MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Moderní nástroje pro měření výkonu v praxi

BAKALÁRSKA PRÁCA

Tomáš Borčin

Brno, jeseň 2014

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Vedúci práce: Mgr. Marek Grác, Ph.D.

Poďakovanie

Rád by som sa poďakoval Mgr. Marekovi Grácovi, Ph.D. za vedenie a rady poskytnuté pri písaní tejto práce. Veľká vďaka patrí aj Mgr. Martinovi Večeřovi a Ing. Pavlovi Macíkovi. V neposlednej rade ďakujem mojej rodine, hlavne mojim rodičom, Emílii a Ladislavovi za ich podporu pri štúdiu.

Zhrnutie

Cílem práce je nastudovat a popsat moderní nástroje na testování výkonu aplikací a velmi detailně porovnat jejich vlastnosti (tj. funkcionalita, použitelnost, podpora v IDE, možnosti nastavení, generování reportů atd.). Student napíše jednoduchou aplikaci, která bude odpovídat na dotazy za použití několika protokolů (především WS, JMS, REST, HTTP), kterou nasadí na JBoss AS 7. Poté navrhne několik testovacích scénářů pro všechny nástroje a spoustí měření vyvinuté aplikace. Každé měření musí být provedeno několikrát a bude sledována spolehlivost výsledků. Student by se měl pokusit odhalit příčinu případných rozptylů ve výsledcích. Dále by měla být změřena teoretická propustnost samotného náastroje.

Nástroje pro porovnání:

- Apache jMeter - jmeter.apache.org
- PerfCake - perfcake.org
- Faban - faban.org
- Gatling - gatling-tool.org

Kľúčové slová

performance testy, testovanie, porovnanie nástrojov, Apache JMeter, PerfCake, Faban, Gatling

Obsah

Kapitola 1

Úvod

Obrovský rozvoj informačných technológií so sebou prináša zvyšovanie nárokov na výpočtový výkon [1]. Tento problém sa dá riešiť dvoma spôsobmi. Jednoduchým riešením je neustále zvyšovanie systémových zdrojov spolu s rastúcimi požiadavkami. Treba podotknúť, že toto riešenie je z dlhodobého hľadiska neefektívne, pretože investície do systémových zdrojov budú s ďalším vývojom a údržbou systému rásť, kým neprekročia únosnú hranicu a systém sa stane neudržateľným [2].

Prácnejším, ale o to efektívnejším riešením je testovanie aplikácií a analýza ich výkonu [3]. Test výkonu aplikácie sa uskutočňuje s cieľom zistiť, ako daná aplikácia pracuje pod určitou záťažou. Taktiež môže slúžiť na vyšetrovanie, meranie, potvrdzovanie alebo overovanie ďalších vlastností systému, ako rozšíriteľnosť a efektívnosť využívania prostriedkov. Najčastejšie používanými testami výkonu sú záťažové testy, únavové testy, testy výdrže, kulminačné testy, konfiguračné testy a separačné testy [3].

Výber správneho nástroja na testovanie je často náročnou úlohou, pričom kriticky dôležitá funkcionalita závisí od rôznych faktorov. Testovací nástroj určený na občasné testovanie jednoduchej aplikácie nemusí poskytovať detailné informácie o priebehu testov, avšak jednoduchosť používania a rýchlosť prípravy testov môže byť kritická. Toto zadanie môže spĺňať aj nástroj, ktorý nepodporuje najnovšie technológie a jeho vývoj prebieha oproti iným nástrojom veľmi pomaly. Naopak tomu môže byť vo firme s desiatkami zamestnancov, ktorá vyvíja nové aplikácie s použitím najmodernejších technológií. Tím skúsených testerov denne vykonávajúci množstvo testov kladie vysoké požiadavky na automatizáciu, dobrú organizáciu a presnosť testov. Výsledky testov generované v užívateľsky čitateľnej forme môžu byť prezentované vedeniu firmy bez nutnosti ďalšej úpravy. Pravidelná aktualizácia, aktívny vývoj a poskytovaná užívateľská podpora zohráva nemenej dôležitú rolu.

Úlohou tejto práce je porovnať nástroje používané k testovaniu výkonu aplikácii v praxi a poskytnúť tak čitateľovi prehľad pri výbere nástroja. V práci sú popísané klady a zápory jednotlivých nástrojov a čitateľ si na základe svojich kritérií môže zvoliť nástroj, ktorý mu najviac vyhovuje.

