě žádného volitelného parametru. Veškeré grafické výstupy zajišťuje program `gnuplot` ze zpracovaných výstupních statistik. Vykreslovat si lze zároveň jednu i více metrik z výstupních statistik.

Kapitola 2

Návrh řešení

2.1 Základní koncepce

Rámcový koncept řešení by měl vycházet z obecných požadavků. Grafická rozhraní lze navrhnout s následujícími principy, která jsou dále rozebrána právě s ohledem na ony požadavky.

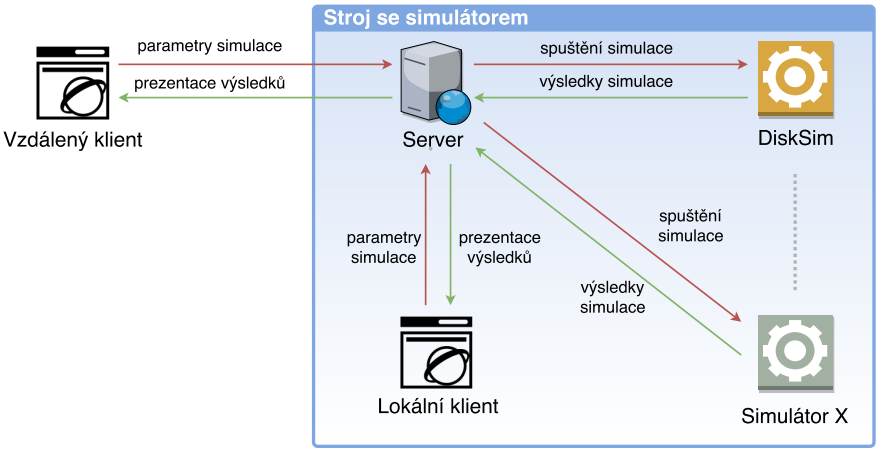
Variantou nejblíže k vlastnímu simulátoru je úprava a rozšíření jeho zdrojových kódů tak, aby kromě simulací zajišťoval i své grafické rozhraní. Takové řešení je na jednu stranu maximálně celistvé, neboť by se jednalo o jediný spustitelný soubor. Avšak na druhou stranu díky rozmanitosti a nekompatibilitě grafických knihoven už jen napříč systémy UNIX nutně komplikovaně až vůbec přenositelné, velmi pracně rozšiřitelné a prakticky bez možnosti začlenění podpory dalších simulátorů. Navíc se u programu DiskSim očekávají další změny a nové vlastnosti vyvíjené v samostatných projektech, v čehož důsledku se zakomponování GUI přímo do simulátoru jeví krajně nevhodné.

Další možností je takovéto rozhraní zkompilovat do samostatného spustitelného souboru, čím se i nadále velmi dobře udrží celistvost, ovšem i zde závislost na konkrétních grafických knihovnách drasticky snižuje přenositelnost i rozšiřitelnost.

Právě pro splnění přenositelnosti je tedy nezbytné použít model úplného oddělení simulátoru od grafického rozhraní tak, aby mohl být DiskSim simulátor dále nezávisle vyvíjen a grafické rozhraní mohlo být zároveň rozšiřováno o další simulátory a nástroje. Takto oddělené grafické rozhraní by také samo o sobě mělo být maximálně přenositelné, což vylučuje přímou závislost na konkrétních grafických knihovnách. V současné době není pro tento úkol nic výhodnějšího, než ponechat uživatelskou část grafického rozhraní na webovém prohlížeči, který nabízí solidní, uniformní, rozšířené a podporované řešení na prakticky libovolné platformě formou HTML jazyka, CSS kaskádových stylů a JavaScript skriptů.

Grafické rozhraní se tak rozpadá do dvou nezávislých, ale samozřejmě spolupracujících celků - server a klient. Serverová část hraje roli běžného webového

serveru, přijímá požadavky uživatele, spouští simulace a vrací její výsledky. Klientská část ve formě dynamických HTML stránek běží v libovolném moderním webovém prohlížeči. Takovéto rozdělení navíc usnadňuje další vývoj rozhraní i testování jeho jednotlivých komponent. Navíc lze poté klienta (webový prohlížeč) spouštět i z jiného stroje než kde běží serverová část spolu s vlastním simulátorem a lze tak realizovat například počítačovou učebnu s jediným silným strojem pro spouštění simulací a s žáky jej používající pouze přes svůj webový prohlížeč bez potřeby jakékoliv instalace dalšího software.



Obrázek 2.1: Základní koncepce

2.2 Serverová část

Serverová část hraje roli běžného webového serveru, která přijímá požadavky uživatele, spouští simulace a vrací její výsledky. Klientem mu může být jak prohlížeč na tom samém stroji, tak prohlížeče na síťově dostupných jiných strojích. Měl by tedy být schopen zároveň obsluhovat i více požadavků od jediného klienta, stejně jako více klientů.

Současnou práci se simulátorem více klienty najednou je nutné řešit oddělením jednotlivých simulací do samostatných úloh. Jedna úloha slouží pro sdílení s ostatními a nelze ji tedy smazat. V úlohách je nejprve nutno vytvořit, nakopírovat nebo nasdílet nezbytné části konfigurace simulátoru, což jsou jeho bloky (typicky typu `disksim_*` nebo `dm_*` a dále i `instantiate` a `topology`) spolu s dalšími vstupními soubory (tzv. `seek` a `defs` soubory). Vybrané bloky se poskládají do hlavní konfigurace, nad níž se již dá položit požadavek na spuštění simulace spolu s volitelným proměnlivým parametrem (typicky počtem procesů provádějící nad simulací diskové operace). Ta se neprovádí rovnou, nýbrž je zpracována nezávislou komponentou – tzv. vykonavatelem – který simulace spouští s ohledem na systémové zdroje a poté ukládá a formátuje výsledky simulace. Tyto výsledky pak klient může zpětně požadovat v různých formátech.

Webový server klientovi zasílá požadované HTML dokumenty a jeho příslušenství, jakým jsou obrázky, styly, skripty atp. Webový klient – prohlížeč – běžně posílá serveru požadavky na dynamický obsah v HTML formulářích. Ty používají HTTP metodu `POST` a jsou vhodné spíše pro jednoduché malé úkoly. Pro komplexní práci s nastavením simulátoru a zpracováním jeho výsledku se ovšem více hodí rozhraní REST, což je architektura navržená pro distribuované prostředí jakým je i toto grafické rozhraní. Sémantika jeho operací nad zdroji je konečná a tvoří ji pouze čtyři typy – vytvořit, získat, upravit, smazat (zkratka CRUD), reprezentované HTTP metodami `GET`, `POST`, `PUT` a `DELETE` nad daty serializovanými do formátu JSON. Toto REST rozhraní umožňuje rychlé začlenění případných dalších úprav, neboť jej lze takto velmi snadno strojově testovat.

Konfigurační bloky jsou deklarativně popsány standardem JSON Schema [12], který následně používá jak klient pro dynamické vytváření, změnu bloků a jejich ověřování, tak i serverová část pro další ověřování těchto bloků od uživatele a pro jejich reprezentaci do přirozeného DiskSim formátu.

2.3 Klientská část

Klientskou část tvoří jak statické HTML stránky, tak dynamicky vytvářený obsah skriptováním v jazyce JavaScript s rozhraním XMLHttpRequest, umožňujícím webovým aplikacím komunikaci mezi serverem a klientem prostřednictvím protokolu HTTP. Je tedy protikusem pro server s REST rozhraním.

Uživatel spouští serverovou část a tím může pokračovat v práci s aplikací z libovolného prohlížeče po zadání příslušné adresy. Uživateli se zobrazí nabídka spuštění scénáře se zjednodušenou konfigurací a zátěží, ale může se podívat i do seznamu úloh, nad nimiž provádí základní úkony. Každá úloha obsahuje vlastní sadu vstupních částí, což jsou konfigurační bloky programu DiskSim a jeho další související nutné vstupní soubory. Těmi jsou *seek* soubory obsahující naměřené průměrné přístupové doby na různá místa konkrétního fyzického disku a *defs* soubory s distribučními rozloženími pro různé statistické výstupy. Jedna úloha je sdílená, nelze ji odstranit a seskupuje nejčastěji používané části tak, aby se zbytečně neduplikovaly.

S každou částí úlohy uživatel provádí běžné operace jejich prohlížení, úprav nebo stažení jako samostatného souboru. Bloky se upravují v pokročilém editoru, který kontroluje syntaxi upravovaného bloku a poskytuje nápovědu a vysvětlení každého konfiguračního parametru.

Z předem připravených vzorových úloh nebo nově vytvořených uživatel zadává instrukce ke spuštění simulace vybráním hlavního konfiguračního bloku, který obsahuje pouze odkazy na další bloky formou direktivy `source`. Až v této chvíli se požadované bloky rekurzivně nahrávají z aktuální úlohy nebo ze sdílené, pokud v té inkriminované nebyly nalezeny. Všechny případně chybějící bloky uživatel vidí v chybovém hlášení a nemůže v simulaci úlohy pokračovat, dokud nebudou všechny bloky a vstupní soubory dané konfigurace konkrétní úlohy nalezeny. V tomto bodě si uživatel ovšem může vybrat i spuštění více simulací nad stejnou konfigurací s tím, že každé pozmění jeden parametr výpisem nebo rozsahem jejich jiných hodnot.

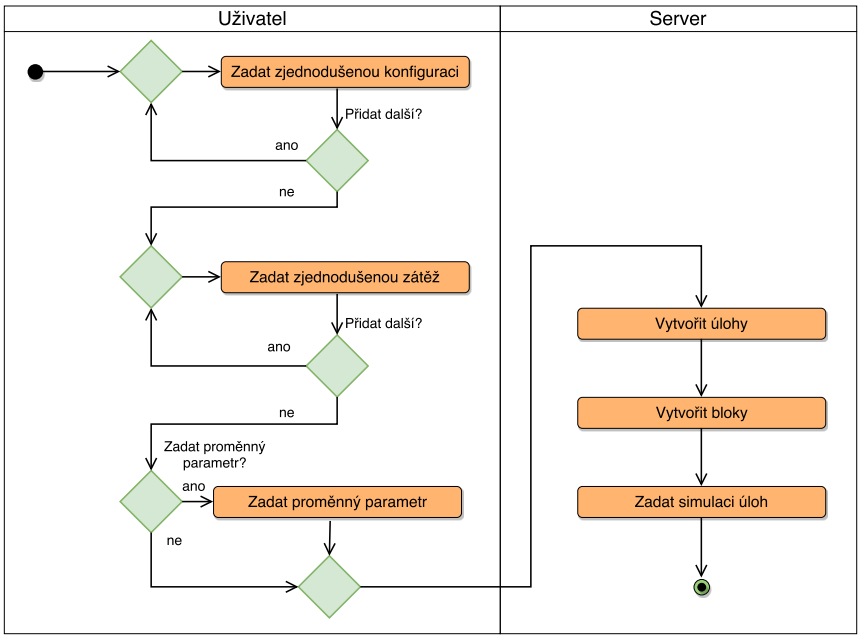
Takto se spustí jediná nebo více simulací dané úlohy na pozadí a její výstupy jsou uloženy pro každou úlohu a každou variantu konfigurace zvlášť. Uživatel vidí stav zadaných simulací u každé úlohy jak čekají na provedení, kolik jich zrovna probíhá a s jakým výsledkem jich kolik skončilo. Nad dokončenými simulacemi si uživatel prohlíží výstupní metriky seskupené pro přehlednost do logicky souvisejících skupin. Pokud zadal více variant simulace, má možnost si vybrat libovolný počet metrik, které si zobrazí do grafu, kde osou X je zadaný pozměňovací parametr. Tento graf si může stáhnout jako rastrový obrázek, vektorový obrázek nebo jako zdrojová data v CSV formátu pro další zpracování.

