VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MERANIE VÝKONU A PROFILOVANIE SYSTÉMU CLOVERETL PRE SPRACOVANIE DÁT

BENCHMARKING AND PROFILING OF CLOVERETL DATA PROCESSING SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ROLAND BOTKA**
AUTHOR

VEDOUCÍ PRÁCE **RNDr. MAREK RYCHLÝ, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/20480/2017/xbotka00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Botka Roland, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Měření výkonu a profilování systému CloverETL pro zpracování dat**
Benchmarking and Profiling of CloverETL Data Processing System
Kategorie: Informační systémy

Pokyny:

1. Prozkoumejte možnosti měření výkonu systémů pro zpracování dat (TPC-DI/H) a jejich profilování, zejména s ohledem na datové transformace a V/V operace. Analyzujte a porovnejte existující přístupy a nástroje.
2. Seznamte se s řešením CloverETL pro integraci dat, prozkoumejte jeho části.
3. Navrhněte či vyberte způsob měření výkonu a profilování systému CloverETL v různých běhových prostředích (fyzický/virtuální stroj, různé technologie cloud, JVM aj.). Cílem je na základě naměřených hodnot zjistit maximální možnou průchodnost systému a jeho částí, sledovat změny výkonnosti a určit omezení (např. najít nejslabší článek). Navrhněte způsob sběru naměřených dat a jejich analýzy.
4. Po konzultaci s vedoucím implementujte a otestujte měřením výkonu různých nasazení a konfigurací CloverETL. Výsledek zveřejněte jako open-source.
5. Proveďte zhodnocení dosažených výsledků a diskutujte další možný vývoj projektu.

Literatura:

- Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears. Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10). ACM, New York, NY, USA, p. 143-154, 2010. ISBN 978-1-4503-0036-0. [https://doi.org/10.1145/1807128.1807152]
- Luo, C., Zhan, J., Jia, Z. et al. CloudRank-D: benchmarking and ranking cloud computing systems for data processing applications. Front. Comput. Sci. (2012) 6: 347. ISSN 2095-2228. [https://link.springer.com/article/10.1007/s11704-012-2118-7]
- Phoronix Test Suite - Linux Testing & Benchmarking Platform. Phoronix Media, 2017. [http://www.phoronix-test-suite.com/]

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% 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: **Rychlý Marek, RNDr., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

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

Táto diplomová práca sa zaoberá meraním výkonu a profilovaním systému CloverETL pre spracovanie dát. Podstatou je navrhnutie spôsobu merania výkonu prostredia a profilovania systému CloverETL. Problém bol vyriešený návrhom a implementáciou aplikácie, ktorá na základe nameraných dát a ich analýzy zistí maximálnu priepustonsť systému a sleduje zmeny výkonnosti systému. Výhodou je možnosť porovnať prostredia na ktorých beží systém CloverETL a využitie aplikácie pri automatizovanom testovaní vo firme. Vytvorené ETL Grafy boli otestované počas vývoja a sú súčasťou výsledku diplomovej práce.

Abstract

This thesis deals with the benchmarking and profiling of CloverETL Data Processing system. The aim of this thesis is to design a way of measuring performance and profiling in different runtime environments. The problem was solved by designing and implementing an application, which is based on measured data and its analysis, detects maximum system perpetuity and monitor performance changes in system performance. The advantage is the ability to compare environments on which the CloverETL System is running and application usage in automated testing in company. Created ETL Graphs have been tested during development and they are part of the result of this thesis.

Kľúčové slová

Meranie výkonu, Profilovanie, Systém CloverETL, CloverETL Grafy, Spracovanie dát, Programovací jazyk Java, MXBean, Testovanie

Keywords

Benchmarking, Profiling, System CloverETL, CloverETL Grafy, Data processing, Java programming language, MXBean, Testing

Citácia

BOTKA, Roland. *Meranie výkonu a profilovanie systému CloverETL pre spracovanie dát.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Marek Rychlý, Ph.D.

Meranie výkonu a profilovanie systému CloverETL pre spracovanie dát

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána RNDr. Marka Rychlého Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Roland Botka
17. mája 2018

Poďakovanie

Chcel by som poďakovať vedúcemu práce RNDr. Marku Rychlému Ph.D. za rady, ktoré mi poskytol pri riešení tejto diplomovej práce. Taktiež chcem poďakovať spoločnosti Javlin a.s., ktorá mi poskytla potrebný softvér a umožnila tak realizovať túto prácu.

Obsah

Kapitola 1

Úvod

V dnešnej dobe, v ktorej je kladený veľký dôraz na efektívnosť spracovávania veľkých objemov dát a je potrebné neustále hľadať a optimalizovať prostredie pre beh nástrojov, ktoré tieto dáta spracovávajú. Zvolenie nesprávneho prostredia môže spomaliť celý systém, ktorý je následne neefektívny. Toto nesprávne rozhodnutie spôsobí veľké straty pre spoločnosť. Môže sa jednať o straty napríklad finančné alebo strata zákazníkov. Je preto veľmi dôležité, aby spoločnosť čo najpresnejšie určila čo je potrebné zlepšiť v ich infraštruktúre, aby bol následne systém pre spracovanie dát efektívny. Rozhodne je však neprípustné žiadať od spoločnosti výmenu všetkých, už zaužívaných komponentov v ich infraštruktúre.

Keďže systém CloverETL je nástroj pre spracovanie veľkých objemov dát, stretáva sa s podobnými problémami. Systém CloverETL je jedinečný a v súčastnosti neexistuje nástroj, ktorý by priamo otestoval prostredie, na ktorom systém CloverETL beží a samotnú výkonnosť tohoto systému. Systém CloverETL sa nasadí do produkčného prostredia a až následne sa riešia problémy, ktoré vzniknú pri používaní a pri spracovávaní dát. Problémom je však aj to, že každá spoločnosť, ktorá využíva systém CloverETL má rozdielnu a špecifickú infraštruktúru, čo sťažuje určovanie najslabších komponentov, ktoré potom systém CloverETL spomaľujú a robia ho neefektívnym.

Táto práca sa zameriava na riešenie tohto problému, a to získaním informácií o voľne dostupných, už existujúcich riešeniach, ktoré by sa mohli využiť aj pri systéme CloverETL, poprípade riešeniach, ktorých spojením by vzniklo jedno komplexnejšie riešenie využiteľné pri špecifikovaní a vyriešení problému. Po vytvorení takéhoto nástroja vznikne sada testovacích súborov, takzvaných ETL Grafov, ktoré sa následne spustia na rôznych infraštruktúrach. Z behu týchto testov sa potom určí rýchlosť celého spracovania dát, poprípade sa určia komponenty infraštruktúry, ktoré sú nevyhovujúce. Toto poskytne spoločnosti lepší pohľad na problém, ktorý potom bráni lepšiemu a rýchlejšiemu spracovávaniu dát.

Práca je realizovaná v spolupráci s firmou Javlin a.s., ktorá vyvíja systém CloverETL a týmto projektom by rada poskytla informácie pre spoločnosti, kde systém CloverETL využívajú. Tento projekt by dokázal odhaliť slabšie komponenty v infraštruktúre zákazníkov firmy Javlin a.s., a to by zlepšilo a pomohlo odhaliť a vyriešiť nejeden problém s rýchlosťou spracovania dát. Keďže systém CloverETL je jedinečný a multiplatformový, je veľmi náročné použiť nejaké existujúce riešenie pre získanie týchto informácií.

Cieľom tejto práce je návrh a implementácia aplikácie, ktorá bude sledovať prostredie, na ktorom systém CloverETL beží a beh jednotlivých ETL Grafov. V kapitole 2 sa nachádzajú teoretické informácie o meraní výkonu a profilovaní Java aplikácií a kapitola 3 obsahuje základné informácie o systéme CloverETL. Existujúce riešenia, ktoré boli skúmané v rámci tejto práce sa nachádzajú v kapitole 4. V tejto kapitole sú taktiež dôvody prečo

konkrétne riešenia nie je možné využiť. Na základe týchto informácií je v kapitole 5 popísaný návrh výslednej aplikácie, za ktorým nasleduje popis implementácie (6). Záverečná kapitola 7 popisuje testovanie rôznych scenárov na novovytvorenej aplikácii a vyhodnotenie výsledkov.

Kapitola 2

Meranie výkonu a profilovanie systémov pre spracovanie dát

V súčastnosti je veľmi potrebné vytvárať výkonné a spoľahlivé systémy. Schopnosť merať a vyhodnocovať výkonnosť systémov je nutné pre následné lepšie optimalizácie operačného systému, na ktorom meraný systém beží alebo optimalizácie meraného systému. S tým súvisí aj efektivita systémov. Pri rozhodovaní, ktorý systém chce užívateľ využiť je jedným z kľúčových faktorov aj efektivita, a to či už pamäťová alebo časová. Nikto predsa nechce kupovať čo najdrahší hardvér, aby mohol čo najefektívnejšie spracovávať dáta. So zlepšením týchto vlastností dochádza k zvýšeniu hodnoty produktov, a to aj zvýšeniu počtu užívateľov. Ďalším faktorom je aj neustály vývoj a implementácia nových vlastností systémov, čo môže spôsobiť pokles výkonnosti. To môže byť ľahko detekovateľné pomocou nástrojov, ktoré vedia vyhodnotiť výkonnosť. V súčasnej dobe čoraz viac užívateľov a spoločností kladie dôraz na čo najlepšiu výkonnosť a spoľahlivosť systémov pre spracovanie dát. Na to existujú dva dôvody, a to úsilie o maximalizáciu využitia systémových zdrojov a maximalizáciu výkonu aplikácie.

Výkonnosť systémov je často meraná v jednotkách času, ale môže byť meraná aj spotrebovanou pamäťou. To znamená, koľko času alebo pamäti je potrebné na vykonanie nejakej operácie v systéme.

V tejto kapitole sú teoretické poznatky potrebné k pochopeniu problematiky, ktorá je riešená v tejto diplomovej práci. Keďže systém CloverETL je naprogramovaný v jazyku Java, celá kapitola je zameraná na meranie výkonu a profilovanie Java aplikácií. Ako prvé som sa zameral na meranie výkonu a profilovanie vo všeobecnosti, následne na monitorovanie výkonu virtuálneho stroja a profilovanie Java aplikácií.

2.1 Meranie výkonu

Porovnávacím nástrojom (angl. benchmark) je spustenie počítačového programu, súborov, programov alebo iných operácii s cieľom posúdiť relatívnu výkonnosť objektu, zvyčajne spustením niekoľkých štandardných testov a skúšok proti nemu. Meranie výkonu (angl. benchmarking) je proces vytvorenia výkonnostných výsledkov na základe behu súborov s testami. Tieto testy reprezentujú výkon celej úlohy alebo časti úlohy. Meranie výkonu sa zvyčajne spája s hodnotením výkonnostných charakteristík počítačového hardvéru, napríklad s operačným výkonom CPU s pohyblivou rádovou čiarkou, avšak niekedy sa táto technika aplikuje aj na softvér. Testy sú navrhnuté tak, aby napodobňovali určitý typ

pracovnej záťaže na komponente alebo systéme. Syntetické porovnávacie nástroje to robia pomocou špeciálne vytvorených programov. Referenčné porovnávacie nástroje pracujú s reálnymi programami v systéme. Zatiaľ čo aplikačné porovnávacie nástroje zvyčajne určujú oveľa lepšiu mieru reálneho výkonu v danom systéme, syntetické porovnávacie nástroje sú užitočné pri testovaní jednotlivých komponentov. Výsledky testov by mali ukázať, ako dobre zvládol systém splniť dané úlohy na danej konfigurácii. Takto sa dajú porovnať zmeny v danom systéme s cieľom určiť účinky prípadnej zmeny [13].

Pri vykonávaní merania výkonu sa často naráža na radu rôznych problémov. Takéto meranie nie je jednoduché, a preto sa často niekoľkokrát opakujú merania, aby sa dospelo k predvídateľným výsledkom. Zhodnotenie výsledkov však býva veľmi zložité, pretože niektorí výrobcovia vytvárajú produkty tak, aby dosiahli čo najlepšie čísla v týchto testoch. Existuje však aj veľa kritérií, ktoré je možné medzi produktami porovnávať.

Typy porovnávacích nástrojov:

- skutočný program
- meranie výkonu menších komponentov
- meranie výkonu jadra
- syntetické porovnávacie nástroje
- vstupné a výstupné porovnávacie nástroje
- meranie výkonu pri databázach – meranie rýchlosti prenosu a rýchlosti reakcie databáz
- paralelné porovnávacie nástroje – používané pri prístrojoch s viacerými jadrami alebo procesormi, poprípade systémy, ktoré sa skladajú z viacerých prístrojov

2.2 Profilovanie

Profilovanie (angl. profiling) formou dynamickej programovej analýzy, pomocou ktorej vieme zmerať pamäťovú alebo časovú zložitosť programu, použitie konkrétnych inštrukcií alebo frekvenciu a trvanie funkčných volaní. Profilovanie sa dosahuje pomocou nástroja s názvom profiler. To môže byť zdrojový kód programu alebo binárny spustiteľný program. Tieto nástroje môžu využívať mnoho rôznych techník, ako sú napríklad metódy založené na udalostiach, štatistikách, inštrukciách a simulačné metódy. Podobne ako porovnávacie testy pri meraní výkonu, tak aj pri profilovaní existujú testy, ktoré produkujú výsledky. Tieto výsledky sa ďalej analyzujú s cieľom určiť časti systému, ktoré sú problematické buď v ich výkonnosti (čas na dokončenie) alebo v ich využívaní zdrojov (pridelenie a využitie pamäte). Výsledky profilovania sa využívajú hlavne na nájdenie menej výkonných miest v systéme.

Profileri využívajú rôzne techniky na zbieranie dát:

- Hardware interrupts – Hardvérové prerušenie je metóda pre asynchrónnu obsluhu udalostí.
- Code instrumentation – Schopnosť monitorovať alebo merať výkon produktu, diagnostikovať chyby a zapisovať informácie o bežiacom programe.
- Instruction set simulation – Simulátor sady inštrukcií je simulačný model, ktorý napodobňuje správanie mikroprocesora pomocou čítania inštrukcií a udržiavania interných premenných, ktoré reprezentujú registre procesora.

- Operating system hooks – Termín „hooks“ pokrýva množstvo techník, ktoré sa používajú na zmenu alebo rozšírenie správania operačného systému.
- Hardware performance counter – Sú to špeciálne registre zabudované do moderných mikroprocesorov, ktoré ukladajú počty aktivít súvisiacich s hardvérom.

Typy profilerov na základe výstupu:

- plochý profiler – vypočíta priemer počtu volaní
- profiler volaní – tento profiler zobrazí počet a frekvenciu volaní určitých funkcií
- profiler citlivý na vstup – tento typ sa vzťahuje na meranie výkonu pri zmenách vstupných faktorov ako sú vstupná veľkosť alebo rôznorodosť vstupných hodnôt. Vytvárajú grafy, ktoré charakterizujú zmenu výkonnosti aplikácie pri zmenách vstupu

Nástroje merania výkonu a profilovania poskytujú platformu, ktorá dokáže identifikovať problémové oblasti systému. Nástroje merania výkonu umožňujú porovnávať zmeny v systéme a nástroje profilovania diagnostikovať miesta, kde sa problémy vyskytujú [13].

2.3 Dva prístupy pre výkonnostné analýzy Java aplikácií

Typicky existujú dva prístupy pre výkonnostné analýzy, a to zhora nadol a zdola nahor. Prístup zhora nadol sa zameriava na najvyššie úrovne aplikácií a prechádza softwérom v ktorom hľadá problémové oblasti a optimalizačné možnosti. Naproti tomu, prístup zdola nahor začína na najnižšej úrovni v softwéri, kde sa snaží nájsť možnosti zlepšenia na úrovni práce s CPU. Prístup zdola nahor je najčastejšie využívaný špecialistami, ktorí analyzujú výkon jednotlivých softwérov. Každý prístup však nájde rôzne odlišné výkonnostné problémy [13].

2.3.1 Prístup zhora nadol

Prístup zhora nadol je najčastejšie používaným prístupom pre ladenie výkonu. Tento prístup sa začína sledovaním aplikácie pod záťažou, pri ktorom sa zameriava na problémy s výkonnosťou. V niektorých prípadoch sa môže aplikácia monitorovať nepretržite, a to najmä z dôvodu zmeny konfigurácie aplikácie alebo zmeny v záťaži aplikácie, pri ktorom nastáva zníženie výkonnosti aplikácie.

Táto sledovacia činnosť môže zhŕňať sledovanie štatistiky úrovne operačného systému, štatistiky virtuálneho stroja, štatistiky kontajnerov Java EE alebo štatistiky výkonnosti inštrukcií. Následne, po zmonitorovaní problémov sa prechádza do ďalšej fázy, a to ladenie garbage collectoru, ladenie virtuálneho stroja alebo profilovanie aplikácie. Profilovanie aplikácie môže viesť k zmene v implementácií aplikácie, identifikácii neefektívnej implementácie tried alebo metód v knižnici Java SE alebo ku identifikovaniu neefektívnej implementácie knižníc tretích strán, ktoré sa v aplikácií používajú [13].

2.3.2 Prístup zdola nahor

Tento prístup sa zvyčajne používa pre dosiahnutie zlepšenia výkonu aplikácie na jednej platforme vzhľadom na inú platformu, kde existujú rozdiely v základnom CPU, architektúre CPU alebo počte procesorov. Prístup zdola nahor sa často používa pre zlepšenie výkonu aplikácie pri migrácii, na podporu iného operačného systému a taktiež vtedy, keď nie je možné vykonať zmenu zdrojového kódu aplikácie.

V tomto prístupe sa začína monitorovanie na najnižšej úrovni, úrovni CPU. Monitorujú sa štatistiky ako počet CPU inštrukcií potrebných na vykonanie daného pracovného zaťaženia a počet CPU cache miss, ktoré sa vyskytnú počas behu aplikácie. Aplikácia je rýchlejšia, ak je schopná vykonávať a precízne škálovať zaťaženie vykonaním menšieho počtu inštrukcií. CPU Cache miss spôsobuje zbytočné cykly procesora, ktorý čaká na dáta, ktoré majú byť vytiahnuté z pamäte. Zaujímavé môžu byť aj iné CPU štatistiky, ale predošlé dve sú najčastejšie pozorovanými v prístupe zdola nahor. Hlavným cieľom tohoto prístupu je zlepšiť využitie procesora bez zmien v aplikácii [13].

V modernom virtuálnom stroji môžu byť implementované rôzne optimalizácie, ktoré by vygenerovali efektívnejší strojový kód na základe modelov prístupu k pamäti. Ďalej sa môžu zmeniť alebo vyladiť aj nastavenia na úrovni operačného systému tak, aby umožňovali lepší výkon (napr. zmena algoritmu plánovania CPU).

2.4 Monitorovanie výkonu operačného systému

Monitorovanie výkonu je nerušivá činnosť zhromažďovania alebo sledovania výkonnostných údajov z aplikácie. Monitorovanie je zvyčajne preventívny alebo proaktívny typ činnosti, ktorý sa zvyčajne vykonáva v produkčnom, kvalifikačnom alebo vývojovom prostredí. Monitorovanie je prvým krokom, keď užívateľ aplikácie nahlásil problém s výkonom, ale neposkytol dostatočné informácie o potenciálnej príčine [13].

Profilovanie výkonnosti je aktom zhromažďovania údajov o výkonnosti aplikácie, ktorý môže narušiť prístupnosť a výkonnosť aplikácie. Profilovanie sa iba zriedka vykonáva vo výrobnom prostredí a zvyčajne sa robí v kvalifikačných, testovacích alebo vývojových prostrediach. Je zvyčajne aktom, ktorý nasleduje po monitorovacej činnosti, ktorá poukazuje na nejaký problém s výkonom aplikácie. Ladenie výkonnosti na rozdiel od monitorovania výkonu a profilovania výkonnosti, je aktom zmeny atribútov, zdrojového kódu alebo atribútov konfigurácie za účelom prístupnosti a výkonnosti aplikácie. Tieto zmeny sú zvyčajne tretím krokom pri procese zlepšovania výkonnosti systému [13].

2.4.1 Využitie CPU

Pre dosiahnutie najvyššieho výkonu je potrebné nielen plné využitie cyklov CPU, ktoré sú k dispozícií, ale aj ich využitie takým spôsobom, aby neboli plytvané. Schopnosť efektívneho využitia cyklov CPU je náročnejšie pri viacvláknových aplikáciach bežiacich na multiprocesorových a viacjadrových systémoch. Okrem toho, aplikácia, ktorá dokázala nasýtiť zdroje procesora, nemusí dosiahnuť svoj maximálny výkon. Využitie CPU sa vo väčšine operačných systémov zaznamenáva ako užívateľské využitie CPU a kernelové alebo systémove využitie CPU. Pri užívateľskom využití CPU je to percento času, ktorý aplikácia strávi v aplikačnom kóde. Naopak kernelové alebo systémove využitie CPU predstavuje percento času, ktorý aplikácia strávi využívaním kernelu alebo systému. Využívanie kernelu alebo systému môže byť indikáciou zdieľania zdrojov alebo veľkého počtu interakcií medzi vstupno/výstupnými zariadeniami. Ideálna situácia pre maximálny výkon aplikácií ma mať využitie 0% kernelu alebo systémového CPU, pretože cykly vyčerpané v kóde kernelu alebo operačného systému by mohli byť využité pomocou kódu aplikácie [13].

Pri aplikáciách, ktoré sú náročné na výpočet, môže ísť monitorovanie výkonu oveľa hlbšie, ako len pre pozorovanie užívateľského a kernelového alebo systémového využitia CPU. Na náročnejších systémoch je možné ďalšie sledovanie počtu inštrukcií CPU na jeden hodinový cyklus (IPC) alebo počet cyklov CPU na jednu CPU inštrukciu (CPI). Tieto

dve dodatočné metriky sú zaujímavé pre aplikácie náročné na výpočet, pretože moderné monitorovacie aplikácie ukazujú využitie procesora a neoznamujú percento cyklov CPU, ktoré procesor vykonal [13].