Kapitola 2

Testy výkonu

Testovanie je dôležitou súčasťou vývoja softvéru. Existuje mnoho testov, ktoré majú za účel odhaliť rôzne typy chýb. Spoločnosti vyvíjajúce softvér si veľmi dobre uvedomujú dôležitosť testovania a preto sú testy nedeliteľnou súčasťou životného cyklu vývoja softvéru. Napriek tomu sú testy výkonu softvéru podceňované a nie je im prikladaný dostatočný dôraz [3]. Testy výkonu sa delia na základné testy, záťažové testy, stresové testy, testy vytrvalosti, testy špičky, konfiguračné testy a izolačné testy.

2.1 Základné testy

Základné testy určujú čas potrebný na vykonanie jednotlivých transakcií jedným klientom. Za účelom získania čo najpresnejších údajov je nutné, aby počas vykonávania testov systém nevykonával žiadne iné aktivity. Pomocou týchto informácií je možné zistiť, v akej miere klesol výkon systému s ohľadom na záťaž systému a počet klientov pracujúcich so systémom.

2.2 Záťažové testy

Najpoužívanejším a najjednoduchším testom je záťažový test. Počas tohto testu je systém vystavený stálej záťaži, pričom je monitorovaná dostupnosť služieb, priepustnosť a čas odozvy. Tento test je najvernejšou reprezentáciou reálneho používania systému. Vkladanie a čítanie dát musí byť oneskorené tak, ako je tomu pri práci užívateľa.

2.3 Stresové testy

Stresové testy vystavujú aplikáciu rastúcej záťaži a tým spôsobia, že celá aplikácia alebo jej časť zlyhajú. Zlyhaním systému môže byť nedostupnosť celej aplikácie, jej služby alebo veľmi dlhý čas odozvy. Následne je možné určiť maximálnu únosnú záťaž systému. Znalosť limít systému je dôležitá pre budúci vývoj aplikácie. Maximálna záťaž systému by mala poskytovať dostatočnú rezervu v závislosti na spôsobe používania aplikácie a množstvo klientov, ktoré ju používa.

2.4 Testy vytrvalosti

Testy vytrvalosti podrobujú systém stálej očakávanej záťaži, pričom sú monitorované rôzne časti systému. Vďaka tomu je možné odhaliť problémy s pamäťou, neuvoľnené zdroje a celkovú degradáciu výkonu v čase. Je potrebné zabezpečiť, aby výkon časom neklesal, ale bol rovnaký ako pri spustení, prípadne sa zvyšoval. Cieľom testu je zistiť, ako sa systém správa pri stálom používaní, tak ako tomu bude po jeho nasadení do praxe.

2.5 Testy špičky

Počas testu špičky dochádza k náhlemu zvýšeniu záťaže. Úlohou tohto testu je zistiť ako systém reaguje na náhle zmeny záťaže.

2.6 Izolačné testy

Izolačné testy pomáhajú odhaliť chybné komponenty opakovaným spúšťaním neúspešných testov.

2.7 Konfiguračné testy

Konfiguračné testy pozorujú, ako zmeny konfigurácie ovplyvňujú výkon, jednotlivé komponenty a celkové správanie systému. Zmeny konfigurácie môžu systému pomôcť lepšie zvládať zmeny záťaže[4].

Kapitola 3

Metodika porovnávania nástrojov

V nasledujúcom zozname popisujem postup práce a definujem kategórie, ktoré zohľadňujem pri porovnávaní a hodnotení nástrojov.

1. Inštalácia, oboznámenie sa s nástrojmi, študovanie dokumentácie, inštalácia nástro-jov do vývojových prostredí IntelliJ IDEA, NetBeans Eclipse, nástroja Apache Maven a nastavovanie generovania výsledkov, pričom sa kladie dôraz na jednoduchosť používania nástroja.

2. Návrh a vytvorenie testov, nastavenie generovania výsledkov. Dôraz sa kladie na jednoduchosť a rýchlosť vytvorenia testovacích scenárov, ako aj prínos dokumentácie pri ich tvorbe a riešení vzniknutých problémov.

