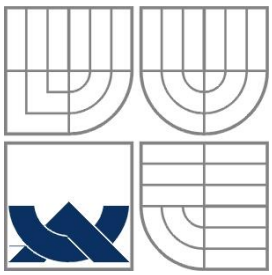
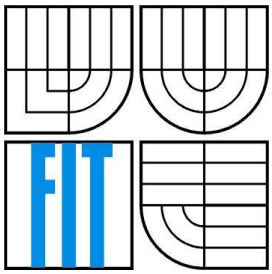


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

REPORTOVACÍ NÁSTROJ PRO GRIDENGINE

GRIDENGINE REPORTING TOOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FRANTIŠEK ROŽEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ KAŠPÁREK

BRNO 2016

Zadání bakalářské práce/8461/2015/xrozek01

Vysoké učení technické v Brně - Fakulta informačních technologií

Centrum výpočetní techniky Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Rožek František**
Obor: Informační technologie
Téma: **Reportovací nástroj pro GridEngine**
GridEngine Reporting Tool
Kategorie: Web

Pokyny:

1. Prostudujte systémy pro HPC výpočty z rodiny Grid Engine.
2. Dle pokynů vedoucího navrhněte nástroj pro reportování, zpracování a visualizaci účtovacích informací z tohoto systému.
3. Implementujte navržený systém jako webovou aplikaci.
4. Otestujte chování aplikace a demonstrujte její přínosy.

Literatura:

- SGE https://arc.liv.ac.uk/trac/SGE, navštíveno 2015-10-27

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kašpárek Tomáš, Ing.**, CVT FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L. S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této práce je sestavit nástroj, který bude odrážet využití výpočetního clusteru postaveného na technologii Grid Engine. Data jsou zpracována pomocí PHP a Shell skriptů a následně uložena do MySQL, či RRD databází. V práci byl vytvořen systém, který zpracovává obrovské množství dat a poskytuje ucelený náhled na využití celého clusteru, ale i jeho konkrétních komponent, či na statistiky jednotlivých uživatelů. Vytvořené řešení poskytuje aktuální i dlouhodobá data. Výsledek této práce umožňuje sledovat výpočetní cluster z jediného nástroje, což dříve nebylo možné.

Abstract

The aim of this work is to build a tool that will reflect the utilization of the computing cluster, built on Grid Engine technology. Data are processed using PHP and Shell scripts and then stored in MySQL, or RRD databases. The work created a system that handles huge amounts of data and provides a comprehensive view on the utilization of the entire cluster, but also its specific components, or statistics of individual users. Created solution provides current and long-term data. The result of this work allows you to watch computing cluster from a single tool, which was not possible before.

Klíčová slova

reportovací nástroj, výpočetní cluster, Grid Engine, HPC, RRDtool, JavascriptRRD

Keywords

reporting tool, computing cluster, Grid Engine, HPC, RRDtool, JavascriptRRD

Citace

ROŽEK, František. *Reportovací nástroj pro GridEngine*. Brno, 2016. 36s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kašpárek Tomáš.

Reportovací nástroj pro GridEngine

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Tomáše Kašpárka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
František Rožek
18. května 2016

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Tomáši Kašpárkovi za jeho odbornou pomoc a materiály, které mi při řešení mé práce poskytl a za čas, který této práci věnoval.

Obsah

1 Úvod

Tato práce se zabývá tvorbou reportovacího nástroje pro Grid Engine. Čtenář se dozví návrh a postup tvorby tohoto reportovacího nástroje.

Grid Engine je systém pro správu dávkového zpracování úloh. Tento systém se typicky používá na výpočetních clusterech pro HPC výpočty (vysoce náročné výpočty). Je nasazen i na Fakultě informačních technologií VUT v Brně a zajišťuje správu výpočetního clusteru a je zodpovědný za přijímání, plánování, odesílání a správu vzdáleného a distribuovaného provádění velkého počtu samostatných, paralelních a uživatelsky interaktivních úloh.

Cílem práce je vytvořit reportovací nástroj pro výše zmíněnou fakultu. V práci je uvedeno z jakých komponent se její výpočetní cluster skládá. Nástroj bude uživatelům poskytovat požadované informace o stavu výpočetního clusteru snadněji pomocí jednoho nástroje. Tento nástroj má být implementován jako webová aplikace. To umožňuje přístup k tomuto nástroji z libovolné platformy. Uživatelé nemusí žádnou aplikaci instalovat, pouze spustí webový prohlížeč a připojí se k adrese, kde bude nástroj umístěn.

Tato práce vychází z dosavadního[1] řešení reportovacího nástroje. Existující řešení je pouhý základ a žádá si větší rozšíření tohoto nástroje.

Prvním krokem při tvorbě takového nástroje je nutné stanovit případy užití výsledného systému. To znamená, jaké požadavky budou uživatelé na tento nástroj klást.

Analýzou těchto požadavků se vytřídí informace, které budou k jejich realizaci vyžadovány. V dalším kroku se určí, zdali jsou potřebné informace dostupné, případně odkud se tyto informace získají. Požadované informace mohou být v tomto nástroji aktuálního nebo historického typu. Z tohoto důvodu budou vytvořeny databáze, ve kterých se budou tato historická data uchovávat.

Po analyzování a získání dat je nutné navrhnout jejich zobrazení tak, aby zobrazení pokrývalo požadavky uživatelů. Poskytovalo jim snadnou orientaci v tomto zobrazení a další možnosti, které jim zpříjemní práci s tímto nástrojem. Tyto možnosti mohou být například filtrování dat, řazení dat nebo přibližování grafu. Je vždy nutné požadavky roztřídit do určitých kategorií, podle kterých se vytvoří základní menu daného systému.

Na závěr se systém otestuje a ověří, zdali podává validní informace a zhodnotí se, do jaké míry je systém použitelný. Také se hodnotí přínos nástroje uživatelům, jak jsou s ním uživatelé spokojeni, a do jaké míry jim usnadňuje práci při získávání jimi požadovaných informací.

[1] Dosavadní řešení reportovacího nástroje

- pohled na cluster - http://merlin.fit.vutbr.cz/sge-stats/index.php
- pohled na statistiky uživatelů - http://merlin.fit.vutbr.cz/sge-stats/users.html

2 Základní informace

2.1 HPC výpočty

HPC výpočty (High-Performance Computing) neboli vysoce náročné výpočty jsou v praxi takové výpočty, které potřebují k vyřešení problému o tolik větší výpočetní výkon, než by se dalo dostat z obyčejného stolního počítače nebo pracovní stanice.

Jak uvádí server TechTarget [1] HPC výpočty využívají paralelního zpracování úloh pro běh pokročilých aplikačních programů efektivně, rychle a spolehlivě. Termín HPC se týká zejména systémů, které pracují nad teraFLOPS (FLoating point Operations Per Second - operací v pohyblivé řádové čárce za sekundu). Termín HPC se občas používá jako synonymum pro počítání na superpočítačích. Tyto výpočty většinou řeší složité problémy z oblasti vědy, techniky nebo podnikaní.

Definice superpočítače na serveru TechTarget [2] uvádí, že superpočítač je takový počítač, který poskytuje nebo se alespoň blíží vždy současné nejvyšší provozní rychlosti pro počítače. Většina superpočítačů je v dnešní době realizována jako více propojených počítačů pomocí vysokorychlostní sítě, které provádějí paralelní zpracování úloh a tvoří počítačový cluster.

Lidé okolo HPC často nazývají jednotlivé počítače v clusterech jako uzly. Tyto uzly[2] využívají vícejádrové serverové procesory (např.: řady Intel Xeon[3] nebo řady AMD Opteron[4]), které obsahují standardně 2-4 tyto procesory, každý takový procesor má 4-18 jader a disponují pamětí až přes 512 GB.

Měření superpočítačů v TOP 500 List[5] je založené na Linpack benchmarku[6], který řeší hustý systém lineárních rovnic. Superpočítače jsou velmi exotické stroje vzhledem k použitým technologiím uvnitř těchto strojů a rozměrům, kterých tyto stroje nabývají. Nejvýkonnější počítač podle Top500 List je v dnešní době národní superpočítač v Kantonu v Číně. Obsahuje 3 120 000 jader, jeho maximální výkon je 33 862,7 TFLOP/s, ve špičce potom 54 902,4 TFLOP/s. Jeho příkon je 17 808 kW.

2.2 Monitorovací systémy

Podle Gridload [3] existuje několik monitorovacích systémů. Dále uvedené systémy byly inspirací pro vytvoření reportovacího nástroje pro Fakultu informačních technologií Vysokého učení technického v Brně.

Monit

Monit nejen monitoruje server, ale také se snaží o nápravu problémů tím, že využívá předem určené akce pro určité situace.

[2] Uzly pro HPC - http://www.analisisysimulacion.com/documentos/productos/pdf/hp_hpc_ays.pdf

[3] Intel Xeon - http://ark.intel.com/products/family/78583/Intel-Xeon-Processor-E5-v3-Family#@Server

[4] AMD Opteron - http://www.amd.com/en-us/products/server/opteron/6000/6200

[5] Top500 List - http://www.top500.org/list/2015/11/

[6] Linpack benchmark - http://www.top500.org/project/linpack/

Ganglia

Ganglia je škálovatelný distribuovaný monitorovací systém pro vysoce výkonné výpočetní systémy, jako jsou clustery. Využívá široce používané technologie jako je XML pro reprezentaci dat a RRDtool pro ukládání dat a vizualizaci.

Ganglia poskytuje pohled na využití celého clusteru. Je to velice užitečný nástroj pro monitorování počítačového clusteru, ale je zbytečné ho využívat na monitorování jednoho serveru.

Munin

Munin monitoruje a produkuje grafy využití systému. Je možné automaticky vytvářet denní, týdenní, měsíční, roční grafy výkonu a zprávy o mnoha důležitých metrikách. Je dodáván s možností monitorovat základní systémové prostředky, jako je paměť, místo na disku, využití procesoru a serverových aplikací, jako je MySQL, Apache a Squid. Jednou z největších předností Munin je, že může být velice jednoduché ho dále rozšířit pomocí pluginů.

Cacti

Cacti je podobný systému Munin v mnoha ohledech. Co Cacti od Muninu odlišuje je to, že umožnuje měnit velikost grafů a zobrazovat data z libovolného rozsahu. Munin má pevně dané denní, týdenní, měsíční a roční grafy.

Nagios

Nagios je, podle svých internetových stránek, průmyslovým standardem pro monitorování IT infrastruktury. Nagios může být složité nainstalovat a nakonfigurovat, ale jeho bohatství funkcí je nesrovnatelné s jakýmkoli nástrojem na trhu. Je zaměřen na zkušené správce sítí. Podporuje sledování více počítačů a může posílat upozornění prostřednictvím e-mailu, pageru nebo textových zpráv. Jako Monit může být také nakonfigurován tak, aby automaticky reagoval na problémy.

2.3 Školní výpočetní cluster

Výpočetní cluster je založen na blade serverech IBM a Dell. Celkem je nyní osazeno 102 modulů, každý má dva 4-16 jádrové procesory a paměť 8-256 GB. Na blade serverech běží 64bitový systém Linux CentOS 6.x. Pro ukládání rozsáhlých dat je k dispozici 12 souborových serverů o celkové kapacitě 400 TB. Je k dispozici také 6 GPU serverů, které všechny disponují čtyřmi grafickými procesory nVidia GeForce GTX 980. Propojení těchto komponent je zajištěno pomocí 10 gigabitové sítě.