2.5 Monitorovanie výkonu virtuálneho stroja

Monitorovanie virtuálneho stroja je činnosť, ktorá by mala byť vykonávaná po celú dobu behu aplikácie. Vzhľadom na to, že virtuálny stroj je kritickou komponentov, mal by byť monitorovaný rovnako ako samotná aplikácia a operačný systém. Analýza monitorovania virtuálneho stroja indikuje to, kedy je potrebné jeho ladenie. Ladenie virtuálneho stroja by sa malo očakávať vždy vtedy, keď nastane zmena verzie virtuálneho stroja, zmena operačného systému alebo veľká zmena v aplikácií [13].

Existuje niekoľko oblastí virtuálneho stroja, ktoré sú vhodné pre monitorovanie výkonu, a to vrátane garbage collectora, aktivity JIT kompilátora a načítávania tried (class loading). Na monitorovanie virtuálneho stroja je k dispozícií veľa nástrojov, či už bezplatných alebo komerčných [13].

2.5.1 Monitorovanie alogritmu garbage collector

Garbage collector je špeciálny algoritmus, ktorý vyhľadáva a uvoľňuje tie úseky pamäte, ktoré už program alebo proces nepoužíva. Monitorovanie algoritmu garbage collector virtuálneho stroja je dôležité, pretože môže mať výrazný vplyv na výkonnosť a latenciu aplikácie. Moderné virtuálne stroje umožňujú sledovať štatistické údaje o algoritme garbage collector v textovej forme, za pomoci logovacích súborov alebo publikovaním štatistických údajov do monitorovacieho užívateľského rozhrania [13].

Údaje, ktoré sa zbierajú v kontexte algoritmu garbage collector sú:

- použitý garbage collector
- veľkosť Java heap
- veľkosť fyzických oblastí mladej, starej generácie a permanentnej generácie
- dĺžka trvania a frekvencia minoritného a úplného garbage collection
- množstvo regenerovaného priestoru pomocou minoritného garbage collection
- obsadenosť Java heap pred spustením a po dokončení garbage collection
- obsadenosť mladej, starej a permanentnej generácie pred spustením a po dokončení garbage collection

2.6 Profilovanie Java aplikácií

Profilovanie Java aplikácií sa zaoberá najmä profilovaním metód a profilovaním pamäte. Profilovanie metód poskytuje informácie o čase vykonávania metód v Java aplikácií. Pre takéto profilovanie už existuje veľa analyzátorov od rôznych spoločností. Na rozdiel od profilovania metód poskytuje profilovanie pamäte informácie o využití pamäte v Java aplikácií, ako napríklad počet a veľkosť alokácií objektov [13].

Medzi bežné pojmy profilovania patria:

- Profiler – Nástroj, ktorý zobrazuje informácie o aplikácii pri jej behu. Môže zobrazovať aj informácie o virtuálnom stroji.
- Profil – Súbor, ktorý obsahuje informácie zozbierané profilerom pri behu aplikácie.
- Režijné náklady – Čas, ktorý profiler strávi zbieraním profilovacích informácií, namiesto behu aplikácie.
- Strom volaní – Zoznam metód, ktorý znázorňuje metódy volané od začiatku spustenia programu.
- Filter – Artefakt, ktorý sa môže použiť na zozbieraný profil a zužuje rozsah zhromaždených informácií.

2.7 Java MBean a MXBean objekty

Mbean je spravovaný Java objekt, ktorý sa riadi návrhovými vzormi uvedenými v špecifikácií JMX (Java Management Extensions). MBean môže reprezentovať zariadenie, aplikáciu alebo akýkošvek zdroj, ktorý je potrebné spravovať. MXBeans sú špeciálny typ MBean, ktorá odkazuje iba na preddefinovanú sadu dátových typov. MXBeans poskytujú pohodlný spôsob spájania súvisiacich hodnôt, bez toho, aby boli klienti špeciálne konfigurovaní. Rovnako ako MBean, aj MXBeans sú definované Java rozhraním s názvonm SomethingMXBean a Java triedy, ktorá implementuje toto rozhranie [23].

Trieda ManagementFactory je továrenská trieda pre získanie beanov pre platformu Java. Táto trieda pozostáva zo statických metód, z ktorých každá vráti jednu alebo viac platformové MXBean-y reprezentujúce rozhranie pre spravovanie virtuálneho stroja(JVM) [18].

Rozhranie MemoryMXBean

MemoryMXBean je rozhranie správy pamäťového systému virtuálneho stroja. Pamäťový systém virtuálneho stroja spravuje nasledujúce typy pamäte, ktoré sú znázornené na obrázku 2.1:

- Pamäť Java Heap

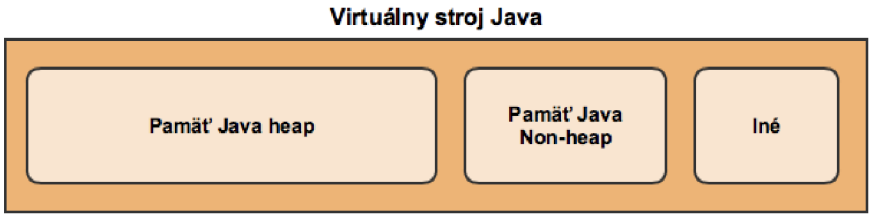
  Virtuálny stroj má Java heap, čo je dátová oblasť, v ktorej sú alokované všetky inštancie tried a polí. Táto dátová oblasť je vytvorená pri spustení virtuálneho stroja. Hromada pre objekty je regenerovaná automatickým systémom spravy pamäte, ktoré jazyk Java poskytuje – garbage collector. Táto pamäť môže mať fixnú veľkosť alebo môže byť rozšírená a zmenšená automaticky [20].

- Pamäť Java non-heap

  Virtuálny stroj má oblasť, ktorá je zdieľaná medzi všetkými vláknami a pamäť potrebnú na interné spracovávanie alebo optimalizáciu virtuálneho stroja. Ukladá štruktúry tried, údaje o poliach a metódach a kód pre metódy a konštruktory. Túto pamäť nesmie ovplyvňovať garbage colector a podobne ako hromada, môže byť fixnej alebo premennej veľkosti [20].

- Iná pamäť

  Tu sa nachádza kód virtuálneho stroja, jeho vnútorné štruktúry a podobne.



Obr. 2.1: Rozdelenie pamäte vo virtuálnom stroji

Monitorovanie používania pamäte slúži na indikáciu využitia pamäte aplikácie, pracovného zaťaženia alebo môže signalizovať únik pamäte. Sledovanie pamäte môže byť vykonané pollingom, prahovým oznámením alebo zberom prahových oznámení.

Rozhranie ThreadMXBean

ThreadMXBean je rozhranie pre správu systému vlákien virtuálneho stroja. Identifikátor vlákna je hodnota, ktorá je jedinečná počas životnosti tohoto vlákna. Po ukončení vlákna môže byť táto identifikačná hodnota opätovne použitá. Niektoré metódy tohoto rozhrania používajú tieto hodnoty ako parameter a následne vrátia informácie o jednotlivých vláknach [22].

Rozhranie OperatingSystemMXBean

OperatingSystemMXBean je rozhranie pre správu operačného systému, na ktorom je spustená. Toto rozhranie definuje niekoľko metód prístupu k systémovým vlastnostiam operačného systému [21].

Rozhranie GarbageCollectorMXBean

Rozhranie GarbageCollectorMXBean slúži pre manažment algoritmu garbage collection. Garbage collection je proces, ktorý využíva virtuálny stroj na vyhľadanie, obnovu nedosiahnuteľných objektov a uvoľnenie pamäťového priestoru. Virtuálny stroj môže mať jednu alebo viac inštancií implementačnej triedy tohto rozhrania [19].

Kapitola 3

Systém CloverETL

CloverETL je softvérový nástroj vyvíjaný českou spoločnosťou Javlin, ktorý slúži na integráciu dát s rýchlym rozvojom. Umožňuje efektívne vyvíjať, nasadzovať a automatizovať transparentné transformácie dát, od načítania súborov z databáz, až po automatizáciu komplexného pohybu dát medzi databázami, súbormi a rozhraniami Web Service APIs. Poskytuje efektívnu zmes rýchleho vizuálneho návrhu transformácií a pracovných postupov s plnohodnotnými prispôsobovacími vlastnosťami kódovania a automatizáciou [4].

Systém CloverETL využívajú hlavne:

- Firmy, ktoré kombinujú údaje z rôznych zdrojov do jednej alebo viacerých cieľových aplikácií (reportovanie, analytika, business, intelligence).
- Tímy na spracovanie a vývoj údajov, ktoré potrebujú transparentné vizuálne riešenie ETL na automatizáciu ich dátových operácií.
- Dátoví architekti a inžinieri, ktorí potrebujú rýchlo reagovať na zmeny v dátových štruktúrach, formátoch a dátových tokoch bez problémov s kódovaním.

Práca s dátami prebieha prostredníctvom transformačného grafu, ktorý je orientovaný, acyklický a obsahuje jednotlivé komponenty. Tieto komponenty sú navzájom poprepájané prostredníctvom hrán, ktorými dáta pretekajú medzi komponentami. Celá práca s dátami prebieha v rámci špecializovaných komponent, ktoré môžu dáta napríklad čítať z disku alebo databázy, filtrovať, meniť ich formát alebo zapisovať na disk. Tieto transformačné grafy je možné vytvárať alebo upravovať za pomoci návrhára grafov, aplikácie CloverETL Designer.

Transformácie sú vykonávané prostredníctvom behu jednotlivých komponent, kde každá komponenta beží vo svojom vlákne. Všetky bežiace komponenty sú riadené pomocou špeciálneho vlákna – WatchDog, ktoré sleduje počet pracujúcich komponent a ich zdravie. Toto vlákno vykonáva aj potrebné upratovanie po zhavarovaní komponenty a taktiež reportuje o tom, čo sa v rámci grafu aktuálne deje. Komponenty si medzi sebou posielajú jednotlivé dátové záznamy, inak o sebe nevedia. Ak sa v priebehu transformácie vyskytne chyba, vlákno WatchDog všetky komponenty ukončí. Štandardné ukončenie transformácie prebieha tak, že počiatočné komponenty posielajú správu o tom, že žiadne ďalšie dáta už nie sú k dispozícií a samé sa ukončia. Táto správa sa šíri naprieč celým grafom, až napokon dôjde k ukončeniu poslednej komponenty. Keď vlákno WatchDog zistí, že už žiadna komponenta nebeží, ukončí sa, čo má za následok zastavenie celého programu.

Grafy

Grafom sú pomenované najmenšie spustiteľné jednotky v systéme. Grafy sa vytvárajú v rámci CloverETL projektov.

Typy grafov

Systém pracuje s rôznymi typmi grafov, z ktorých každý typ má iné použitie.

- ETL Grafy

  ETL Graf je najmenšia spustiteľná jednotka v systéme CloverETL. Definícia ETL Grafu je uložená v samostatnom súbore s príponou *.grf.

- Podgrafy

  Podgraf (Subgraph) je užívateľom definovaný graf, ktorý môže byť opakovane použiteľný ako komponent s logikou implementovanou ako ETL Graf. Podgraf môže použiť akékoľvek prvky grafu. Podgrafy môžu byť vnorené, to znamená, že podgraf môže pracovať s inými podgrafami. Podgrafy sa využívajú na zjednodušenie komplexnej ETL logiky ako aj pre vizuálne zníženie počtu komponentov v komplexných ETL grafoch. Vytváraním podgrafov vznikajú opakovane použiteľné bloky logiky. Definícia podgrafu je uložená v samostatnom súbore s príponou *.sgrf.

- Jobflow

  Jobflow modul umožňuje kombinovať ETL Grafy spolu s ďalšími aktivitami do zložitých procesov – zabezpečenie orchestrácie, podmieneného vykonávania úloh a spracovania chýb. V module jobflow sa môžu zúčastňovať rôzne akcie ako CloverETL ETL Grafy, aplikácie a skripty, webové služby (REST/SOAP) a operácie s lokálnymi a vzdialenými súbormi. Okrem týchto akcií, ktoré sú k dispozícií ako vyhradené komponenty modulu jobflow, môže modul obsahovať aj komponenty ETL. Definícia Jobflow je uložená v samostatnom súbore s príponou *.jbf.

- Dátové služby

  Dátové služby(Data Services) umožňujú nasadenie/publikovanie dát alebo vystavanie dátových transformácií vo forme rozhrania REST API. Logika služby je implementovaná pomocou CloverETL Grafu, ktorý má úplný prístup k HTTP kontextu volaní: prichádzajúce dáta, parametre a hlavičky. Služba môže odpovedať s dátami, HTTP hlavičkami a stavovými kódmi. Definícia dátovej služby je uložená v samostatnom súbore s príponou *.rjob.

Komponenty

Najdôležitejšími elementami grafov sú komponenty, ktoré slúžia na spracovávanie dát. Väčšina z nich má porty, cez ktoré sa dáta príjmajú a porty, pomocou ktorých sa spracované dáta odosielajú. Existujú rôzne typy komponentov, ktoré sa rozdeľujú do niekoľkých skupín.

- komponenty pre čítanie – Načítajú dáta zo vstupných súborov (lokálne alebo vzdialene), prijímajú ich z pripojeného vstupného portu, čítajú ich zo slovníka alebo generujú údaje.

- komponenty pre zápis – Prijímajú dáta zo vstupných portov a zapisujú ich do súborov (taktiež lokálne alebo vzdialene), posielajú ich cez pripojený výstupný port, dokážu posielať e-maily, zapisovať údaje do slovníka alebo vyradia nepotrebné dáta.
- komponenty pre transformácie – Prijímajú dáta a kopírujú ich na všetky výstupné porty, filtrujú alebo zoraďujú dáta, zlučujú, zhromažďujú alebo zlúčia dáta z viacerých portov alebo slúžia pre rôzne zložitejšie transformácie.
- komponenty pre spájanie – Prijímajú dáta z dvoch alebo viacerých zdrojov, spájajú dáta podľa zadaného kľúča a odosielajú tieto dáta cez výstupné porty.
- komponenty pre kontrolovanie úloh – Vykonávajú a sledujú rôzne typy úloh. Umožňujú spustiť ETL Grafy, jobflow a rôzne skripty. Následne tieto ETL Grafy a jobflow môžu byť monitorované.
- komponenty pre súborové operácie – Spracovávajú súbory v lokálnom alebo vzdialenom súborovom systéme.
- komponenty klastra – Umožňujú distribuovať dátové záznamy medzi rôzne uzly klastra CloverETL clustrov.
- komponenty vykonávajúce rôzne úlohy na základe kvality dát – Určujú informácie o údajoch, zisťujú a odstraňujú problémy a podobne.
- ostatné komponenty – Heterogénna skupina komponentov, ktoré vykonávajú rôzne úlohy.
- Zastarané komponenty

Hrany

Hrany prenášajú dáta z jednej komponenty do druhej. Komponenty musia byť prepojené hranami, kde hrana musí prepájať práve dva komponenty. V systéme CloverETL existujú štyri typy hrán:

- priame hrany – majú buffer v pamäti, ktorá pomáha rýchlejšiemu toku dát. Predvolený typ hrany pre ETL Grafy
- bufrované hrany
- priame propagované hrany
- bufrované propagované hrany

Metadáta

Keďže každá hrana nesie nejaké dáta, je potrebné tieto dáta opísať pomocou metadát. Metadáta opisujú štruktúru dát, ktoré prechadzajú z jednej komponenty do druhej. Záznam je možné popísať ako riadok dátového súboru alebo ako riadok databázovej tabuľky. Záznamy môžu mať rôzne typy a každé pole záznamu môže obsahovať iný typ dát.

CloverETL transformačný jazyk

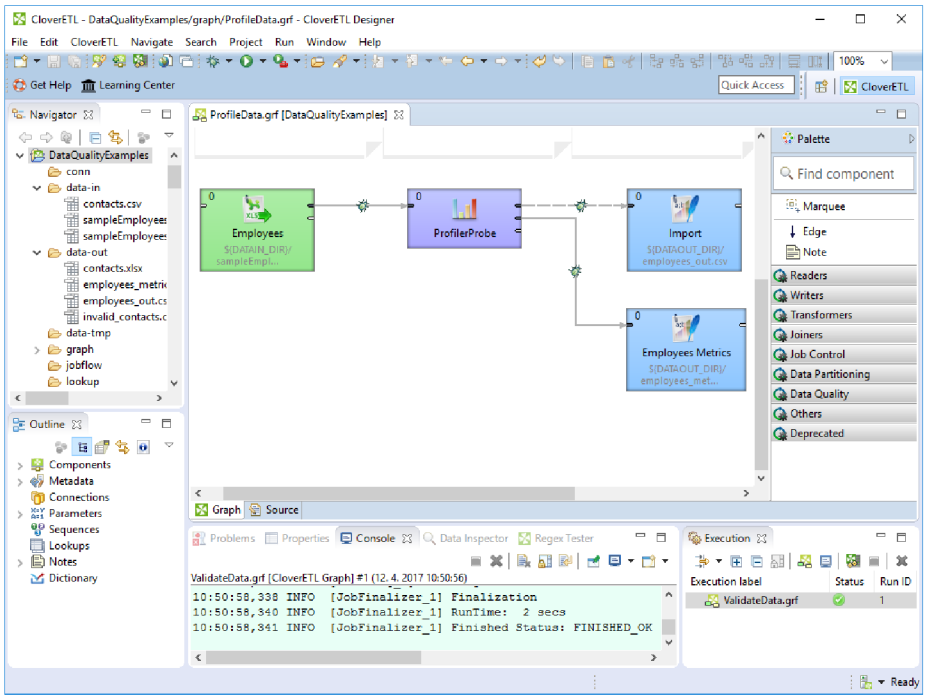
CloverETL Transformation Language (CTL) je proprietárny skriptovací jazyk orientovaný na spracovanie údajov v transformačných komponentách systému ClovetETL. Je navrhnutý tak, aby umožňoval jednoduché a jasné zaznamenanie spôsobu spracovávanie údajov a poskytoval dostatočné prostriedky na ich manipuláciu. Jazyková syntax pripomína programovací jazyk Java, s niektorými konštrukciami bežnými v skriptovacích jazykoch. CTL je jazyk vysokej úrovne v abstrakcii spracovaných údajov a v prostredí, v ktorom sa vykonáva. Jazyk chráni programátora pred zložitosťou celkovej transformácie dát a súčasne ho zameriava na vytvorenie jedného transformačného kroku ako súboru operácií použiteľných na všetky spracované dátové záznamy. CTL poskytuje bohatú sadu funkcií pre validáciu a manipuláciu s dátami. Počas vykonávania transformácie každá zložka, ktorá používa CTL kód, používa samostatnú inštanciu prekladača, čím sa zabráni možným kolíziám v paralelnom multi-vláknovom prostredí systému CloverETL [3]. V súčastnosti sa používa druhá verzia CloverETL transformačného jazyku. Vo výpise 3.1 sa nachádza príklad funkcie napísanej v CloverETL transformačnom jazyku (CTL2).

```
//#CTL2
function integer transform() {
	if ( $in.0.hasDetail ) {
		$out.0.name = $in.0.name;
		$out.0.email = $in.0.email;
		return 0;
	}

	$out.1.name = $in.0.name;

	return 1;
}
```

Výpis 3.1: Príklad CTL2 kódu

3.1 CloverETL Designer

CloverETL Designer je samostatná aplikácia na vytváranie a prevádzku ETL Grafov. Aplikácia sa využíva na vytváranie projektov, grafov a všetkých ostatných zdrojov, ktoré je možné spúšťať lokálne priamo cez aplikáciu. CloverETL Designer umožňuje ľahko pracovať s CloverETL Serverom. Designer je postavený na rozšíriteľnej platforme Eclipse [3]. Na obrázku 3.1 sa nachádza užívateľské rozhranie aplikácie CloverETL Designer.



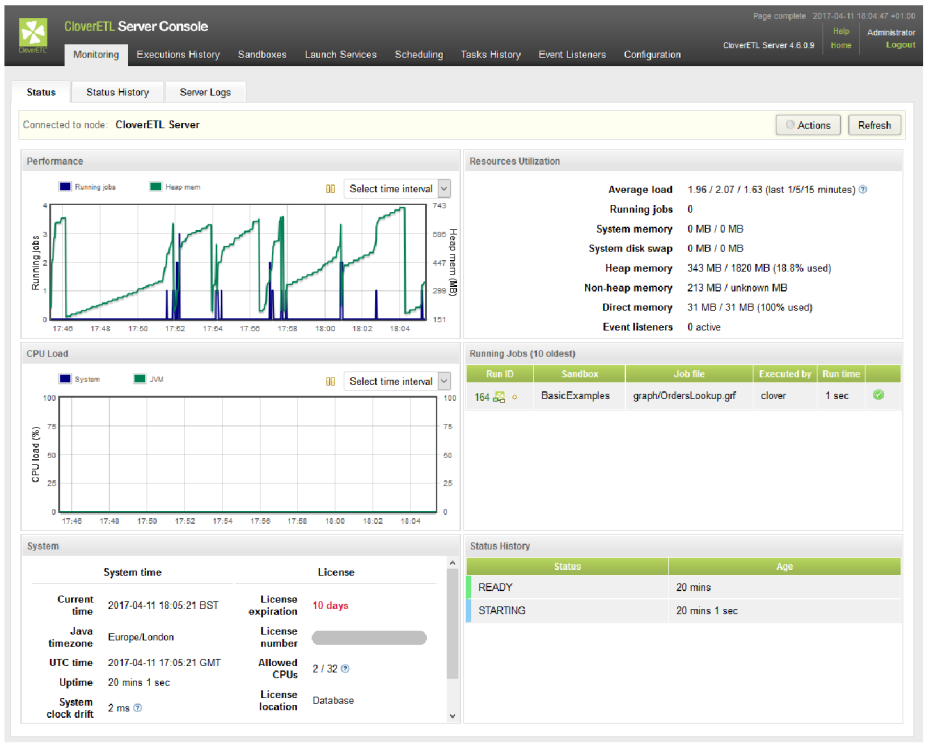
Obr. 3.1: Prostredie aplikácie CloverETL Designer [1]

3.1.1 Typy CloverETL projektov

- CloverETL (Lokálne) Projekty – sú projekty, ktorých celá štruktúra sa nachádza na lokálnom počítači. Všetky dáta a grafy sú lokálne v projekte v pracovnom priestore (workspace) a grafy je možné spúšťať lokálne pomocou CloverETL Runtime. Lokálne projekty je možné jednoducho verzovať pomocou ľubovolného verzovacieho systému.
- CloverETL Serverové Projekty – sú projekty, ktoré patria do takzvaných CloverETL Servrových sandboxov. Celá štruktúra projektu je na serveri a taktiež sa súbory nachádzajú lokálne na počítači. Pri vytváraní a editovaní súborov lokálne sa automaticky synchronizuje obsah s CloverETL Serverom. Naopak, pri vytvorení súboru na serveri sa automaticky stiahne obsah súboru a zobrazí sa v CloverETL Designeri.