Nabídka spuštění scénáře uživatele vyzývá v jedné nebo více zjednodušených konfigurací, jedné nebo více zjednodušené zátěže a volitelně nastavením jednoho proměnného parametru. Výsledky spuštění takového scénáře si uživatel zobrazuje v grafech porovnání více konfigurací zároveň.

2.4 Popis aktivit

2.4.1 Aktivity scénáře

Uživatel zadá požadavek vytvořit nový scénář.



Obrázek 2.2: Aktivity scénáře

2.4.1.1 Zadat zjednodušenou konfiguraci

Uživatel musí ve scénáři zadat alespoň jednu zjednodušenou konfiguraci, ve které si vybírá typ diskového pole, typ disku, počet disků a velikost chunku. Jednotlivé konfigurace jsou navzájem nezávislé a každá vytvoří samostatnou úlohu.

2.4.1.2 Zadat zjednodušenou zátěž

Uživatel musí ve scénáři zadat alespoň jednu zjednodušenou zátěž, ve které určuje počet požadavků, pravděpodobnost operací čtení versus zápis, náhodný nebo sekvenční přístup a variabilitu velikosti požadavků.

2.4.1.3 Zadat proměnný parametr

Uživatel si může vybrat jeden následujících parametrů s více hodnotami určenými rozsahem a jejich odstupem:

- počet zátěží,
- velikost chunku,
- počet požadavků,
- pravděpodobnost operací čtení versus zápis.

2.4.1.4 Vytvořit úlohy

Podle zjednodušených konfigurací scénáře klient zadá serveru vytvořit nové úlohy. Všechny budou mít v názvu unikátní prefix a tím se logicky spojí v jeden scénář.

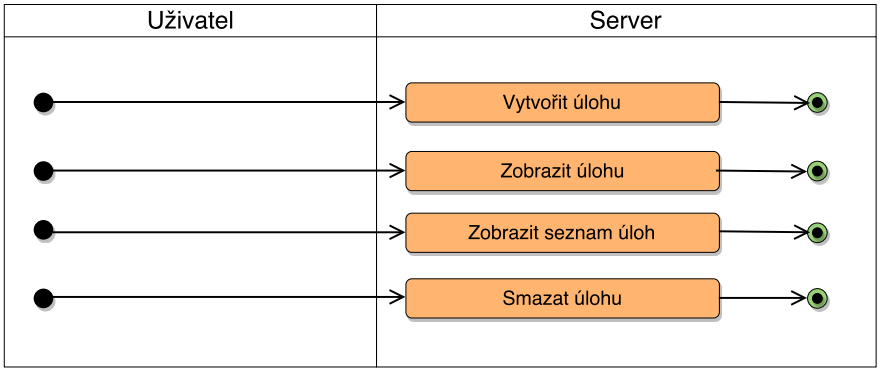
2.4.1.5 Vytvořit bloky

Do těchto nových úloh klient zadá serveru vytvořit bloky popisující zjednodušenou konfiguraci a zjednodušené zátěže scénáře s maximálním využitím společných obecných bloků ze sdílené úlohy.

2.4.1.6 Zadat simulaci úloh

Tyto nové úlohy klient zadá serveru ke spuštění, volitelně s proměnnými parametry scénáře.

2.4.2 Aktivity úloh



Obrázek 2.3: Aktivity úloh

2.4.2.1 Vytvořit úlohu

Uživatel zadá požadavek vytvořit novou úlohu. Nelze přepsat již existující. Lze použít jinou jako vzor bloků a vstupních souborů, jinak bude prázdná.

2.4.2.2 Zobrazit úlohu

Uživatel chce vědět detaily o již existující úloze – kolik a jakých je v ní bloků, seek souborů, defs souborů a zadaných simulací.

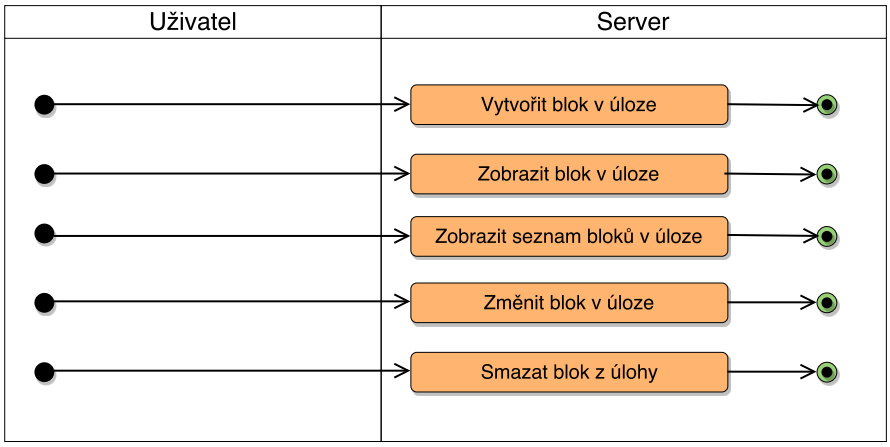
2.4.2.3 Zobrazit seznam úloh

Uživatele zajímá seznam všech úloh s vyznačením té, která je sdílená.

2.4.2.4 Smazat úlohu

Uživatel chce smazat úlohu, tedy včetně všech jejich bloků, vstupních souborů, souborů simulace i jejich výsledků. Neexistující ani sdílená úloha smazat nelze.

2.4.3 Aktivity bloků



Obrázek 2.4: Aktivity bloků

2.4.3.1 Vytvořit blok v úloze

Uživatel chce nahrát nový konfigurační blok do úlohy. S požadovaným jménem nesmí v úloze ještě existovat. K pozdější simulaci je takových bloků potřeba několik a některé jsou i povinné, to se ale v této akci nekontroluje. Stejně tak se zatím nekontrolují zanořené odkazy na jiné bloky formou DiskSim direktivy `source`. Naopak již nyní se ověřuje syntaxe nahrávaného bloku.

2.4.3.2 Zobrazit blok v úloze

Uživatel si chce prohlédnout blok úlohy. Ten musí existovat v dané úloze nebo v úloze sdílené. Blok lze obdržet jak v obvyklém JSON formátu, tak i v originálním DiskSim formátu. V obojím lze nastavit požadovanou úroveň odsazení jednotlivých řádek. Lze též požádat o nahrazení odkazovaných bloků jejich obsahem a to i rekurzivně.

2.4.3.3 Zobrazit seznam bloků v úloze

Uživatel dostane seznam bloků v dané úloze.

2.4.3.4 Změnit blok v úloze

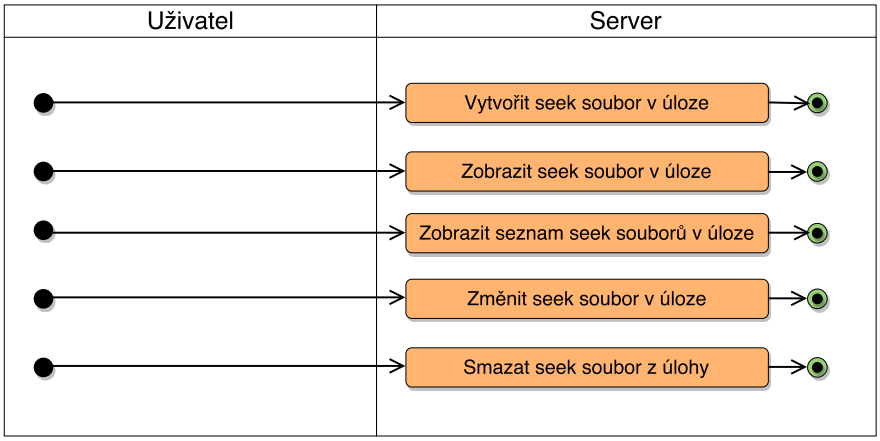
Uživatel chce do úloh nahrát nový konfigurační blok nebo přepsat již existující. Zatím se nekontrolují zanořené odkazy na jiné bloky formou DiskSim direktivy `source`. Naopak již nyní se ověřuje syntaxe nahrávaného bloku.

2.4.3.5 Smazat blok z úlohy

Uživatel chce smazat blok z úlohy. Musí v ní existovat. Nijak se nekontroluje, není-li blok odkazován jiným blokem.

2.4.4 Aktivity seek souborů

Seek soubor je povinným vstupním souborem v bloku `dm_mech_g1` popisujícím vybavovací časy pro různé bloky na skutečném disku. Má vlastní formát, odlišný od bloků, a jeho obsah nemůže být součástí konfiguračního souboru.



Obrázek 2.5: Aktivity seek souborů

2.4.4.1 Vytvořit seek soubor v úloze

Uživatel chce nahrát nový seek soubor do úlohy. S požadovaným jménem nesmí v úloze ještě existovat. Ověřuje se jeho syntaxe.

2.4.4.2 Zobrazit seek soubor v úloze

Uživatel získá požadovaný seek soubor, pokud existuje v dané nebo sdílené úloze, tak jak je.

2.4.4.3 Zobrazit seznam seek souborů v úloze

Uživatel dostane seznam seek souborů v dané úloze.

2.4.4.4 Změnit seek soubor v úloze

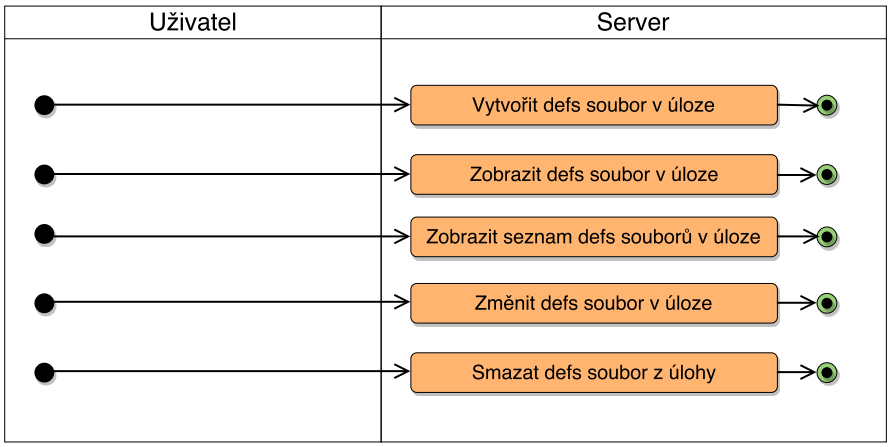
Uživatel nahrávaný seek soubor do úlohy přidává jako nový nebo přepisuje již existující. Ověřuje se jeho syntaxe.

2.4.4.5 Smazat seek soubor z úlohy

Uživatel chce smazat seek soubor z úlohy. Musí v ní existovat. Nijak se nekontroluje, není-li seek soubor odkazován nějakým blokem.