Zpracování úloh v clusteru je řízeno systémem Sun Grid Engine (SGE). Požadovaný výpočet musí být popsán skriptem shellu, který je předán do SGE. SGE na základě aktuálního stavu uzlů v clusteru rozhodne, kdy a na kterých z dostupných uzlů úlohu spustí. Maximální paralelní výpočetní kapacita clusteru je dynamická, závisí na stavu jednotlivých blade serverů, teoreticky cca 2400 procesů.

Předcházející údaje pochází z webových stránek fakulty informačních technologií [4] a správce výpočetního clusteru pana Ing. Tomáše Kašpárka.

3 Použité technologie

V této kapitole je uvedeno, které technologie byly pro vytvoření reportovacího nástroje použity. U každé technologie bude její popis a vysvětleno, proč byla zvolena.

3.1 Základní technologie

Reportovací nástroj je vytvořen jako webová aplikace, bylo tedy nutné použít technologie, které slouží k vytvoření webových stránek. Tyto technologie jsou v této kapitole jenom letmo zmíněny s krátkým popisem, na co byly použity.

HTML

HTML (Hyper Text Markup Language) je značkovací jazyk pro vytvoření základní struktury webových stránek. Vytváří se dílčí elementy, které mohou mít své atributy. Tyto elementy poté tvoří jako celek výslednou strukturu dokumentu. K těmto elementům je možné dále přistupovat a měnit jejich obsah či atributy dynamicky např. pomocí JavaScriptu.

CSS

CSS (Cascading Style Sheets) je jazyk pro popis způsobu zobrazení elementů HTML dokumentu. CSS technologie programátorovi usnadňuje práci. Nemusí psát ke každému elementu styl zobrazení zvlášť, ale programátor ho napíše pouze na jednom místě. Tento styl potom mají všechny elementy stejného typu. To má široké využití i při úpravách designu. Změní-li se jedna hodnota v CSS souboru, změna se projeví u všech elementů.

PHP

PHP (Hypertext Preprocessor, původně Personal Home Page) je skriptovací jazyk, který zajišťuje zpracování skriptu na straně serveru. Na klientský webový prohlížeč je přenesen už pouze vyhodnocený skript ve formátu HTML dokumentu. Poskytuje dynamiku webových stránek, kdy se podle zadaných podmínek může webová stránka zobrazit pokaždé jiným způsobem. Nabízí také vhodné a jednoduché rozhraní pro komunikaci s MySQL databází.

MySQL

MySQL (My Structured Query Language) je databázový systém. Pro komunikaci s ním se používá jazyk SQL. Pomocí něho lze s databází manipulovat. Umožňuje vkládat, upravovat a mazat data nebo data z databáze číst a také vytvářet, mazat či upravovat celé databáze.

JavaScript

JavaScript je objektově orientovaný skriptovací jazyk. V tomto nástroji je využit pro ovládání interaktivních prvků GUI. Umožňuje nám reagovat na události a podle nich provádět dynamicky změnu dokumentů HTML.

AJAX

AJAX (Asynchronous JavaScript and XML) technologie umožňuje vytvořit asynchronní webové aplikace. To znamená načítat data asynchronně (na pozadí) a měnit dynamicky obsah stránky bez jejího opětovného načtení.

3.2 RRDtool

Tobias Oetiker [5] uvádí, že RRDtool (Round-robin database tool) je volně šiřitelný průmyslový standard, který umožňuje uchování dat a vykreslování dat pomocí grafů pro vysoce náročné časově závislé údaje. Tato technologie se dá jednoduše naimplementovat do skriptovacích jazyků jako je Shell, Perl, Python, Ruby nebo jako u tohoto nástroje PHP.

Struktura RRD souboru

RRD soubor je tvořen datovými řadami DS, u kterých se specifikuje typ řady GUAGE, COUNTER, DERIVE nebo ABSOLUTE, parametr *heartbeat*, který určuje po jakém intervalu je hodnota určena jako neznámá, a v jakých mezích by se jejich hodnoty měly vyskytovat, jinak je uložena nedefinovaná hodnota. RRD soubor dále obsahuje archivy RRA pro ukládání těchto řad. U archivu se musí specifikovat 4 parametry. První je konsolidační funkce AVERAGE, MAX, MIN nebo LAST. Dále parametr *xff*, který určuje část konsolidačního intervalu, která se může skládat z neznámých hodnot, zatímco výsledná konsolidační hodnota je stále považována za známou. Třetí parametr definuje kolik hodnot DS se použije k vytvoření konsolidační hodnoty a poslední parametr určuje velikost archivu.

Rozdíly mezi RRD a ostatními databázemi:

- V případě lineárních databází se nová data připojí na konec tabulky databáze. Z tohoto důvodu velikost databáze neustále roste, zatímco velikost databáze RRD je určena v okamžiku vytvoření a je po celou dobu konstantní.
- Jiné databáze aktualizují hodnoty těmi, které jsou dodávány. RRD mohou být konfigurovány pro ukládání míry změny mezí předchozí a současnou hodnotou.
- Databáze RRD je strukturována tak, že potřebuje data v předem určených časových intervalech. Pokud není dodána nová hodnota během tohoto intervalu, uloží se neznámá hodnota do tohoto intervalu.
- Při použití databáze RRD je nutné použít skripty, které běží v pravidelných intervalech, aby zajistily stálý tok dat do databáze.

3.3 JavascriptRRD

JavascriptRRD [6] je javascriptová knihovna pro čtení a interpretaci RRD souborů. Využívá technologie založené na technologii AJAX, aniž by na vzdáleném serveru spustila nějaký kód. To znamená, že čistě na klientské straně poskytuje přístup k souborům RRD. JavascriptRRD disponuje funkcemi pro čtení těchto souborů a pomocnou knihovnou Flot pro jejich zobrazení.

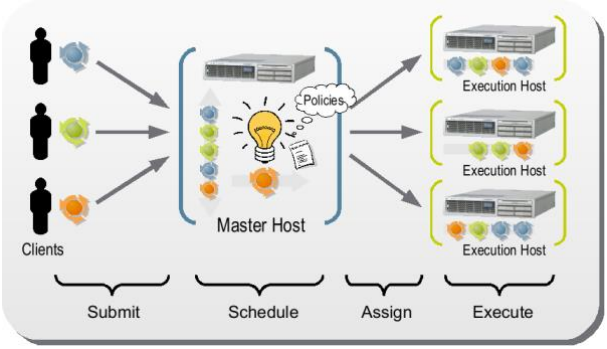
Z této knihovny byly využity moduly RrdFlot pro zobrazení grafů založených na jediném RRD souboru a modul RrdFlotMatrix, který vykresluje více RRD souborů do jednoho grafu.

3.4 Grid Engine

Univa Grid Engine, dříve známý jako Oracle Grid Engine nebo Sun Grid Engine (SGE), je systém pro správu dávkového zpracování úloh. Na základě tohoto systému už vznikla spousta odnoží, které disponují různými vlastnostmi, ale základ je víceméně stejný. V této práci se budeme zabývat odnoží *Son of Grid Engine*, která navazuje na projekt SGE od společnosti Sun.

Oracle [7] uvádí, že se Grid Engine používá obvykle na výpočetních clusterech a je zodpovědný za přijímání, plánování, odesílání a správu vzdáleného a distribuovaného provádění velkého počtu samostatných, paralelních nebo interaktivních uživatelských úloh. DRM (Distributed Resource Manager) řídí a plánuje přidělování dostupných prostředků. Jeho úkolem je přijmout seznam úloh od uživatelů a efektivně ho distribuovat na dostupné stroje. DRM usnadňuje práci uživatelům, správci a hlavně organizacím, protože se snaží držet všechny jejich stroje vytížené tak, aby bylo dosaženo maximálního efektivního využití.

Grid Engine se skládá z výkonných uzlů, master uzlu a ze záložních stínových master uzlů, kdyby se master uzel porouchal, viz obrázek 1 - struktura Grid Enginu. Na všech uzlech běží bez pozornosti uživatele příslušní démoni (execution daemon, qmaster daemon a shadow daemon). Qmaster démon je srdcem celého clusteru a bez něho by nemohly být žádné úlohy předloženy ke zpracování ani naplánovány. Na výkonných uzlech se provádějí úlohy.



Obrázek 1 - struktura Grid Enginu
Zdroj: https://blogs.oracle.com/templedf/entry/sun_grid_engine_for_dummies

Uživatel předkládá úlohu ke zpracování použitím příkazu k předložení úlohy (např.: *qsub*, *qmon*) a přikládá také důležité informace o úloze: co by měla skutečně dělat, jaký druh výkonného uzlu potřebuje, kolik spotřebuje paměti, jak dlouho bude běžet, atd. Tyto informace použije qmaster démon k plánování a správě dané úlohy. Dostupné sloty jsou přidělovány úlohám podle priority úlohy. Při určování priority záleží na politice daného systému. Slot umožňuje spustit úlohu. Počet slotů na výkonném uzlu je dán počtem CPU jader, která tento uzel má. Poté co se úloha v daném slotu dokončí, démon na výkonném uzlu po ní uklidí a informuje o tom qmaster démona. Qmaster démon zapíše informace do účtovacího modulu a odstraní úlohu ze seznamu aktivních úloh.

Klíčové vlastnosti Grid Engine:

- škálovatelnost – lze použít pro správu 100 počítačů v clusteru, tak i pro tisíce počítačů v clusteru
- flexibilita – lze nastavit tak, aby vyhovoval všem potřebám
- pokročilý plánovač – poskytuje různé politiky, jak budou úlohy distribuovány
- spolehlivost – zákazníci potvrzují spolehlivost systému. Po první inicializaci se o něho téměř není nutné starat.

4 Návrh reportovacího nástroje

Pro další pochopení textu si je potřeba ujasnit dále používané názvosloví. Uživatelé do clusteru posílají ke zpracování úlohy (*anglicky: job*). Jedna úloha se může skládat z několika dílčích úloh (*anglicky: task*). V textu bude dále použito počeštěné slovo *task* nebo *podúloha*, protože pro dílčí úlohu nebylo vhodně nalezeno jiné české vyjádření.

4.1 Případy užití reportovacího nástroje

Prvním krokem návrhu reportovacího nástroje je zjistit, jak by uživatelé chtěli tento nástroj používat. K tomu slouží případy užití systému. Je nutné zvážit, jaká data budou k poskytnutí daného případu užití potřeba a jakým způsobem tato data budou prezentována. Dále jsou uvedeny konkrétní případy užití reportovacího nástroje.

4.1.1 Zpoplatnění služeb clusteru

Popis případu užití

Na fakultě bylo zavedeno, že používání clusteru bude zpoplatněno. Pověřená osoba fakulty jde za správcem výpočetního clusteru a potřebuje zjistit, jak uživatelé zatěžovali cluster za nějaké období. Ze získaných statistik bude poté vyúčtováno zpoplatnění služeb z pohledu vytížení prostředků výpočetního clusteru.

Analýza případu užití

V tomto případě bude nutné zjistit u každého uživatele, jaké úlohy v daném období vykonal. U těchto úloh se musí zjistit, jak vytížily dané prostředky (CPU, GPU, datové servery) a z těchto hodnot určit, jak uživatelé v daném období vytěžovali prostředky a spočítat přibližný odběr elektřiny.