3.2 CloverETL Server

CloverETL Server je serverová aplikácia pre spustenie, plánovanie a monitorovanie ETL Grafov. Táto aplikácia spolupracuje s aplikáciou CloverETL Designer, v ktorej je možno vytvárať taktiež projekty, grafy a všetky ostatné zdroje. S týmito projektami sa potom pracuje rovnakým spôsobom, ako keby sa pracovalo so štandardným CloverETL Designerom iba lokálne. CloverETL Server je Java aplikácia postavené na štandardoch J2EE, ktorá podporuje širokú škálu aplikačných serverov (Apache Tomcar, Jetty, IBM WebSphere, SunGlassfish, JBoss AS a Oracle WebLogic). CloverETL Cluster zahrňuje niekoľko inštancií CloverETL Serverov, ktoré bežia paralelne na iných uzloch [3]. Na obrázku 3.2 sa nachádza administračné rozhranie CloverETL Serveru.



Obr. 3.2: Prostredie CloverETL Serveru [2]

3.3 CloverETL Community Edition

CloverETL Community Edition je voľne dostupná verzia CloverETL Designéra pre nekomerčné účely alebo komerčné s obmedzenou funkcionalitou. Táto verzia je limitovaná maximálnym počtom komponentov v grafe, ktorých môže byť 20. Taktiež je obmedzená rôzna funkcionalita ako napríklad sekvencie, niektoré pokročilejšie funkcie v jazyku CTL, exportovanie, logovanie a rôzne iné [3]. Pre porovnanie v tabuľke 3.1 sú uvedené rozdiely v počte komponentov v CloverETL Community Edition a CloverETL Designer.

Components	CloverETL Community Edition	CloverETL Designer
Readers	10	26
Writers	7	31
Transformers	10	22
Joiners	2	7
Job Controls	1	7
File Operations	0	5
Others	5	10

Tabuľka 3.1: Porovnanie počtu komponentov CloverETL Community Edition a CloverETL Designer [3]

3.4 CloverETL Open Source Engine

CloverETL Engine je knižnica programovacieho jazyka Java, ktorú využívajú produkty systému CloverETL pre beh ETL Grafov. Všetky vydania systému CloverETL sú postavené na základe knižnice CloverETL Engine. CloverETL Engine je základná knižnica systému CloverETL, na ktorej môžu byť púšťané grafy a je voľne dostupná avšak neobsahuje žiadne grafické užívateľské rozhranie, akým je napríklad grafový dizajnér. Môže však byť použitá v rôznych aplikáciach v obmedzenom režime. Pre vývoj ETL Grafov sa predovšetkým využíva nástroj CloverETL Designer, kde sa vytvárajú a upravujú špeciálne súbory, ktoré môžu byť spustené už vopred vytvorenými skriptami z tejto knižnice.

Kapitola 4

Porovnanie existujúcich riešení

Existuje veľa nástrojov pre porovnávanie výkonnosti častí procesu spracovania dát. Tieto nástroje sú však často zamerané na meranie výkonnosti jednotlivých procesov, ktoré sú veľmi špecifické alebo na meranie výkonu webových aplikácií.

V tejto kapitole sa nachádza prehľad existujúcich nástrojov a riešení, ktoré boli skúmané v rámci tejto práce. Mnoho z týchto nástrojov však nie je možné využiť, pretože sú veľmi špecifické a zameriavajú sa na iné problémy, ktoré nesúvisia s touto prácou.

4.1 Yahoo! Cloud serving benchmark

YCSB (Yahoo! Cloud Serving Benchmark) je porovnávací framework, ktorý slúži pre porovnávanie výkonu nových generácií cloudových systémov, slúžiacich na poskytovanie dát v cloude. Často sa používa na porovnanie relatívneho výkonu systémov správy databáz NoSQL. Ako je už z názvu dané, tento framework bol vytvorený pracovníkmi vo výskumnej divízií spoločnosti Yahoo! v roku 2010 [6].

Porovnávacie úrovne

1. Úroveň – Výkonnosť

   Prvá úroveň sa zameriava na latenciu požiadaviek pri zaťaženej databáze. Výkonnosť je dôležitá, pretože ľudia sú netrpezliví pri čakaní načítania stránky. Existuje však kompromis medzi latenciou a priepustnosťou, kde pri danom nastavení hardvéru sa zvyšuje zaťaženie a latencia jednotlivých požiadaviek sa zvyšuje. Pri návrhu aplikácie sa potom rozhoduje o prijateľnej latencii aplikácie.

   Táto prvá úroveň má za cieľ určiť tento kompromis pre každý databázový systém meraním latencie pri zvyšovaní výkonnosti až do bodu, v ktorom sa zastaví výkonnosť. Pre vykonanie tejto úrovne sa využíva generátor pracovnej záťaže, ktorý slúži dvom účelom, a to: 1. definuje množinu údajov a načítava ich do databázy, 2. vykonáva operácie proti dátovému súboru pri meraní výkonu. Pre tieto účely slúži klient YCSB. Množina parametrových súborov definuje charakter údajov a operácie (transkacie), ktoré sa vykonávajú voči údajom. Klient YCSB umožňuje užívateľovi definovať ponúkanú priepustnosť ako parameter príkazového riadku a hlásiť výslednú latenciu [5].

2. Úroveň – Škálovateľnosť

   Keďže aplikáciám rastie popularita a pridáva sa do nich nová funkcionalita, cloudové systémy sú schopné meniť sa v závislosti od záťaže. Úroveň škálovateľnosti pri databázach skúma dopad na výkon pri pridaní serverov do systému [5]. V tejto vrstve sa merajú dve metriky:

   (a) Zvyšovanie mierky

   V prvom prípade sa načítava daný počet serverov s dátami na ktorom sa spúšťa určité zaťaženie. Po tomto teste sa vymažú údaje, pridajú ďalšie servery a načítava sa väčšie množstvo dát do väčšieho clustra, kde sa potom spúšťa znova zaťaženie. Ak má databázový systém dobré vlastnosti z hľadiska veľkosti, výkonnosť by mala zostať konštantná.

   (b) Elastické zrýchlenie

   V druhom prípade sa zisťuje to, ako funguje databáza ak počet serverov narastá, kým systém beží. Najprv sa načíta daný počet serverov s dátami a opäť sa spustí zaťaženie. Postupne sa pridá jeden alebo viac serverov a pozoruje sa ich vplyv na výkonnosť. Systém, ktorý ponuka dobru elasticitu, by mal ukázať zlepšenie výkonu pri pridávaní nových serverov s krátkym alebo neexistujúcim obdobím narušenia, kým sa systém sám prekonfiguruje na používanie nového servera.

Zhrnutie

Tento framework nebudem môcť využiť pri svojej práci, pretože YCSB je špecifický framework zameraný na cloudové databázy a NoSQL systémy. Táto práca sa zameriava na meranie výkonu systému CloverETL a nie meranie výkonu rôznych databází. Z toho vyplýva, že by bolo zbytočné využívať tento porovnávací nástroj, keďže výsledná aplikácia má poskytovať hromadné výsledky o behu systému CloverETL. Je potrebné pokryť rôznorodosť systému CloverETL, ktorý nie je databázový nástroj, ale nástroj pre spracovanie dát.

4.2 CloudRank-D

CloudRank-D je porovnávacia sada používaná na hodnotenie súkromného cloudového systému, ktorý je zdieľaný pre bežiace aplikácie na spracovanie dát. Súkromný cloud je typ cloud computingu, ktorý poskytuje podobné výhody ako verejný cloud vrátane škálovateľnosti a samoobsluhy. Na rozdiel od verejných cloudov, ktoré poskytujú služby viacerým organizáciám, je súkromný cloud venovaný potrebám a cieľom jednej organizácie [17]. CloudRank-D poskytuje dve metriky týkajúce sa spracovania údajov:

- Údaje spracované za sekundu – je definovaná ako celkové množstvo dátových vstupov všetkých úloh vydelených celkovým časom prevádzky od času odoslania prvej úlohy až po konečný čas poslednej úlohy.
- Údaje spracované za joul – je definovaná ako celkové množstvo dátových vstupov všetkých úloh vydelených celkovou spotrebou energie počas trvania od času odovzdania prvej úlohy do dokončenia poslednej úlohy.

Z definícií metód je jasné, že namerané metriky výkonnosti testovaného systému závisia nielen od výpočtovej rýchlosti procesorov, ale aj od efektívnosti prístupu k pamäti, disku a

sieti. CloudRank-D poskytuje súbor 13 reprezentatívnych nástrojov na analýzu dát. Na základe charakteristík výpočtu zameraného na údaje, klasifikujeme kritériá CloudRank-D do štyroch základných kategórií: transformácia, agregácia, zhrnutie, expanzia a jedna odvodená kategória hybrid. Tento porovnávací balík sa využíva pre tri účely [17].

- Kvantitatívne metriky merajú výkonnosť rôznych systémov a najmä merajú, o koľko jeden systém prekonáva iný systém.
- Metriky môžu usmerniť optimalizáciu testovaného systému.
- Pomocou metrík môžeme ohodnotiť rôzne systémy.

Zhrnutie

Táto porovnávacia sada, slúži na hodnotenie súkromného cloudového systému. Výsledná aplikácia tejto práce však beží lokálne a slúži pre spracovanie dát. Z toho vyplýva, že CloudRank-D nie je vhodný pre použitie pri riešení problému, ktorým sa zaoberá táto diplomová práca.

4.3 Phoronix Test Suite

Phoronix Text Suite (PST) je najkomplexnejšia testovacia a porovnávacia platforma dostupná pre operačné systémy Linux, OS X, Windows, Solaris a BSD. Phoronix Test Suite umožňuje vykonať testy plne automatizovaným spôsobom, od inštalácie testov až po vykonanie a reportovanie. Všetky testy sú ľahko reprodukovateľné, ľahko použiteľné a podporujú plne automatizované vykonávanie. Phoronix Test Suite je open source pod licenciou GNU GPLv3 a bol vyvinutý firmou Phoronix Media v spolupráci s partnermi [25].

Phoronix Test Suite je open-source framework pre vykonávanie automatizovaných testov, oznamovanie výsledkov týchto testov, detekovanie nainštalovaného systémového softvéru/hardvéru a ďalších funkcií. Moduly tohto frameworku tiež podporujú regresné testy. Tento framework je navrhnutý tak, aby bol rozšíriteľný, čo znamená, že pridávanie nových testovacích profilov a súborov je jednoduché. Tieto nové testovacie profily a súbory sa potom môžu používať pre výkonnostné testovanie, unit testovanie a kvalitatívne a kvantitatívne testovanie. V spolupráci s platformou OpenBenchmarking.org je štandardne k dispozícií viac ako 200 individuálnych testovacích profilov a viac ako 60 testovacích súborov. Tieto testy sú zamerané na rôzne prvky systémov ako je rýchlosť pamätí, vstupno/výstupné operácie, testovanie grafických operácií a rôzne iné. Testovací profil pozostáva zo súboru Bash/Shell skriptov a súborov XML a testovací súbor je jediný XML súbor. OpenBenchmarking.org umožňuje uskutočňovanie vedľajších porovnávaní výsledkov, centrálne správu pre ukladanie a zdieľanie výsledkov testov a spoluprácu na testovacích dátach [24].

Phoromatic je doplnková platforma pre OpenBenchmarking.org a Phoronix Test Suite pre rozhranie programu Phoronix Test Suite na automatické vykonávanie testovacích cyklov na základe časového spúšťanie, spúšťania po commite alebo iných spúšťaní. Phoromatic umožňuje jednoduchú správu viacerých sieťových systémov s klientmi Phoronix Test Suite prostredníctvom jedného webového rozhrania [25].

Zhrnutie

Phoronix Test Suite je vhodný pre testovanie prostredia na ktorom bude následne systém CloverETL evaluovať ETL Grafy. Je skutočne výhodné, že Phoronix Test Suite spolupracuje

s OpenBenchmarking.org, kde sa nachádzajú už vytvorené a voľne dostupné testy pre rôzne platformy. Ďalšou výhodou je možnosť vytvárania testovacích scenárov, ktorými si budú môcť užívatelia otestovať svoj systém. Výsledky týchto testov sú reportované v užívateľsky čitateľných súboroch HTML. Jediným problémom však môže byť závislosť na skriptovacom jazyku PHP. Táto platforma môže pomôcť odhaliť problémy, ktoré priamo ovplyvňujú spracovanie dát pri používaní systému CloverETL.

4.4 Profiler Profiler4j

Profiler4j je jednoducho použiteľný CPU Java profiler, ktorý podporuje vzdialené profilovanie a zmenu konfigurácie za behu [11]. Tento profiler pracuje na Jave 5 a novšej. Profiler Profiler4j vznikol ako študentský projekt v roku 2006. Posledná aktualizácia zdrojových kódov bola v roku 2013.

Hlavnými rysmi tohoto profilera sú:

- založený na dynamických bytcode inštrukciách
- 100% Java
- nepotrebuje žiadnu natívnu knižnicu alebo spustiteľný súbor
- poskytuje zobrazenia s volaniami, pamäťové monitorovanie a zoznam tried
- možnosť upraviť konfiguráciu za behu, bez potreby reštartovania profilovanej JVM.

Zhrnutie

Tento profiler používa grafické užívateľské rozhranie a pripojí sa na bežiacu JVM, ktorú chce profilovať. Toto však pri grafových testoch v tomto projekte nebude možné, keďže CloverETL Engine sa spustí a ihneď začne spracovávať graf. Tým by sa prišlo o značnú časť výsledkov, ktoré by mohli mať veľký vplyv na konečné výsledky.

4.5 Profiler JAMon API

Java Application Monitor (JAMon) je bezplatný pod BSD licenciou, vysoko výkonný, vlákonovo bezpečný Java API profiler, ktorý umožňuje vývojárom ľahko monitorovať produkčné aplikácie. JAMon sa môže použiť na určenie prekážok v oblasti výkonnosti aplikácií, interakcií medzi používateľmi aplikácií a škálovateľnosti aplikácií.[15].

- JAMon umožňuje developerom sledovať výkonnosť aplikácií a správanie pomocou preddefinovaných modulov ako sú SQL, HTTP žiadostí o stránky, Springové beany, spúšťanie metód, Log4j a výnimiek [15].
- JAMon zhromažďuje súhrne štatistické udaje, ako sú napríklad prístupy, časy vykonávania (celkový, priemerný, minimálny, maximálny, štandardná odchýlka) [15].
- JAMon je rýchly a nepoužíva veľké množstvo pamäte, takže ho je možné použiť v produkcii [15].

Zhrnutie

Profiler JAMon bol primárne vyvinutý pre monitorovanie webových aplikácií. Pri testovaní tohoto profileru boli poskytnuté dáta o dĺžke behu a taktiež o veľkosti použitia pamäte, čo sa zisťovalo za pomoci rozhrania JMX. Pre presnejšie výsledky by bolo nutné prekompilovať knižnicu CloverETL Engine, aby sa mohli monitory spustiť pri spustení konkrétnych CloverETL komponentov. To by nám poskytlo presnejšie časové údaje behu jednotlivých CloverETL komponentov.

4.6 Profiler JMeasurement

JMeasurement je voľne dostupné Java API pre monitorovanie behu a sledovanie užívateľsky definovaných aspektov v produkčnom kóde. Tento profiler je jednoduchý na použitie, ľahko rozšíriteľný a podporuje JMX. JMeasurement môže pomôcť ukázať v bežiacej aplikácii, ako dlho sú spustené niektoré definované časti a ako často sa volali. Tento softvér je open-source a nachádza sa pod licenciou Apache 2.0 [16]. ProfilerJMeasurement je projekt, ktorý vznikol už v roku 2005.

Vlastnosti:

- možnosť filtorvania v monitorovaní
- perzistencia v XML
- monitorovanie rozhraní
- monitorovanie výnimok
- spustenie a vypnutie za behu
- dáta su prístupné za behu
- automatické obnovenie po zmene konfiguračného súboru, ktorý je na classpath
- podpora JMX

Zhrnutie

Profiler JMeasurement po testovaní poskytol podobné informácie ako predchádzajúci profiler JAMon. Tento profiler však neposkytol informácie o pamäti, ale iba o dĺžke behu. Taktiež by bolo nutné prekompilovanie knižnice CloverETL Engine, aby sa mohli monitorovať jednotlivé CloverETL komponenty.

4.7 Profiler Infrared

InfraRED je nástroj pre monitorovanie výkonu J2EE aplikácií a diagnostiku výkonnostných problémov. Zhromažďuje metriky o rôznych aspektoch výkonnosti aplikácie a sprístupňuje ich kvantitatívnu analýzu. Tento nástroj má schopnosť monitorovať komplexnú architektúru J2EE aplikácií, poskytovať podrobné informácie pre analýzu, upozorňovať na problémy súvisiace s výkonom a vedie k tomu, aby bolo možné určiť hlavnú príčinu problému. InfraRED používa aspketovo-orientované programovanie na väzbu kódu monitorovanie výkonu do aplikácie [14].

Vlastnosti:

- nevyžaduje žiadne programovanie
- štatistiky výkonnosti
- štatistiky JDBC API a SQL
- kolerácia štatistických údajov medzi jednotlivými vrstvami
- užívateľské GUI
- podpora pre viac aplikačných serverov
- export štatistík do Excel tabuliek

Zhrnutie

Profiler InfraRED sa používa pre monitorovanie výkonu J2EE aplikácií. Tento profiler však v súčastnosti už nie je aktuálny, pretože posledná aktualizácia bola ešte v roku 2006.

Kapitola 5

Návrh aplikácie

Cieľom tejto práce je vytvoriť aplikáciu, ktorá vykoná meranie výkonu a profilovanie systému CloverETL. Po zozbieraní nameraných dát sa vykoná analýza, ktorej cieľom bude zistiť priepustnosť systému a jeho častí. Ďalšou súčasťou je sledovanie zmeny výkonnosti, na základe ktorých je potom možné určiť obmedzenia systému. To by malo pomôcť pre zvýšenie efektívnosti systému CloverETL. Pri nesprávnom návrhu a následnej nesprávnej implementácii môže byť výstup aplikácii zavádzajúci.

5.1 Popis problému

Výsledná aplikácia by mala ohodnotiť výkonnosť systému pre dátovú transformáciu a následne porovnať hodnoty rovnakého systému v inom prostredí. Systémy, ktoré vykonávajú dátové transformácie je zložité priamo porovnať podľa nameraných hodnôt, pretože okolité prostredie má priamy dopad na výkonnosť systému. Preto vznikla táto práca, ktorej cieľom je na základe nameraných hodnôt odhadnúť maximálnu priechodnosť systému, sledovať možné zmeny výkonnosti systému a určiť obmedzujúce hodnoty behového prostredia.

Problémy, ktoré sú známe:

- Podporovanie platforiem Windows, MacOS, a Linux, pretože systém CloverETL taktiež podporuje tieto platformy. Prostredie, na ktorom systém beží má priamy dopad na výkonnosť systému.
- Závislosť a komplexita prostredia, na ktorom systém beží, kde môže nastať problém napríklad načítaní alebo ukladaní dát do súborov, ktoré môžu mať rozdielny formát.
- Určenie ako sa vyhodnotí systém z nameraných hodnôt a či bude možné vždy dosiahnuť presné a jasné výsledky.
- Vytvorenie testovacích ETL Grafov, ktoré má systém pri testovaní spúšťať tak, aby boli schopné pokryť širokú škálu použiteľnosti systému CloverETL.

5.2 Architektúra

Návrh aplikácie pozostáva z troch rôznych modulov, kde každý modul bude spustiteľný samostatne. Okrem hlavných úloh na meranie výkonu budú prvé dva moduly obsahovať časti, ktoré konkrétny modul nainštalujú. Po nainštalovaní tieto dva moduly získajú dáta, o prostredí a o behu systému CloverETL. Následne tretí modul bude schopný vyhodnotiť a vizualizovať namerané dáta. Každý modul bude spustiteľný samostatne len vtedy, ak bude spustený s požadovanými parametrami.

- Meranie výkonu prostredia, na ktorom bude systém CloverETL pracovať – Získanie informácií o prostredí. Výsledkom je súbor vo formáte HTML.
- Profilovanie systému CloverETL – Určenie efektívnosti spracovania systému CloverETL na testovacích transformačných ETL Grafoch. Výsledkom je súbor vo formáte JSON.
- Analýza výstupov, určenie výsledkov a vizualizácia – Vyhodnotenie celého behu predchádzajúcich modulov a určenie obmedzujúcich prvkov systému CloverETL na základe behu na danom prostredí. Výsledkom bude HTML súbor z prvého modulu a súbor vo formáte XML, z ktorého sa vytvorí pomocou XSL šablóny HTML stránka s výsledkami, z druhého modulu. Tieto výsledky bude možné porovnať s referenčnými a tak zistiť rozdiely medzi aktuálnymi a referenčnými výsledkami.