2.4.5 Aktivity defs souborů

Defs soubor je povinným vstupním souborem v bloku `disksim_global` popisujícím statistické rozložení pro agregaci naměřených veličin. Má vlastní formát, odlišný od bloků, a jeho obsah nemůže být součástí konfiguračního souboru.



Obrázek 2.6: Aktivity defs souborů

2.4.5.1 Vytvořit defs soubor v úloze

Uživatel chce nahrát nový defs soubor do úlohy. S požadovaným jménem nesmí v úloze ještě existovat.

2.4.5.2 Zobrazit defs soubor v úloze

Uživatel získá požadovaný defs soubor, pokud existuje v dané nebo sdílené úloze, tak jak je.

2.4.5.3 Zobrazit seznam defs souborů v úloze

Uživatel dostane seznam defs souborů v dané úloze.

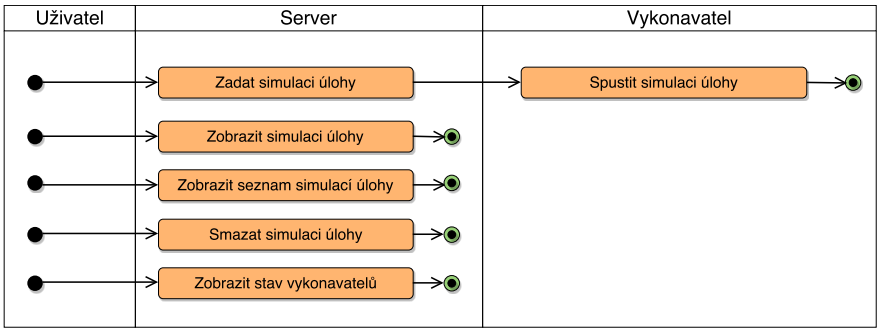
2.4.5.4 Změnit defs soubor v úloze

Uživatel nahrávaný defs soubor do úlohy přidává jako nový nebo přepisuje již existující.

2.4.5.5 Smazat defs soubor z úlohy

Uživatel chce smazat defs soubor z úlohy. Musí v ní existovat. Nijak se nekontroluje, není-li defs soubor odkazován nějakým blokem.

2.4.6 Aktivity simulací



Obrázek 2.7: Aktivity simulací

2.4.6.1 Zadat simulaci úlohy

Uživatel chce spustit simulaci nad připravenými bloky pospojovanými speciálním blokem `main` a souvisejícími vstupními soubory. Každý odkazovaný i zanořený blok, stejně jako vstupní soubory, musí existovat buď v dané nebo ve sdílené úloze. Takto vytvořený konfigurační soubor v DiskSim formátu i se

všemi potřebnými vstupními soubory se vykopíruje bokem a tak případné následné změny v původních souborech neovlivní zadanou simulaci.

Uživatel může změnit libovolný jeden konfigurační parametr jinou hodnotou, výčtem nebo rozsahem hodnot. Podle toho se vytvoří jeden či více samostatných adresářů ve kterých bude spuštěn vlastní simulátor. Tím sice všechny tyto varianty jedné simulace sdílí jednu sadu konfiguračního a vstupních souborů, ovšem přesto jsou spuštěny s rozdílným jedním pozměňovacím parametrem. Každá varianta vytváří výstupní soubory do vlastních adresářů, takže se nijak navzájem neovlivňují.

Všechny případné předešlé simulace se před žádostí o novou smažou.

2.4.6.2 Spustit simulaci úlohy

Připravené simulace spouští paralelně běžící vlákna – vykonavatelé, jejichž počet je konstantní, ale konfigurovatelný. Každou simulaci připravenou ke spuštění spustí, počkají na její dokončení a podle jejího úspěchu výstupní log soubor rozeberou, zanalyzují a připraví z něj strukturovaný soubor s naměřenými metrikami pro jejich následnou snadnou, rychlou a pravděpodobně i opakovanou prezentaci.

2.4.6.3 Zobrazit simulaci úlohy

Uživatel chce vidět informace o zadané jedné simulaci dané úlohy. Ty jsou mu poskytnuty.

2.4.6.4 Zobrazit seznam simulací úlohy

Uživatel chce vidět informace o zadaných simulacích dané úlohy. Ty jsou mu poskytnuty.

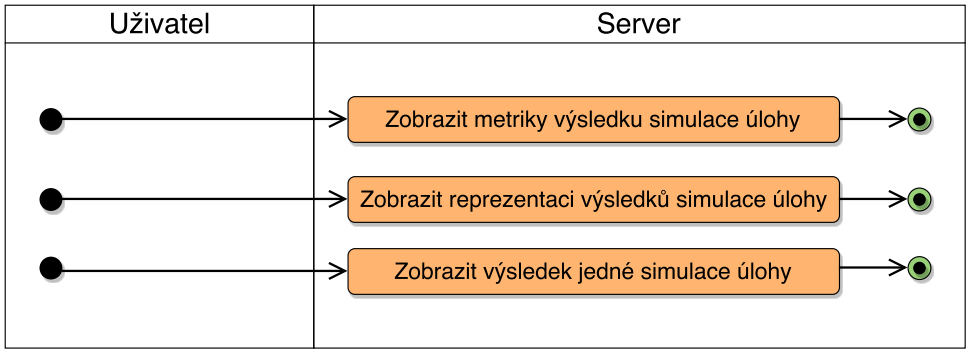
2.4.6.5 Smazat simulaci úlohy

Uživatel chce smazat výsledky dané úlohy. To se provede, pokud úloha existuje a pokud nad ní neběží nějaká simulace.

2.4.6.6 Zobrazit stav vykonavatelů

Uživatel se dozví počet vykonavatelů a kolik jich zrovna spouští simulace.

2.4.7 Aktivity výsledků



Obrázek 2.8: Aktivity výsledků

2.4.7.1 Zobrazit metriky výsledku simulace úlohy

Uživatel poptává seznam dostupných metrik proběhlé simulacemi dané úlohy. Ten je mu poskytnut.

2.4.7.2 Zobrazit reprezentaci výsledků simulace úlohy

Nad proběhnutými simulacemi dané úlohy může uživatel pokládat dotazy na hodnoty jednotlivých nebo i více metrik. Pokud byl zadán výčet nebo rozsah pozměňovacího parametru, lze získat i tabulku závislostí jedné nebo více metrik na onom parametru i ve formátech CSV, GIF, PNG nebo SVG. Stejně tak lze stáhnout inkriminovaný konfigurační soubor i jeho související vstupní soubory. Výsledky scénáře si uživatel může prohlédnout jako porovnání více zjednodušených konfigurací pohromadě v jednom grafu, pro každou metriku jeden graf a pokud nebyl zvolen proměnný parametr, tak ve formě sloupcového grafu.

2.4.7.3 Zobrazit výsledek jedné simulace úlohy

Uživatel poptává výsledky proběhlé simulacemi dané úlohy. Ty jsou mu poskytnuty.

Kapitola 3

Popis implementace

Pro realizaci navrženého řešení bylo třeba udělat několik klíčových rozhodnutí:

- zvolit prostředí webového serveru,
- zvolit skriptovací jazyk na straně serveru,
- zvolit aplikační prostředí serveru,
- zvolit vizualizační nástroj,
- zvolit aplikační prostředí klienta.

3.1 Serverová část

Z důvodu bezproblémové přenositelnosti a snadné rozšiřitelnosti je vhodné vybrat skriptovací řešení před kompilovaným. Nejrozšířenější skriptovací aplikační rámce webového serveru jsou psány v jazycích `PHP`, `Ruby`, `Perl` nebo `Python`. Ovšem pouze poslední dva bývají součástí standardní distribuce většiny UNIX systémů a tak u nich není potřeba žádná další instalace závisejícího software. `Python` je navíc čistě objektově orientovaný a přitom programy v něm napsané mají čistý vzhled, jsou snadno čitelné a jasně srozumitelné. Je tak lehký i pro začátečníky a vychází vstříc případným dalším úpravám a rozšířením díla. To vše dává hlas pro volbu jazyka `Python` před ostatními.

Z nabídky snad desítek aplikačních rámců webového serveru psaných v jazyce `Python` se velmi výhodně jeví rámec `Bottle` [13], jehož jediný skriptovací soubor dobře vyhovuje požadavku celistvosti a díky podpoře `Python` verze od 2.6 až do současné 3.6 je i snadno přenositelný. Navíc je rozšiřitelný o zásuvné moduly obohacující jeho funkčnost a jeho vytvářený obsah je také snadno šablonovatelný.

Rámec `Bottle` je ve skutečnosti spojovníkem (tzv. middleware) mezi WSGI serverem a požadovanou aplikací - serverovou částí grafického rozhraní. WSGI

je standardizované, výkonné a hojně rozšířené rozhraní, díky čemuž `Bottle` umí spolupracovat s asi 15 různými WSGI servery, což opět dále zvětšuje jeho přenositelnost. Velmi stabilním a vícevláknovým WSGI serverem se jeví `waitress` [14] se záložním `wsgiref` [15] pro `Python` verze starší než 2.7. Tato na první pohled přílišná dekompozice se ovšem mnohem lépe vyvíjí, udržuje i rozšiřuje, neboť požadovaná aplikace se může plně věnovat pouze svému hlavnímu úkolu a přenechat starost o webovou část mnoha uživateli prověřených, důkladně otestovaných, průběžně vyvíjených WSGI komponent.

Obrázek 3.1: Serverová část

3.1.1 WSGI server

WSGI server se spouští z hlavního adresáře projektu příkazem `./disksimui.sh`. Ten najde nejnovější nainstalovaný interpretr `Python`, nastaví cestu na projekt a spustí `disksimui/__main__.py`. Konfiguruje se souborem `./disksimui.conf`. Je k němu dostupná tato nápověda:

```
usage: ./disksimui.sh [-h] [-v] [--bind ADDRESS]
                      [--reload SECONDS]
                      [--verbose] [--debug]

DiskSim web user interface

optional arguments:
  -h, --help          show this help message and exit
  -v, --version       show version and exit
  --bind ADDRESS      bind socket to ADDRESS,
                      by default localhost:8080
  --reload SECONDS    reload on file changes each SECONDS,
                      by default 0
  --verbose           print verbose output to stdout
                      and stderr
  --debug             start server in debug mode
```

Příklad jeho výstupu na obrazovku:

```
Python version 3.5.2+ (default, Sep 22 2016, 12:18:14)
[GCC 6.2.0 20160927]
Main PID: 11613
DiskSim Worker 0: started
DiskSim Worker 1: started
DiskSim Worker 2: started
DiskSim Worker 3: started
DiskSim Worker 4: started
DiskSim Worker 5: started
DiskSim Worker 6: started
DiskSim Worker 7: started

Bottle v0.12.13 server starting up (using WaitressServer())...
Listening on http://localhost:8080/
Hit Ctrl-C to quit.

Serving on http://localhost:8080
```

Jak vidno, server nyní běží včetně 8 vláken vykonavatelů a je dostupný na uvedené webové adrese.

3.1.2 Python prostředí

Použité Python knihovny i aplikace fungují pod Python od verze 2.6 a vyšší.