Zobrazení případu užití

Tento případ užití bude zobrazen tabulkou. Tabulka bude obsahovat jméno uživatele, jeho zařazení (např. ústav) a odběr elektřiny v kWh. Tato data budou vztažena ke zvolenému období.

4.1.2 Vylepšení clusteru

V tomto případě jsou dvě možnosti jak nějaký obnos peněz investovat do vylepšení výpočetního clusteru. Buď pořízením nových uzlů s určitými prostředky, nebo vyřazením starých nepoužívaných prostředků a jejich nahrazením za nové prostředky.

4.1.2.1 Koupě nového uzlu s určitými parametry

Popis případu užití

Správce obdržel sumu peněz a chystá se jí utratit na vylepšení clusteru. Zjistí, které prostředky chybí uživatelům nebo které prostředky jsou málo dostupné. Pořídí nové uzly s těmito prostředky a zařadí je do clusteru.

4.1.2.2 Vylepšení clusteru výměnou starých prostředků ve stávajících uzlech

Popis případu užití

Správce obdržel určitý obnos peněz na vylepšení clusteru a rozhodl se vyměnit prostředky ve starých uzlech za lepší. Zjistí, které prostředky se nejméně využívají a které prostředky uživatelům nestačí. Nakoupí nové díly a vymění je za ty, co se nejméně používaly ve starých uzlech.

Analýza těchto případů užití

Bude potřeba zjistit data o využití daných prostředků. To znamená, jak jsou využívány například nějaké typy grafických karet, jak jsou využívány dané typy procesorů, zdali stačí kapacita operační paměti apod.

Zobrazení těchto případů užití

Oba tyto případy budou zobrazeny pomocí grafů, kde budou tato data dobře vidět. Ke každému uzlu bude vytvořen graf, kde bude v čase promítnuto využití jeho prostředků (CPU, paměť, počet úloh).

4.1.3 Čekající úlohy

Popis případu užití

Správce potřebuje zjistit, které úlohy čekají na zpracování a proč, případně kolik jich je. Správce zjistí důvod a se získanými informacemi se pokusí problém vyřešit.

Analýza případu užití

V tomto případě je nutné zjistit úlohy odeslané ke zpracování a jejich stav. Stav značí, jestli je úloha prováděna nebo čeká na zpracování. Bude také potřeba zjistit, který uživatel tuto úlohu spustil, aby ho mohl správce případně kontaktovat. Úlohy mohou v clusteru čekat hned z několika důvodů. Například pokud jsou všechny sloty pro provádění úloh obsazeny nebo pokud cluster nemá k dispozici prostředky, které úloha k provedení potřebuje.

Zobrazení případu užití

Tento případ užití bude zobrazen tabulkou. V prvním sloupci bude identifikátor úlohy, v dalším sloupci bude uživatel, který úlohu spustil. V dalších sloupcích se zobrazí základní informace o úloze a důvod, proč není prováděna.

4.1.4 Využití celého clusteru

Popis případu užití

Správce potřebuje zjistit využití clusteru za nějaké období. Účelem případu užití může být, že správce naplánuje údržbu clusteru v nejméně využívaném období.

Analýza případu užití

Bude potřeba zjistit, kolik bylo volných a kolik obsazených slotů v daném období. Kolik úloh bylo prováděno a kolik jich v tomto období bylo čekajících na zpracování.

Zobrazení případu užití

Tento případ užití bude zobrazen sloupcovými grafy. Jeden graf pro využití uzlů (slotů) a druhý pro čekající úlohy. Na ose x bude časový údaj a na ose y hodnota. Grafy bude pro lepší orientaci možné do určité míry přibližovat.

4.1.5 Zatížení jednotlivých uzlů

Popis případu užití

Správce nebo uživatel potřebuje zjistit, jak jsou zatížené jednotlivé uzly v clusteru.

Analýza případu užití

K tomuto případu užití bude potřeba zjistit, jak byly na tomto uzlu vytíženy jeho prostředky za dané období.

Zobrazení případu užití

Zobrazení bude stejné jako u případu užití Vylepšení clusteru tedy graf s časovou osou. Zobrazí se využití paměti, disku, CPU, GPU.

4.1.6 Statistiky uživatelů

Popis případu užití

Správce si potřebuje zobrazit statistiky jednotlivých uživatelů, které chce u každého z nich vidět. Správce si vyhledá uživatele a zobrazí si jeho celkové statistiky v daném období nebo statistiky k jeho jednotlivým úlohám v daném období.

Analýza případu užití

Je nutné zjistit statistiky jednotlivých uživatelů. Zjistíme reálný čas a čas CPU potřebný k výpočtu jeho úloh, průměrný čas zpracovaní jedné úlohy, počet úloh (job), počet podúloh (task), využití GPU a další.

Zobrazení případu užití

Zobrazí se seznam uživatelů s jejich statistikami formou tabulky. Uživatel si bude moct zvolit, které statistiky, chce sledovat. Statistiky bude možné filtrovat pomocí filtrů.

4.1.7 Blokující úlohy

Popis případu užití

Správce potřebuje zjistit, které úlohy využívají dané prostředky a brání v dalším využití clusteru ostatním uživatelům. Správce z nástroje dostane seznam těchto úloh a prostředků, jaké blokují. V případě, že úloha blokuje mnoho prostředků, správce kontaktuje uživatele, který tuto úlohu spustil. Společně potom rozhodnou, co se bude s úlohou dále dělat.

Analýza případu užití

Bude nutné zobrazit úlohy, které blokují zadané prostředky. U těchto úloh se zjistí, který uživatel je odeslal ke zpracování a jaké prostředky úloha skutečně zabírá.

Zobrazení případu užití

Seznam úloh zobrazený v tabulce, kde budou informace o úloze, uživateli a údaje o využitých prostředcích.

4.1.8 Plýtvání prostředky

Popis případu užití

Správce prochází vykonané úlohy a zjišťuje, že úloha má alokovaných více prostředků než doopravdy využila. Z těchto údajů poté může kontaktovat uživatele, aby si dal větší práci s výběrem prostředků, které úloha opravdu potřebuje, a výpočetní cluster se tím lépe využil.

Analýza případu užití

K tomuto případu užití bude potřeba zjistit, které potřebné prostředky uživatel zadal ke zpracování úlohy a které prostředky úloha doopravdy využila. Tyto údaje poté porovnáme.

Zobrazení případu užití

Případ užití bude zobrazen formou tabulky. V tabulce budou alokované prostředky a skutečně využité prostředky. Úlohy, které mají alokováno více prostředků, než potřebují, budou barevně odlišeny pro lepší orientaci v tabulce. Dobře alokované úlohy budou zobrazeny zeleně a špatně alokované budou zobrazeny červeně.

4.1.9 Neběží úloha

Popis případu užití

Uživatel kontaktuje správce, že se mu nepovedlo spustit úlohu. Správce zjišťuje, proč tomu tak je. Zjistí, jaké prostředky úloha ke spuštění potřebuje a další podrobnosti. Se získanými výsledky poté kontaktuje uživatele.

Analýza případu užití

Bude potřeba vyhledat úlohu podle uživatele nebo jejího identifikátoru v čekajících úlohách. O této úloze se poté zjistí další podrobnosti jako například prostředky, které potřebuje ke zpracování. To, že se úloha nezpracovává, může být z několika důvodů. Buď potřebnými prostředky cluster nedisponuje, nebo je úloha ještě ve frontě ke zpracování, protože čeká na volný slot pro zpracování, nebo došlo k dalším chybám.

Zobrazení případu užití

Správci bude zobrazeno vyhledávací pole, podle kterého si bude schopen zobrazit konkrétní úlohu. Správci bude zobrazena tabulka informací o úloze. Tabulka bude obsahovat identifikátor úlohy, uživatele a další informace o ní (například: potřebné prostředky).

4.1.10 Efektivní využití prostředků

Popis případu užití

Uživatel si chce zobrazit údaje o specifické úloze. Kolik zabrala prostředků (GPU, CPU, RAM). Další úlohy poté spouští se zjištěnými parametry a minimalizuje tak zabrané prostředky.

Analýza případu užití

Opět bude potřeba zobrazit konkrétní úlohu a k ní zjistit skutečně zabrané prostředky.

Zobrazení případu užití

Uživatel vyhledá požadovanou úlohu pomocí vyhledávacího pole. Jedno pole pro vytřídění úloh podle uživatele a další bude třídit podle identifikátoru úlohy. Úlohy budou zobrazeny formou tabulky.

4.2 Návrh uživatelského rozhraní nástroje

Z výše uvedených případů užití bylo zjištěno, že se uživatelé nástroje budou zajímat o informace o výpočetním clusteru jako celku, o uživatelích, kteří tento cluster využívají, o úlohách, které jsou na clusteru prováděny a o prostředcích, kterými cluster disponuje, zejména jejich využitím.

Proto bylo rozhodnuto vytvořit základní menu tak, že bude obsahovat položky cluster, uživatelé, úlohy a prostředky. Toto budou čtyři základní pohledy, kde se uživatelé dozví nejdůležitější informace. Každá obrazovka bude obsahovat tlačítko s nastavením, kde bude uživateli umožněno upravit, jaká data se mají na obrazovce zobrazit. Umístění prvků v těchto obrazovkách bude probráno v následujících kapitolách.

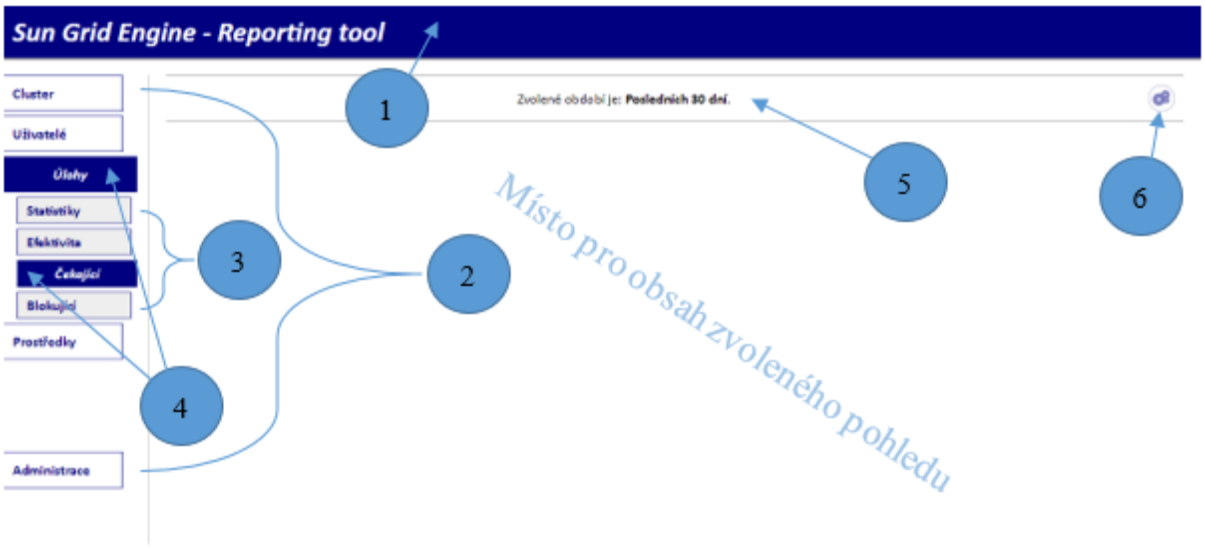
V menu bude umístěna také položka administrace, kam bude umožněn přístup pouze správci pod heslem. Správce zde bude moci konfigurovat některé vlastnosti nástroje.