3. Vytvorenie testovacej architektúry, ktorá spúšťa a riadi priebeh testov. Tu je dôležité si všímať, nakoľko nástroje podporujú automatizáciu testov a problémy, ktoré pri jej vytváraní vznikli pri jednotlivých nástrojoch.

4. Spustenie testov. Sem patrí monitorovanie testov a sledovanie chýb, ktoré sa pri testovaní vyskytli.

5. Vyhodnotenie výsledkov. Tu je dôležité si všímať nie len výkon nástrojov, ale aj spoľahlivosť výsledkov. Veľký rozdiel vo výsledkoch znamená znižuje relevantnosť výsledkov.

6. Konečné zhodnotenie nástrojov na základe hodnotiacich kritérií.

Kapitola 4

Nástroje

4.1 Apache JMeter

Apache JMeter je písaný v jazyku Java a dostupný pod open-source licenciou. Nástroj poskytuje podporu pre testovanie statických, ako aj dynamických zdrojov a podporuje širokú škálu protokolov [5]. Najnovšia verzia 2.12 je dostupná na stránke www.jmeter.apache.org/download_jmeter.cgi. Ja som používal verziu 2.11.

4.1.1 Inštalácia

Keďže je Apache JMeter aplikácia písaná v jazyku Java, vyžaduje k svojmu behu systém s nainštalovaným Java behovým prostredím JRE[1] verzie 6 a vyššej. V operačnom systéme Ubuntu 14.04 LTS je možné Apache JMeter nainštalovať aj cez softvérové centrum alebo v termináli pomocou príkazu *sudo apt-get install jmeter*. Inštalácia cez softvérové centrum alebo pomocou príkazu *sudo apt-get install jmeter* je jednoduchšia, ale obsahuje dva roky starú verziu 2.8.1.

4.1.2 Používanie

Spustenie programu je možné z príkazového riadka pomocou príkazu *jmeter.bat* pre Windows a *./jmeter* pre Linux a MacOS, prípadne univerzálnym príkazom *java -jar ApacheJMeter.jar*. Otvorí sa užívateľské rozhranie, v ktorom je možné vytvárať testy a nastavovať generovanie výsledkov. Dokumentácia nepopisuje vytváranie testov v textovom editore tak, ako pri zvyšných nástrojoch, preto je nutné k tvorbe testov používať toto rozhranie. Spúšťanie testov v prebieha spustením príkazu *./jmeter* s prepínačom "-n", ktorý zabezpečí, aby sa test spustil bez grafického rozhrania.

4.1.3 Dokumentácia

Užívateľská dokumentácia je dostupná na stránke http://jmeter.apache.org/usermanual/index.html. Apache JMeter je pomerne robustný nástroj, čo sa odrazilo aj na rozsahu dokumentácie. Ako pozitívum hodnotím wiki stránku http:

1. Java Runtime Environment(JRE) je prostredie umožnujúce spúšťať Java aplikácie.

//wiki.apache.org/jmeter/, ktorá obsahuje návody, popisuje rozšírenia a tiež poskytuje informácie pre vývojárov. Nachádza sa tu aj návod popisujúci nastavenie Apache JMetra pre používanie v Eclipse IDE. Pozitívom je aj FAQ stránka http://wiki.apache.org/jmeter/JMeterFAQ ponúkajúca odpovede na najčastejšie problémy, s ktorými sa môže užívateľ pri používaní Apache JMetra stretnúť. Dokumentácia je napriek svojej rozsiahlosti prehľadná a pre každý typ testu poskytuje samostatnú kapitolu popisujúcu vytvorenie testu od úplného začiatku.

4.1.4 Použiteľnosť vo vývojovom prostredí

Eclipse
Návod http://wiki.apache.org/jmeter/JMeterAndEclipseHowTo z wiki strán-ky čitateľa hneď v úvode presmeruje na stránku http://people.apache.org/~mkostrze/jmeter-eclipse/jmeter-eclipse.html, ktorá bola naposledy upravená 10.03.2004. Stránka odkazuje na neexistujúce zdroje a nepodarilo sa mi podľa nej nastaviť Apache JMeter tak, aby sa dal jednoducho použiť v Eclipse IDE verzie 4.4.1.