Při spuštění se jednou zkompilované Python skripty ukládají do souborů `*.pyc` u verze 2 nebo do adresáře `__pycache__/` u verze 3. To zrychlí následné další spuštění Python aplikace, nikoli ovšem její chod samotný. Tyto soubory lze kdykoliv smazat - vytvoří se při dalším spuštění znovu. Tyto soubory není ale nutné ani vhodné ukládat do repositářů.

Adresář `lib/` obsahuje projektem používané Python knihovny, které však nejsou nebo nebývají součástí distribucí operačních systémů a bylo by je tak tedy navíc třeba samostatně instalovat. Jde o následující moduly:

- `waitress` ... WSGI server [14],
- `bottle` ... WSGI middleware [13],
- `six` ... knihovna pro zastření nekompatibilit mezi Python verzemi [16],
- `jsonschema` ... ověřování datových struktur podle zadaných schémat [17],
- `lockfile` ... atomické zamykání souborů [18].

3.1.3 WSGI middleware - Bottle

Adresář `disksimui/` obsahuje serverový aplikační rámec Bottle.
Soubor `disksimui/__main__.py` zpracovává parametry spustitelného souboru `disksimui.sh` a konfigurační soubor `disksimui.conf`, spouští WSGI server `waitress` nebo `wsgiref`, pod ním upravenou aplikaci JSONBottle přístupnou na URL adrese `/`.

Přehled dalších podadresářů:

- `tasks` ... adresář s úlohami a jejími simulacemi,
- `locks` ... adresář pro zamykaní úloh,
- `static` ... statické HTML stránky, styly a skripty,
- `util` ... pomocné skripty.

Složka `static/` obsahuje statické HTML části, čili i kaskádové styly, JavaScript skripty, fonty a ikony. Jejich popis následuje později v kapitole implementace klienta.

Adresář `util/` obsahuje pomocné a testovací nástroje. Nejsou pro chod aplikace vůbec potřeba, ale velmi usnadňují vývoj aplikace. Způsob jejich použití je popsán v jejich hlavičkách.

3.1.4 Aplikace - serverová část GUI

Základní přehled podadresářů DiskSim serverové části `disksimui/`:

- `lib` ... funkce a třídy pro práci se simulátorem a se schématy,
- `api` ... REST rozhraní,
- `worker` ... funkce a třídy pro práci s vykonavateli.

V adresáři `disksimui/lib` jsou uloženy tyto knihovny:

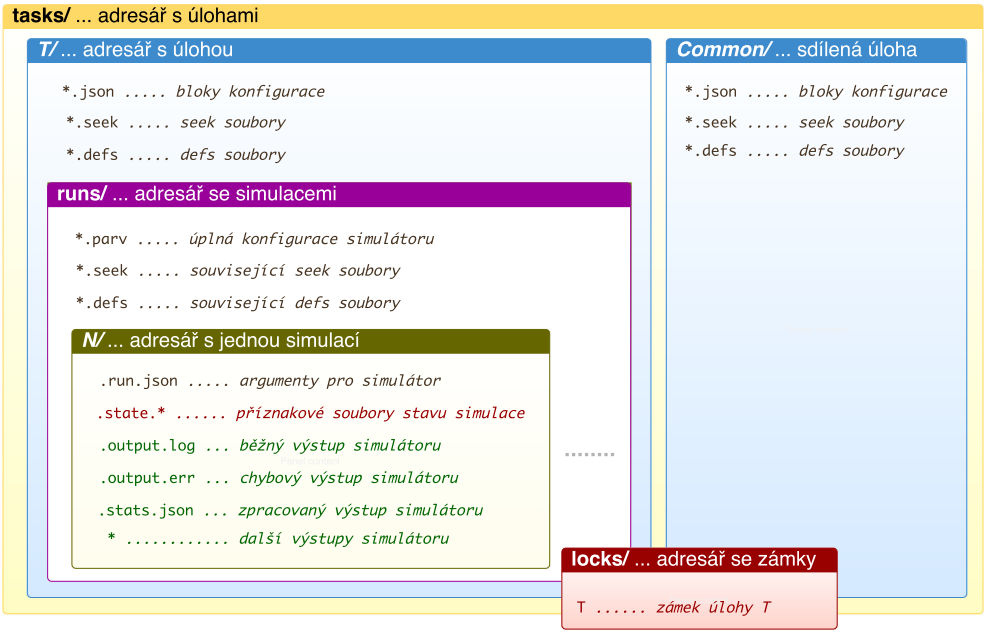
- `csv.py` ... funkce pro práci s CSV soubory,
- `data.py` ... funkce pro manipulaci s datovými strukturami,
- `files.py` ... funkce pro práci se soubory a zámky,
- `gnuplot.py` ... třída pro ovládání programu `gnuplot`,
- `jsonbottle.py` ... upravená aplikace Bottle pro práci s JSON objekty,
- `schema.py` ... třída pro práci s validačními schématy,
- `disksim.py` ... třída pro převod JSON konfigurace do nativní pro DiskSim.

Popis bloků použitých pro validaci na straně klienta i serveru je obsahem adresáře `disksimui/lib/schema/` ve formě JSON Schemat pro lepší přehlednost a udržovatelnost uložených ve formátu YAML. Navíc schémata obsahují i další údaje použité JSON Editorem na straně klienta. Aplikace podporuje všechny typu bloků a některé další pro lepší vnitřní interpretaci: `disksim_bus_stats`, `disksim_bus`, `disksim_cachedev`, `disksim_cachemem`, `disksim_ctlr_stats`, `disksim_ctlr`, `disksim_device_stats`, `disksim_disk`, `disksim_global`, `disksim_iodriver_stats`, `disksim_iodriver`, `disksim_iomap`, `disksim_ioqueue`, `disksim_iosim`, `disksim_logorg`, `disksim_pf_stats`, `disksim_pf`, `disksim_simpledisk`, `disksim_stats`, `disksim_syncset`, `disksim_synthgen`, `disksim_synthio`, `dm_disk`, `dm_layout_g1`, `dm_layout_g1_zone`, `dm_layout_g2`, `dm_layout_g2_zone`, `dm_layout_g4`, `dm_mech_g1`, `instantiate`, `main`, `memsmodel_mems`, `source`, `ssdmodel_ssd`, `topology_disksim_bus_ctlr`, `topology_disksim_bus_disk`, `topology_disksim_bus_iodriver`, `topology_disksim_ctlr_2`, `topology_disksim_ctlr`, `topology_disksim_disk`, `topology_disksim_iodriver`, `topology_memsmodel_mems`, `topology_ssdmodel_ssd`, `topology`.

REST rozhraní v adresáři `disksimui/api/` u sebe drží i YAML soubory popisující toto rozhraní, které jsou použity nástrojem Swagger UI [19] pro interaktivní dokumentaci a testování, stejně jako pro případnou dokumentaci tištěnou.

Adresář s úlohami `tasks/` je úložištěm všech úloh, tedy i té sdílené. Co úloha, to stejnojmenný adresář v něm. Úloha obsahuje konfigurační bloky a vstupní soubory. Pokud se na nějaký takovýto blok nebo soubor odkazuje a neexistuje v dané úloze, zkusí se najít v úloze sdílené. Zadání simulace dané úlohy znamená vytvoření podadresáře `runs/`, do kterého se vytvoří celá konfigurace a nakopírují všechny závislé vstupní soubory, aby případná další manipulace s nimi již neovlivnila zadanou simulaci. K tomu se navíc vytvoří pro každou zvolenou variantu simulace, minimálně jednu, podadresáře číslované od nuly, které obsahují argumenty příkazového řádku simulátoru, veškeré jeho výstupy a příznakové soubory stavu simulace pro synchronizaci jednotlivých komponent díla.

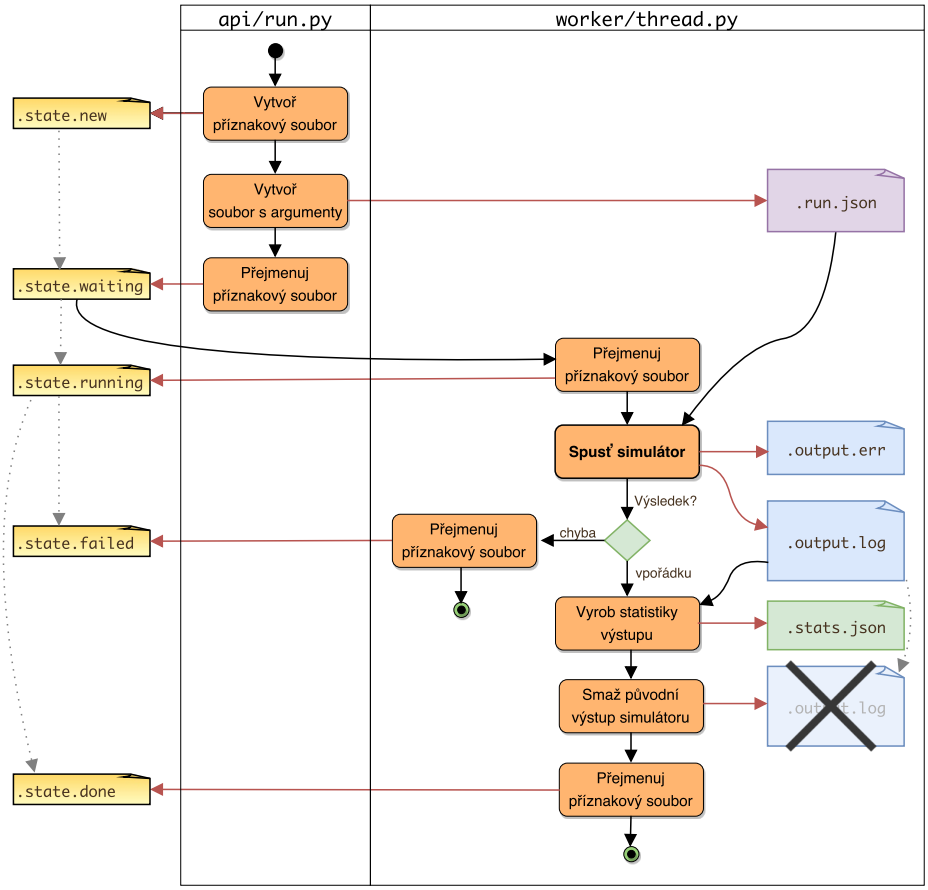
Veškeré pozměňovací operace nad úlohou jsou proti neočekávaným důsledkům souběžných operací chráněny atomickými zámky realizovanými soubory či adresáři ve vyhrazeném adresáři `locks/`. Ten je tedy kritický pro zachování konzistence úloh i když je drtivou většinu času prázdný.



Obrázek 3.2: Adresáře a soubory úloh

Spolu s webovým serverem se startují vlákna vykonavatelů umístěných v adresáři `disksimui/worker/` v počtu zadaném konfigurací, typicky podle počtu jader CPU. Ty v pravidelných intervalech prohledávají úkoly na nově zadané simulace a ty potom i pokud možno paralelně spouští. Synchronizace jejich chodu mezi sebou i dalšími komponentami serverové části probíhá skrze příznakové soubory.