Celkový návrh reportovacího nástroje viz Příloha A – návrh reportovacího nástroje.

4.2.1 Základní pohled na nástroj

Reportovací nástroj byl navržen tak, aby uživateli poskytoval přehledné uživatelské rozhraní a usnadňoval mu práci, viz obrázek 2 - základní pohled na nástroj.

1) Hlavička
2) Menu
3) Podmenu
4) Aktivní položka menu
5) Informační panel
6) Ikona s nastavením pohledu



Obrázek 2 - základní pohled na nástroj

5 Realizace reportovacího nástroje

V této kapitole je rozebráno, jaká data byla při realizaci nástroje k dispozici. Jsou zde popsány techniky, které byly použity pro získání potřebných dat a jakým způsobem tyto techniky omezily nástroj. Dále jsou zde popsány realizace jednotlivých případů užití.

5.1 Analýza dat

5.1.1 Qstat

Qstat [8] je nástroj technologie Grid Engine, který poskytuje informace o aktuálním stavu clusteru. Nástroj se spustí pomocí terminálu příkazem *qstat*. Je možno zadat další parametry, které vybírají pouze některá data z tohoto nástroje. Tento nástroj je klient, který se dotazuje master uzlu. Každý takový dotaz tedy zatěžuje hlavní plánovač, proto není vhodné ho vyvolávat příliš často.

Pro reportovací nástroj byly využity příkazy *qstat –F* pro detailnější výpis aktuálního stavu clusteru, *qstat –s psz* pro výpis čekajících úloh ve frontách, *qstat –j id_ulohy* pro zjištění detailnějších informacích o úloze a *qstat –r resource_list* pro zjištění úloh, které využívají zadaný seznam prostředků. Výpis tohoto nástroje bylo nutné zpracovat, aby z něho byla získána data v takové formě, v jaké jsou požadována.

K získání těchto údajů je tedy vždy nutné vyvolat příkaz, data zpracovat a uložit je. O tuto činnost se stará Shell skript, který každý definovaný interval tento příkaz vyvolá a data předá ke zpracování. Tyto definované intervaly budou dále v textu specifikovány u realizací konkrétních případů užití.

Z tohoto nástroje byla získána data o jednotlivých uzlech (počet běžících úloh, využití CPU, volná paměť, volné místo na disku, využité GPU), čekající úlohy ve frontách, využité a volné sloty clusteru a využití globálních prostředků jako jsou souborové servery a grafické karty.

5.1.2 Accounting

Accounting [9] je nástroj technologie Grid Engine, který poskytuje informace o dokončených úlohách. Dokončených úloh je v clusteru obrovské množství, jedná se tedy o obrovské množství dat.

Tento nástroj funguje tím způsobem, že s každou dokončenou úlohou respektive podúlohou (taskem) zapíše její vlastnosti do souboru *accounting*. Tento soubor je po dosažení určitého časového intervalu zkomprimován a uložen. Další data se pak ukládají do nového souboru.

Soubory *accounting* mají po dekomprimaci řádově jednotky gigabajtů a práce s nimi je velice náročná. Zpracování všech těchto souborů vyžaduje velké množství času, proto jsou tato data zpracována na požadavek správce zhruba jednou měsíčně a to tak, že se do databáze kumulují, tak aby se nezpracovávala data, která už byla dříve zpracována. Uživatelům je tedy vždy k dispozici náhled na jejich statistiky s maximálně měsíčním zpožděním.

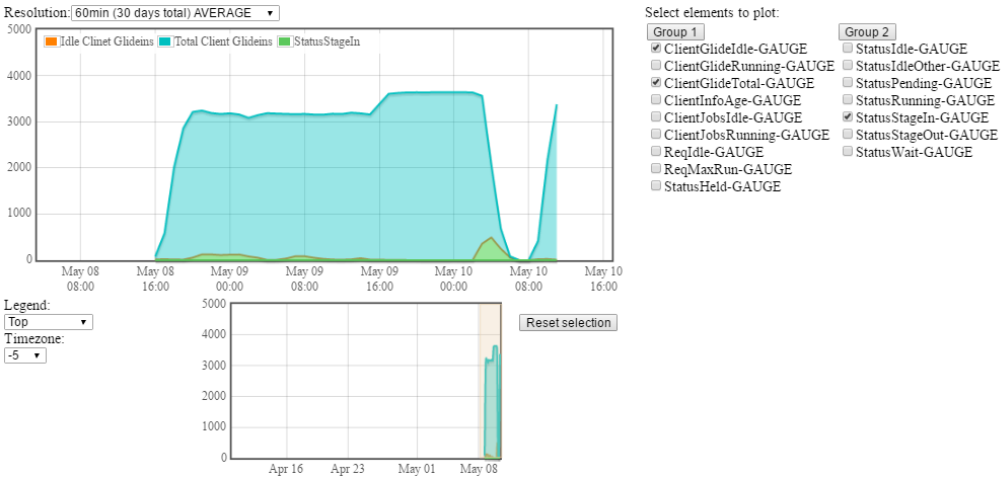
Z *accounting* souborů byla použita data, jako jsou například reálný čas úlohy, čas běhu úlohy na CPU, alokované prostředky a využité prostředky, čas kdy byla úloha spuštěna, který uživatel ji spustil atd. Dále jsou některé hodnoty dopočítány jako například průměrné časy úloh a podúloh (tasků), efektivita úloh a spotřeba.

5.2 Úprava knihovny JavascriptRRD

K zobrazení RRD souborů byla využita knihovna JavascriptRRD. Knihovna sama o sobě pracuje velmi spolehlivě, ale pro reportovací nástroj nevyhovoval její původní vzhled a její funkce na zobrazení více souborů v jednom grafu. Z těchto důvodů bylo nutné knihovnu upravit.

Knihovna standardně nabízí dva vzhledy. Buď vzhled s nastavením (viz obrázek 3 - původní vzhled s nastavením knihovny JavascriptRRD), nebo vzhled bez nastavení, který zobrazuje pouze graf. Ani jedno řešení nebylo vhodné. Nastavení grafu zabíralo velkou část prostoru, proto byl zvolen vzhled bez nastavení a nastavení bylo implementováno odděleně. Ke grafu bylo přiděleno pouze tlačítko „reset zoom“, které funguje lokálně pro jeden graf. Jelikož ve většině případů reportovacího nástroje je na jedné stránce zobrazeno více grafů zároveň, nastavení bylo naimplementované globálně pro všechny grafy na stránce a ne pro každý graf zvlášť.

Při implementaci grafů, které se skládají z více RRD souborů bylo nutné implementovat vlastní funkci na zobrazení těchto grafů. Knihovna v tomto modulu standardně datové řady sčítala (vrstvila na sebe) a zobrazovala špatnou časovou zónu, což bylo v našem případě nežádoucí. Vlastní funkce byla vytvořena na základě původní funkce tak, že bylo odstraněno nežádoucí sčítání datových řad a byla upravena časová osa. Tato úprava[7] byla nahrána na GitHub, kde z ní mohou čerpat další vývojáři.



Obrázek 3 - původní vzhled s nastavením knihovny JavascriptRRD

5.3 Realizace RRD souborů

Round-robin databáze byly po konzultaci s vedoucím práce, správcem výpočetního clusteru, nastaveny tak, že obsahují hned čtyři archivy. Archiv pro 5 dní, kde jsou data vkládaná každou minutu. Archiv pro 3 týdny, kde jsou data za 5 minut zprůměrována do jednoho záznamu. Archiv na 2 měsíce, kde jsou data za 30 minut zprůměrována do jednoho záznamu. A archiv na 10 let, kde jsou každé 2 hodiny zprůměrovány do jednoho záznamu.

RRD databáze pro sledování využití souborových serverů je doplněna ke každému výše zmíněnému archivu ještě o archiv, který ukládá maximální hodnotu v těchto časových úsecích, což umožňuje správci mít ještě větší přehled o jejich využití.

O aktualizaci těchto souborů se stará Shell skript, který tyto databáze plní po minutových intervalech.

[7] Úprava knihovny - https://github.com/rozekfr/javascriptRRD

5.4 Realizace MySQL databáze

Tato databáze uchovává statistická data o uživatelích a úlohách. Dále v textu bude popsáno, která data se uchovávají.

5.4.1 Tabulka pro statistiky uživatelů

Tato tabulka uchovává statistiky o uživatelích za celou dobu, po kterou je výpočetní cluster spuštěn. Jako primární klíč je v tabulce zvolen login uživatele, který je pro každého jedinečný.

Dalšími hodnotami, které se v této tabulce uchovávají, jsou počet úloh, počet podúloh (tasků), reálný čas a čas zatížení CPU, po který jeho úlohy běžely na výpočetním clusteru. Dále jsou spočítány hodnoty jako průměr využitých GPU na úlohu, průměrný čas potřebný na vykonání úlohy, průměrný čas na vykonání podúlohy (tasku), efektivita úlohy a její spotřeba.

Tato tabulka je propojena s tabulkou skupin, podle které je možno uživatele filtrovat podle skupin.

5.4.2 Tabulka pro statistiky úloh

Tato tabulka uchovává statistiky o jednotlivých úlohách za celou dobu, po kterou je výpočetní cluster spuštěn. Primární klíč v této tabulce je identifikátor úlohy, který je pro každou úlohu jedinečný.

Tabulka dále uchovává hodnoty jako uživatel, který úlohu spustil, počet podúloh (tasků), ze kterých se úloha skládala, počet GPU, která úloha využila, čas, kdy byla úloha spuštěna, reálný čas, po který úloha běžela a čas na CPU, který úloha zabrala na výpočetním clusteru při svém běhu, kolik paměti uživatel pro úlohu alokoval, kolik paměti úloha skutečně zabrala a maximum paměti, kolik úloha zabrala ve špičce. Dále jsou spočítány hodnoty, jako jsou průměrný potřebný čas na vykonání podúlohy, efektivita úlohy a její spotřeba.

Z této tabulky se vybírají data do tabulek specifických období, pokud uživatel v nástroji vybere období, ze kterého chce získat statistiky. Nově vytvořené tabulky se po dobu, než dojde k dalšímu zpracování *accounting* souboru uchovávají, protože stejná data může chtít i jiný uživatel a nemusí se tak znovu zpracovávat. Nové tabulky se tvoří z této také z důvodu, že SQL dotaz nad databázovou tabulkou je zpravidla rychlejší, než lineární zpracování tak velkých souborů jako jsou soubory *accounting*.

5.4.3 Tabulka pro čekající úlohy

Tato tabulka uchovává čekající úlohy ve frontách clusteru. Jako primární klíč je opět zvolen identifikátor úlohy.

Dále se v tabulce ukládají data o uživateli, kterému tato úloh patří, čas, kdy byla úloha odeslána ke zpracování, stav úlohy, který vyjadřuje, zdali úloha čeká na zpracování nebo byla z nějakého důvodu pozastavena a zařazena do fronty. V této tabulce je také uchováno, kolik podúloh (tasků) dané úlohy čeká ve frontě.