5.2.1 Meranie výkonu prostredia

Meranie výkonu prostredia bude vykonané za pomoci platformy Phoronix Test Suite, ktorá je open source. Prvým z dôvodov je to, že Phoronix Test Suite je platforma dostupná pre všetky potrebné operačné systémy. Druhým dôvodom je existencia rôznych testov a ďalším dôvodom je aj rozšíriteľnosť tohoto frameworku, takže existuje možnosť prispôsobiť testy potrebám tejto práce alebo možnosť vytvorenia nových testov. Táto platforma spolupracuje s platformou OpenBenchmarking.org, kde je už štandardne k dispozícií viac ako 200 testovacích profilov a viac ako 60 testovacích súborov. Výsledkom jedného testu je HTML stránka, na ktorej sa nachádzajú informácie o stroji (typ procesora, chipset, veľkosť a typ pamäte RAM, veľkosť a typ pevného disku, typ operačného systému a iné.), na ktorom testy bežali a dĺžka behu jednotlivých testov.

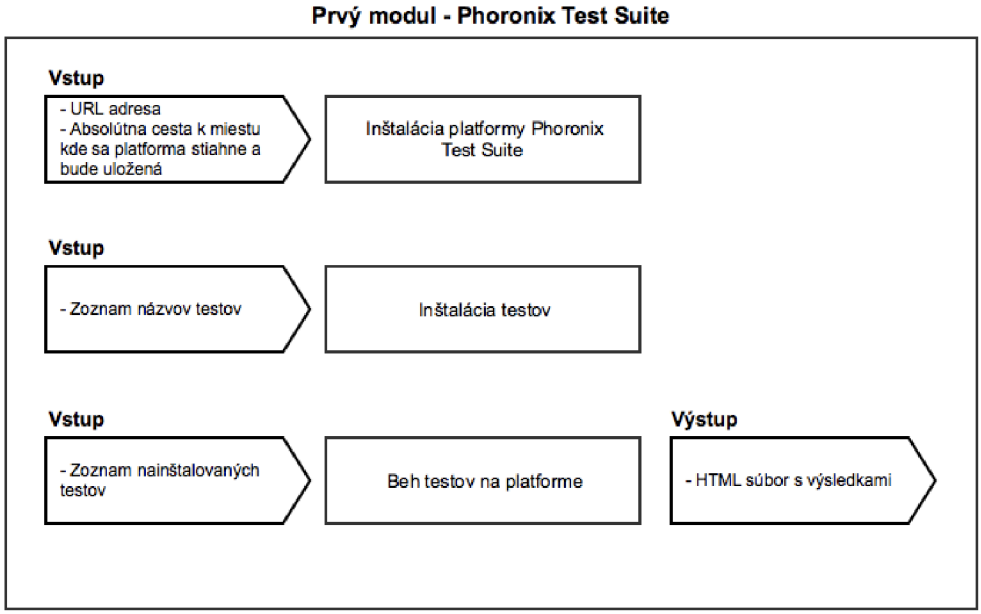
Jediná závislosť, ktorú platforma Phoronix Test Suite potrebuje je skriptovací jazyk PHP pre príkazovú riadku od verzie 5.3 alebo novšej. Tento skriptovací jazyk používa Phoronix Test Suite pre XML operácie, kompresiu a dekompresiu súborov, HTTPS komunikáciu medzi Phoronix Test Suite a servermi OpenBenchmarking.org, multivláknové monitorovanie, získavanie informácií o systéme, na ktorom test beží, podporu sieťovej výkonnosti a sťahovanie testovacích súborov, porovnanie dát, generovanie obrázkov pri grafických testoch a generovanie PDF reportov.

Pri prvom spustení bude potrebné stiahnuť a nainštalovať platformu Phoronix Test Suite. To bude možné priamo cez túto aplikáciu, kde vstupom je URL adresa, z ktorej sa má Phoronix Test Suite stiahnuť a absolútna cesta k miestu na súborovom systéme, kde bude platforma uložená.

Ďalším krokom je nainštalovanie testov, ktoré budú spustené. Tento krok bude taktiež plne automatizovaný a jediným vstupom je zoznam testov, ktoré bude potrebné nainštalovať. Niektoré testy však nie sú podporované pre všetky platformy, preto je vhodné si vyhľadať tieto informácie na stránke OpenBenchmarking.org.

Phoronix Test Suite je platforma, ktorá už má k dispozícii radu rôznych testov. Niektoré z týchto testov, ako sú napríklad testy pamäte alebo procesora, sú vhodné využiť pri behu tohto modulu. Sám užívateľ si určí, ktoré z týchto testov budú spustené na systéme. Vstupom je zoznam testov, ktoré už sú stiahnuté a nainštalované pre potreby platformy. Následne prebehne beh týchto testov a výstupom tohto modulu budú výsledky, ktoré budú pozostávať z informácií o stroji, na ktorom platforma bežala a taktiež času behu jednotlivých testov. Výsledky budú prezentované ako HTML stránka, ktorá je priamo vygenerovaná platformou Phoronix Test Suite.

Obrázok 5.1 znázorňuje vstupy a výstupy jednotlivých častí prvého modulu, ktorý meria výkonnosť prostredia.



Obr. 5.1: Vstupy a výstupy jednotlivých častí prvého modulu

5.2.2 Profilovanie systému CloverETL

Samotné profilovanie systému CloverETL bude vykonávané za pomoci Java profileru a Java MXBean objektov. V tejto časti bude možné použiť profiler Java Application Monitor (JAMon). MXBean objekty sa budú využívať pre zisťovanie aktuálnych informácií ohľadne pamäti, zaťaženia procesora a monitorovanie deskriptorov súborov.

V tejto časti je potrebné vytvoriť ETL Grafy, ktoré budú priamo spúšťané cez knižnicu CloverETL Engine čo znamená, že sa nebude používať grafické užívateľské rozhranie (CloverETL Designer). Tieto ETL Grafy je potrebné vytvoriť tak, aby ich spracovanie obsahovalo čo najväčšie množstvo rôznych operácií. Preto sa vytvorí niekoľko ETL Grafov, ktoré budú používať rôzne komponenty pre testovanie rôznych operácií. To by malo zabezpečiť celkový pohľad na prostredie, na ktorom bude aplikácia spustená.

Typy úloh, ktoré budú testované týmto nástrojom:

- čítanie dát zo súboru (komponenta FlatFileReader)
- zápis dát do súboru (komponenta FlatFileWriter)

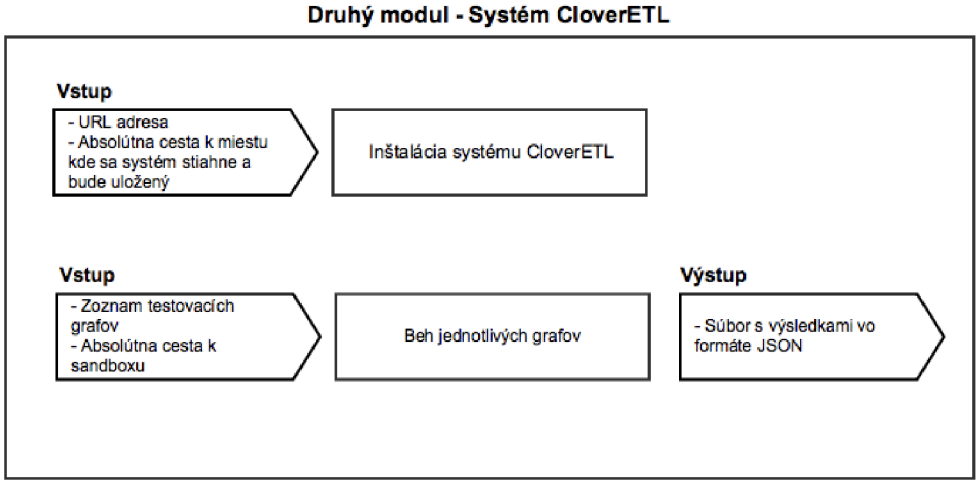
- radenie dát (komponenta FastSort)
- agregovanie dát (komponenta Aggregate)
- kombinovanie dát (komponenta Combine)
- spracovanie, pretransformovanie a vytvorenie súboru JSON (ETL Graf z CloverETL príkladov)
- spracovanie, pretransformovanie a vytvorenie súboru XML (ETL Graf z CloverETL príkladov)

Tieto grafy je však možné voľne upravovať alebo vytvárať, a to po stiahnutí a nainštalovaní aplikácie CloverETL Designer s trialovou licenciou.

Pri prvom spustení bude potrebné podobne ako pri predchádzajúcom module stiahnuť a nainštalovať knižnicu CloverETL Engine. Pre vykonanie tejto inštalácie bude potrebné zadať URL adresu, z ktorej sa má stiahnuť knižnica CloverETL Engine a absolútnu cestu k miestu na súborovom systéme, kde bude knižnica uložená.

Hlavnou časťou tohto modulu je určite testovanie behu jednotlivých ETL Grafov. V tejto časti budú postupne evaluované jednotlivé ETL Grafy, ktorých beh bude monitorovaný pomocou MXBean objektov. Každý jeden ETL Graf bude spustený defaultne trikrát (užívateľsky definovaný počet), a to preto, aby sa lepšie odhadli výsledky vypočítaním priemerných hodnôt. Informácie o pamäti, zaťaženia procesora a počtu deskriptorov súborov sa budú aktualizovať defaultne každú pol sekundu, avšak tento interval bude konfigurovateľný. Vstupom tohto modulu je zoznam ETL Grafov, absolútnej cesty k sandboxu a absolútnej cesty k miestu na súborovom systéme, kde bude súbor s výsledkami uložený. Následne sa budú evaluovať tieto ETL Grafy a výsledky z behu týchto grafov budú výstupom tohto modulu. Tieto výsledky budú uložené v súboroch vo formáte JSON, kde v jednom súbore budú všetky informácie ohľadne behu jedného ETL Grafu. Súčasťou týchto výsledkov budú čas behu ETL Grafu, informácie o pamäti, zaťaženia procesora a počte deskriptorov súborov.

Obrázok 5.2 určuje vstupné a výstupné dáta pre rôzne časti druhého modulu.



Obr. 5.2: Vstupy a výstupy jednotlivých častí druhého modulu

5.2.3 Analýza výsledkov a vizualizácia

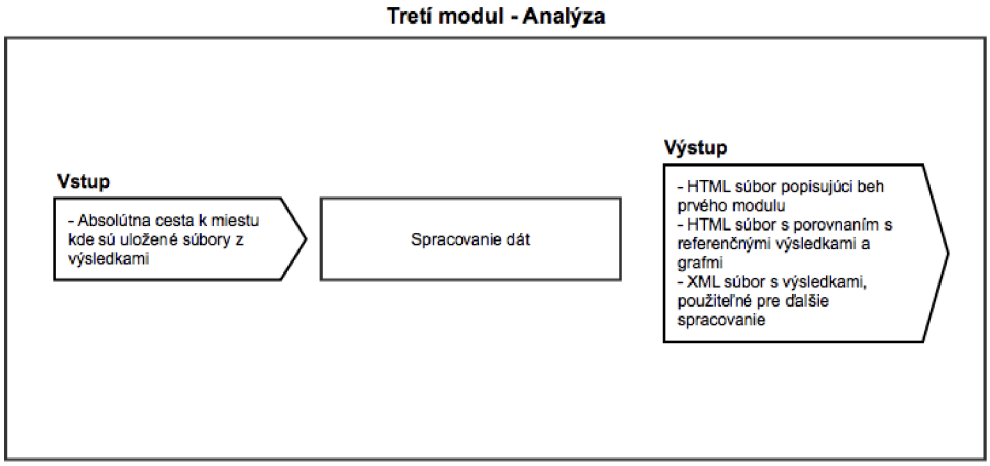
Posledným modulom aplikácie bude modul, ktorý zanalyzuje výsledky a vytvorí vizualizáciu pre určité typy dát. Táto vizualizácia by mala byť zrozumiteľná pre užívateľa.

Výstupom prvého modulu budú HTML súbory. Keďže je možné vytvoriť si vlastné testy a niektoré testy sú podporované len pre určité konkrétne operačné systémy, tento súbor bude už priamo jedným z výsledkov aplikácie. Súbor HTML dobre prezentuje výsledky prvého modulu. Každý jeden test, ktorý bol spustený v prvom module, bude reprezentovať výsledok uložený v súbore. V týchto výsledkoch sú očakávané informácie o systéme, na ktorom aplikácia beží. Ďalšími výsledkami prvej časti sú výsledky z testov, ktoré sú vytvorené a voľne dostupné pre platformu Phoronix Test Suite. Tieto testy sa budú používať pre testovanie vstupno/výstupných operácií alebo pamäte RAM. Na základe týchto informácií sa bude dať určiť, ktoré prvky systému sú nevyhovujúce, takzvaný najslabší článok, a ktoré prvky by bolo potrebné vymeniť pre zlepšenie efektívnosti spracovania dát.

Výstupom druhého modulu budú súbory vo formáte JSON. V tomto súbore sa bude nachádzať množina dát, ktorá sa bude merať každú pol sekundu (ak nie je konfiguráciou určené inak). Tieto výsledky by už priamo ukázali, ktoré časti spracovania dát v systéme CloverETL sú najefektívnejšie, a ktoré nie sú. Tieto výsledky odhalia problémy s priepustnosťou dát poprípade určia obmedzenia. Taktiež bude na základe výsledkov možné sledovať zmenu výkonnosti a to vďaka zmene konfigurácie systému a opätovnému spusteniu aplikácie.

Súbory vo formáte JSON postupne prejdu spracovaním, kde sa vypočíta priemerná hodnota pre každý element a určia sa štatistiky z behu ETL Grafu. Na základe údajov z druhého modulu, sa vytvoria rôzne grafy, ktoré budú reprezentovať zaťaženie procesora, zmena Java heap pamäte a aktivitu algoritmu garbage collector počas behu ETL Grafu. Výstupným súborom je XML súbor, kde budú uložené dáta, ktoré sa budú dať ďalej automatizovane spracovať. Vďaka formátu XML sa tento súbor bude dať ďalšími nástrojmi (napríklad Jenkins) spracovať a použiť pre regresné testy. Tento súbor zlepší regresné testovanie, a tým sa zlepší a urýchli vývoj softvéru. Ďalším výstupným súborom bude HTML súbor, ktorý bude pozostávať z informácii o danom systéme, z údajov z XML súboru a vytvorených grafov. Tento súbor bude automaticky vygenerovaný na základe XSL šablóny z výstupného XML súboru a obrázkových grafov. Výsledné dáta budú zmena počtu vlákien počas behu, maximálna a minimálna veľkosť pamäte, zaťaženia procesora, počet deskriptorov súborov. HTML súbor je možné rozšíriť o referenčné výsledky a na základe týchto hodnôt sa vypočíta odchylka medzi nameranými a referenčnými. Ak odchylka prevýši prah určený v konfiguračnom súbore, tieto dáta sa vyznačia červenou (veľká odchýlka - chyba) alebo žltou (menšia odchýlka, potencionálny problém - upozornenie) farbou. Vždy teda bude viditeľný rozdiel medzi aktuálnymi dátami a nameranými dátami.

Na obrázku 5.3 sa nachádzajú vstupné a výstupné súbory pri behu tretieho modulu aplikácie.



Obr. 5.3: Vstupy a výstupy tretieho modulu

5.2.4 Metriky

Metriky, ktoré budú použité v tejto aplikácii, môžeme rozdeliť na dve kategórie a to metriky prvého modulu (meranie výkonu prostredia) a metriky druhého modulu (profilovanie systému CloverETL). Na získanie výsledných metrík slúži tretí modul.

- Prvý modul:

  Čas behu testov Phoronix Test Suite – Užívateľsky definované testy vygenerujú HTML súbory, v ktorom sa nachádzajú časy behov jednotlivých testov. Tieto výsledky je možné porovnať s rôznymi inými systémami pomocou platformy OpenBenchmarking.org, kde sa už nachádzajú výsledky testov.

- Druhý modul:

  Čas behu ETL Grafu – Určuje čas behu jedného ETL Grafu od spustenia systému CloverETL.

  Veľkosť fyzickej pamäte – Celková veľkosť voľnej fyzickej pamäte.

  Veľkosť heap a non-heap pamäte – Určuje celkovú veľkosť heap a non-heap pamäte virtuálneho stroja.

  Počet vlákien – Určenie počtu vlákien a ich prípadnému únik počas behu ETL Grafov.

  Veľkosť CPU času – Celková veľkosť času spotrebovaná vláknom, ktoré spúšťa ETL Grafy.

  CPU čas procesu – Veľkosť CPU času použitého procesom, na ktorom virtuálny stroj beží.

  Deskriptory súborov – Počet otvorených deskriptorov súborov reprezentuje reálny počet otvorených súborov v systéme a maximálny počet deskriptorov súborov určuje maximálny možný počet na systéme.

  Aktivita algoritmu garbage collector – Znázorneňuje periodicitu spúšťania algoritmu garbage collector pre potreby uvoľnenia pamäte. Rozlišuje sa mladá a stará generácia algoritmu garbage collection.

Veľkosť voľnej fyzickej pamäte - Zmena veľkosti fyzickej pamäte pred a po spustení ETL Grafov.

Zaťaženie procesora – Zaťaženie procesora v percentách reprezentuje, akou časťou beh ETL Grafov zaťažil procesor systému.

Pamäť heap – Znázornenie zmeny veľkosti pamäte heap počas behu ETL Grafov.

Aktivita algoritmu garbage collector – Znázornenie periodicity spúšťania algoritmu garbage collector pre potreby uvoľnenia pamäte.

Kapitola 6

Implementácia

Podľa návrhu z kapitoly 5 je aplikácia rozdelená na tri moduly. Prvý modul, ktorý slúži na testovanie prostredia, je zavislý na platforme Phoronix Test Suite. Ďalší modul, ktorý profiluje systém CloverETL, využíva voľne dostupný CloverETL Engine, na ktorom sa spúšťajú ETL Grafy. Posledným modulom je modul na analýzu, spracovanie a vyhodnotenie výsledkov. Celá výsledná aplikácia je naprogramovaná v programovacom jazyku Java. Úvod tejto kapitoly 6.1 obsahuje popis použitých technológií, nástrojov a knižníc. V ďalšej časti 6.2 je vysvetlená štruktúra samotnej aplikácie.

6.1 Použité nástroje

Aplikácia pre profilovanie systému CloverETL je implementovaná v programovacom jazyku Java. To, že celý systém CloverETL je napísaný v jazyku Java, bolo hlavnou motiváciou k použitiu tohto programovacieho jazyka aj pri implementovaní tejto práce. To dáva lepšiu možnosť prepojenia tejto aplikácie so systémom CloverETL a lepšie možnosť využiť a profilovať tento systém. Hlavnou časťou pre evaluovanie ETL Grafov je CloverETL Engine, ktorý je použitý aj pre spracovanie ETL Grafov v tejto aplikácii. Výsledná aplikácia ponúka sadu ETL Grafov, ktoré boli testované v rámci tejto práce. Tieto ETL Grafy boli vytvorené pomocou aplikácie CloverETL Designer. Pre lepšie meranie výkonu prostredia, na ktorom bude systém CloverETL spracovávať dáta, táto aplikácia umožní spustiť testy pomocou platformy Phoronix Test Suite. V rámci implementácie boli využité rôzne voľne dostupné knižnice, a to či už pre prácu s JSON súbormi, pre vytváranie výsledných grafov alebo prácu s logmi. Implementácia CloverETL Enginu od spoločnosti Javlin využíva rôzne iné knižnice, ktoré však neboli využité v tejto aplikácii, preto tu nebudú uvedené.

Programovací jazyk Java

Jazyk Java je jeden z mnohých programovacích jazykov, ktorého kľúčové vlastnosti sú jednoduchosť, multiplatformovosť a objektová orientácia. Štandardný spôsob spracovania programu v Jave prechádza piatimi fázami a to editovaním, prekladom, zavedením, overovaním a vykonávaním.

Virtuálny stroj (JVM - Java Virtual Machine) je modul, ktorý spracováva Java bajtkód. Bajtkód je jazyk virtuálneho stroja, ktorý je nezávislý na platforme. Trieda je základným stavebným kameňom, v ktorom sú definované premenné a metódy. Trieda je šablóna, ktorá nemá pridelenú pamäť a iba reprezentuje skupinu objektov, ktoré majú rovnaké vlastnosti. Názvy tried sú zložené z názvu balíčka, v ktorom sa trieda nachádza a názvu triedy. Objekt

je dátový prvok, ktorý je vytvorený podľa vzoru triedy. Objekt je inštancia triedy a má alokovanú pamäť. Trieda môže mať ľubovoľný počet inštancií. Objekty sa posielajú medzi sebou správy, ktoré spracovávajú metódami. Balíček slúži k združovaniu tried, ktoré spolu logicky súvisia. Balíčky sú hierarchicky usporiadané a môžu obsahovať ďalšie balíčky [12]. Výsledná aplikácia bola vyvíjaná na Java 8.

Platforma Phoronix Test Suite

Ako už bolo spomínané v návrhu aplikácie (5), hlavnou úlohou platformy Phoronix Test Suite v tejto aplikácii je zmerať výkonnosť prostredia. K existujúcim testom, ktoré sú voľne dostupné pre rôzne platformy, ponúka Phoronix Test Suite aj možnosť vytvorenia vlastných testov. Táto aplikácia ponúka možnosť inštalácie a spustenia testov tejto platformy.

CloverETL Designer

CloverETL Designer, popísaný v časti 3.1, je nástroj, v ktorom boli implementované ETL Grafy. Tento nástroj je voľne dostupný v trialovej licencii a obmedzenej funkcionalite. CloverETL Designer umožňuje ľahko vytvoriť ETL Grafy, ktoré následne môžu byť vstupom pre druhý modul tejto aplikácie, ktorý bude daný ETL Graf profilovať.