NetBeans
Apache JMeter doplnok je možné do NetBeans IDE nainštalovať v menu Nástroje (Tools) → Doplnky (Plugins). V okne doplnkov je potrebné vybrať panel Dostupných doplnkov (Available Plugins), vyhľadať JMeter doplnok a spustiť inštaláciu. Následne je možné do projektu pridať nový JMeter plán. Doplnok vytvorí v adresári projektu nový priečinok "jmeter", ktorý obsahuje jednoduchý návrh testu. V okne služieb vznikne nová služba Generátory záťaže (Load Generators) obsahujúca generátor JMeter, ktorému je nutné priradiť testovací plán. Testy je možné spustiť samostatne alebo automaticky pri kompilácii projektu.

IntelliJ IDEA
Doplnok pre IntelliJ IDEA je nutné stiahnuť zo stránky https://plugins.jetbrains.com/plugin/7013. V IntelliJ IDEA v menu Nastavenia (Settings) → IDE nastavenia (IDE Settings) → Doplnky (Plugins) vyberieme možnosť Nainštalovať doplnok z disku (Install plugin from disk). Po dokončení inštalácie a reštartovaní IDE je nutné nastaviť v menu Nastavenia → Nastavenia projektu (Project settings) → JMeter domovský adresár pre Apache JMeter. Následne je možné do projektu vložiť JMeter súbor a v IDE ho aj upravovať. Po spustení testovacieho súboru sa otvorí okno Apache JMeter, v ktorom je možné spustiť testy.

Apache Maven[2]
Chýbajúci doplnok pre Eclipse IDE je možné nahradiť doplnkom pre Maven. Návod je k dispozícii na wiki stránke Apache JMeter https://wiki.apache.org/jmeter/

2. Apache Maven je softvér určený na riadienie životného cyklu zostavovania aplikácií.

JMeterMavenPlugin. Po nastavení doplnku a vytvorení JMeter testovacích scenárov sa tieto testy spúšťajú automaticky pri kompilácii projektu.

4.1.5 Generovanie výsledkov

Apache JMeter poskytuje širokú škálu spôsobov, pomocou ktorých je možné monitorovať priebeh testov. Monitor zaznamenávajúci využitie procesora a pamäte RAM nie je súčasťou Apache JMeter, ale je poskytovaný zdarma ako rozšírenie. Inštalácia doplnku prebieha skopírovaním knižnice JMeterPlugins-Standard.jar do adresára lib/ext, ktorý sa nachádza v domovskom adresári nástroja. Následne je nutné nastaviť aj Apache agenta, ktorý bude na strane servera monitorovať využitie procesora a pamäte. Adresár obsahujúci agenta je nutné priložiť k serveru tak, aby bolo možné pri štarte servera spustiť aj agenta, ktorý sa štartuje spustením skriptu startAgent.sh. Agent používa na komunikáciu port číslo 4444.

4.2 Faban

Faban je napísaný v jazyku Java a jeho licencia je open-source. Skladá sa z dvoch hlavných komponent: Faban Driver Framework a Faban Harness. Driver Framework slúži na definovanie záťaže, riadenie životného cyklu a nastavenie výpisu výsledkov. Faban Harness poskytuje užívateľské rozhranie k spúšťaniu a kontrolovaniu testov, generovaniu výsledkov a ukladaniu naplánovaných testov do fronty [6]. Aktuálna verzia Fabanu je 1.2 a dá sa získať na stránke projeku `www.faban.org/`. Používam túto verziu Fabanu. Faban sa nachádza aj na GitHub-e: `www.github.com/akara/faban`.

4.2.1 Inštalácia

Faban vyžaduje plnú JDK[3] inštaláciu verzie 1.5 a vyššej. Nutnosť inštalácie JDK hodnotím ako negatívum. Najnovšia verzia programu sa nachádza na stránke `www.faban.org/download.html`.