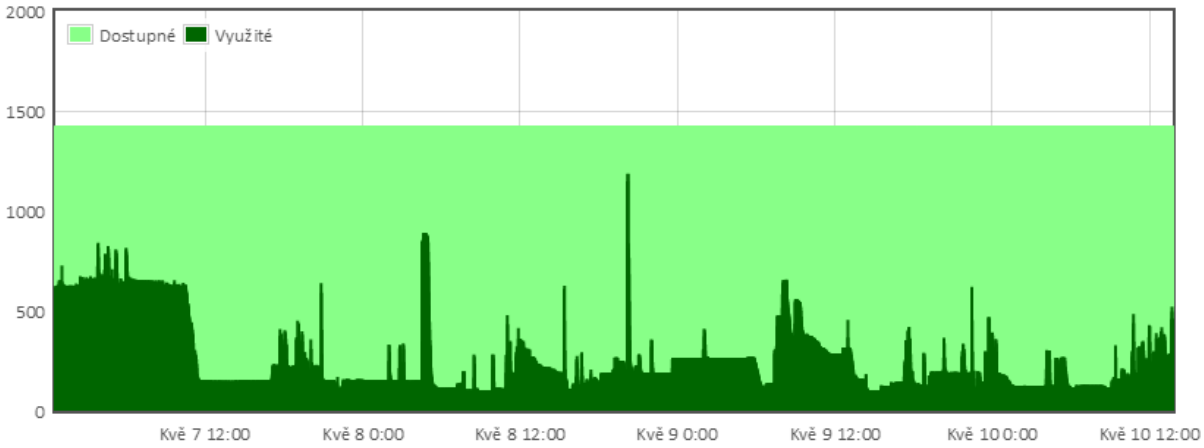
5.4.4 Další tabulky

V databázi jsou ještě další tabulky, kde jsou uchovány skupiny uživatelů a nastavení uzlů, co se týče jejich konfigurace a spotřeby.

5.5 Realizace pohledu na cluster

Tento pohled pokrývá případ užití Využití celého clusteru. V tomto pohledu jsou uživateli reportovacího nástroje poskytnuty dva grafy s časovou osou. Jeden pro využití slotů clusteru, kde jsou barevně odděleny využité sloty a dostupné sloty, viz obrázek 4 - pohled na cluster (sloty). Druhý graf zobrazuje využití clusteru z pohledu prováděných úloh a úloh, které zatím čekají ve frontách ke zpracování. Oba tyto údaje jsou barevně odděleny.

V nastavení si uživatel na této stránce může vybrat, jaké období a které datové řady chce v grafech zobrazit.



Obrázek 4 - pohled na cluster (sloty)

5.6 Realizace pohledu na uživatele

Tento pohled pokrývá případy užití Zpoplatnění služeb clusteru a Statistiky uživatelů. Uživateli reportovacího nástroje je zobrazena interaktivní tabulka uživatelů, viz obrázek 5 - interaktivní tabulka uživatelů.

V nastavení si uživatel může zvolit, které sloupce, chce na stránce zobrazit a z kterého období záznamy požaduje. Jsou dvě možnosti jak zvolit období, a to buď výběrem z předem definovaného seznamu období, nebo zvolením konkrétního období pomocí dvou zadaných dat a časů.

Nad tabulkou je implementováno stránkování, kde si uživatel může vybrat, kolik záznamů chce na stránku vykreslit a kterou stránku chce zobrazit. Vedle stránkování je umístěno tlačítko pro aplikaci filtrů, které vybere z databáze pouze vyhovující záznamy. Více o filtrech viz Implementace filtrů. Podle každého zobrazeného sloupce lze tabulku seřadit vzestupně, či sestupně.

Každá manipulace s tabulkou jako řazení, filtrace, výběr stránky či změna období vyvolá událost, kterou zpracovává AJAX funkce, která ze zadaných parametrů provede SQL dotaz nad příslušnou tabulkou databáze a vrátí vygenerovanou HTML tabulku na základě výsledku daného SQL dotazu.

Počet položek: 5 Stránka: 1 Filtry: aplikovat filtry

Uživatel	Skupina	Počet úloh	Počet tasků	Reálný čas	CPU čas
baskar	-	85	986	0r 6d 23h 16m 58s	0r 7d 6h 45m 20s
grezl	-	4	4	0r 11d 3h 39m 19s	0r 10d 20h 53m 12s
iegorova	-	9	9	0r 5d 5h 13m 52s	0r 5d 5h 14m 55s
imrazek	-	292	1 520	0r 86d 9h 39m 17s	0r 85d 22h 2m 11s
inovoton	-	5	5	0r 1d 12h 19m 22s	0r 0d 1h 29m 28s

Obrázek 5 - interaktivní tabulka uživatelů

5.6.1 Implementace filtrů

Filtry u interaktivních tabulek slouží k výběru jen určitých dat z databázové tabulky. U každého zobrazeného sloupce je vytvořeno pole pro vložení filtru. Formát zápisu filtru do pole je specifický, viz dále v textu. Pokud filtr nesplňuje povolený formát, tak je uživateli zvýrazněn červeně a dotaz do databáze se nevykoná. Uživatel už při psaní filtru s každou stisknutou klávesou vidí, zdali je filtr zapsán správně, či nikoliv.

Význam použitých zkratek:

P_O – porovnávací operátor
L_O – logický operátor

Povolené formáty filtrů:

1) seznam
2) P_O hodnota
3) P_O hodnota L_O P_O hodnota

Hodnoty ve filtru

Pokud se jedná o celočíselnou hodnotu nebo textový řetězec, může uživatel hodnotu zapsat bez žádných potřebných úprav.

Pokud jde o desetinnou hodnotu, může ji uživatel zapsat jak pomocí desetinné čárky, tak desetinné tečky.

Porovnání časového údaje ve formátu *x*r *x*d *x*h *x*m *x*s, kde *x* je celočíselné číslo, *r* je počet roků, *d* je počet dní, *h* je počet hodin, *m* je počet minut a *s* je počet vteřin, si vyžaduje menší úpravu. Tento údaj uživatel musí zadat bez mezer. Zadává tedy například takto: *10r5d2h0m0s* nebo zkráceně *10r5d2h*. Tato hodnota je potom převedena na sekundy a dále v databázi porovnávána.

Stejné platí i pro hodnoty paměti, kde uživatel zadává jednotky opět bez mezery. Tedy například *100MB*, *4GB* a *6kB*. Tyto hodnoty jsou převáděny na MB a následně porovnány v databázi.

Porovnávací operátory

Uživatel může použít znaménka z následující tabulky:

Znaménko	Význam
<	menší než
>	větší než
<=	menší nebo rovno než
>=	větší nebo rovno než
=	rovno
!=	různé od (nerovno)

Tabulka 1 - přehled použitelných znamének ve filtrech

Logické operátory

Jako logické spojky mezi jednotlivými podmínkami může uživatel použít logické spojky AND nebo OR. Psány velkými písmeny, jak je uvedeno.

1) Formát filtru seznam

Tento filtr umožňuje zadat jednu nebo více hodnot. Pokud uživatel chce zadat více hodnot, musí je oddělit pomocí čárek. Takovýto filtr je pak transformován do SQL následovně: *sloupec IN (seznam).*

Příklady zápisu filtru:

Zápis filtru	SQL transformace
xrozek01	sloupec IN (‘xrozek01‘)
xrozek01, xlogin00	sloupec IN (‘xrozek01‘,‘xlogin00‘)
xrozek01,xlogin00 *(bez mezer)*	sloupec IN (‘xrozek01‘,‘xlogin00‘)

Tabulka 2 - příklady zápisu filtru seznam

2) Formát filtru P_O hodnota

Tento filtr umožňuje porovnat sloupce vzhledem k nějaké hodnotě. Je nutné, aby ve filtru byla mezera mezi operátorem a hodnotou. Takovýto filtr se poté transformuje do SQL tímto způsobem: *sloupec P_O hodnota.*

Příklady zápisu filtru:

Zápis filtru	SQL transformace
< 5	sloupec < 5
>= 1r10d6h	sloupec >= 32421600
= xrozek01	sloupec = ‘xrozek01‘

Tabulka 3 - příklady zápisu filtru P_Z hodnota

3) Formát filtru P_O hodnota L_O P_O hodnota

Tento filtr umožňuje porovnat sloupce všeobecně na více hodnot, nejen pouze na dvě. Je nutné, aby vždy mezi P_O a hodnotou byla mezera a L_O byl z obou stran ohraničen mezerami a zapsán velkými písmeny.

Příklady zápisu filtru:

Zápis filtru	SQL transformace
> 1 AND < 5	sloupec > 1 AND sloupec < 5
= xrozek01 OR = xlogin00 OR = xlogin01	sloupec = ‘xrozek01‘ OR sloupec = ‘xlogin00‘ OR sloupec = ‘xlogin01‘

Tabulka 4 - příklad zápisu filtru P_Z hodnota L_O P_Z hodnota

5.7 Realizace pohledu na úlohy

Tento pohled byl rozdělený na další pohledy, kvůli odlišným možnostem, jak budou uživatelé na úlohy nahlížet. Toho bylo docíleno vytvořením podmenu. Podmenu je tvořeno položkami Statistiky, Efektivita, Čekající a Blokující. Položka statistiky tvoří pohled na různé statistiky úloh. Položka Efektivita se zabývá efektivitou úloh z hlediska výpočtu i z hlediska efektivního využití paměti. Položka Čekající zobrazuje úlohy, které jsou v clusteru zařazeny ve frontách, tedy nejsou prováděny. Položka Blokující zobrazuje úlohy, které blokují uživatelem zvolené prostředky.

5.7.1 Statistiky úloh

Pohled na statistiky úloh realizuje část případu užití Statistiky uživatelů je implementován stejným způsobem jako pohled na statistiky uživatelů ale nad jinou tabulkou databáze. Opět umožňuje filtraci, řazení a stránkování záznamů, viz obrázek 6 - pohled na statistiky úloh. V nastavení lze také volit sloupce a období, které chceme zobrazit.

Počet položek: 5 Stránka: 1 Filtry: aplikovat filtry

ID úlohy	Uživatel	Reálný čas	CPU čas	Počet tasků	Alokovaná paměť
4526370	ipesan	0r 14d 0h 0m 1s	0r 0d 0h 0m 34s	1	4 GB
4539916	grezl	0r 11d 1h 9m 25s	0r 10d 18h 36m 34s	1	10 GB
4586878	inovoton	0r 1d 7h 25m 15s	0r 0d 0h 8m 2s	1	10 GB
4621943	baskar	0r 0d 18h 10m 33s	0r 0d 18h 5m 45s	1	-
4626434	isilnova	0r 1d 14h 48m 26s	0r 8d 23h 6m 14s	229	4 GB

Obrázek 6 - pohled na statistiky úloh

Výpočet spotřeby

Pro statistiky úloh je mimo jiných hodnot, které se počítají standardním způsobem, například: průměrný čas podúlohy (tasku), vypočítána spotřeba úlohy následujícím způsobem.

Spotřebu jednotlivých uzlů zadává do nástroje správce - viz Administrace. Spotřeba úlohy se liší tím, jaké prostředky využívá. Pokud zatěžuje pouze CPU a nepotřebuje ke své práci například GPU nebo některý ze souborových serverů, pak je spotřeba úlohy značně menší než kdyby je využívala. Pro výpočet spotřeby se tedy využijí následující rovnice.

$$Spotřeba\ CPU = příkon\ CPU \times CPU\ čas \quad (1)$$

$$Spotřeba\ GPU = příkon\ GPU \times reálný\ čas \quad (2)$$

$$Spotřeba\ souborových\ serverů = příkon\ serveru \times míra\ využití \times reálný\ čas \quad (3)$$

$$Spotřeba = Spotřeba\ CPU + Spotřeba\ GPU + Spotřeba\ souborových\ serverů \quad (4)$$

Z výše uvedeného plyne, že výpočet spotřeby je pouze orientační. Nemáme totiž k dispozici žádné informace o spotřebě celého clusteru, ani jednotlivých uzlů. Pomocí zmíněného výpočtu lze určit, které úlohy, respektive kteří uživatelé spotřebovali více elektrické energie než ostatní.