Výstupy programu DiskSim mohou být podle zvolené konfigurace dosti velké - běžně jedna varianta vytvoří řádově 1MiB soubor, což při obvyklých řádově 100 variantách znamená 100MiB zabraného místa na disku serverové části jen pro jednu skupinu simulací jedné úlohy. Proto dané vlákno ihned po úspěšném skončení simulace výsledný objemný nestrukturalizovaný výstupní soubor daným vláknem zpracuje a hodnoty systematicky uloží do výstupního souboru s metrikami tak, aby byly rychle a spolehlivě k dispozici klientům v očekávaných dotazech na výsledky. Původní výstupní se log se pak smaže a uvolní tím značné místo na disku.



Obrázek 3.3: Diagram zadání a spuštění simulace

3.2 Klientská část

Strana klienta je realizovaná dynamickou HTML stránkou, kterou a jejíž data servíruje serverová část. Pro volbu jednotlivých komponent jsou rozhodující obecné požadavky, zejména přenositelnost. Proto se nevyplatí psát JavaScript

kód ani kaskádové styly přímo, ale je velmi vhodné použít nějaký hojně rozšířený aplikační rámec. Ten již sám o sobě řeší detekci a podporu nejrůznějších prohlížečů v jejich široké paletě verzí a hojná komunita vývojářů jej udržuje funkční, spolehlivý, bezpečný a neustále podporující nejnovější prohlížeče.

Obrazovka uživatele je logicky rozdělena do panelů, jejichž popis rozebírám níže. Většina tlačítek a některé další prvky uživateli poskytnu krátkou nápovědu k čemu jsou, pokud nad ně najede kurzorem myši a počká méně než sekundu. Po odjetí kurzorem mimo daný prvek nápověda opět rychle zmizí.

Jednotlivé prvky obrazovky se snaží ergonomicky přizpůsobovat šířce obrazovky tak, aby aplikace byla přehledná a ovladatelná jak na velkých monitorech tak třeba i na malých displayích tabletů.

3.2.1 Aplikační rámce a použité knihovny

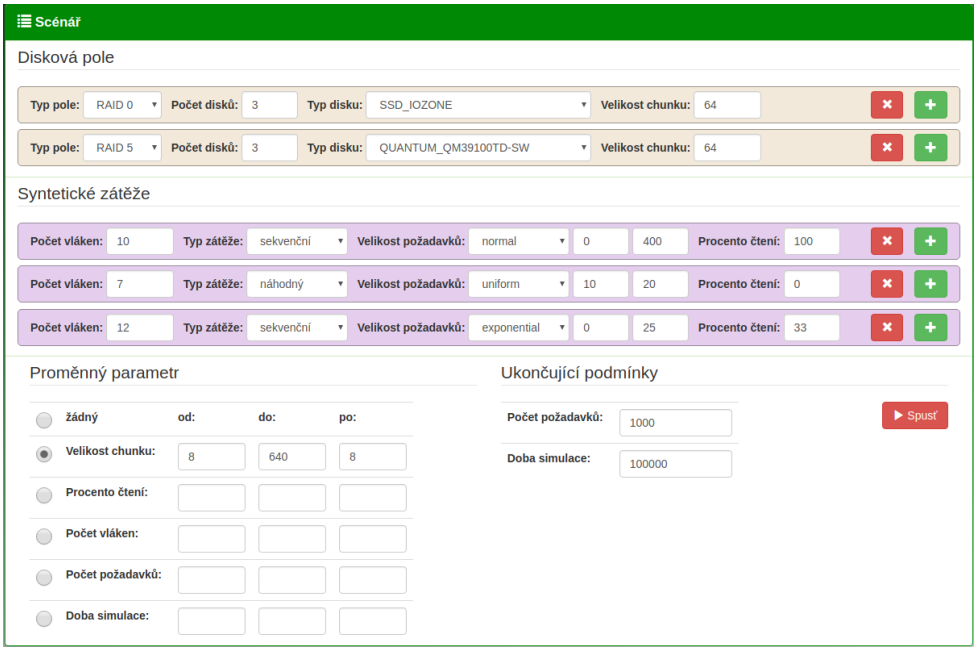
Vzhled moderních HTML stránek je určován kaskádovými styly CSS a pro konzistentní a čistý výsledek jsem vybral zřejmě nejznámější sadu připravených stylů - Bootstrap [20]. Jeho rozšířením TreeView [21] dosahuji opravdu příjemného a přehledného zobrazování nespočet metrik výsledků simulace.

Pro práci s DOM strukturou HTML dokumentu bohatě postačí rámec jQuery [22], který navíc umí i pracovat s CSS styly, zvládá základní animace a efekty, poskytuje užitečné nástroje jako např. `each()` a hlavně podporuje potřebný AJAX pro komunikaci s REST rozhraním serverové části.

Díky modulárnosti rámce jQuery dále vylepšuji uživatelskou zkušenost použitím zásuvného modulu Select2 [23] zpříjemňujícím chování HTML elementu `SELECT`. Pro snadné třídění HTML elementů používám malou knihovnu TinySort [24]. Pro úpravu bloků jsem vybral nástroj JSON Editor [25], který umožňuje dynamické vytváření HTML formulářů podle zadaných JSON schémat. Stránkování mezi jednotlivými výsledky simulací zajišťuje jQuery plugin simplePagination [26].

Všechny tyto styly a skripty jsou sice veřejně dostupné, ovšem pro maximální celistvost, nezávislost aplikace a její chod i v uzavřených sítích jsou všechny jejich komponenty součástí aplikace v adresáři `static/`.

3.2.2 Panel scénáře



Obrázek 3.4: Panel scénáře

Panel scénáře slouží pro zadání simulace se zjednodušenou konfigurací diskového pole, zjednodušenou syntetickou zátěží, zjednodušeným proměnným parametrem a ukončujícími podmínkami. Je hlavním panelem, který se automaticky a zatím jako jediný uživateli zobrazí po prvním načtení stránky aplikace. Kliknutí na horní pruh panelu jeho schová nebo opět ukáže. Na každou konfiguraci pole se spustí všechny zátěže a pokud je zadán proměnný parametr, tak i ve všech jeho hodnotách. Výsledky těchto simulací se poté dají zobrazit společně v porovnání mezi jednotlivými konfiguracemi.

Počet konfigurací polí je libovolný, uživatel je může přidávat stisknutím zeleného tlačítka nebo odebírat stisknutím červeného tlačítka, ale nejméně jedna musí zůstat. Uživatel volí typ pole, počet disků, typ disku, a velikost chunku každé konfigurace pole.

Počet zátěží je libovolný, uživatel je může přidávat stisknutím zeleného tlačítka nebo odebírat stisknutím červeného tlačítka, ale nejméně jedna musí zůstat. Uživatel volí počet vláken, typ zátěže, velikost požadavků a procento operací čtení vůči zápisu.

Pro požadované simulace lze vybrat jeden proměnný z předvybraných parametrů - velikost chunku, procento čtení, počet vláken, počtu požadavků nebo dobu simulace. Pokud je počet vláken menší než součet vláken nakonfigurovaných zátěží, použijí se jen ty odshora uvedené, dokud se nenaplní počet zadaný volitelným parametrem. Pokud je počet vláken zátěže větší než součet

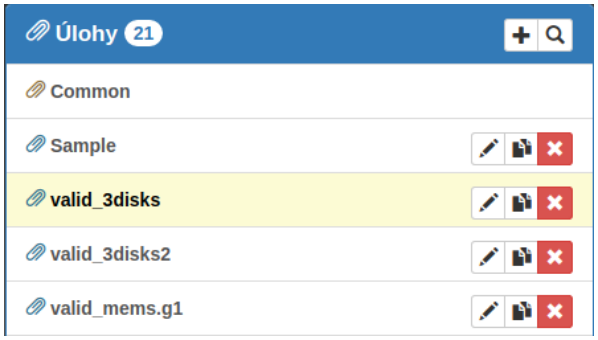
vláken nakonfigurovaných zátěží, použijí se všechny uvedené a poslední pro každou další požadovanou.

Simulace končí dosažením první ze dvou možných ukončujících podmínek - provedením zadaného počtu diskových operací nebo celkovou dobou simulace. Obojí může uživatel nastavit.

Jakmile uživatel klikne na tlačítko Spusť, klient vytvoří všechny potřebné úlohy a bloky v nich a spustí nad nimi simulace. Jakmile všechny v pořádku skončí, panel scénáře se překreslí na výsledky scénáře tak, že v liště vpravo nahoře přibude červené tlačítko pro smazání všech úloh i simulací scénáře a opět se uživateli vykreslí prvky pro zadání scénáře nového.

V panelu scénáře potom uživatel dostane k nabídce pár vybraných metrik z vyrobených statistik, jež se potom vykreslí jako samostatný graf pro každou metriku s tím, že každý graf obsahuje vždy výsledky každé konfigurace scénáře. Tyto grafu jsou sloupcové pokud nebyl zadán žádný proměnný parametr jinak je osou X tento proměnný parametr.

3.2.3 Panel úloh



Obrázek 3.5: Panel úloh

Panel úloh slouží pro práci s úlohami. Horní modrý pruh obsahuje celkový počet úloh, tlačítko pro přidání nové úlohy a tlačítko pro filtrování zobrazených úloh. Kliknutí na horní pruh panelu jeho položky schová nebo opět ukáže.

Každá úloha je zobrazena v seznamu pod horním pruhem, spolu s tlačítky pro přejmenování, kopírování a smazání úlohy. Kliknutím uživatel vybírá danou úlohu, čímž se mu otevře panel částí dané úlohy spolu s panelem simulací a daná úloha se žlutě podbarví aby bylo zřejmé, že právě s ní se zrovna pracuje. Sdílená úloha má jinak zabarvenou ikonu před svým názvem a nelze ji přejmenovat, kopírovat ani smazat.

Počet i seznam úloh se průběžně synchronizuje se serverem a tak se změny v nich jiným oknem prohlížeče daného uživatele nebo zcela jiným uživatelem bezprostředně ukáží ve všech otevřených oknech prohlížeče nad touto aplikací.

Stisk tlačítka pro filtrování zobrazených úloh otevře formulářové políčko, které při každém zadaném či smazaném písmenu omezuje seznam níže zobrazených úloh pouze na ty, které obsahují zadanou sekvenci znaků, na velikost malých a velkých písmen však nehledě. Tím se dá potencionálně dlouhý a nepřehledný seznam úloh zkrátit i na jednu jedinou, požadovanou.

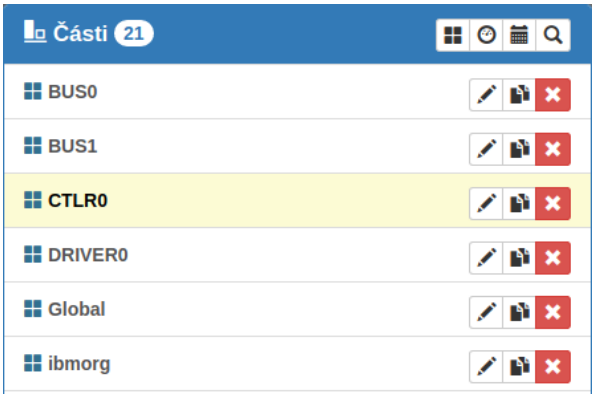
Stisk tlačítka přidání úlohy otevře formulářové políčko pro název úlohy. Nesmí být prázdné a samozřejmě ani nelze vytvořit úlohu, která již existuje.