Knižnica Jackson

Jackson je efektívna a populárna Java knižnica, ktorá slúži na serializáciu a mapovanie Java objetov do formátu JSON. Okrem toho slúži táto knižnica aj na parsovanie JSON súborov [7]. Jackson podporuje mnoho spôsobov pre prácu, vrátane jednoduchého mapovania objektov až po zložitejšie mapovania za pomoci anotácií. Knižnica Jackson je voľne dostupná knižnica pod licenciou Apache 2.0. V rámci tejto aplikácie je táto knižnica využitá na mapovanie objektov do formátu JSON, vytvorenie a parsovanie JSON súborov, v ktorých sa poskytujú dáta medzi druhým a tretím modulom.

Knižnica JAXB

Java Architecture for XML Binding (JAXB) je API pre prácu s dokumentami XML. JAXB umožňuje prístup k údajom z XML, a to bez potreby spracovania XML a nevyžaduje prechádzanie stromom tak, ako je to napríklad v protokole DOM. V rámci tohto projektu je táto knižnica využitá pre vytvorenie XML súboru s dátami a taktiež pre transformáciu XML súboru na HTML súbor za pomoci XSL schémy.

Knižnica JFreeChart

JFreeChart je Java knižnica, ktorá umožňuje jednoducho vytvárať a zobrazovať grafy v Java aplikáciách. Knižnica môže byť použitá aj na strane klienta, aj na strane serveru a ponúka širokú škálu rôznych grafov, flexibilný dizajn a podporuje výstupy rôznych typov. JFreeChart je voľne dostupná knižnica pod licenciou GNU LGPL (Lesser General Public License) [10]. Táto knižnica je využitá pre vytváranie grafových výstupov aplikácie, kde sú znázornené rôzne merané hodnoty ako napríklad veľkosť pamäte alebo využitie CPU.

Knižnica Log4j

Vkladanie logovacích hlášok do aplikácií je dôležité, a to najmä pre následné ladenie aplikácií. Používanie logov môže byť niekedy jediným spôsobom hľadania chýb v aplikáciách [9]. Táto knižnica je taktiež voľne dostupná pod licenciou Apache Software License. V rámci výslednej aplikácii tohto projektu je pre logovanie využitá knižnica Log4j. Táto knižnica loguje informačné a chybové hlášky na štandardný výstup a do aplikačného logu.

Knižnica Commons Compress

Apache Commons Compress knižnica definuje API pre prácu s ar, cpio, Unix dump, tar, zip, gzip, XZ, Pack200, bzip2, 7z, arj, lzma, snappy, DEFLATE, lz4, Brotli, Zstandard, DEFLATE64 a Z files [8]. Knižnica rozdeľuje komponenty na kompresory a archivátory. Zatiaľ čo kompresory (un) komprimujú prúdy, ktoré zvyčajne ukladajú jediný záznam, archivátori sa zaoberajú archívmi, ktoré obsahujú štruktúrovaný obsah reprezentovaný inštanciami `ArchiveEntry`, ktoré zvyčajne zodpovedajú jednotlivým súborom alebo adresárom. V rámci tohoto projektu je knižnica využitá pri rozbaľovaní Phoronix Test Suite tar archívu a CloverETL Engine gzip archívu.

6.2 Štruktúra aplikácie

V tejto časti bude popísaná štruktúra implementovanej aplikácie a spôsob, akým sú prepojené jednotlivé moduly. Vytvorená aplikácia `profilingSystem` má nasledujúcu štruktúru balíkov:

```
profilingSystem
profilingSystem.properties
profilingSystem.module
profilingSystem.module.phoronix
profilingSystem.module.clover
profilingSystem.module.clover.data
profilingSystem.module.analysis
profilingSystem.module.analysis.data
profilingSystem.utils
```

Baliček `profilingSystem`

V balíčku `profilingSystem` sa nachádza hlavná trieda aplikácie (`Main.java`), ktorá je vstupným bodom do aplikácie. Následne sa v tomto balíku nachádzajú triedy, ktoré spracovávajú parametre aplikácie a na základe týchto spracovaných dát sa ďalej rozhoduje v triede `Program.java`, ktorá akcia bude nasledovať. Parametre aplikácie sú popísané v podkapitole 6.2.1.

Balíček `profilingSystem.properties`

Balíček obsahuje dve triedy, a to triedu `PropertiesManager.java` pre spracovanie konfiguračného súboru `config.properties` a triedu `SystemProperties.java`, ktorá reprezentuje načítané dáta. Tieto údaje, ako napríklad adresáre alebo konfiguračné parametre aplikácie,

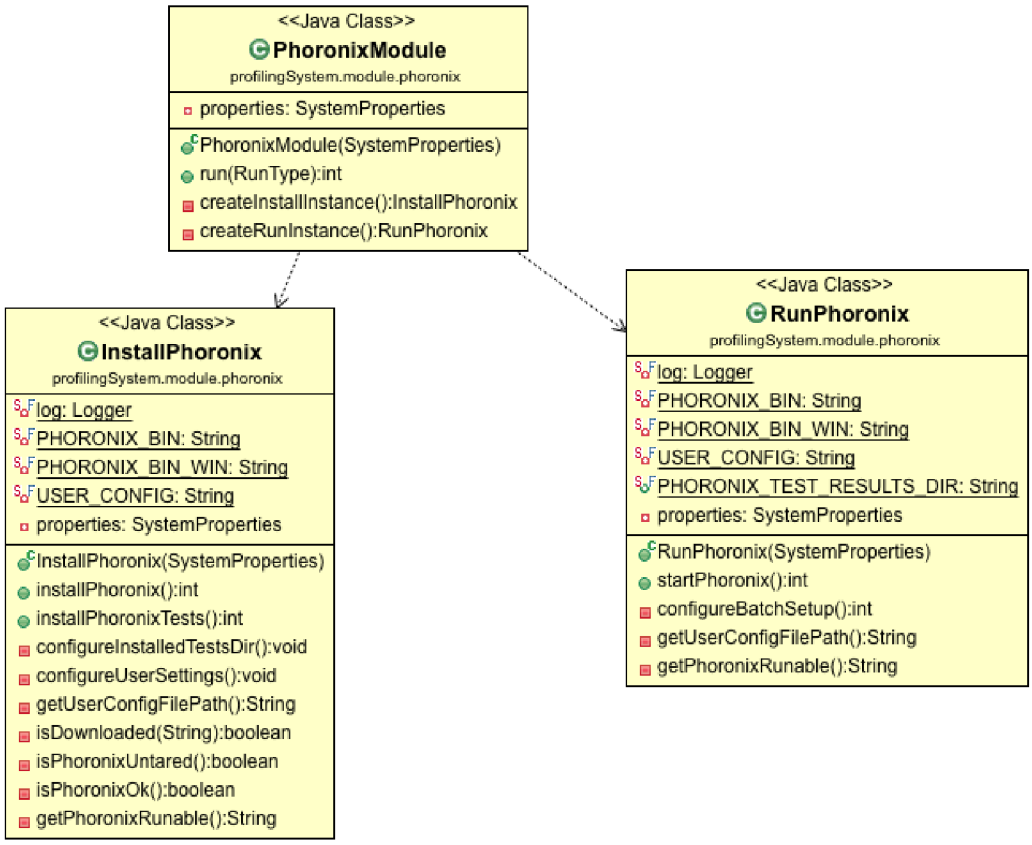
sú ďalej využité pri behu jednotlivých modulov. Spracovanie konfiguračného súboru prebieha vždy na začiatku behu aplikácie. Trieda `PropertiesValidator.java` slúži pre validovanie zadaných parametrov. V prípade chýbajúcich parametrov nebude daný modul spustený. Konfiguračný súbor bude bližšie popísaný v podkapitole 6.2.1.

Balíček `profilingSystem.module`

Toto je základný balíček aplikácie, ktorý rozhoduje a predáva riadenie už konkrétnym modulom. Pre túto činnosť existuje trieda `ModuleManager.java`, ktorá na základe vstupných parametrov vytvorí požadovaný modul a spustí jeho konkrétnu časť.

Balíček `profilingSystem.module.phoronix`

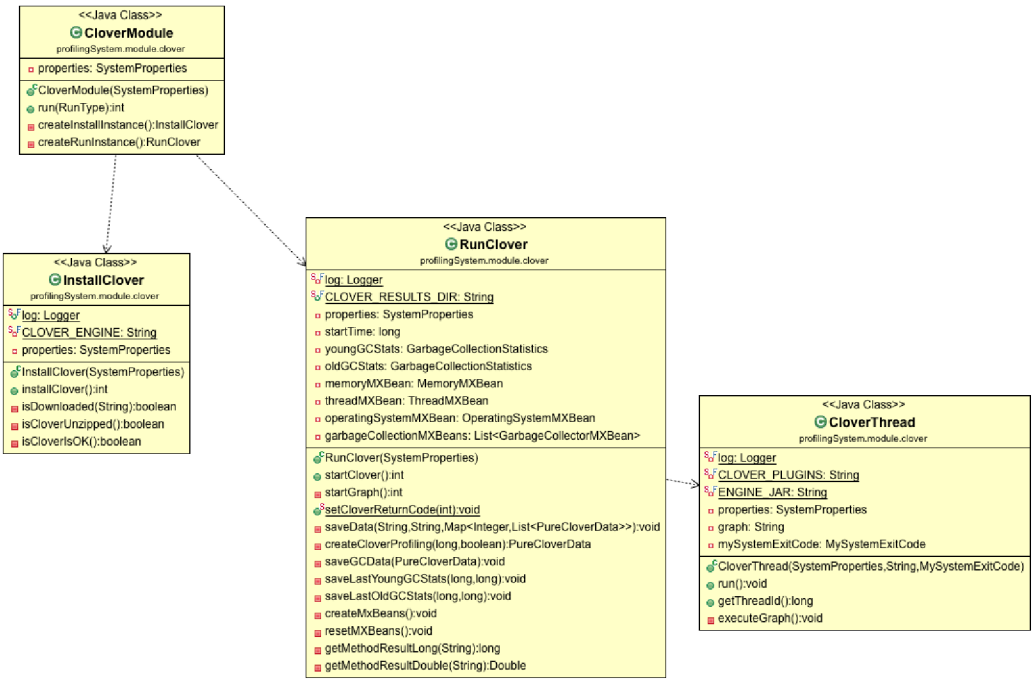
Prvým modulom aplikácie je modul pre testovanie prostredia na ktorom systém beží, ktorý reprezentuje trieda `PhoronixModule.java`. Ako je možné vidieť z obrázku 5.1, tento modul sa skladá z viacerých častí. Prvou časťou je inštalácia platformy Phoronix Test Suite. Táto platforma sa stiahne z webovej stránky a dekomprimuje do adresára definovaného v konfiguračnom súbore. Ďalšou časťou je inštalovanie testov platformy Phoronix Text Suite definovaných v konfiguračnom súbore aplikácie. Všetky potrebné inštalácie prebiehajú v triede `InstallPhoronix.java`. Následné spustenie týchto testov je implementované v triede `RunPhoronix.java`. Každú jednu časť tohto modulu je možné spustiť samostatne. Diagram hlavných tried prvého modulu je znázornený na obrázku 6.1.



Obr. 6.1: Diagram hlavných tried prvého modulu

Balíček `profilingSystem.module.clover` a balíček `profilingSystem.module.clover.data`

Profilovanie systému CloverETL prebieha v druhom module (trieda `CloverModule.java`). Tento modul sa skladá z dvoch častí (obrázok 5.2). Prvá časť nainštaluje CloverETL Engine z požadovanej webovej stránky (možnosť vybrať si verziu CloverETL Engine), ktorý sa následne dekomprimuje do požadovaného adresára. Inštaláciu obstaráva trieda `InstallClover.java`. Spustenie profilovania systému CloverETL sa nachádza v triede `RunClover.java`. Toto je jedna z najzložitejších tried tejto aplikácie, pretože sa tu nachádza pridávanie knižníc na classpath aplikácie za behu, spustenie ETL Grafu v inom vlákne (trieda `CloverThread.java`, kde prebieha proces behu ETL Grafu) a sledovanie požadovaných hodnôt za pomoci MXBean. Namerané dáta reprezentujú triedy v balíku `profilingSystem.module.clover.data`. Po úspešnom dobehnutí ETL Grafov sa vytvorí a uloží JSON súbor s nameranými dátami. Taktiež aj v tomto module sa každá časť spúšťa samostatne. Na obrázku 6.2 je znázornený diagram riadiacich tried druhého modulu aplikácie.

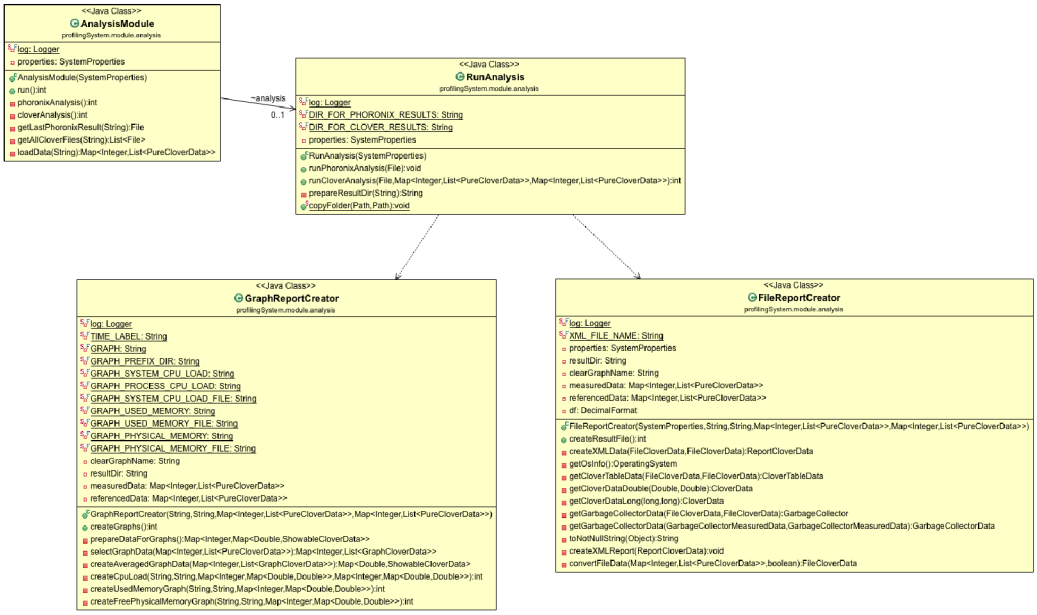


Obr. 6.2: Diagram hlavných tried druhého modulu

Balíček `profilingSystem.module.analysis` a balíček `profilingSystem.module.analysis.data`

Posledným modulom aplikácie je samotné vyhodnotenie meraní a vytvorení výsledkov. Implementácia tohto modulu sa nachádza v triede `AnalysisModule.java`. Z tejto triedy sa deleguje riadenie aplikácie do triedy `RunAnalysis.java`, ktorá vytávra výsledné adresáre. Výsledným reportom z prvého modulu je HTML stránka a výsledkom druhého modulu je XML súbor a HTML stránka. HTML report prvého modulu sa prekopíruje do adresára s výsledkami aplikácie. Trieda `GraphReportCreator.java` vytvára grafy, ktoré sú následne

súčasťou HTML reportu. Tieto grafy sa vytvárajú vypočítaním priemeru z hodnôt dodaných druhým modulom. Trieda `FileReportCreator.java` je zodpovedná za vytvorenie výsledného súhrnného XML a HTML reportu aplikácie na základe dát z druhého modulu. HTML report je vygenerovaná stránka pomocou XSL schémy z XML súboru. Nachádzajú sa v ňon výsledky o behu ETL Grafu, aktivita algoritmu garbage collection a grafy prezentujúce zaujímavé štatistiky. V konfiguračnom súbore je možné nastaviť hranicu, od ktorej sa majú výsledky počítať ako chyby (error) alebo podozrivé zmeny (warning). To umožňuje lepší prehľad v tomto súbore. Každá z týchto dvoch tried si spracováva potrebné dáta sama. Dáta, ktoré sú spracovávané v tomto module sú reprezentované triedami v balíčku `profilingSystem.module.analysis.data`. Obrázok 6.3 znázorňuje diagram hlavných tried tohto modulu. Ukážky všetkých grafických výstupov sa nachádzajú v prílohe D.



Obr. 6.3: Diagram hlavných tried tretieho modulu

Balíček `profilingSystem.utils`

V tomto balíčku sa nachádzajú rôzne praktické metódy, ktoré sú využité napriek celou aplikáciou. Prebieha tu sťahovanie súborov z webových stránok (trieda `DownloadUtil.java`), ďalej dekomprimovanie súborov (`ExtractUtil.java`) a nachádzajú sa tu taktiež aj rôzne metódy pre prácu so súbormi (`FileUtils.java`).

6.2.1 Spustenie aplikácie

Pre spustenie aplikácie je potrebné mať nainštalovanú Javu 8, skriptovací jazyk PHP (pre spustenie prvého modulu). Spustenie samotnej aplikácie prebieha z príkazovej riadky spustením aplikácie `profilingSystem.jar` s požadovanými parametrami. Pred prvým spustením je potrebné si definovať konfirguračné parametre do konfiguračného súboru. Ak chce užívateľ spustiť meranie výkonu prostredia za pomoci platformy Phoronix Test Suite, bude potrebné nainštalovať skriptovací jazyk PHP, ktorý táto platforma využíva. Pre spustenie tohto modulu na systéme Windows je potrebné definovať cestu k skriptovaciemu jazyku PHP v adresári `/phoronix-test-suite` v súbore `phoronix-test-suite.bat`. Príklad inštalácie a spustenia je popísaný v podkapitole 6.2.3. Pre spustenie modulov je však vždy potrebné dodržať vstupné požiadavky daných modulov popísané v kapitole 5 a znázornené v tabuľkách 6.1, 6.2, 6.3.

Celý beh aplikácie je sprevádzaný logovacími výpismi do príkazového riadku. Podľa týchto informácii je teda možné jednoducho určiť to, čo sa práve vykonáva na pozadí a aj to, v ktorej časti sa aplikácia práve nachádza. Tieto logovacie výpisy sú taktiež využiteľné pre odhalenie chýb spôsobených užívateľom. Logovacie výpisy taktiež vytvárajú súbor `application.log` v aktuálnom adresári, kde sa nachádzajú posledné informácie o behu aplikácie.

Parametre aplikácie

Aplikácia pracuje s nasledujúcimi parametrami:

`install [-config:<file>]` – Inštalácia Phoronix Test Suite, následne budú nainštalované testy pre túto platformu a ako posledné sa nainštaluje CloverETL Engine.

`installPhoronix [-config:<file>]` – Inštalácia Phoronix Test Suite.

`installPhoronixTests [-config:<file>]` – Inštalácia testov pre platformu Phoronix Test Suite.

`installClover [-config:<file>]` – Inštalácia systému CloverETL, konkrétne CloverETL Engine.

`start [-config:<file>]` – Spustenie nainštalovaných testov platformy Phoronix Test Suite po ktorých sa spustí profilovanie systému CloverETL a následne analýza dát.

`startPhoronix [-config:<file>]` – Spustenie nainštalovaných testov platformy Phoronix Test Suite.

`startClover [-config:<file>]` – Spustenie profilovania systému CloverETL a následnej analýzi týchto dát.

`startAnalysis [-config:<file>]` – Spustenie analýzi dát.

`-help` – Vypíše nápovedu aplikácie.

`-info` – Vypíše informácie o aplikácii.

Parameter `[-config:<file>]` je nepovinný parameter, v prípade jeho nepoužitia bude aplikácia očakávať konfiguračný súbor `config.properties` v aktuálnom adresári.

Konfiguračný súbor

Konfiguračný súbor je potrebný pri spustení každého modulu, pretože sa v ňom nachádzajú všetky potrebné parametre, ktoré ovplvyňujú beh aplikácie. Konfiguračný súbor reprezentuje parametre ako kľúč=hodnota. Aplikácia pracuje s rôznymi parametrami, z ktorých existujú parametre povinné a nepovinné. To, či je parameter povinný alebo nepovinný určuje aj to, ktorá časť modul je práve spustená.

`phoronix.dir` – Cesta k adresáru, kde sa má nainštalovať platforma Phoronix Test Suite.

`phoronix.download` – Odkaz na stiahnutie platformy Phoronix Test Suite.

`phoronix.tests` – Zoznam testov platformy Phoronix Test Suite, ktoré sa nainštalujú a potom spustia v prvom module.

`clover.dir` – Cesta k adresáru, kde sa má nainštalovať CloverETL Engine.

`clover.download` – Odkaz na stiahnutie CloverETL Engine.

`clover.sandbox` – Cesta k projektu systému CloverETL, kde sa nachádzajú ETL Grafy.

`clover.graphs` – Zoznam ETL Grafov, ktoré sa budú profilované.

`clover.reference.dir` – Cesta k adresáru, kde sú uložené výsledky z iného behu v JSON súbore. Výsledky sa na seba mapujú pomocou názvov súborov. Ak nie je tento parameter zadaný, výsledky sa nebudú porovnávať.

`results.dir` – Cesta k adresáru, kde majú byť uložené priebežné a výsledné súbory aplikácie. Pri analýze sa načitávajú z tohto adresára všetky súbory typu JSON.

Voliteľné parametre:

`monitoring.period` – Čas v milisekundách, za aký sa má vykonávať meranie dát pri behu ETL Grafu. Defaultná hodnota je 500 milisekúnd.

`monitoring.attempts` – Počet opakovaní merania jednotlivých ETL Grafov. Defaultná hodnota je 3.

`clover.threshold.error` – Určuje prah, od ktorého sa majú výsledné hodnoty druhého modulu označiť ako chybné (error) po porovnaní s referenčnými výsledkami vo výslednom HTML súbore. Defaultná hodnota je 100.

`clover.threshold.warn` – Určuje prah, od ktorého sa majú výsledné hodnoty druhého modulu označiť ako podozrivé (warning) po porovnaní s referenčnými výsledkami vo výslednom HTML súbore. Defaultná hodnota je 100.