5.7.2 Efektivita úloh

Pohled na efektivitu úloh realizuje případy užití Plýtvání prostředky a Efektivní využití prostředků. Poskytuje porovnání jak dobře, který uživatel využívá výpočetní cluster. Ke každé úloze je zobrazena efektivita výpočtu a efektivita alokace paměti. Ostatní data k porovnání nebyla k dispozici.

V reportovacím nástroji je efektivita uživatelům zobrazena barevně, a to následovně: zeleně, pokud je úloha v daném ohledu efektivní, oranžově, pokud je méně efektivní a červeně neefektivní, viz obrázek 7 - pohled na efektivitu úloh.

ID úlohy	Uživatel	Efektivita výpočtu	Alokace paměti
4526370	ipesan	0,00%	4 GB (využito: 181 MB, max: 245 MB)
4539916	grezl	97,53%	10 GB (využito: 5 GB, max: 5 GB)
4586878	inovoton	0,43%	10 GB (využito: 2 GB, max: 3 GB)
4621943	baskar	99,56%	-
4626434	isilnova	554,29%	4 GB (využito: 3 GB, max: 4 GB)

Obrázek 7 - pohled na efektivitu úloh

5.7.2.1 Efektivita výpočtu úlohy

Efektivita výpočtu je vypočítána z reálného času, po který úloha běžela, a času, po který tato úloha zatěžovala CPU. Čím déle úloha při svém běhu zatěžuje CPU, tím je efektivnější. V opačném případě čeká na nějaké zdroje a procesor nepracuje. Hodnota efektivity je zobrazena na dvě desetinná místa.

Reálný čas, po který úloha běžela, je získán z *accounting* souboru a je uveden v celých sekundách. Čas běhu úlohy na CPU je desetinné číslo s přesností na 6 desetinných míst. Z důvodu, že reálný čas není přesný, ale je zaokrouhlen na celé číslo, je nutné počítat s chybou. V některých případech může také dojít k tomu, že je reálný čas nebo čas strávený na CPU nulový. V tomto případě není možné efektivitu úlohy spočítat.

Reportovací nástroj považuje efektivitu výpočtu za efektivní, pokud je v rozmezí 50-110 % kvůli možné chybě. Pokud je efektivita menší než 50 %, je úloha označena jako neefektivní, pokud je efektivita větší než 110 %, je u takovéto úlohy podezření na špatnou specifikaci úlohy - označeno oranžově.

Výpočet:

$$efektivita = \frac{čas\ na\ CPU}{reálný\ čas} \times 100 \tag{5}$$

Příklad:

Z *accounting* souboru byla vybrána úloha, na které bude výpočet demonstrován.

$Získané\ hodnoty$:
$čas\ na\ CPU = 200{,}861000$
$reálný\ čas = 321$

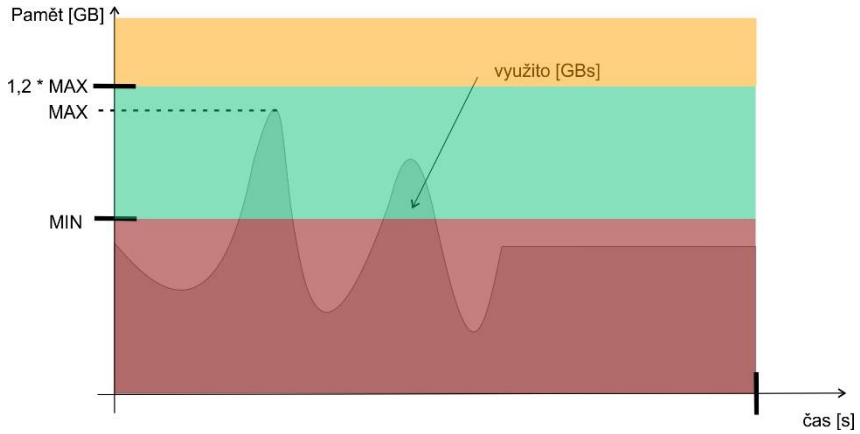
$$efektivita = \frac{200{,}86100,}{321} \times 100 \doteq \underline{\underline{62{,}57\%}}$$

5.7.2.2 Efektivita alokace paměti

Accounting soubor poskytuje údaje o tom, kolik uživatel alokoval místa v paměti pro úlohu, kolik úloha spotřebovala paměti, což je dáno integrálem využité paměti v GBs (gigabajt CPU sekundách), a maximální hodnotu zabrané paměti v kB - MAX. Tyto hodnoty jsou převedeny na stejnou jednotku. Dalším potřebným parametrem je čas, po který úloha běžela na CPU.

Způsob usouzení, že je alokace efektivní lze vidět na obrázku 8 - efektivita alokace paměti. V červené oblasti je efektivita alokace považována za nevhodnou, protože má alokováno málo paměti a úloha tak neustále čeká na její uvolnění. V oranžové oblasti uživatel alokoval příliš mnoho paměti, kterou úloha nevyužívá a tak plýtvá prostředky výpočetního clusteru. V zelené oblasti je alokace v pořádku. Hodnota *MIN* na obrázku se spočítá pomocí následující rovnice. V reportovacím nástroji se bere v úvahu 20% tolerance alokace paměti, proto je zelená oblast od *MIN* do *1,2 × MAX*.

$$MIN = \frac{využitá\ paměť}{doba\ běhu\ úlohy\ na\ CPU} \tag{6}$$



Obrázek 8 - efektivita alokace paměti

Příklad

Ze souboru *accounting* byla vybrána úloha pro demonstraci výpočtu.

$doba\ běhu\ úlohy\ na\ CPU = 774\ 374\ s$

$využitá\ paměť = \ 2\ 705\ 573{,}061\ GBs$

$maximální\ využitá\ paměť = 3{,}7\ GB$

$alokovaná\ paměť = 4\ GB$

Z těchto hodnot se vypočítá hodnota MIN.

$$MIN = \frac{využitá\ paměť}{doba\ běhu\ úlohy\ na\ CPU} = \frac{2\ 705\ 573{,}061}{774\ 374} = \underline{\underline{3{,}49\ GB}}$$

Dále se určí hodnota MAX s 20% tolerancí alokace paměti.

$$1{,}2 \times MAX = maximální\ využitá\ paměť \times 1{,}2 = 3{,}7 \times 1{,}2 = \underline{\underline{4{,}44\ GB}}$$

Alokovaná hodnota paměti je 4 GB. Tato hodnota leží mezi hodnotami MIN a 1,2×MAX. Z toho vyplývá, že v tabulce efektivity alokace paměti bude zobrazena zelenou barvou.

5.7.3 Čekající úlohy

Pohled na čekající úlohy realizuje případ užití Čekající úlohy a je implementován pomocí několika skriptů a funkcí.

Byl vytvořen Shell skript, který se každou minutu dotazuje výpočetního clusteru na čekající úlohy. Zjištěný výsledek uloží do souboru.

Pokud si uživatel chce zobrazit čekající úlohy, tak pokud je potřeba aktualizovat data v databázi o čekajících úlohách, spustí se skript, který zpracuje vytvořený soubor a data uloží do MySQL databáze, aby mohla být řazena, filtrována a stránkována.

Data jsou na stránce zobrazena stejným způsobem jako v případě statistik úloh či uživatelů. Existuje tedy AJAX funkce, která se stará o načtení dat z databáze podle požadavků uživatele. Je zde navíc sloupec, který umožňuje zjistit o úloze více informací, viz obrázek 9 - pohled na čekající úlohy - poslední sloupec.

Počet položek: 5 Stránka: 1 Filtry: aplikovat filtry

ID úlohy	Uživatel	Čas odeslání	Stav	Počet tasků	Další informace
5064840	ibrejcha	2016-04-18 18:10:32	qw	119	zjistit
5064841	ibrejcha	2016-04-18 18:10:32	hqw	122	zjistit
5604001	karafiat	2016-04-27 22:24:58	qw	1	zjistit
5604343	karafiat	2016-04-27 22:30:21	qw	1	zjistit
5604477	karafiat	2016-04-27 22:32:32	qw	1	zjistit

Obrázek 9 - pohled na čekající úlohy

Zjištění více informací o úloze

Tento úkon je realizován Shell skriptem, který periodicky po 15 sekundách smaže staré vygenerované soubory s informacemi o úlohách, provede vygenerovaný soubor žádostí, které uživatelé odeslali (vytvoří se soubory s odpověďmi na žádosti o informace o daných úlohách) a vyčistí soubor žádostí.

Uživatel z aplikace stiskne tlačítko ke zjištění informací o úloze. Pomocí AJAX funkce se vygeneruje do souboru se žádostmi jeho žádost. Dále uživatel čeká, dokud se neprovede výše zmíněný skript, který vytvoří soubor s odpovědí na jeho žádost. Po vytvoření žádosti o informace o úloze se spustí další AJAX funkce, která každou sekundu kontroluje, zdali už existuje soubor s odpovědí na žádost. V čase, kdy existuje, zpracuje data ze souboru a zobrazí je uživateli, viz obrázek 10 - výřez z informací o úloze.

Informace o úloze 5657939

job_number	5657939
exec_file	job_scripts/5657939
submission_time	Fri Apr 29 16
owner	ibrejcha
uid	30141
group	fit
gid	201
sge_o_home	/homes/kazi/ibrejcha
sge_o_log_name	ibrejcha

Obrázek 10 - výřez z informací o úloze

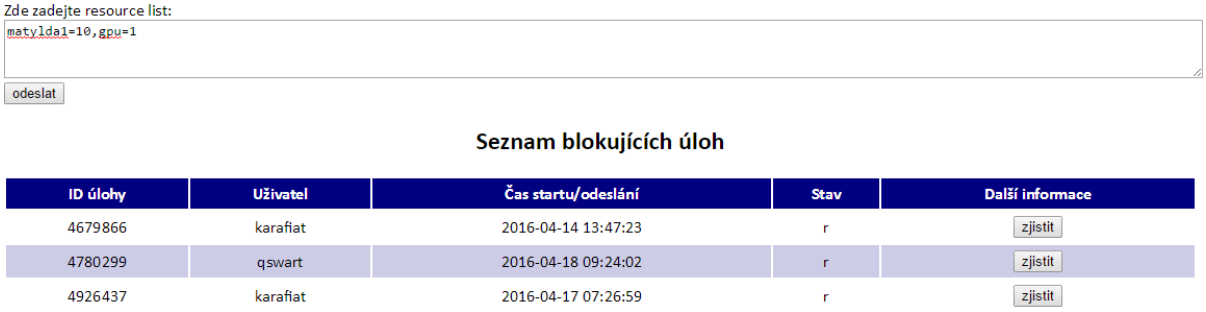
5.7.4 Blokující úlohy

Tento pohled realizuje případ užití Blokující úlohy a poskytuje uživateli textové pole pro zadání prostředků, u kterých chceme zjistit, kterými úlohami jsou blokovány.