4.2.2 Používanie

Najskôr je nutné spustiť Faban Master, ktorý sa spúšťa v Linuxe pomocou príkazu *./startup.sh*, pre Windows príkazom *startup-using-launcher.bat*. Súbory sa nachádzajú v priečin-ku master/bin a je nutné ich spustiť cez príkazový riadok, pretože po spustení program vypíše adresu, na ktorej beží Faban server. Túto adresu je nutné zadať do prehliadača. Otvorí sa rozhranie, v ktorom je nutné nastaviť vytvorený ovládač definujúci základné parametre testu. Následne je možné upravovať, spúšťať, ukončovať testy a prezerať si vygenerované výsledký testov. Syntax pre spúšťanie testov z príkazového riadku je *FABAN_HOME/bin/fabancli submit benchmark profile /configuration/run.xml*, kde FABAN_-HOME je adresár obsahujúci Faban, fabancli je skript umožňujúci interakciu s Faban Harness. Prvý argument spúšťaného skriptu definuje príkaz, ktorý sa má vykonať. V tomto prípade sa má zaregistrovať nový test. Druhý argument určuje ovládač použitý k spúšťaniu testov, nasledujúci argument určuje profil použitý k spusteniu testov a posledným argumentom je konfiguračný súbor, ktorý obsahuje základné informácie o teste, teda dĺžku testu, počet klientov a verziu Javy použitú v testoch.

4.2.3 Dokumentácia

Stručný návod popisujúci inštaláciu a spustenie prvých testov je dostupný na stránke `http://faban.org/docs/QuickStartTutorial.html`. Návod popisujúci vytvára-nie záťaže je k dispozícii na stránke `http://faban.org/docs/CreatingWorkloadTutorial.html`. Ďalšie návody sú k dispozícii na stránke `http://faban.org/1.2/docs/index.html`.

3. Java Development Kit (JDK) je vývojová platforma pre programovací jazyk Java. Súčasťou balíka JDK sú komponenty a nástroje umožňujúce vývoj Java aplikácií a JRE umožňujúce ich spúšťanie.

4.2.4 Použiteľnosť vo vývojovom prostredí

Žiadne doplnky pre vývojové prostredia Eclipse, NetBeans, IntelliJ IDEA a nástroj Maven som nenašiel.

4.2.5 Generovanie výsledkov

Vytváranie výsledkov a logovacích súborov prebieha automaticky. Pri štarte testu dostane každý test identifikačný reťazec, ktorý sa skladá z názvu profilu použitého pri testovaní, bodky, čísla a velkého písmena. Číslo a písmeno slúži ako jednoznačný identifikátor testu a Faban ho generuje automaticky. Výsledky testu je možné zobraziť v grafickom prostredí Faban Harness. V menu View Results je nutné vybrať test identifikovaný identifikačným reťazcom, čím sa zobrazí stručný výpis výsledku. Detailnejšie výsledky sú k dispozícii v hornom menu v záložkách Detailed Results, Run Log a Statistics. Výsledky testov sa ukladajú do adresára output, ktorý sa nachádza v adresári Fabanu. Pre každý test Faban vytvára adresár, ktorého názov je zhodný s identifikačným reťazcom testu a do tohto adresára ukladá všetky výsledky a logy, ktoré patria k danému testu.

4.3 Gatling

Gatling je open-source, podobne ako všetky predchádzajúce nástroje. Kým predchádzajúce nástroje boli písané výhradne, alebo z veľkej časti v jazyku Java, Gatling je vyvíjaný v jazyku Scala, ktorý beží v tom istom virtuálnom stroji, ako Java a preto je nevyhnutné ju mať nainštalovanú. Gatling ako jediný nástroj nepoužíva jazyk XML[4] pre písanie testovacích scenárov. Scenáre sú písané v jazyku Scala. Jadro nástroja nie je závislé na konkrétnom protokole a je ľahko rozšíriteľné. V súčasnosti poskytuje podporu pre HTTP a JMS protokol [7]. Stránka projektu `www.gatling-tool.org/` je veľmi jednoduchá a obsahuje len pár odkazov, väčšinou na GitHub stránku projektu `www.github.com/excilys/gatling`.

4.3.1 Inštalácia

Staré verzie programu sa nachádzajú na stránke `www.github.com/excilys/gatling/wiki/Downloads`. Najnovšia stabilná verzia 2.1.2 je dostupná na stránke `http://gatling.io/download/`. Nachádza sa tu aj verzia 2.2.0-SNAPSHOT, ktorá je vo vývoji. Ja používam verziu 2.0.1. Základným predpokladom pre používanie programu Gatling je JDK verzie 6, pričom sa odporúča mať nainštalovanú najnovšiu verziu JDK.