`clover.gc.threshold.error` – Určuje prah, od ktorého sa majú výsledné hodnoty algoritmu garbage collection označiť ako chybné (error) po porovnaní s referenčnými výsledkami vo výslednom HTML súbore. Defaultná hodnota je 100.

`clover.gc.threshold.warn` – Určuje prah, od ktorého sa majú výsledné hodnoty algoritmu garbage collection označiť ako podozrivé (warning) po porovnaní s referenčnými výsledkami vo výslednom HTML súbore. Defaultná hodnota je 100.

`log.level` – Parameter určujúci úroveň logovania. Defaultná hodnota je INFO. Dostupné hodnoty ALL | TRACE | DEBUG | INFO | WARN | ERROR | FATAL | OFF.

Ukážka konfiguračného súboru sa nachádza v prílohe C, kde sa nachádza konfigurácie výslednej aplikácie na systéme MacOS, ktorá bola použitá pri testovaní.

Nasledujúce tabuľky (6.1, 6.2, 6.3) ukazujú povinné a nepovinné parametre pre všetky časti každého modulu.

Konfiguračný parameter	Inštalácia Phoronix Test Suite	Inštalácia testov Phoronix Test Suite	Beh testov Phoronix Test Suite
`phoronix.dir`	✓	✓	✓
`phoronix.download`	✓	×	×
`phoronix.tests`	×	✓	✓
`results.dir`	×	×	✓

Tabuľka 6.1: Povinné a nepovinné konfiguračné parametre pri spustení rôznych častí prvého modulu.

Konfiguračný parameter	Inštalácia CloverETL	Beh grafov CloverETL
`clover.dir`	✓	✓
`clover.download`	✓	×
`clover.sandbox`	×	✓
`clover.graphs`	×	✓
`results.dir`	×	✓

Tabuľka 6.2: Povinné a nepovinné konfiguračné parametre pri spustení rôznych častí druhého modulu.

Konfiguračný parameter	Analýza
`results.dir`	✓

Tabuľka 6.3: Povinné a nepovinné konfiguračné parametre pri spustení rôznych častí tretieho modulu.

6.2.2 Ukončenie aplikácie

V prípade chýbajúcich alebo nevalidných parametrov skončí aplikácia chybou s návratovou hodnotou 1. Nevalidným parametrom môže byť chyba užívateľa (neznalosť parametrov alebo pravopisná chyba) alebo neexistujúca cesta ku konfiguračnému súboru. Ďalším problémom môžu byť chýbajúce požadované parametre v konfiguračnom súbore, taktiež neexistujúce cesty k adresárom alebo súborom (zoznam ETL Grafov). Za užívateľskú chybu sa považuje aj použitie testov platformy Phoronix Test Suite, ktoré sú nevhodné pre danú platformu. Pre úspešné ukončenie aplikácie musia byť všetky parametre aplikácie a konfiguračné parametre validne zadané. Každý modul po úspešnom skončení vracia návratovú hodnotu 0.

6.2.3 Príklad spustenia

Keďže táto aplikácia je multiplatformová, v tejto podkapitole je popísaný postup pri prvom spustený aplikácie na platformách MacOS, Linux a Windows. Všetky ďalšie spustenia sú následne už rovnaké na všetkých platformách.

MacOS, Linux

1. Definovanie adresárov v konfiguračnom súbore (`phoronix.dir` a `results.dir`), odkazu pre stiahnutie Phoronix Test Suite a testov pre platformu Phoronix Test Suite.
2. Definovanie adresárov v konfiguračnom súbore (`clover.dir` a `clover.sandbox`), odkazu pre stiahnutie CloverETL Engine a testovacích grafov.
3. Spustenie inštalácie Phoronix Test Suite, jeho testov a CloverETL Engine pomocou parametru `install`. Po nainštalovaní Phoronix Test Suite je potrebné potvrdiť licenčné podmienky tohto frameworku.
4. Spustenie troch modulov – parameter `start`.

Widnows

V prostredí Windows sú pokyny pre inštaláciu takmer rovnaké avšak po nainštalovaní patformy Phoronix Test Suite je potrebné upraviť súbor `phoronix-test-suite.bat`. V tomto súbore je potrebné pridať cestu k skriptovaciemu jazyku PHP a zadať absolútnu cestu do premennej `%PHP%` k súboru `.../pts-core/phoronix-test-suite.php`.

6.2.4 Štruktúra vytvorených adresárov

V tejto časti je znázornená štruktúra adresárov, ktoré sa vytvárajú pri behu tohto modulu.

Adresár `phoronix.dir` a `clover.dir`

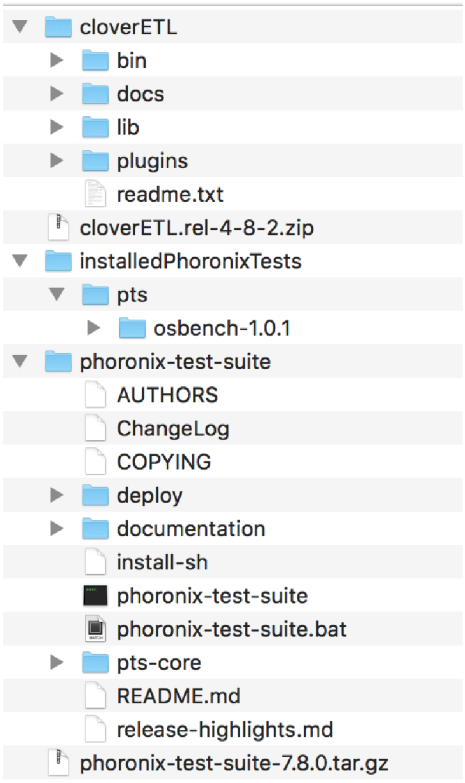
Ak sú tieto dva adresáre zhodné, budú sa v tomto adresári nachádzať stiahnutý Phoronix Test Suite a CloverETL Engine. Po nainštalovaní Phoronix Test Suite testov sa tu nachádzajú aj tieto testy.

Adresár `/phoronix-test-suite` – Všetky potrebné súbory pre spustenie Phoronix Test Suite. V prípade spustenia tejto aplikácie na systéme Windows, je potrebné v tomto adresári zmeniť cestu k PHP v súbore `.../phoronix-test-suite/phoronix-test-suite.bat`.

Adresár `/installedPhoronixTests` – Nainštalované testy platformy Phoronix Test Suite.

Adresár `/cloverETL` – Všetky potrebné súbory pre spustenie CloverETL Engine.

Ukážka štruktúry adresáru, kde je nainštalovaná platforma Phoronix Test Suite a CloverETL Engine prezentuje obrázok 6.4. V tomto prípade boli zhodne zvolené parametre `phoronix.dir` a `clover.dir` v konfiguračnom súbore aplikácie.



Obr. 6.4: Štruktúra adresára `phoronix.dir` a `clover.dir`

Adresár `results.dir`

Adresár `phoronixTestResults` – Nachádzajú sa v ňom výsledky prvého modulu.

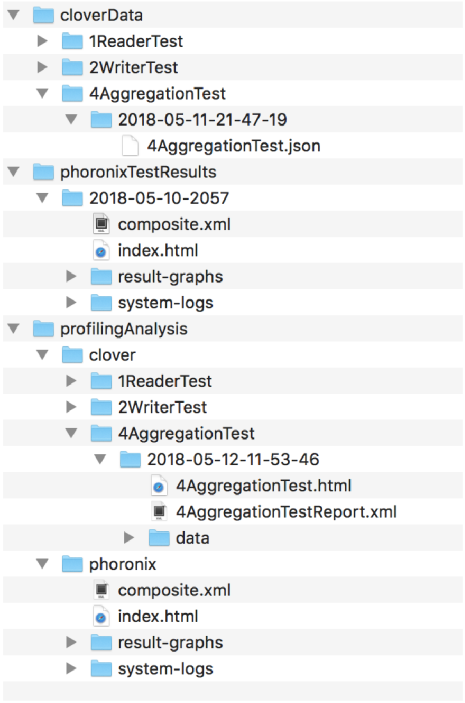
Adresár `cloverData` – Tento adresár obsahuje podadresáre s menom ETL Grafu, ktorý bol spustený. V každom z týchto adresárov sa nachádzajú výsledky behu týchto ETL Grafov v JSON súbore.

Adresár `profilingAnalysis`

Adresár `phoronix` – Výsledky prvého modulu, kde sa nachádza výsledný HTML súbor.

Adresár `clover` – Podadresáre s menom ETL Grafu, ktorý obsahuje výsledné dáta v XML súbore vizualizované v HTML súbore.

Ukážka štruktúry adresáru `results.dir` sa nachádza na obrázku 6.5.



Obr. 6.5: Štruktúra adresára `results.dir`

Kapitola 7

Testovanie a experimentovanie

Po implementácii prebehlo testovanie, ktoré malo určiť kvalitu vytvorenej aplikácie. Testovanie Phoronix Test Suite Testov prebehlo na platformách MacOS a Windows s rôznymi parametrami. Toto sa zvolilo preto, aby sa dali porovnať jednotlivé platformy a ich rozdiel vo výkonnosti jednotlivých operácií. Následne profilovanie systému CloverETL bude prebiehať na systéme MacOS, kde sa bude postupne zvyšovať zaťaženie procesora alebo nastane zmena v konfigurácii ETL Grafu. To ukáže, ako je možné odhaliť chyby pri spracovaní dát. Testovacie scenáre sú rozdelené na dve časti a to testovanie dát z platformy Phoronix Test Suite a testovanie ETL Grafov pri behu na systéme CloverETL.

V nasledujúcej kapitole sa nachádza popis jednotlivých testov s platformou Phoronix Test Suite a testovaných ETL Grafov, ktoré boli vytvorené pre účel experimentovania. Pre vytvorenie týchto testovacích ETL Grafov bol využitý CloverETL Designer. Pri testovaní bol využitý CloverETL Engine s plnou funkcionalitou pre možnosť otestovania zložitejších ETL Grafov, v ktorých sa nachádzajú pokročilejšie funkcie systému CloverETL. ETL Grafy, ktoré je možné spúšťať aj na CloverETL Open Source Engine budú súčasťou výslednej aplikácie.

Aplikácia je napísaná v programovacom jazyku Java 8, čo znamená prenositeľnosť medzi platformami. Pri testovaní boli vyskúšané tri najčastejšie platformy a to MacOS, Windows a Ubunutu vo virtuálnom prostredí. Všetky platformy sú funkčné, avšak systém Windows nepodporuje niektoré typy MXBean, čo znamená to, že výsledné dáta nebudú kompletné v rámci tejto platformy. Sú to dáta ako zaťaženie CPU a informácia o počte deskriptorov súborov. Prezentácia výsledkov pomocou HTML stránky je taktiež platformovo nezávislá, kde pre zobrazenie výsledných reportov postačí ľubovolný webový prehliadač.

7.1 Meranie výkonu prostredia

Prvým scenárom bol test `pts/osbench`, ktorým bolo testované prostredie, na ktorom systém CloverETL beži. Tento test bol vytvorený spoločnosťou Phoronix Test Suite. OSBench je zbierka mikro-benchmarkov na meranie základných operácií operačného systému, ako je čas na vytváranie vlákien/procesov, spustenie programov, vytváranie súborov a prideľovanie pamäte. Test bol vybraný z mnoho ďalších dostupných testov najmä preto, že beží na všetkých požadovaných platformách. Výsledky testu majú dať pohľad na základné operácie operačného systému a rýchlosť ich spracovania. To môže signalizovať pomalý priebeh spracovania dát systémom CloverETL a tým pádom aj to, že systém na ktorom sa majú spracovávať dáta je nedostačujúci.

Pre tento test boli zvolené tri rôzne systémy s dvomi rozdielnymi operačnými systémami.

Parametre a špecifikácia systémov

V nasledujúcej tabuľke 7.1 sa nachádza porovnanie parametrov systémov, na ktorých boli spúšťané testy pre meranie výkonu.

	Apple MacBook	Toshiba Tecra	Lenovo IdeaPad
Operačný systém	macOS High Sierra	Windows 7	Windows 10
Pamäť RAM	16 GB	8 GB	8 GB
Procesor	2,9 GHz Intel Core i7	1,6 GHz Intel Core i5	2,3 GHz Intel Core i7
Veľkosť pamäte	500 GB	256 GB	256 GB

Tabuľka 7.1: Porovnanie systémov, na ktorých boli vykonané testy

Beh testu

Tento test bežal na spomínaných konfiguráciách, kde ako výsledok vznikol HTML súbor. V tomto súbore sa nachádza výsledok behu tohto testu, kde sa merala rýchlosť vytvárania súborov, vytvárania vlákien, spúšťania programov, vytvárania procesov a alokovania pamäte. Nasledujúca tabuľka 7.2 porovnáva jednotlivé časy behu podtestov tohoto testu.

	Apple MacBook	Toshiba Tecra	Lenovo IdeaPad
Vytváranie súborov [μs]	283,98	1264,08	605,01
Vytváranie vlákien [μs]	14,71	43,41	23,84
Spúšťanie programov [μs]	410,12	1434,49	1051,05
Alokácia pamäte [ns]	142,58	106,44	109,06

Tabuľka 7.2: Čas behu jednotlivých podtestov Phoronix Test Suite Testu

Z daných podtestov sa ukázal ako najrýchlejší systém na počítači Apple MacBook, čo jasne dokazujú aj parametre systémov z tabuľky 7.1. Výsledky tohto testu potvrdzuje aj rýchlosť behu rovnakého ETL Grafu na týchto systémoch. Tento test abecedne radí milión záznamov. Najrýchlejšie tento test prebehol na počítači Apple MacBook (5,8s), pomalší bol počítač Lenovo IdeaPad (10,5s) a najpomalší bol Toshiba Tecra (14s). Ukážka výsledného súboru zo systému MacOS sa nachádza v prílohe D.

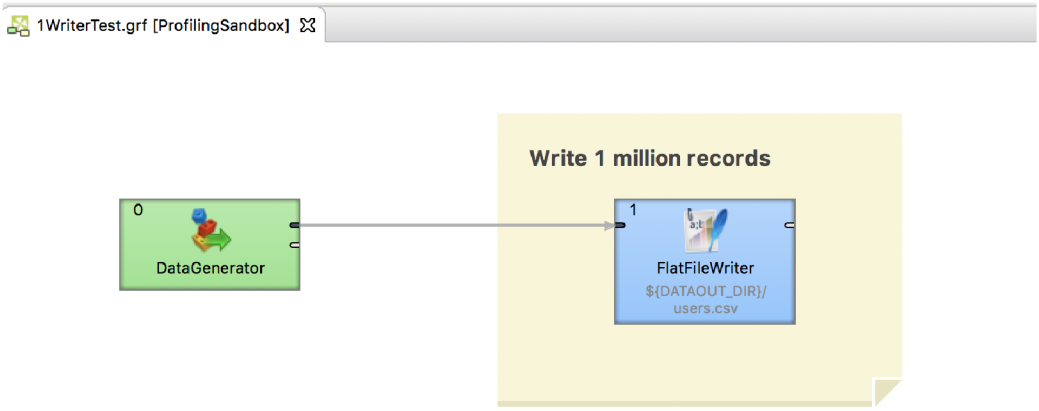
7.2 Profilovanie systému CloverETL

Ďalšou časťou testovania je samotné testovanie systému CloverETL za pomoci vytvorených ETL Grafov. Tak, ako bolo spomenuté v úvode kapitoly, pre testovanie rôznych ETL Grafov bol využitý CloverETL Engine s plnou funkcionalitou. To umožňuje otestovať a porovnať viacej druhov používaných grafových komponent. Každý test bude spustený na počítači Apple MacBook minimálne dva krát, kde druhý z týchto testov je test pri úplnom zaťažení procesora, fyzickej pamäte, disku alebo po zmene konfigurácie ETL Grafu. Po ukončení testov sa výsledky porovnajú a výsledný HTML súbor by mal ukázať rozdiely pri behu týchto ETL Grafov. Ukážka takéhoto výsledného súboru sa nachádza v prílohe D. Niektoré z týchto testov budú meniť počet spracovávaných záznamov a použité komponenty, aby

sa ukázalo ako vie výsledná aplikácia detekovať problémy pri spracovávaní dát. Meranie informácií počas behu ETL Grafu bude prebiehať každých sto milisekúnd a každý ETL Graf bude spustený päťkrát. Referenčné výsledky sa určia po spustení niekoľkých testov na nezaťaženom systéme. Prah pre chyby a podozrivé zmeny bol zvolený rovnako pre dáta z druhého modulu, a taktiež pre hodnoty algoritmu garbage collection. Pre chyby (error) bol zvolený 50% prah a pre podozrivé zmeny (warning) bol 25%. Príloha C ukazuje nastavenie konfiguračného súboru pri behu týchto testov.

7.2.1 Zapisovanie dát do súboru

Prvým testovaným ETL Grafom bol graf s komponentou FlatFileWriter v grafe `1WriterTest.grf`. Tento ETL Graf (7.1) vygeneruje milión záznamov, ktoré následne zapíše do CSV súboru. V tomto teste sa okrem testovania pri plne zaťaženom a nezaťaženom CPU a následného porovnania s referenčnými výsledkami, vykonávali experimenty s fázami jednotlivých komponent. To znamená že komponenta DataGenerator bola vo fáze 0 a komponenta FlatFileWriter vo fáze 1. Dáta sa teda nespracovávali paralelne ale sekvenčne.



Obr. 7.1: Testovací ETL Graf `1WriterTest.grf`

Vyhodnotenie

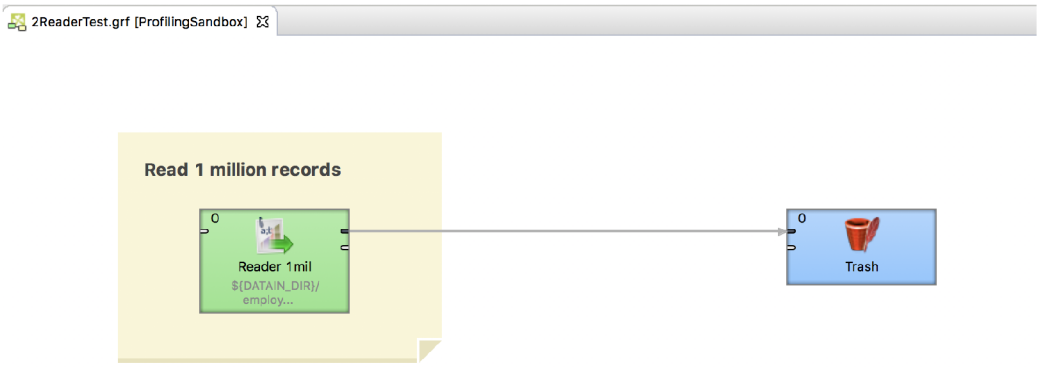
Tak ako sa predpokladalo, výsledky odhalili spomalenie behu ETL Grafu a zvýšenie aktivity algoritmu garbage collection. Zaťaženie CPU bolo porovnateľné s referenčnými hodnotami. Tabuľka 7.3 ukazuje porovnanie hodnôt týchto údajov. Tieto výsledky dokazujú, že rýchlosť spracovania dát záleží nielen na systéme, ale aj na konfigurácii samotného ETL Grafu.

	Sekvenčný beh komponent	Paralelný beh komponent
Čas v sekundách	5.5 s	3.8 s
Priemerné zaťaženie procesora v %	12.1 %	16.9 %
Voľná fyzicka pamäť v MB	1619.9 MB	1630.5 MB
Aktivita mladej generácie algoritmu garbage collection počet/čas	4 / 19 ms	2 / 8 ms

Tabuľka 7.3: Porovnanie výsledkov z testu ETL Grafu `1WriterTest.grf`

7.2.2 Čítanie dát zo súboru

Ďalším ETL Grafom (obrázok 7.2) bol graf, ktorý používa komponentu FlatFileReader pre načítanie milión záznamov z externého CSV súboru. Pri behu tohto testu sa vykonali merania pri plne zaťaženom CPU a disku.



Obr. 7.2: Testovací ETL Graf `2ReaderTest.grf`

Vyhodnotenie

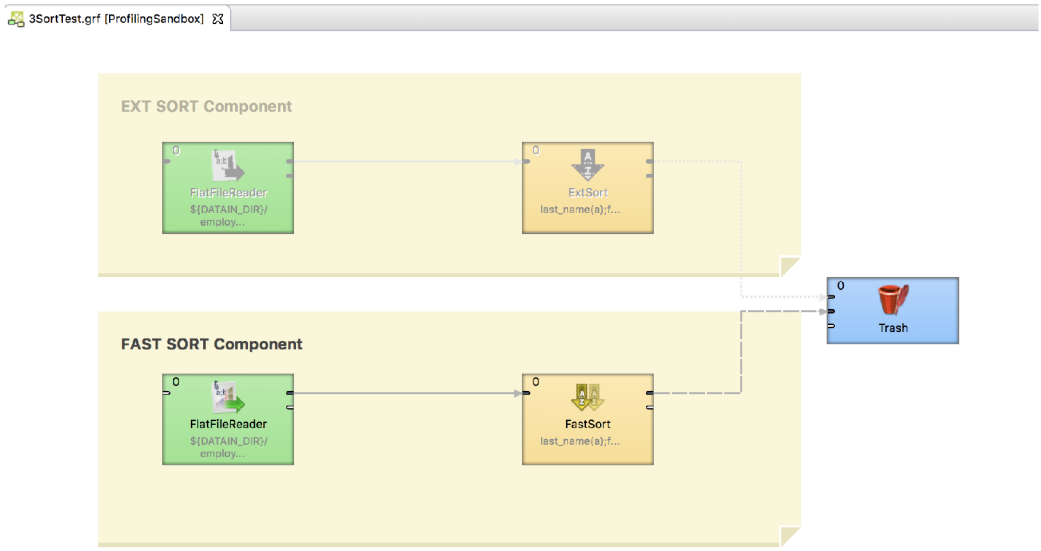
Pri porovnaní s referenčnými výsledkami sa odhalili zmeny vo viacerých meraných dátach, ako napríklad čas behu celého ETL Grafu alebo vyššie zaťaženie CPU. Dáta, ktoré presiahli prah chyby boli podfarbené červenou a podozrivé odchylky žltou farbou. Zmeny nastali aj v počte opakovaní a rýchlosti behu algoritmu garbage collection. V tabuľke 7.4 sa nachádzajú dáta, v ktorých nastala najväčšia odchylka pri tomto teste. Na základe týchto dát je možné určiť problém v infraštruktúre (napríklad málo voľnej fyzickej pamäte), keďže rýchlosť načítania dát sa znížila na polovicu, a taktiež sa vyhodnotilo zaťaženie CPU ako neprimerané.