Stejně jako u čekajících úloh je zde Shell skript, který se spouští periodicky každých 15 sekund a vyřizuje vygenerované žádosti na zjištění, které úlohy zabírají uživatelem zadané prostředky.

Pokud uživatel vyplní pole prostředků a odešle žádost, čeká, až se vytvoří soubor s jeho odpovědí. Existenci souboru opět testuje AJAX funkce každou vteřinu. Po vytvoření souboru jsou data z tohoto souboru zpracována a zobrazena formou tabulky, viz obrázek 11 - pohled na blokující úlohy.

Poslední sloupec slouží opět ke zjištění detailnějších informací o úloze, které je implementováno stejně jako u čekajících úloh.



Obrázek 11 - pohled na blokující úlohy

5.8 Realizace pohledu na prostředky

Tento pohled realizuje případy užití Vylepšení clusteru a Zatížení jednotlivých uzlů. Zde má uživatel přehled o tom, jak jsou jednotlivé uzly a jejich prostředky využívány.

Uživateli jsou tato data zobrazena pomocí grafů, které berou data z RRD souborů.

5.8.1 Rozdělení prostředků do kategorií

Výkonných uzlů je ve fakultním výpočetním clusteru mnoho. Proto bylo nutné tyto uzly rozdělit do 4 hlavních kategorií. Jednou takovou kategorií jsou globální zdroje, jako jsou souborové servery matylda1-6 a scratch1-6 a grafické karty. Dalšími kategoriemi byly výkonné uzly, bloky a učebny. Výkonné uzly obsahují uzly jako blade servery, gpu servery, výpočetní servery, apod. Bloky obsahují všechny počítače nacházející se v těchto blocích a učebny obsahují všechny počítače v dané učebně.

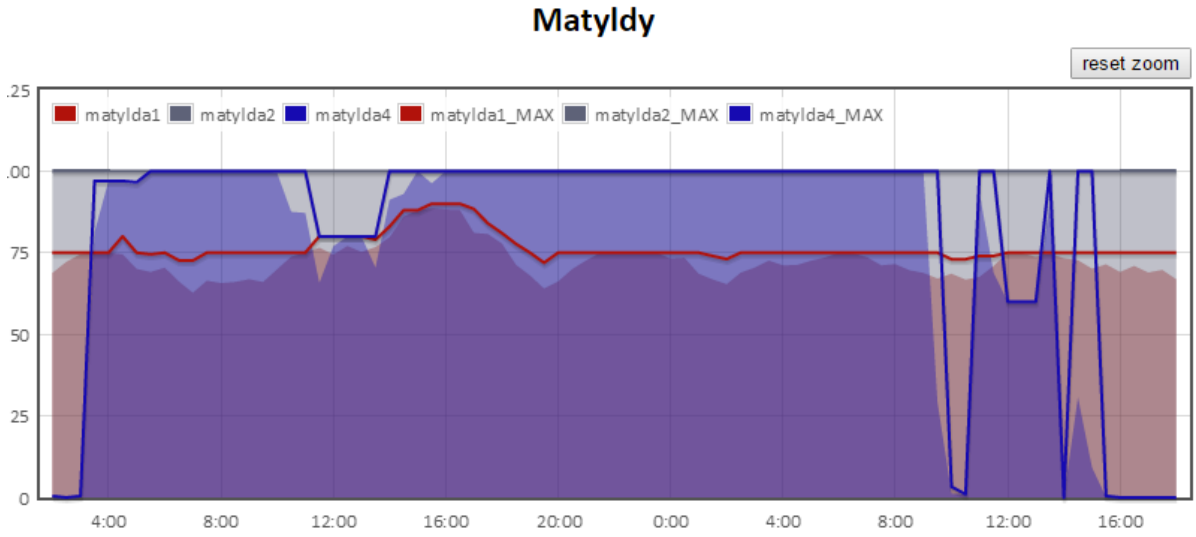
Tyto kategorie a další podkategorie byly získány pomocí nástroje *qconf*. *Qconf* je jeden z nástrojů Grid Engine. Pomocí příkazů *qconf – shgrpl* pro získání skupin a příkazu *qconf -shgrp skupina* pro získání uzlů ve skupině byl vytvořen soubor se skupinami a uzly náležícími do těchto skupin. S tímto souborem pak reportovací nástroj pracuje, podle svých potřeb.

5.8.2 Globální prostředky

Souborové servery

Tento pohled je realizován dvěma grafy s časovou osou. Jeden zobrazuje dostupnost souborových serverů matylda1-6 a druhý zobrazuje souborové servery scratch1-6. Každý tento graf je složen ze dvanácti RRD souborů – každý server má svůj soubor pro průměrné hodnoty a soubor pro maximální hodnoty v daných obdobích. Tyto hodnoty pak uživateli umožňují vidět jejich dostupnost v širším kontextu. Maximální hodnoty jsou zobrazeny čarou a průměrné hodnoty jako oblast, viz obrázek 12 - pohled na dostupnost souborových serverů.

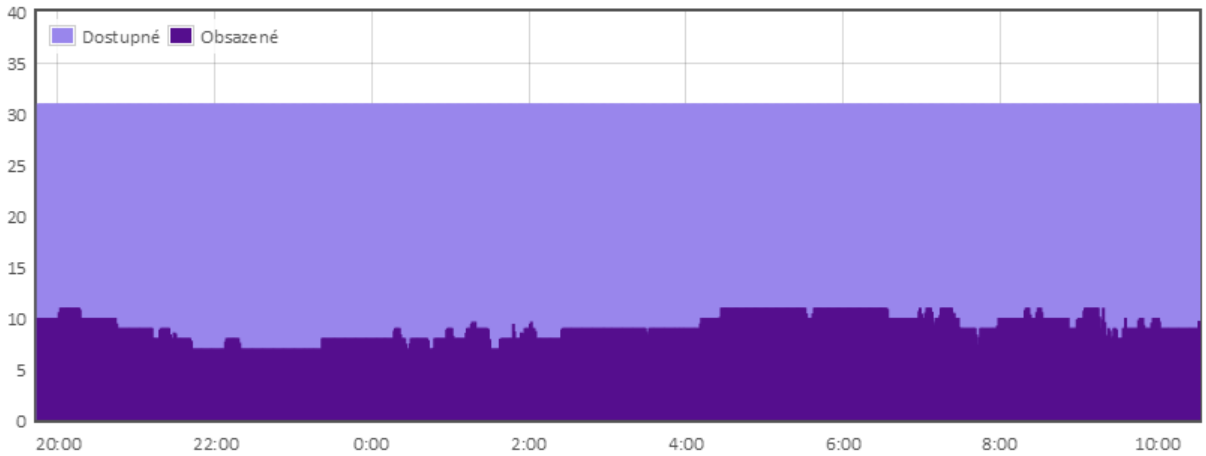
V nastavení lze vybrat, které datové řady chce uživatel v grafech zobrazit a z jakého období data požaduje.



Obrázek 12 - pohled na dostupnost souborových serverů

Grafické karty

Tento pohled nabízí uživateli přehled o tom, kolik je ve výpočetním clusteru volných grafických karet a jaké je jejich využití. Zobrazení je realizováno pomocí sloupcového grafu, viz obrázek 13 - využití grafických karet.

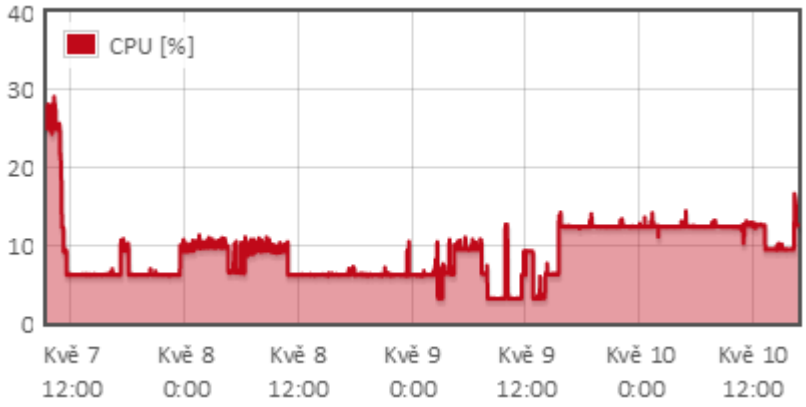


Obrázek 13 - využití grafických karet

5.8.3 Výkonné uzly

Tento pohled uživateli poskytuje možnost porovnat, které uzly jsou zatíženy více, či méně. Nabízí pohled na počet úloh, který je na uzlu prováděn, na využití jeho CPU, na využití paměti, velikost volného místa na disku a dostupnost jeho grafických karet. Bohužel ne u všech uzlů jsou všechna tato data dostupná. U počítačů v učebnách je možno sledovat pouze využití paměti. Grafické karty se mohou sledovat pouze na grafických serverech, atd.

Standardně je zobrazen počet prováděných úloh na výkonném uzlu, respektive využité a volné sloty uzlu. Další datové řady a jiná období si uživatel může změnit v nastavení, viz obrázek 14 - pohled na využití CPU výkonného uzlu.



Obrázek 14 - pohled na využití CPU výkonného uzlu

5.9 Administrace

Do administrace má přístup jen pověřená osoba – správce. Jeho ověření spočívá v autentizaci pomocí jména a hesla. V administraci je správci umožněno konfigurovat některé vlastnosti reportovacího nástroje a jsou zde také informace, co dělat pro aktualizaci nástroje (aktualizování uzlů, statistik, atd).

Vytváření skupin uživatelů

Správce má možnost každého uživatele přiřadit do několika skupin. Podle těchto skupin je pak možno uživatele filtrovat.

Každý uživatel patří do výchozí skupiny, která je získána ze souboru *etc/passwd* a nemůže být změněna. Správce si vytvoří například skupiny hříšníků, které si poté ve statistikách může vyfiltrovat.

Nastavení konfigurace uzlu

Každému uzlu se přiřadí konfigurace, kterou je poté možno u konkrétního uzlu zobrazit. Podle těchto konfigurací se správce dozví například, který typ grafických karet nebo procesorů je využit nejvíce.

Ve výpočetním clusteru existují skupiny uzlů se stejnými konfiguracemi. Zadávání konfigurací je řešeno tak, že se konfigurace zapíše pouze jednou, vyberou se uzly, kterých se daná konfigurace týká a jim se daná konfigurace přiřadí. Díky tomuto způsobu správce nemusí stále dokola psát to samé.

Nastavení spotřeby uzlu

Nastavení spotřeby se opět jako u nastavení konfigurace dá aplikovat na více uzlů současně. Správce každému uzlu zadává parametry: příkon CPU a příkon GPU. Z těchto hodnot se poté počítá spotřeba jednotlivých úloh a uživatelů.

6 Testování nástroje

Reportovací nástroj byl testován během jeho vývoje na skutečných datech přímo ze školního výpočetního clusteru.

Pro testování statistik úloh a uživatelů byla zpracována pouze menší část *accounting* souboru. Pokud by byl nástroj nasazen k reálnému použití, bylo by nutné otestovat jeho rychlost se všemi *accounting* soubory. Při případných problémech s rychlostí nástroje by bylo nutné navrhnout efektivnější způsob, jak pracovat s tak velkým množstvím dat.

Pro ostatní případy užití se pracovalo se skutečnými daty v plném rozsahu. Pokud to bylo možné, získaná data byla ověřena pomocí starého reportovacího nástroje nebo byla ověřena pomocí nástrojů Grid Engine.