Stisk tlačítka přejmenování úlohy otevře formulářové políčko pro nový název úlohy. Nesmí být prázdné a samozřejmě ani nelze úlohu přejmenovat na takovou, která již existuje a došlo-li by tedy k jejímu přepsání.

Stisk tlačítka kopírování úlohy otevře formulářové políčko pro název kopie úlohy. Nesmí být prázdné a samozřejmě ani nelze úlohu zkopírovat na takovou, která již existuje a došlo-li by tedy k jejímu přepsání. Kopírují se všechny části úlohy, ale bez případných zadaných nebo i již hotových simulací.

Stisk tlačítka smazání úlohy okamžitě a bez potvrzování danou úlohu smaže, tedy i všechny její části a případné simulace.

3.2.4 Panel částí



Obrázek 3.6: Panel částí

Panel částí slouží pro práci s částmi vybrané úlohami, tedy s konfiguračními bloky, vstupními seek a defs soubory, rozlišenými rozdílnou ikonou před svým názvem. Zobrazí se až po vybrání konkrétní úlohy v panelu úloh a sama mizí, pokud žádná úloha není zrovna vybrána – např. po smazání úlohy. Horní modrý pruh obsahuje celkový počet částí úlohy, tlačítka pro přidání nové části a tlačítko pro filtrování zobrazených částí. Kliknutí na horní pruh panelu jeho položky schová nebo opět ukáže.

Každá část úlohy je zobrazena v seznamu pod horním pruhem, spolu s tlačítky pro přejmenování, kopírování a smazání části. Kliknutím uživatel vybírá danou část úlohy, čímž se mu otevře panel detailů části dané úlohy a daná část úlohy se žlutě podbarví aby bylo zřejmé, že právě s ní se zrovna pracuje.

Počet i seznam částí úlohy se průběžně synchronizuje se serverem a tak se změny v nich jiným oknem prohlížeče daného uživatele nebo zcela jiným uživatelem bezprostředně ukáží ve všech otevřených oknech prohlížeče nad touto aplikací.

Stisk tlačítka pro filtrování zobrazených částí úlohy otevře formulářové políčko, které při každém zadaném či smazaném písmenu omezuje seznam níže zobrazených částí úlohy pouze na ty, které obsahují zadanou sekvenci znaků, na velikost malých a velkých písmen však nehledě. Tím se dá potencionálně dlouhý a nepřehledný seznam částí úlohy zkrátit i na jednu jedinou, požadovanou.

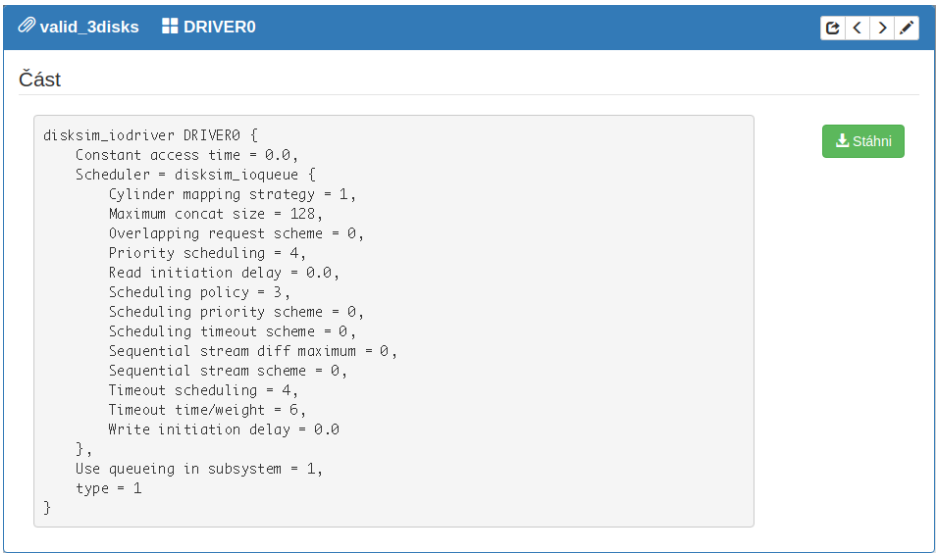
Stisk některého z tlačítka přidání části úlohy otevře formulářové políčko pro název části úlohy. Nesmí být prázdné a samozřejmě ani nelze vytvořit část úlohy, která již existuje.

Stisk tlačítka přejmenování části úlohy otevře formulářové políčko pro nový název části úlohy. Nesmí být prázdné a samozřejmě ani nelze část úlohy přejmenovat na takovou, která již existuje a došlo-li by tedy k jejímu přepsání.

Stisk tlačítka kopírování části úlohy otevře formulářové políčko pro název kopie části úlohy. Nesmí být prázdné a samozřejmě ani nelze část úlohu zkopírovat na takovou, která již existuje a došlo-li by tedy k jejímu přepsání.

Stisk tlačítka smazání části úlohy okamžitě a bez potvrzování danou část úlohy smaže bez ohledu na to, je-li na ní jiným blokem odkazováno či nikoliv.

3.2.5 Panel detailu části



Obrázek 3.7: Panel detailu části v režimu prohlížení

Panel detailu části slouží pro práci s konkrétní částí vybrané úlohy, tedy s konfiguračním blokem, vstupním seek nebo defs souborem. Zobrazí se až po vybrání konkrétní části dané úlohy v panelu částí a sama mizí, pokud žádná část úlohy není zrovna vybrána – např. po smazání části úlohy. Horní modrý pruh obsahuje název dané úlohy a dotyčné části, tlačítko pro změnu části a bloky navíc i tlačítka pro rozbalování odkazovaných bloků, odsazení doleva a doprava o jeden znak. Kliknutí na horní pruh panelu detail části schová nebo opět ukáže.

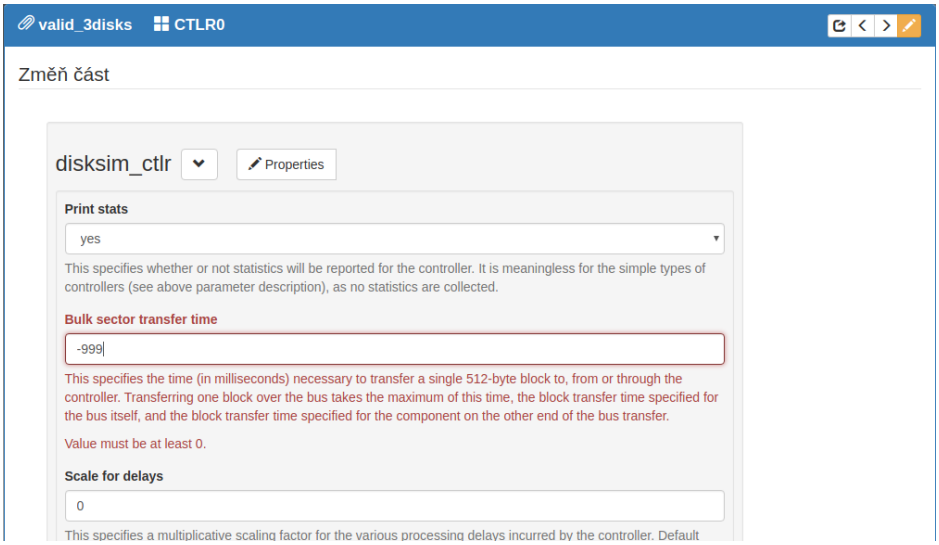
V režimu prohlížení je daná část celá zobrazena a uživatel si ji i může stáhnout kliknutím na tlačítko Stáhni. Pokud je prohlíženou částí blok, lze tlačítkem pro rozbalování odkazovaných bloků přepínat mezi zobrazením bloku tak jak je nebo rekurzivně nahrazovat deklarace `source` obsahem jím odkazovaného bloku, který se hledá v dané úloze nebo potom v úloze sdílené. Nastavení zobrazování s rozbalováním je barevně signalizováno na tlačítku pro rozbalování.

V režimu prohlížení bloků lze i měnit velikost odsazení tlačítky pro odsazení. Výslednou reprezentaci bloků vždy generuje server a klient je již jen zobrazuje.

Stisknutím tlačítka pro změnu části se panel detailu části přepne do režimu změny části či naopak zpět do režimu prohlížení. I to je signalizováno barevnou změnou tlačítka pro změnu části.

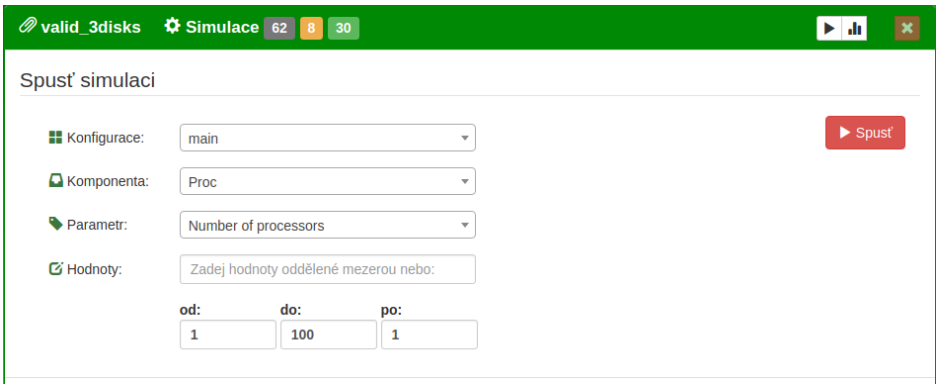
Změna části – vstupního souboru – ji otevře pro úpravy a poté jej lze uložit. Změna bloku ale otevře JSON Editor, který vykreslí uživateli HTML formulář

podle typu upravovaného bloku, hojně jej doplní vysvětlivkami a kontroluje i typy a rozsahy zadaných vstupů. Pokud není něco v pořádku, v editoru se chyba zobrazí i s popisem. Až když uživatel zadá všechny údaje správně, zpřístupní se mu tlačítko Ulož a může tak upravený blok uložit.



Obrázek 3.8: Panel detailu části v režimu změny bloku

3.2.6 Panel simulací



Obrázek 3.9: Panel simulací v režimu zadávání simulací

Panel simulací slouží pro zadávání simulací dané úlohy a prohlížení jejích výsledků. Zobrazí se až po vybrání konkrétní úlohy v panelu úloh a sama mizí, pokud žádná úloha není zrovna vybrána – např. po smazání úlohy. Horní zelený pruh ukazuje název dané úlohy, ukazatele počtů simulací, tlačítka pro schování nebo opětovné zobrazení pod-panelů pro spouštění simulací a pro

prohlížení výsledků a tlačítko smazání simulací. Kliknutí na horní pruh panelu celý jeho obsah schová nebo opět ukáže.