	Zaťažený CPU a Disk	Zaťažené CPU	Referenčné dáta
Čas v sekundách	4.2 s	3.3 s	1.6 s
Priemerné zaťaženie procesora v %	57.9 %	50.6 %	13.2 %
Voľná fyzicka pamäť v MB	430.9 MB	3300.1 MB	4020.5 MB
Aktivita starej generácie algoritmu garbage collection počet/čas	1 /149 ms	4 / 153 ms	1 / 64 ms

Tabuľka 7.4: Porovnanie výsledkov z testu ETL Grafu `2ReaderTest.grf`

7.2.3 Radenie dát

Testovací ETL Graf `3SortTest.grf` načitáva databázu zamestnancov z externého súboru a následne ju radí podľa priezviska a mena pomocou komponenty ExtSort alebo FastSort. Tento test bol spúšťaný s rôznymi komponentami pri rôznom zaťažený CPU. To znamená, že buď sa na radenie používala komponenta ExtSort alebo FastSort [1].



Obr. 7.3: Testovací ETL Graf `3SortTest.grf`

Vyhodnotenie

Ako sa už očakáva z názvu komponenty FastSort, skutočne bol ETL Graf s touto komponentou rýchlejší. Avšak použitie tejto komponenty prináša o niečo väčšie zaťaženie procesora a taktiež viac použitia fyzickej pamäte. Naopak, rozdiel aktivity algoritmu garbage collection nie je nijak výrazný. Porovnanie nameraných a referenčných dát môžeme vidieť v tabuľke 7.5. Tento test dokázal to, že použitie iných rýchlejších komponent môže sprevádzať nárok na výkonnosť systému.

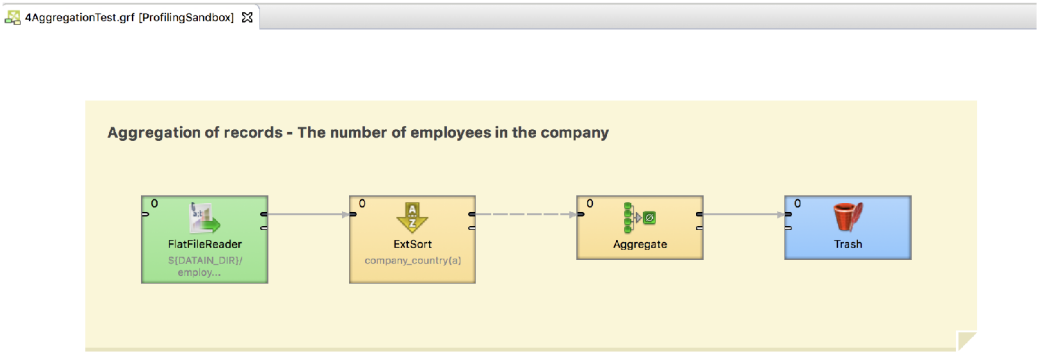
	FastSort	ExtSort
Čas v sekundách	3.3 s	5.5 s
Priemerné zaťaženie procesora v %	19.2 %	12.1 %
Voľná fyzicka pamäť v MB	367.5 MB	800.5 MB
Použitá veľkosť heap pamäte v MB	236.5 MB	179.8 MB

Tabuľka 7.5: Porovnanie výsledkov z testu ETL Grafu `3SortTest.grf`

[1] Obrázok 7.3 znázorňuje tento ETL Graf, kde sú šedé komponenty, čo znamená že sú deaktivované.

7.2.4 Agregovanie dát

ETL Graf `4AggregationTest.grf` agreguje dáta na základe počtu zamestnancov v spoločnosti. Pred samotným agregovaním pomocou komponenty Aggregate, je potrebné načítať dáta zo súboru a zoradiť ich.



Obr. 7.4: Testovací ETL Graf `4AggregationTest.grf`

Vyhodnotenie

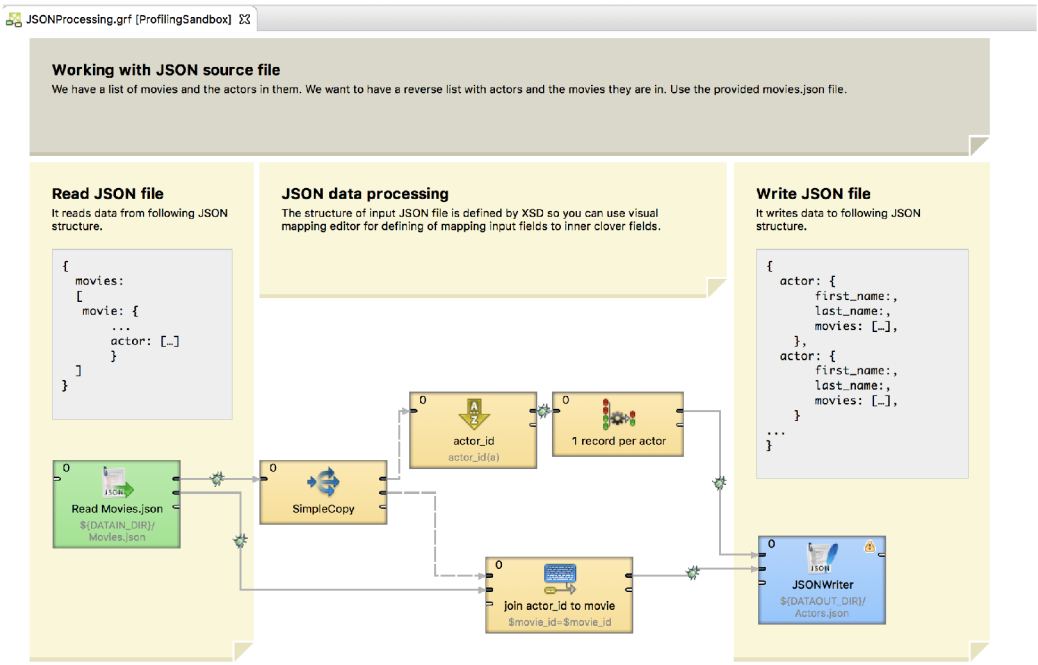
Výsledky tohto testu dosiahli odchýlky znázornené v tabuľke 7.6. Pri zaťaženom CPU, fyzickej pamäte a disku sa dĺžka behu ETL Grafu výrazne zvýšila, taktiež algortimus garbage collection pracoval častejšie pri zaťaženom CPU. To vyplýva už zo samotnej konfigurácie ETL Grafu, kde sa používa viac komponent. Zaťaženie týchto častí malo výrazný vplyv na beh ETL Grafu čo znamená, že efektívnosť spracovania dát závisí priamoúmerne na zaťažení systému.

	Zaťažený systém	Zaťažené CPU	Referenčné dáta
Čas v sekundách	19.7 s	12.6 s	6.2 s
Priemerné zaťaženie procesora v %	59.9 %	50.2 %	11.2 %
Voľná fyzicka pamäť v MB	256.9 MB	780.1 MB	1343.5 MB
Aktivita mladej generácie algoritmu garbage collection počet/čas	1 / 21 ms	4 / 25 ms	1 / 13 ms

Tabuľka 7.6: Porovnanie výsledkov z testu ETL Grafu `4AggregationTest.grf`

7.2.5 Spracovanie súboru typu JSON

ETL Graf `JSONProcessing.grf` je graf, ktorý transformuje dáta vo formáte JSON. Ako prvé sa dáta načítajú zo vstupného súboru, následne prebehne spracovanie dát po ktorom sa výsledné dáta zapíšu opäť do JSON súboru. Tento ETL Graf bol spustený týždeň po nameraní referenčných výsledkov, čím sa snažila dokázať stabilita implementovanej aplikácie.



Obr. 7.5: Testovací ETL Graf `JSONProcessing.grf`

Vyhodnotenie

Ako ukazujú hodnoty z tabuľky 7.7, dáta ako dĺžka behu ETL Grafu a zaťaženie procesora sú po týždni veľmi podobné. Dáta voľnej fyzickej pamäte a aktivity mladej generácie algoritmu garbage collection sú rozdielne, pretože počas týždňa sa znížilo využitie fyzickej pamäte a zvýšila rýchlosť algoritmu garbage collection. Keďže nedošlo k zhoršeniu, ale naopak k zlepšeniu, tieto výsledky nebránia tomu, aby bol referenčný súbor aktualizovaný novými hodnotami, čo by zabezpečilo lepšiu stabilitu testov pri ďalších iteraciách.

	Merané dáta	Referenčné dáta
Čas v sekundách	1.2 s	1.2 s
Priemerné zaťaženie procesora v %	26.6 %	27.4 %
Voľná fyzicka pamäť v MB	1600.9 MB	235.6 MB
Aktivita mladej generácie algoritmu garbage collection počet/čas	1 / 21 ms	1 / 41 ms

Tabuľka 7.7: Porovnanie výsledkov z testu ETL Grafu `JSONProcessing.grf`

7.3 Využitie aplikácie

Aplikácia slúži na meranie výkonu prostredia a profilovanie systému CloverETL, čo je systém pre spracovanie dát. Naimplemetovaná aplikácia slúži na získanie informácií o prostredí, na ktorom systém beží, ale aj o získaní informácií o spôsobe využitia prostredia systémom CloverETL počas spracovávania dát. V tejto podkapitole je popísaný spôsob využitia aplikácie.

Spôsob využitia aplikácie

- Meranie výkonu prostredia, na ktorom má byť systém CloverETL nasadený.
- Profilovanie systému CloverETL, čo znamená monitorovanie prostredia počas behu ETL Grafov, ktoré spracovávajú dáta. Následné porovnanie týchto výsledkov s referenčnými výsledkami vie odhaliť možné príčiny neefektívnosti spracovávania dát.
- Využitie aplikácie pri automatizovanom testovaní samotného systému CloverETL, čo by pomohlo odhaliť regresné chyby vo výkone, ktoré by mohli vzniknúť počas vývoja systému.

Aplikáciu je možné spustiť priamo cez príkazový riadok pomocou parametrov, na základe ktorých sa následne spustí požadovaná časť aplikácie. Keďže aplikácia podporuje maximálne dva parametre, je jednoduché ju spustiť užívateľom. Ďalšou výhodou je použiteľnosť aplikácie pri automatizovanom testovaní. Jednoducho stačí nakonfigurovať súbor s parametrami aplikácie a začleniť ju medzi automatizované testy. Po dobehnutí testov sa výstupný XML súbor môže využiť pre ďalšie spracovanie a HTML súbory priamo prezentujú a porovnávajú výsledky meraní.

Súčasťou tohto projektu je aj sandbox s naimplementovanými ETL Grafmi, ktoré boli priamo testované počas vývoja tejto aplikácii. Sú to testy komponent pre zápis a načítanie dat do/z súborov (ETL Grafy `1WriterTest.grf` a `2ReaderTest.grf`), testovanie komponenty pre radenie dát (ETL Graf `3SortTest.grf` s komponentou ExtSort) a testovanie agregovania údajov podľa špecifikovaného kľúča (ETL Graf `4AggregationTest.grf`). Ďalšie ETL Grafy, ktoré boli testované v rámci vývoja tejto aplikácii, využívali pokročilejšiu funkcionalitu systému CloverETL. Základné ETL Grafy je však možné jednoducho vytvoriť v aplikácii ClvoerETL Desginer, ktorú je možné stiahnuť s trialovou licenciou.

7.4 Námety k rozšíreniu práce

Rozšírenie testovacích Phoronix Test Suite testov a ETL Grafových testov by zväčšilo pohľad na prostredie, kde aplikácia beží a odhalilo by viacero možných problémov, ktoré by mohli znižovať efektivitu spracovania dát systémom CloverETL.

Ak by sa aplikácia mala využívať potencionálnymi zákazníkmi, ktorí by chcel otestovať svoje prostredie, bolo by vhodné implementovať užívateľské rozhranie pre konfiguráciu parametrov a prezentovanie výsledkov aplikácie. Veľa aplikácií už implementuje toto rozhranie, pretože ľudia sa lepšie orientujú v niečom, čo im umožní konfigurovať a zároveň vidieť výsledky na jednom mieste. To by malo za následok lepšie a rýchlejšie konfigurovanie potrebných parametrov a taktiež krajšiu vizuálnu stránku celej aplikácie. Následne by sa v tomto rozhraní mohlo nachádzať viac vysvetľujúcich nápovedí na jednotlivé konfiguračné parametre a rôznu funkcionalitu.

Pre použitie tejto aplikácie v testovacom prostredí a zlepšenie priebežnej integrácie, by sa mohlo zlepšiť reportovanie chýb smerom k ďalším častiam automatizovaného testovania. Momentálne aplikácia vygeneruje dva HTML súbory a jeden XML s dátami. To už priamo podporuje využiteľnosť aplikácie pri automatizovanom testovaní systému CloverETL. Vždy sa ale dajú nájsť rôzne iné možnosti, ktoré je vhodné sledovať a mohli by pomôcť otestovať aplikáciu dôkladnejšie.

Keďže pri testovaní systémov je veľa rôznych vonkajších faktorov, ktoré priamo, ale aj nepriamo ovplyvňujú výkonnosť aplikácie, vytvorením ďalších konfiguračných parametrov sa môže zlepšiť presnosť aplikácie. Či už by to boli rôzne parametre pre získanie cenných dát z behu systému alebo parametre, ktoré by lepšie určili výsledky vzhľadom na priebeh testovania.

Keďže táto aplikácia pracuje s knižnicou CloverETL Open Source Engine, bolo by vhodné rozšíriť použitie aplikácie aj na ďalšie produkty firmy Javlin. To je napríklad beh ETL Grafov na CloverETL Serveri alebo Clustri. Výrazne by to zvýšilo použiteľnosť aplikácie v produkčnom prostredí, keďže veľa zákazníkov tejto spoločnosti využíva tieto produkty.

Kapitola 8

Záver

Cieľom tejto diplomovej práce bolo vytvorenie aplikácie pre meranie výkonu a profilovanie systému CloverETL pre spracovanie dát a taktiež implementácia a otestovanie rôznych konfigurácií CloverETL Grafov. Riešením tejto práce sa podarilo navrhnúť a implementovať aplikáciu, ktorá sa skladá z troch rôznych modulov. Tieto moduly na seba naväzujú a spoločne poskytujú výstupné dáta. Výsledná aplikácia dokáže otestovať prostredie a vytvoriť dátové a grafické výstupy, na základe ktorých je možné určiť problémové časti prostredia alebo sledovať zmenu výkonnosti. Dôležitým výsledkom je to, že aplikáciu bude možné využiť užívateľmi pri manuálnom testovaní, ale aj v automatizovanom testovacom prostredí, čo pomôže priebežnej integrácií, a tým aj urýchleniu vývoja systému CloverETL. Aplikácia môže slúžiť pre regresné testovanie, kde sa porovnajú referenčné výsledky s aktuálne nameranými.

Ako prvé bolo potrebné naštudovať teóriu merania výkonu a profilovaniu Java aplikácií a následne sa lepšie zoznámiť so systémom CloverETL. Keďže na problematiku merania výkonu a profilovaniu aplikácií existuje veľa riešení, bolo potrebné ich prejsť a rozhodnúť, či už niečo podobné neexistuje. Po preštudovaní existujúcich riešení sa podarilo vybrať aspoň jednu platformu, ktorá sa následne využila pri implementácií jednej časti výslednej aplikácie. Návrh aplikácie v kapitole 5 prezentuje výslednú architektúru aplikácie ako aj metriky, ktoré táto aplikácia poskytuje. Implementácia výslednej aplikácie popísaná v kapitole 6 zahŕňa implementačné detaily každého balíka a jednotlivé diagramy tried najzložitejších častí aplikácie. Ďalšou neoddeliteľnou súčasťou tejto práce je vysvetlenie konfiguračných možností. V rámci testovania bol vyskúšaný každý modul aplikácie, čo je popísané kapitolou 7. Pre účel profilovania systému CloverETL boli vytvorené testovacie CloverETL Grafy, ktorými sa otestovala funkčnosť aplikácie. Testy ukázali použiteľnosť aplikácie pri porovnávaní rôznych prostredí, ako aj samotnom testovaní počas behu CloverETL Grafov. Keďže sa jedná o voľne dostupnú aplikáciu, je možné použitie aplikácii vo firme Javlin v ich automatizovanom testovacom prostredí a taktiež mimo nej. CloverETL Grafy, ktoré je možné spúšťať aj na voľne dostupnej verzii systému CloverETL sú súčasťou výsledku diplomovej práce.

Systém CloverETL je špecifický a existuje veľa možností, akým by sa mohol rozšíriť tento projekt. Jedným z nich je umožniť užívateľom konfiguráciu parametrov pomocou užívateľského rozhrania. To by umožnilo používanie tejto aplikácie viacerým ľuďom, pretože by to zjednodušilo konfiguráciu, ktorá je teraz zabezpečená prostredníctvom konfiguračného súboru. Keďže testovanie systémov ovplyvňuje veľa vonkajších faktorov, ďalšou možnosťou je rozšírenie konfiguračných parametrov alebo sledovacích metrík. Tieto zmeny by zlepšili presnosť meraní výkonu a pomohli by vylepšiť a zefektívniť získavanie dát. Veľkou výzvou

by bolo rozšírenie aplikácie na ďalšie komerčné produkty spoločnosti Javlin, ktoré sú využívané po celom svete. Tým by sa zväčšila použiteľnosť aplikácie a pomohlo by sa detekovať problémy zákazníkov firmy priamo v produkčnom prostredí.

Literatúra

[1] CloverETL: Designer GUI. [Online; navštívené 03.01.2018].
URL http://doc.cloveretl.com/documentation/UserGuide/topic/com.cloveretl.gui.docs/docs/figures/cloveretl-designer.png

[2] CloverETL: Server GUI. [Online; navštívené 03.01.2018].
URL http://doc.cloveretl.com/documentation/UserGuide/topic/com.cloveretl.gui.docs/docs/figures/cloveretl-server.png

[3] CloverETL: User's Guide. [Online; navštívené 03.01.2018].
URL http://doc.cloveretl.com/documentation/UserGuide/index.jsp

[4] CloverETL: Website. [Online; navštívené 03.01.2018].
URL https://www.cloveretl.com

[5] Cooper, B. F.: Yahoo! Cloud Serving Benchmark. [Online; navštívené 10.12.2017].
URL
https://www2.cs.duke.edu/courses/fall13/cps296.4/838-CloudPapers/ycsb.pdf

[6] Cooper, B. F.; Silberstein, A.; Tam, E.; aj.: Benchmarking Cloud Serving Systems with YCSB. *Proceedings of the 1st ACM Symposium on Cloud Computing*, 2010: s. 143 – 154.

[7] FasterXML: Jackson project. [Online; navštívené 20.4.2018].
URL https://github.com/FasterXML/jackson

[8] Foundation, T. A. S.: Apache Commons Compress. [Online; navštívené 28.4.2018].
URL https://commons.apache.org/proper/commons-compress/

[9] Foundation, T. A. S.: Log4j. [Online; navštívené 26.4.2018].
URL https://logging.apache.org/log4j/2.x/manual/index.html

[10] Gilbert, D.: JFreeChart. [Online; navštívené 25.4.2018].
URL http://www.jfree.org/jfreechart/

[11] Gomes, A. S.: Profiler4j. [Online; navštívené 5.2.2018].
URL http://profiler4j.sourceforge.net

[12] Herout, P.: *Učebnice jazyka Java*. Kopp, 2011, ISBN 978-80-7232-398-2.

[13] Hunt, C.; John, B.: *Java Performance*. Addison-Wesley, 2011, ISBN 978-0-13-714252-1.

[14] InfraRED: InfraRED. [Online; navštívené 10.2.2018].
URL http://infrared.sourceforge.net/versions/latest/

[15] JAMon: JAMon. [Online; navštívené 20.2.2018].
URL http://jamonapi.sourceforge.net

[16] JMEasurement: JMeasurement. [Online; navštívené 7.2.2018].
URL http://wkla.no-ip.biz/mcs/jmeasurement/

[17] Luo, C.; Zhan, J.; Jia, Z.; aj.: CloudRank-D: benchmarking and ranking cloud computing systems for data processing applications. *Frontiers of Computer Science*, ročník 6, aug 2012: s. 347 – 362.

[18] Oracle: Class ManagementFactory. [Online; navštívené 04.3.2018].
URL https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/management/ManagementFactory.html

[19] Oracle: Interface GarbageCollectorMXBean. [Online; navštívené 10.4.2018].
URL https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/management/GarbageCollectorMXBean.html

[20] Oracle: Interface MemoryMXBean. [Online; navštívené 08.4.2018].
URL https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/management/MemoryMXBean.html

[21] Oracle: Interface OperatingSystemMXBean. [Online; navštívené 08.4.2018].
URL https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/management/OperatingSystemMXBean.html

[22] Oracle: Interface ThreadMXBean. [Online; navštívené 08.4.2018].
URL https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/management/ThreadMXBean.html

[23] Oracle: Introducing MBeans. [Online; navštívené 04.3.2018].
URL https://docs.oracle.com/javase/tutorial/jmx/mbeans/index.html

[24] Phoronix: Test Suite. [Online; navštívené 12.12.2017].
URL http://www.phoronix-test-suite.com

[25] Phoronix: Test Suite 7.6.0. [Online; navštívené 12.12.2017].
URL https://github.com/phoronix-test-suite/phoronix-test-suite

Prílohy

Zoznam príloh

Príloha A

Obsah priloženého pamäťového média

Zložka so zdrojovými kódmi – profilingSystem

Tu sa nachádzajú zdrojové kódy aplikácie.