Postup tvorby nástroje byl iterativní. Byly například vytvořeny skripty na zpracování souborů, ty se nejprve otestovaly, zdali hodnoty z nich získané korespondují s hodnotami v *qstat*, či *accounting*. Pokud se data jevila jako správná, pokračovalo se dále k vytvoření samotného pohledu. Vždy se vytvořil základ pohledu, poté se k němu přidávaly další funkce. Vše bylo náležitě v každém kroku testované na korespondenci s existujícími nástroji.

Všechny navržené případy užití byly splněny, implementovány a jsou funkční. Nástroj byl průběžně předváděn vedoucímu práce Ing. Tomáši Kašpárkovi.

Dále byl předán několika uživatelům k nahlédnutí a většina z nich nástroj hodnotila jako přínos pro uživatele, co s výpočetním clusterem denně pracují. Pro uživatele s menšími zkušenostmi s výpočetním clusterem, může být nástroj také zajímavý, protože většina dat je zobrazena formou grafů, kde uživatel vidí, jak je výpočetní cluster využit, ale nástroj nevyužijí naplno.

Současnou verzi reportovacího nástroje si je možno vyzkoušet na mých školních stránkách[8].

[8] Školní stránky - http://www.stud.fit.vutbr.cz/~xrozek01/BP/

7 Závěr

Cílem této práce bylo vytvořit reportovací nástroj pro školní výpočetní cluster založený na technologii Grid Engine. Práce vycházela z dosavadního řešení reportovacího nástroje, které podstatně rozšířila o některé funkce.

Zpočátku byly stanoveny případy užití reportovacího nástroje. Tyto případy užití byly zanalyzovány a následně byla získána data, která budou k jejich realizaci potřeba. V dalším kroku byly prostudovány zdroje dat a určeno, jaká data byla k realizaci nástroje k dispozici. Při samotné realizaci nástroje jsme se setkali s určitými limity použitých technologií. Byl brán ohled na konstrukci a velikost RRD databází, aby tyto databáze nebyly příliš velké pro jejich další zobrazení v nástroji. Dalším limitem byla doba zpracování accounting souboru, která trvá příliš dlouho díky obrovskému množství dat, kterým disponuje. To bylo vyřešeno tím, že toto zpracování bude spouštěno správcem zhruba po měsíčních intervalech. Po zpracování skutečných dat se přešlo k realizaci jednotlivých pohledů na využití clusteru. Během realizace těchto pohledů bylo vše náležitě testováno.

Nový reportovací nástroj umožňuje sledovat jak dlouhodobá data, tak do jisté míry aktuální data. Uživatelé mají v nástroji k dispozici pohled na využití celého clusteru, z pohledu počtu běžících a čekajících úloh, případně volných a obsazených slotů. Nástroj poskytuje náhled na statistiky uživatelů, ale i na statistiky jednotlivých úloh. Uživatel dokonce může zjistit, jak byla jeho úloha efektivní z hlediska využití procesoru a alokace paměti. V nástroji lze přímo sledovat využití globálních prostředků, jako jsou souborové servery či grafické karty. Je možno sledovat i jednotlivé komponenty výpočetního clusteru z pohledu počtu úloh na nich prováděných, využití jejich procesorů, využití jejich grafických karet, využití paměti a množství volného místa na disku. Uživatelé si mohou přímo v nástroji zobrazit úlohy, které čekají ve frontách, a zjistit si o nich podrobné informace nebo si mohou zobrazit úlohy, které blokují prostředky, které vyžaduje jejich úloha.

Nástroj je vhodný jak pro běžného uživatele výpočetního clusteru, tak pro správce, který tak má většinu důležitých dat k dispozici pouze z jediného nástroje a má tak o něm ucelený přehled. Správce navíc může nástroj konfigurovat. Může například uspořádat uživatele do skupin, měnit údaje o spotřebě uzlů nebo vkládat jejich hardwarové konfigurace.

Vytvořený nástroj uživatelům urychlí práci, poskytne jim informace, které dříve neměli k dispozici, nebo je museli shánět na různých místech či dokonce svými požadavky zatěžovali správce výpočetního clusteru. Nástroj také poskytuje informace o tom, jak celkově zefektivnit využití výpočetního clusteru.

Dalším možným rozšířením mé práce by bylo najít efektivnější metody zpracování souboru *accounting*, aby jeho zpracování nebylo tak náročné a práce s takovým objemem dat byla rychlejší. Nástroj by se dalo také rozšířit o další informace například o využití propojovací sítě, o právě běžící úlohy a další. Také by se dal rozšířit samotný cluster, který by poskytoval více informací, které bychom následně využili i v nástroji, například využití jednotlivých počítačů ve všech učebnách. Veškeré zdrojové soubory byly nahrány na server GitHub[9] pro jejich další rozvoj.

Při implementaci nástroje jsem využil své dosavadní znalosti o tvorbě webových stránek a prohloubil znalosti o UNIX/Linux systémech, ale také jsem se přiučil technologiím jako Grid Engine či Round-Robin databáze, se kterými jsem se dosud během bakalářského studia nesetkal.

[9] Zdrojové soubory - https://github.com/rozekfr/Grid-Engine-reporting-tool

Literatura

[1] ROUSE, Margaret. High-performance computing (HPC) definition. In: *TechTarget* [online]. 2007 [cit. 2016-01-09]. Dostupné z: http://searchenterpriselinux.techtarget.com/definition/high-performance-computing

[2] ROUSE, Margaret. Supercomputer. In: *TechTarget: WhatIs.com* [online]. 2008 [cit. 2016-01-09]. Dostupné z: http://whatis.techtarget.com/definition/supercomputer

[3] 10 Free Server and Network Monitoring Tools. *Gridload* [online]. Gridload, 2014 [cit. 2016-05-11]. Dostupné z: http://www.gridload.com/2013/09/10-free-server-network-monitoring-tools/

[4] Výpočetní cluster. In: *Vysoké učení technické: Fakulta informačních technoligií* [online]. 2015 [cit. 2016-01-12]. Dostupné z: https://www.fit.vutbr.cz/CVT/cluster/

[5] OETIKER, Tobias. *RRDtool: About RRDtool* [online]. Aarweg (Švýcarsko): OETIKER+PARTNER AG, 2014 [cit. 2016-04-21]. Dostupné z: http://oss.oetiker.ch/rrdtool/

[6] *JavascriptRRD: Client-side access to RRD files* [online]. San Diego (USA): Slashdot Media, 2013 [cit. 2016-04-21]. Dostupné z: http://javascriptrrd.sourceforge.net/

[7] Sun Grid Engine for Dummies. In: *Oracle* [online]. 2009 [cit. 2016-01-12]. Dostupné z: https://blogs.oracle.com/templedf/entry/sun_grid_engine_for_dummies

[8] Qstat: show the status of Sun Grid Engine jobs and queues. *Grid Engine: Open Grid Scheduler* [online]. Santa Clara, Kalifornie, USA: Oracle Corporation, 2012, 2013 [cit. 2016-04-28]. Dostupné z: http://gridscheduler.sourceforge.net/htmlman/htmlman1/qstat.html

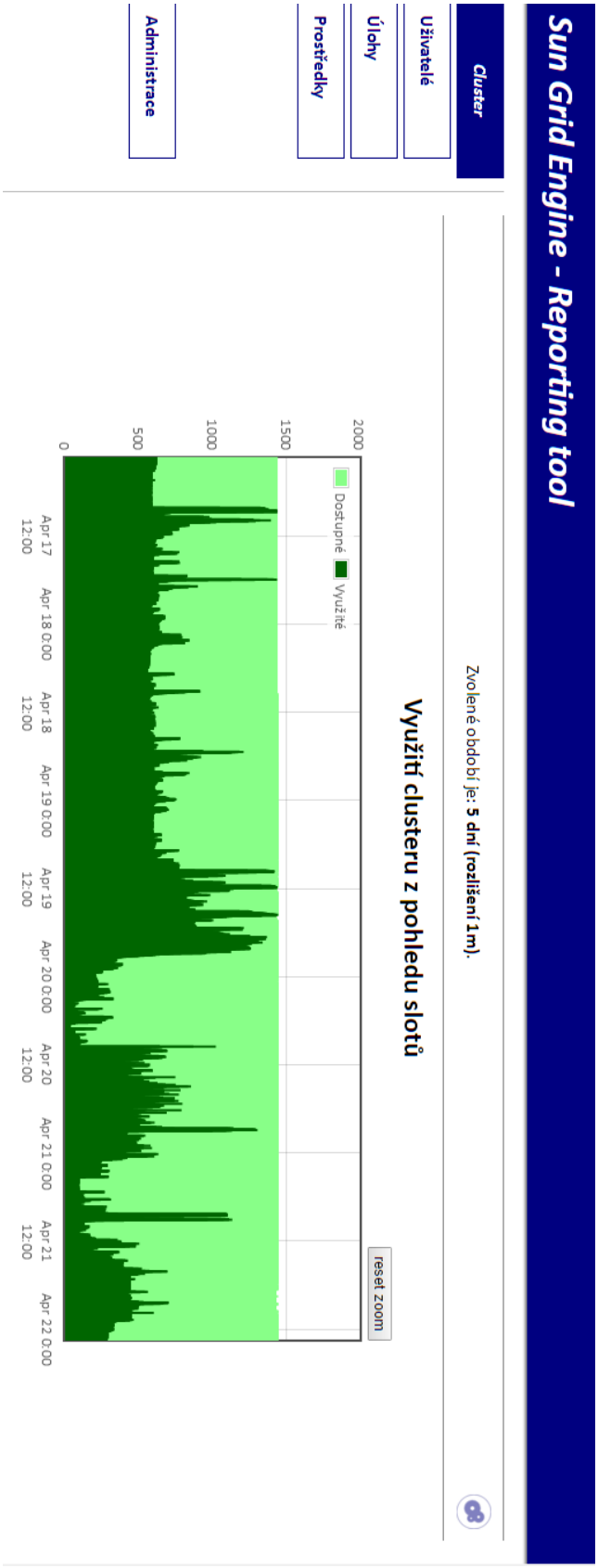
[9] Accounting: Sun Grid Engine accounting file format. *Grid Engine: Open Grid Scheduler* [online]. Santa Clara, Kalifornie, USA: Oracle Corporation, 2012, 2013 [cit. 2016-04-28]. Dostupné z: http://gridscheduler.sourceforge.net/htmlman/htmlman5/accounting.html

Seznam obrázků

Seznam zkratek

zkratka	anglický význam	český význam
CPU	central processing unit	centrální procesorová jednotka
CSS	Cascading Style Sheets	kaskádové styly
DRM	Distributed Resource Manager	správce distribuovaných prostředků
GPU	graphic processing unit	grafický procesor
GUI	Graphic User Interface	grafické uživatelské rozhraní
HPC	High-Performance Computing	vysoce náročné výpočty
HTML	Hyper Text Markup Language	Hypertextový značkovací jazyk
PHP	Hypertext Preprocessor, Personal Home Page	-
RRD	Round-robin database	Round-robin databáze
SGE	Sun Grid Engine	-
SQL	Standard Query Language	standardní dotazovací jazyk

Příloha A – návrh reportovacího nástroje



Příloha B – Obsah CD

`./src/` ..... složka se zdrojovými soubory
`./BP.docx` ..... pracovní text bakalářské práce
`./BP.pdf` ..... text bakalářské práce v elektronické podobě
`./dokumentace.pdf` ..... dokumentace vytvořeného nástroje