Ukazatele se průběžně synchronizují se serverem a tak se změny v nich jiným oknem prohlížeče daného uživatele nebo zcela jiným uživatelem bezprostředně ukáží ve všech otevřených oknech prohlížeče nad touto aplikací. Ukazatele zobrazují následující počty danou barvou, pokud je ovšem daný počet větší než nula:

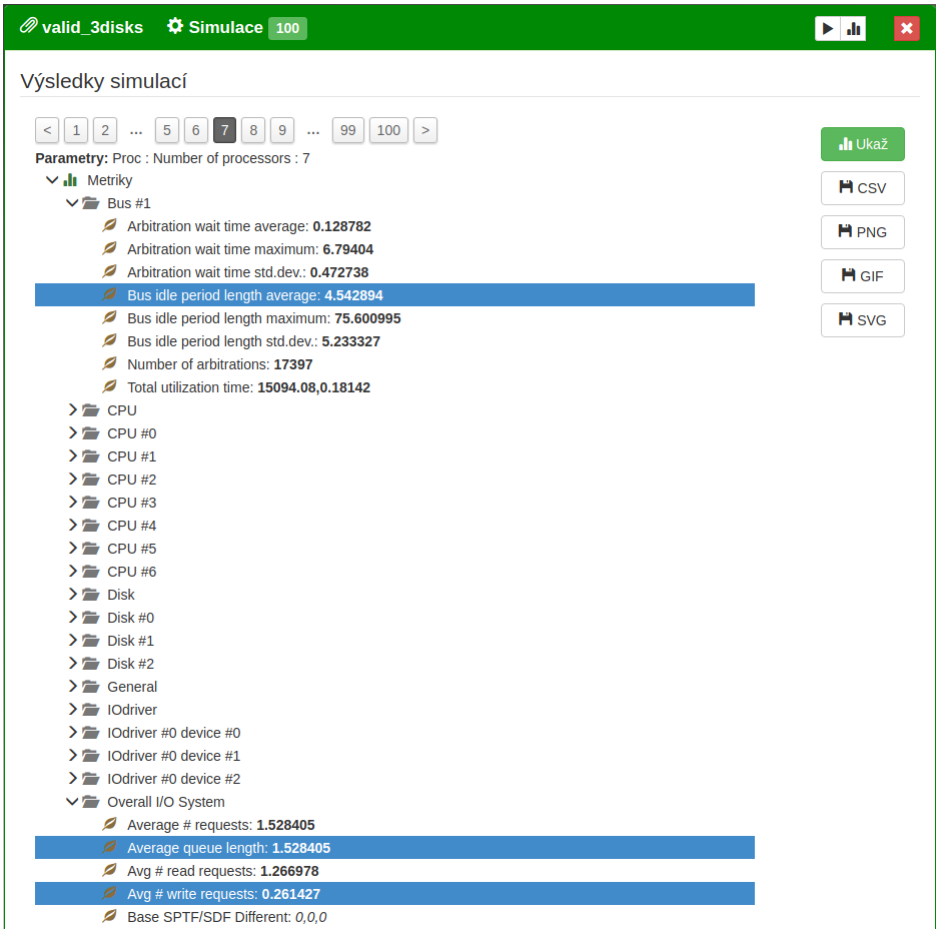
- šedý ukazuje počet simulací čekajících na spuštění,
- žlutý ukazuje počet právě probíhajících simulací,
- zelený ukazuje počet úspěšně dokončených simulací,
- červený ukazuje počet neúspěšně dokončených simulací.

Stisk tlačítka smazání simulací úlohy okamžitě a bez potvrzování zadané i již provedené simulace úlohy smaže.

Povinným parametrem pro spuštění simulace úlohy je volba konfigurace typu bloku `main`, která se vybírá z existujících v dané úloze a která pouze odkazuje na jednotlivé bloky které mají tvořit konfigurační soubor zvolené simulace. Teprve poté se uživateli zpřístupní tlačítko Spusť a uživatel může zadat jednu takovouto simulaci ke spuštění.

Také ovšem může vybrat jednu z komponent použitých v dané konfiguraci a z ní jeden parametr, který změní za jinou hodnotu, seznam hodnot nebo rozsah hodnot. Podle toho se pak spustí jedna nebo i více simulací dané úlohy.

Kliknutím na tlačítko pro zobrazení pod-panelu pro spouštění simulací je dostupné pokud má úloha alespoň jednu úspěšně dokončenou simulaci.

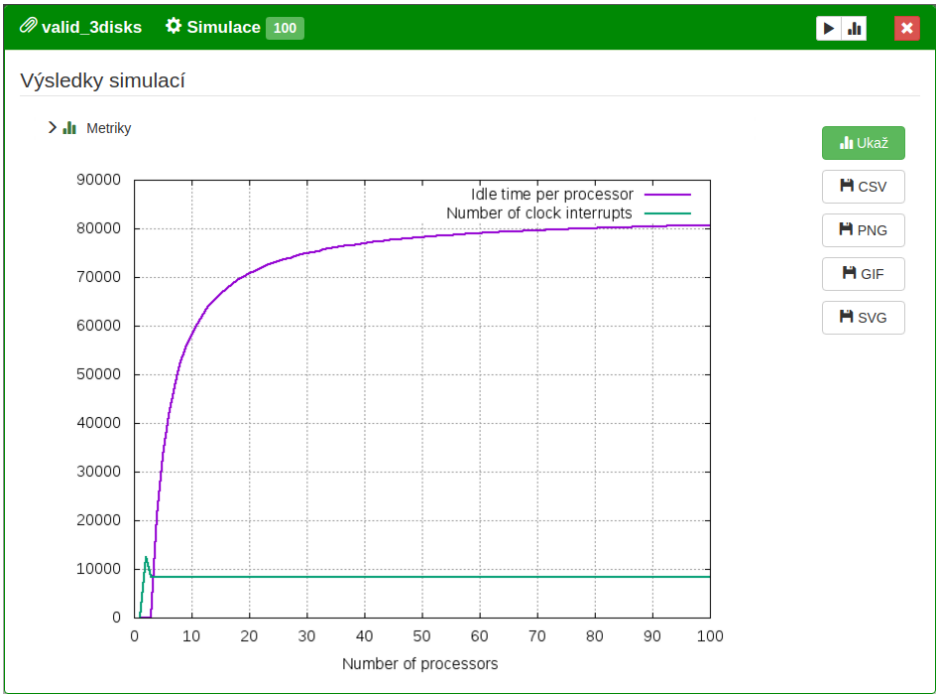


Obrázek 3.10: Panel simulací v režimu výsledků simulací

Seznam všech simulací nabízí uživateli možnost prohlédnout si každou jednotlivě zvlášť.

Výsledky simulací jsou shrnuty do přehledného rozbalovacího stromu, ve kterém jsou získané metriky seskupeny do logických celků. Pokud bylo výsledkem jedné metriky více hodnot, jsou odděleny čárkou.

Pokud bylo spuštění úlohy parametrizováno více než jednou hodnotou, lze označit jednu nebo více metrik a ty si nechat tlačítkem Ukaž ukázat jako graf s osou X měněného parametru, takový graf stáhnout v rastrové i vektorové grafice i všechny jejich tabulované hodnoty v CSV formátu. Velikost obrázku, který generuje server, je pružně určena velikostí okna prohlížeče uživatele.



Obrázek 3.11: Panel simulací s grafickou reprezentací výsledků

3.3 Instalace

Uživatelské rozhraní pro svoji správnou funkci vyžaduje programy `python`, `gnuplot` a `disksim`.

Programy `python` i `gnuplot` bývají snadno instalovatelnou částí UNIX systémů, ale konkrétní příkaz k instalaci se liší na každé distribuci a tak jej zde neuvádím. Leckde bývají rovnou i nainstalovány.

Program `disksim` není binárně distribuován a tak je nutné jej zkompilovat. Doporučuji použít projekt DiskSim Builder [27]:

```
git clone https://gitlab.fit.cvut.cz/jakubkam/disksim.git
cd disksim
make
```

Nejjednodušší instalací uživatelského rozhraní je naklonování repozitáře příkazem:

```
git clone https://gitlab.fit.cvut.cz/jakubkam/disksimui.git
cd disksimui
```

a poté případná úprava konfiguračního souboru `disksimui.conf` o cesty k programům `gnuplot` a `disksim`.

Závěr

Zadání práce požadovalo editovat diskovou konfiguraci, čehož jsem dosáhl jak na detailní úrovni konfigurace libovolné komponenty i jejího parametru, tak zjednodušenou konfigurací ve scénáři.

Další požadavek scénáře se mi podařilo implementovat zcela podle zadání a navíc podporuje i pole typu RAID 4 a libovolný počet zátěží.

Zadání parametrizace simulace jsem splnil volbu jednoho nepovinného parametru před spuštěním simulace a to jak zcela libovolného na úrovni spouštění úlohy nebo předdefinovaného ve scénáři. V obou případech se jeho hodnotám nastavuje rozpětí a velikost skoku mezi nimi.

Požadavek na spouštění je posledním krokem při spouštění úlohy nebo scénáře a nakonec se spouští paralelně na serverové části a uživatel tak může dále pracovat s aplikací.

Zadané vyhodnocování výstupů jsem provedl stromovým procházením všech metrik získaných z výstupu simulátoru nebo jen předdefinovaných ve scénáři.

Graficky zobrazuji výsledky formou přehledných grafů generovaných serverovou částí, které lze stáhnout v rastrovém i vektorovém formátu, stejně jako lze stáhnout generující data v CSV formátu.

Saturační analýza zkoumá, jak se mění průběh sledovaných veličin v závislosti na zátěži, jak se mění v oblasti saturace, ale i za ní, a to lze dobře sledovat při zadání vhodného proměnného parametru simulace ve výsledných grafech s výstupy simulace.

Projekt je nadále otevřen dalším rozšířením a vylepšením, např. podporou pro diskové konfigurace RAID 1+0, RAID 6 nebo pro konfiguraci výpadků a rekonstrukce disků.

Literatura

[1] Ganger, G.: The DiskSim Simulation Environment (v4.0). *The Parallel Data Lab*, květen 2015, [cit. 2017-03-25]. Dostupné z: http://pdl.cmu.edu/DiskSim/index.shtml

[2] Kim, Y.; Tauras, B.; Gupta, A.; aj.: FlashSim: A Simulator for NAND Flash-Based Solid-State Drives. *IEEE Xplore Digital Library*, 2009, doi: 10.1109/SIMUL.2009.17, [cit. 2017-03-25].