Zložka s aplikáciou – profilingSystemApp

V tejto zložke sa nachádza spustiteľný JAR súbor, názorná ukážka konfiguračného súboru a testovací sandbox s CloverETL Grafmi, ktoré je možné spustiť pomocou CloverETL Open Source Engine.

Zložka tex

Obsahuje zdrojové kódy a grafický materiál pre vytvorenie dokumentu `meranie_vykonu_a_profilovanie_systemu_cloveretl_pre_spracovanie_dat.pdf`

Súbor meranie_vykonu_a_profilovanie_systemu_cloveretl_pre_spracovanie_dat.pdf

Tento dokument.

Súbor readme.txt

Obsahuje základné informácie o aplikácii.

Súbor credits

Obsahuje výpis použitých knižníc a odkazy na stránky, z ktorých bol prevzatý kód pre niektorú funkcionalitu (napríklad sťahovanie súborov z internetu).

Príloha B

Manuál

Pre spustenie všetkých modulov aplikácie je potrebné mať nainštalovanú *Javu 8* a skriptovací jazyk *PHP 5.3*. Súbor `profilingSystem.jar` je spustiteľná command-line aplikácia, ktorá sa spúšťa s vhodnými parametrami.

Parametre aplikácie

- -help – nápoveda
- -info – informácie o aplikácii
- install [-config:<filePath>] – stiahne a nainštaluje Phoronix Test Suite, testy pre túto platformu a CloverETL Engine
- installPhoronix [-config:<filePath>] – stiahne a nainštaluje Phoronix Test Suite
- installPhoronixTests [-config:<filePath>] – stiahne a nainštaluje testy pre platformu Phoronix Test Suite
- installClover [-config:<filePath>] – stiahne a nainštaluje CloverETL Engine
- start [-config:<filePath>] – spustí testy Phoronix Test Suite, profilovanie systému CloverETL a analýzu nameraných dát
- startPhoronix [-config:<filePath>] – spustí testy Phoronix Test Suite
- startClover [-config:<filePath>] – spustí profilovanie systému CloverETL
- startAnalysis [-config:<filePath>] – spustí analýzu dát a vytvorí výstupné súbory

Parameter `[-config:<filePath>]` je nepovinný parameter určujúci adresár konfiguračného súboru, pri jeho nezadaní sa očakáva konfiguračný súbor v aktuálnom adresári s názvom `config.properties`.

Ukážka spustenia aplikácie: `java -jar profilingSystem.jar startClover`

Príloha C

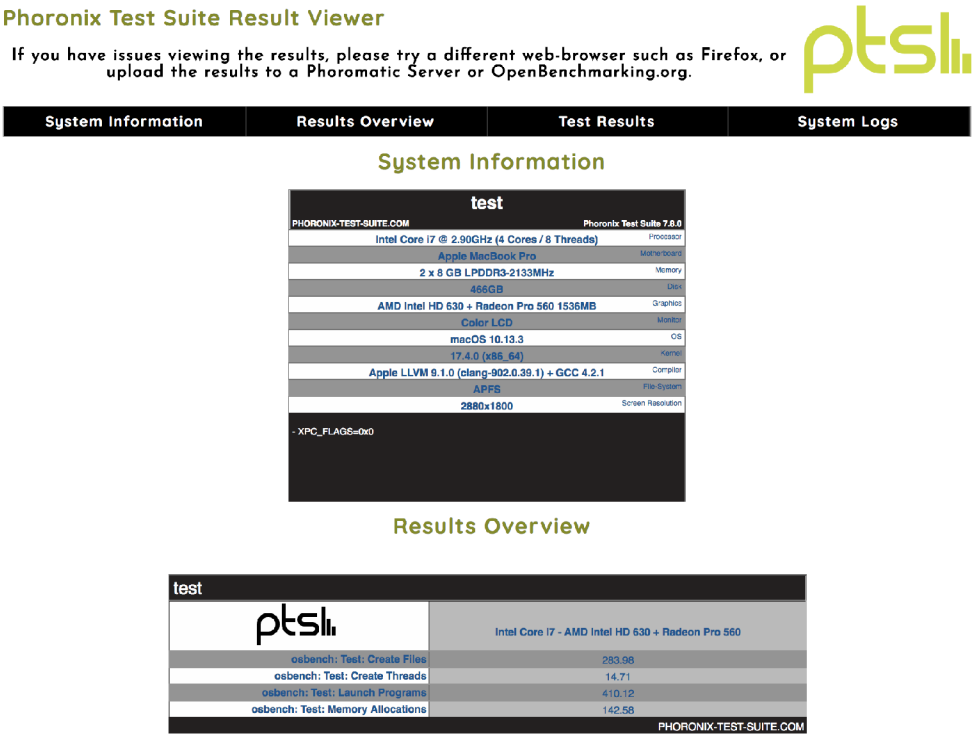
Konfiguračný súbor

```
#########################################################################
### PHORONIX SETTINGS

# Phoronix and phoronix tests instalation directory
phoronix.dir = /Users/roland/Desktop/ProfilingSystem/downloads
# Phoronix test suite download link
phoronix.download = http://phoronix-test-suite.com/releases/phoronix...
# Tests seprated by a comma
phoronix.tests = pts/osbench

#########################################################################
### CLOVER SETTINGS

# Clover instalation directory
clover.dir = /Users/roland/Desktop/ProfilingSystem/downloads
# CloverETL Engine download link
clover.download = https://datapacket.dl.sourceforge.net/project/cloveretl/...
# Project sandbox location
clover.sandbox = /Users/roland/Desktop/ProfilingSystem/ProfilingSandbox
# Test ETL Graphs seprated by a comma
clover.graphs = graph/1WriterTest.grf, graph/2ReaderTest.grf...

#########################################################################
### Monitoring settings

# ETL Graphs monitoring period in ms
monitoring.period = 100
# Number of iteration evaluating ETL Graphs
monitoring.attempts = 5
# Final result directory
results.dir = /Users/roland/Desktop/ProfilingSystem/results
```

```
# Threshold for errors in HTML report
clover.threshold.error = 50
# Threshold for warnings in HTML report
clover.threshold.warn = 25
# Threshold for errors in HTML report for GarbageCollector
clover.gc.threshold.error = 50
# Threshold for warnings in HTML report for GarbageCollector
clover.gc.threshold.warn = 25

# Directory whith referenced results
clover.reference.dir = /Users/roland/Desktop/ProfilingSystem/referenced

######################################################################
### Logging settings
# Application logging
log.level = INFO
```

Príloha D

Výstupy aplikácie



Phoronix Test Suite Result Viewer

If you have issues viewing the results, please try a different web-browser such as Firefox, or upload the results to a Phoromatic Server or OpenBenchmarking.org.

System Information | Results Overview | Test Results | System Logs

System Information

test

PHORONIX-TEST-SUITE.COM	Phoronix Test Suite 7.6.0
Intel Core i7 @ 2.90GHz (4 Cores / 8 Threads)	Processor
Apple MacBook Pro	Motherboard
2 x 8 GB LPDDR3-2133MHz	Memory
466GB	Disk
AMD Intel HD 630 + Radeon Pro 560 1536MB	Graphics
Color LCD	Monitor
macOS 10.13.3	OS
17.4.0 (x86_64)	Kernel
Apple LLVM 9.1.0 (clang-902.0.39.1) + GCC 4.2.1	Compiler
APFS	File-System
2880x1800	Screen Resolution

- XPC_FLAGS=0x0

Results Overview

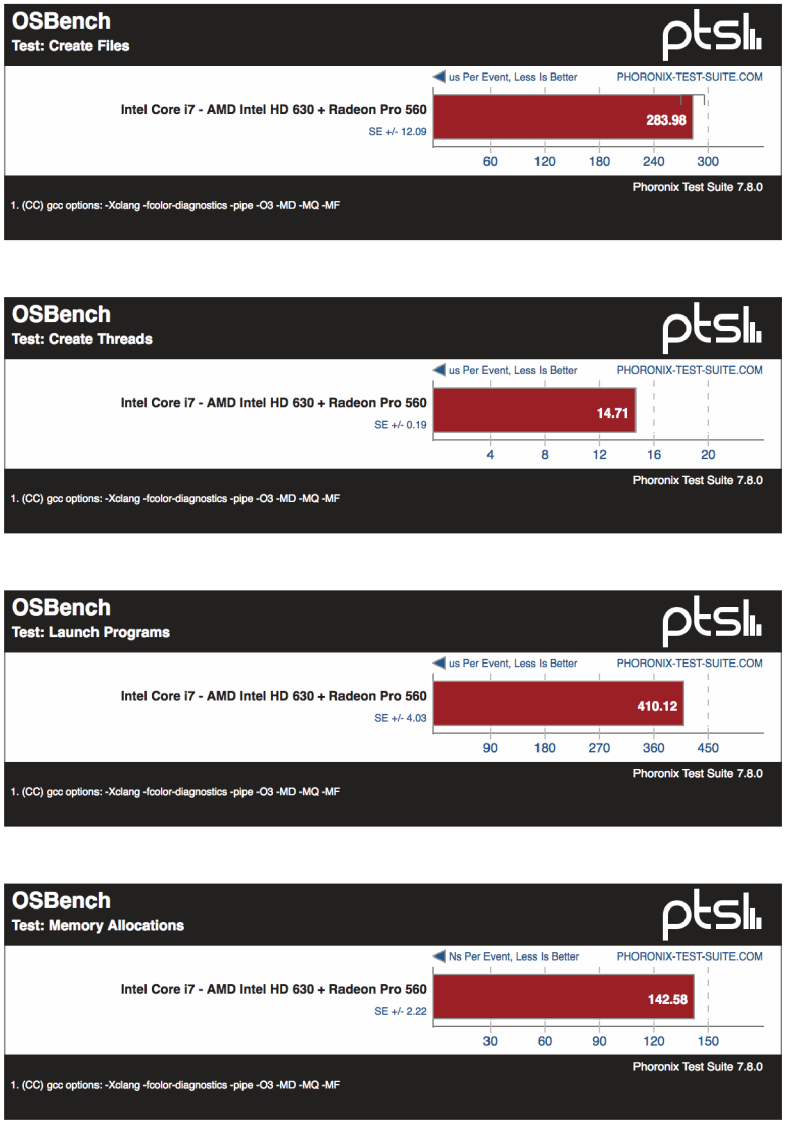
test

	Intel Core i7 - AMD Intel HD 630 + Radeon Pro 560
osbench: Test: Create Files	283.98
osbench: Test: Create Threads	14.71
osbench: Test: Launch Programs	410.12
osbench: Test: Memory Allocations	142.58

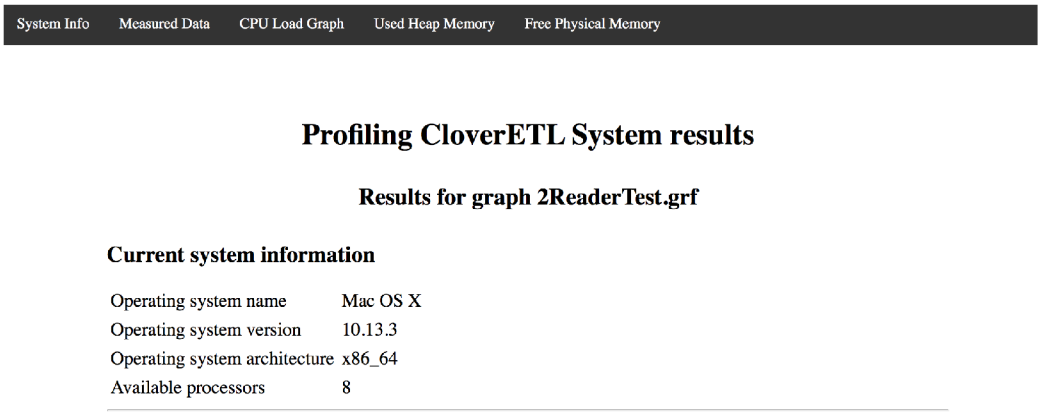
PHORONIX-TEST-SUITE.COM

Obr. D.1: HTML report reprezentujúci Phoronix Test Suite test OSBench. Informácie o systéme.

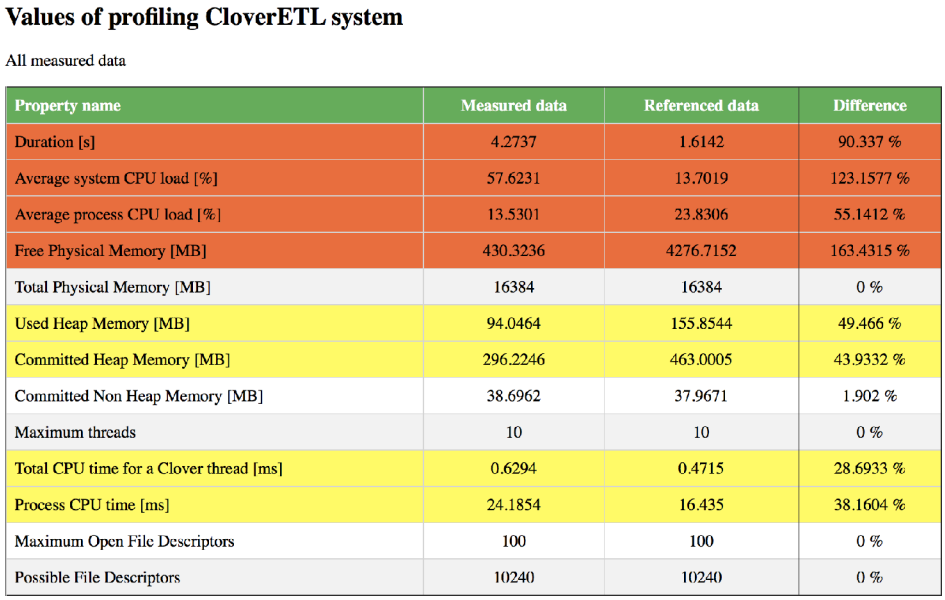
Test Results



Obr. D.2: HTML report reprezentujúci Phoronix Test Suite test OSBench. Grafy s výsledkami jednotlivých podtestov.



Obr. D.3: HTML report reprezentujúci test ETL Grafu. Informácie o systéme.



Values of profiling CloverETL system

All measured data

Property name	Measured data	Referenced data	Difference
Duration [s]	4.2737	1.6142	90.337 %
Average system CPU load [%]	57.6231	13.7019	123.1577 %
Average process CPU load [%]	13.5301	23.8306	55.1412 %
Free Physical Memory [MB]	430.3236	4276.7152	163.4315 %
Total Physical Memory [MB]	16384	16384	0 %
Used Heap Memory [MB]	94.0464	155.8544	49.466 %
Committed Heap Memory [MB]	296.2246	463.0005	43.9332 %
Committed Non Heap Memory [MB]	38.6962	37.9671	1.902 %
Maximum threads	10	10	0 %
Total CPU time for a Clover thread [ms]	0.6294	0.4715	28.6933 %
Process CPU time [ms]	24.1854	16.435	38.1604 %
Maximum Open File Descriptors	100	100	0 %
Possible File Descriptors	10240	10240	0 %

Obr. D.4: HTML report reprezentujúci test ETL Grafu. Namerané dáta porovnané s referenčnými hodnotami a s vyznačenými veľkými odchylkami.

Garbage collection statistics

Information about Garbage Collection during the run of all tests.

Young generation of garbage collection

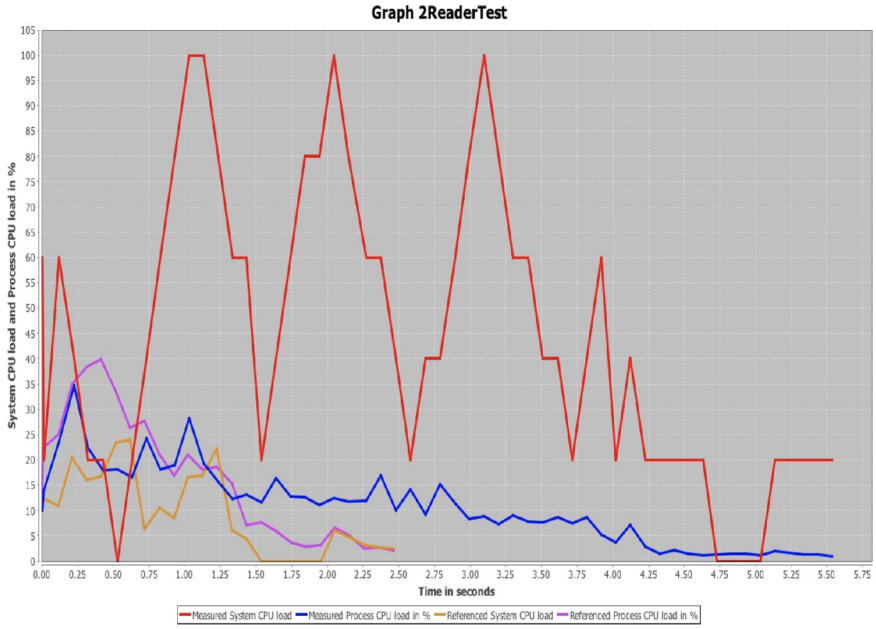
Property name	Measured data	Referenced data	Difference
Garbage collection name	PS Scavenge	PS Scavenge	Same type
Garbage collection count	2.0	2.0	0.0000 %
Garbage collection time [ms]	24.0	28.0	15.3846 %

Old generation of garbage collection

Property name	Measured data	Referenced data	Difference
Garbage collection name	PS MarkSweep	PS MarkSweep	Same type
Garbage collection count	1.0	1.0	0.0000 %
Garbage collection time [ms]	149.0	64.0	79.8122 %

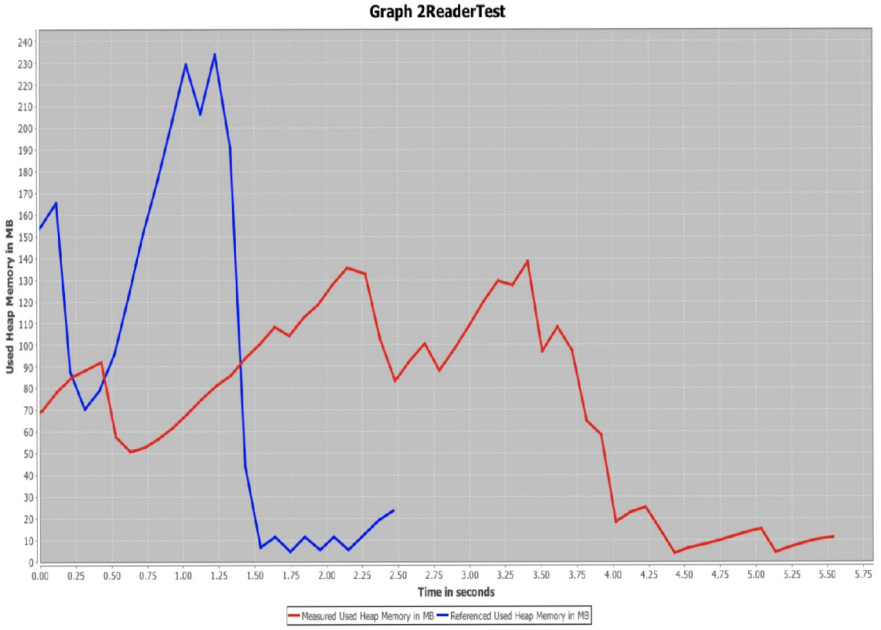
Obr. D.5: HTML report reprezentujúci test ETL Grafu. Nameraná aktivita algoritmu garbage collection porovnaná s referenčnými hodnotami a s vyznačenými veľkými odchylkami.

System and process CPU load



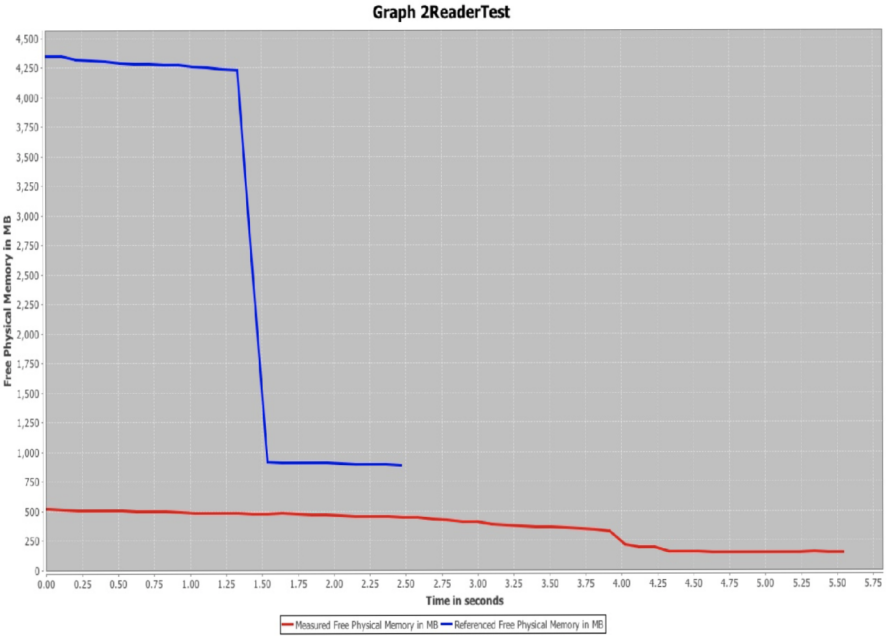
Obr. D.6: HTML report reprezentujúci test ETL Grafu. Graf porovnávajúci zaťaženie CPU nameraných a referenčných dát.

Used heap memory



Obr. D.7: HTML report reprezentujúci test ETL Grafu. Graf porovnávajúci veľkosť použitej heap pamäte nameraných a referenčných dát.

Free physical memory



Obr. D.8: HTML report reprezentujúci test ETL Grafu. Graf porovnávajúci veľkosť voľnej fyzickej pamäte nameraných a referenčných dát.