[3] Liu, O.: *Parallel File System Simulator for I/O Scheduling Algorithm Designers*. únor 2014, [cit. 2017-03-25]. Dostupné z: https://github.com/myidpt/PFSsim

[4] Hsieh, G.; Rodrigues, A.: *SST Overview*. únor 2012, [cit. 2017-03-25]. Dostupné z: http://comparch.gatech.edu/hparch/tutorial_slides/hpca2012/HPCA_SST-Macsim_tutorial.pdf

[5] Maher, U.: Disksim web. leden 2017, [cit. 2017-03-25]. Dostupné z: https://gitlab.fit.cvut.cz/maherula/disksim_web

[6] Intel: RAID Interactive Tutorial for Intel® Embedded Software Raid Technology 2 (ESRT2). srpen 2014, [cit. 2017-03-25]. Dostupné z: https://downloadcenter.intel.com/download/23572

[7] Smith, P.: RAID Simulator. 2017, [cit. 2017-03-25]. Dostupné z: http://www.coastalworks.com/raid/raid.html

[8] Synology: RAID Simulator. 2017, [cit. 2017-03-25]. Dostupné z: https://www.synology.com/en-us/support/RAID_calculator

[9] Ganger, G.: Designing Systems with MEMS-based Storage. *The Parallel Data Lab*, srpen 2012, [cit. 2017-03-25]. Dostupné z: http://pdl.cmu.edu/MEMS/index.shtml

[10] Microsoft: SSD Extension for DiskSim Simulation Environment. březen 2009, [cit. 2017-03-25]. Dostupné z: http://research.microsoft.com/research/downloads/Details/b41019e2-1d2b-44d8-b512-ba35ab814cd4/Details.aspx

[11] Bucy, J. S.; Schindler, J.; Schlosser, S. W.; aj.: *The DiskSim Simulation Environment Version 4.0 Reference Manual*. srpen 2008, [cit. 2017-03-25]. Dostupné z: http://www.pdl.cmu.edu/PDL-FTP/DriveChar/CMU-PDL-08-101.pdf

[12] JSON Schema. 2017, [cit. 2017-03-25]. Dostupné z: http://json-schema.org

[13] Hellkamp, M.: Bottle: Python Web Framework. duben 2017, [cit. 2017-03-25]. Dostupné z: https://bottlepy.org

[14] McDonough, C.: Production-quality pure-Python WSGI server with very acceptable performance. březen 2017, [cit. 2017-03-25]. Dostupné z: https://waitress.readthedocs.io

[15] Foundation, P. S.: WSGI Utilities and Reference Implementation. březen 2017, [cit. 2017-03-25]. Dostupné z: https://docs.python.org/3/library/wsgiref.html

[16] Peterson, B.: Six: Python 2 and 3 Compatibility Library. 2015, [cit. 2017-03-25]. Dostupné z: http://pythonhosted.org/six/

[17] Berman, J.: jsonschema 2.6.0: An implementation of JSON Schema validation for Python. 2017, [cit. 2017-03-25]. Dostupné z: https://pypi.python.org/pypi/jsonschema

[18] Montanaro, S.: lockfile 0.9.1 documentation. 2014, [cit. 2017-03-25]. Dostupné z: http://pythonhosted.org/lockfile/

[19] SmartBear: SWAGGER UI. 2017, [cit. 2017-03-25]. Dostupné z: http://swagger.io/swagger-ui/

[20] The F., jQuery jQuery - write less, do more. 2017, [cit. 2017-03-25]. Dostupné z: http://getbootstrap.com/

[21] Miles, J.: Tree View for Twitter Bootstrap. 2017, [cit. 2017-03-25]. Dostupné z: https://github.com/jonmiles/bootstrap-treeview

[22] The F., jQuery jQuery - write less, do more. 2017, [cit. 2017-03-25]. Dostupné z: http://jquery.com

[23] Otto, M.; Thornton, J.: Bootstrap - The world's most popular mobile-first and responsive front-end framework. 2017, [cit. 2017-03-25]. Dostupné z: https://select2.github.io/

[24] Valstar, R.: TinySort - a small script that sorts HTMLElements. 2017, [cit. 2017-03-25]. Dostupné z: http://tinysort.sjeiti.com

[25] Dorn, J.: JSON Schema Based Editor. 2017, [cit. 2017-03-25]. Dostupné z: https://github.com/jdorn/json-editor

[26] Matis, F.: simplePagination - a simple jQuery pagination plugin, 3 CSS themes and Bootstrap support. 2017, [cit. 2017-03-25]. Dostupné z: http://flaviusmatis.github.io/simplePagination.js

[27] Jakubovič, K.: Disksim builder and scripts. únor 2017, [cit. 2017-03-25]. Dostupné z: https://gitlab.fit.cvut.cz/jakubkam/disksim

Příloha A

Konfigurační soubor DiskSim grafického rozhraní

Soubor `disksimui.conf` konfiguruje výchozí nastavení WSGI serveru i DiskSim serverovou aplikaci. Zde je jeho obsah včetně vysvětlujících komentářů:

```
# Konfigurace vychozich hodnot WSGI serveru.
[server]

# Sitove rozhrani, na kterem ma server poslouchat
# a ocekavat prichozi HTTP pozadavky.
# Hodnota "localhost" zpristupni aplikaci jen
#   na tom samem stroji
# Hodnota "*" zpristupni aplikaci odkudkoliv
host = localhost

# TCP port na kterem ma server poslouchat
# a ocekavat prichozi HTTP pozadavky.
# Na OS UNIX muze neprivilegovany uzivatel otevirat
# TCP porty pro poslouchani pouze vyssi nez 1024
port = 8080

# Interval v sekundach pro kontrolu zmeny souboru pouzivanych
# serverem s jeho naslednym automatickym restartem.
# Hodnota "0" tuto vlastnost vypina, coz je i doporuceno
# pro bezne pouziti.
interval = 0


# Konfigurace cest
[path]

# adresar s ulohami, vcetne te sdilene
# - absolutni nebo relativni vuci
#   adresari s touto konfiguraci
# - pozor, zabira velmi mista na disku
tasks = tasks
```

```
# nazev podadresare pro spustene simulace ulohy
runs = runs

# adresar pro zamykani uloh, vcetne te sdilene
# - absolutni nebo relativni vuci
#   adresari s touto konfiguraci
# - zabira mimum mista na disku
locks = locks


# Nastaveni REST aplikacniho rozhraniho
[api]

# cele cislo verze rozhrani
# - pozadavky klienta explicitne pouzivajici jine
#   cislo verze jsou odmitnuty
version = 1

# nastaveni CORS (Cross-Origin Resource Sharing)
# - hodnota nastavuje HTTP hlavicku odpovedi
#   'Access-Control-Allow-Origin'
# - hodnota "*" povoluje pristup odkukoliv
# - jine hodnoty omezuji klienty pouze z dane URL
cors    = *


# Definice MIME typu
[type]

# JSON pozadavky a odpovedi
json = application/json

# nativni format DiskSim konfiguracniho souboru
parv = text/plain

# nativni format DiskSim seek souboru
seek = text/plain

# nativni format DiskSim defs souboru
defs = text/plain

# CSV tabulky
csv  = text/csv

# grafy v rastrovem PNG formatu
png  = image/png

# grafy v rastrovem GIF formatu
gif  = image/gif

# grafy ve vektorovem SVG formatu
svg  = image/svg+xml
```

```
# Nazvy pripon ruznych typu soboru
[ext]

# pripona bloku konfigurace
block = .json

# pripona seek vstupnich souboru simulatoru
seek  = .seek

# pripona defs vstupnich souboru simulatoru
defs  = .defs

# pripona konfiguracniho souboru simulatoru
parv  = .parv


# Nastaveni uloh
[task]

# nazev sdilene ulohy
common = Common


# Konfigurace vykonavatelu
[worker]

# cesta ke spustitelnemu souboru simulatoru
# - absolutni nebo relativni vuci
#   adresari s touto konfiguraci
bin           = /usr/bin/disksim

# pocet vlaken vykonavatelu
# - hodnota "auto" jej nastavi na pocet jader CPU
threads       = auto

# cekaci interval pred hledanim uloh ke spusteni
# po stavu s (jiz) zadnou takovou ulohou
# - v sekundach
sleep         = 2

# jmeno suboru s parametry pro spusteni
# programu simulatoru
run           = .run.json

# jmeno vystupniho souboru simulatoru
stdout        = .output.log

# jmeno chyboveho vystupniho souboru simulatoru
stderr        = .output.err

# jmeno souboru ze zpracovanymi vystupnimi
# statistikami dokoncene simulace
stats         = .stats.json
```

```
# jmena souboru signalizujicich stavy simulace,
# mezi kterymi je prechazeno
# - prvni "state" je jen prefixem tech ostatnich
state         = .state
state_new     = .state.new
state_waiting = .state.waiting
state_running = .state.running
state_failed  = .state.failed
state_done    = .state.done

# nazev klice s metadaty pridavanych
# do odpovedi s metrikami a statistikami vystupu
key_meta      = .meta

# nazev klice s chybami  simulace pridavanych
key_error     = .error


# Nastaveni nastroje pro graficke vystupy
[gnuplot]

# cesta ke spustitelnemu souboru GNUPLOT
# - absolutni nebo relativni vuci
#   adresari s touto konfiguraci
bin = /usr/bin/gnuplot
```

Příloha **B**

Seznam zkratek a pojmů

AJAX *Asynchronous JavaScript and XML*, technologie interaktivních webových aplikací měnící svůj obsah bez nutnosti kompletního znovunačítání

aplikační rámec *framework*, softwarová struktura podporující programování, vývoji a organizaci jiných softwarových projektů

bus přenosová sběrnice, skupina vodičů přenášející řídící, adresní anebo datové informace mezi systémy

CLI *Command-Line Interface*, uživatelské rozhraní příkazového řádku

controller diskový řadič, elektronická řídící jednotka disku

CRUD *create, read, update, delete*; vytvořit, získat, upravit, smazat; postačující typy operací nad zdroji

CSS *Cascading Style Sheets*, jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML a podobných

CSV *Comma-Separated Values*, hodnoty oddělené čárkami, jednoduchý souborový formát určený pro výměnu tabulkových dat

DOM *Document Object Model*, objektový model dokumentu

driver ovladač zařízení, software umožňující operačnímu systému s hardware

GUI *Graphical User Interface*, grafické uživatelské rozhraní

HTML *HyperText Markup Language*, značkovací jazyk používaný pro tvorbu webových stránek

HTTP *Hypertext Transfer Protocol*, internetový protokol určený pro výměnu hypertextových dokumentů

chunk souvislý elementární blok dat na disku nebo diskovém poli

JavaScript multiplatformní objektově orientovaný skriptovací jazyk

JOBD *Just a Bunch Of Disks*, zřetězení disků bez prokládání dat

JSON *JavaScript Object Notation*, způsob zápisu dat nezávislý na počítačové platformě, určený pro přenos dat, která mohou být organizována v polích nebo agregována v objektech

MEMS *MicroElectroMechanical Systems*, nevolatilní mikroelektromechanické úložné systémy

middleware specializovaný software poskytující aplikacím služby nad rámec těch poskytovaných operačním systémem

parita nadbytečná informace pro detekci a opravu chyb

patch textový soubor popisující změny zdrojového programu, opravující chyby nebo rozšiřující funkcionalitu

Perl interpretovaný programovací jazyk převážně pro práci s textem

PHP *Personal Home Page*, skriptovací programovací jazyk převážně pro programování dynamických internetových stránek

POSIX *Portable Operating System Interface*, standard jednotného rozhraní operačních systémů - převážně UNIX

Python univerzální objektový interpretovaný skriptovací programovací jazyk

queue fronta, datová struktura pro zpracování požadavků

RAID *Redundant Array of Independent Disks*, vícenásobné diskové pole nezávislých disků

REST *Representational State Transfer*, architektura rozhraní pro distribuovaná prostředí

Ruby interpretovaný skriptovací programovací jazyk

scheduler plánovač, rozvrhuje úkoly, zpracovává požadavky

SSD *Solid-State Disk*, nevolatilní disk držící data v integrovaných obvodech

UNIX rodina operačních systémů odvozených z původního AT&T Unix

WSGI *Web Server Gateway Interface*, jednoduché a univerzální rozhraní mezi webovým serverem a webovou aplikací

YAML *YAML Ain't Markup Language*, formát pro serializaci strukturovaných dat