4.3.2 Používanie

Program je možné spustiť v príkazovom riadku príka-zom *./gatling.sh* pre Linux a príkazom *gatling.bat* pre Windows. Spúšťacie skripty sa nachádzajú v priečinku bin. Gatling nemá grafické rozhranie, preto testy vytváram v textovom editore podľa dokumentácie. Spúšťa-nie prebieha

4.3.3 Dokumentácia

Zoznam užívateľských manuálov je k dispozícii na stránke `http://gatling.io/docs/2.0.1/`. Nachádza sa tu stručný a prehľadný návod popisujúci inštaláciu a spustenie prvých testov, ako aj pokročilejšie návody detailne popisujúce nastavenie nástroja a vytváranie testov. Dokumentácia je prehľadná, dobre sa v nej orientuje a pozitívom je aj vzhľad, ktorý je kvalitne spracovaný.

4.3.4 Použiteľnosť vo vývojovom prostredí

Eclipse, NetBeans, IntelliJ IDEA
Gatling neposkytuje podporu pomocou samostatných doplnkov pre jednotlivé vývojové prostredia. Integrácia Gatlingu je realizovaná pomocou Apache Maven

4. Extensible Markup Language je všeobecný značkovací jazyk vyvinutý a štandardizovaný konzorciom W3C.

nástroja, ktorý tieto vývojové prostredia podporujú.

Maven
Dokumentácia obsahuje sekciu Extensions, ktorá popisuje nastavenie Maven Plugin a Maven Archetype.

Maven Plugin je možné použiť v každom projekte, ktorý využíva Maven ako nástroj pre správu tvorby aplikácií. Maven je možné použiť v Eclipse, IntelliJ IDEA, ako aj NetBeans. Pri nastavovaní Maven konfiguračného súboru podľa stránky http://gatling.io/docs/2.0.1/extensions/maven_plugin.html je nutné zmeniť hodnotu *io.gat-ling* v elemente *groupId* na hodnotu *io.gatling.highcharts*, pretože závislosť (Dependency) s hodnotou *io.gatling* v maven repozitári nebola k dispozícii.

Gatling je možné pomocou Maven Archetype iba do vývojových prostredí podporujúcich jazyk Scala a obsahujúcich Maven. Aj keď testované vývojové prostredia neobsahujú natívnu podporu jazyka Scala je možné túto funkcionalitu implementovať pomocou doplnkov dostupných pre každé vývojové prostredie. Manuál popisujúci integráciu Gatling do vývojového prostredia pomocou Maven Archetype sa nachádza na stránke http://gatling.io/docs/2.0.1/extensions/maven_archetype.html.

4.3.5 Generovanie výsledkov

Po spustení testu sú výsledky priebežne vypisované do terminálu a zároveň sa vytvorí adresár, ktorý umožňuje zobraziť výsledky v prehľadnej grafickej podobe. Adresár obsahuje aj logovací súbor, ktorý zaznamenáva výsledok každej pre každú vykonanú požiadavku. Výsledky som z terminálu presmeroval do súboru a pri spúšťaní testu nastavujem umiestnenie adresára s výsledkami na adresár, v ktorom sa spúšťa daný test.

4.4 PerfCake

PerfCake je open-source nástroj písaný v jazyku Java. V súčasnosti je k dispozícii verzia 3.3, ktorú aj používam. Dátum vydania verzie 4.0 je naplánovaný na 20.3.2015. Stránka projektu: www.perfcake.org. Github: www.github.com/PerfCake/PerfCake. Na stránke www.github.com/PerfCake sú dostupné pluginy pre Maven, IntelliJ IDEA, Eclipse a rôzne ukážkové testovacie scenáre.

4.4.1 Inštalácia

Základným predpokladom pre používanie PerfCake je systém s Java Runtime Environment verzie 7 alebo vyššej. Tiež je nutné mať správne nastavenú $JAVA_HOME premennú prostredia. Pre operačné systémy Windows a Linux je na stránke www.perfcake.org/download/ pripravená binárna distribúcia, ktorú je možné používať hneď po stiahnutí a rozbalení archívu. Užívatelia operačného systému Mac si musia stiahnuť zdrojový kód a skompilovať ho. Týmto je program pripravený na používanie. Inštalácia je veľmi jednoduchá, prebehla bez komplikácií a zaberie len pár minút.

4.4.2 Používanie

Práca s programom prebieha prostredníctvom príkazoveho riadka. V systéme Windows spustíme PerfCake príkazom *perfcake.bat*, v systémoch Linux a MacOS *./perfcake.sh*. Po spustení týchto príkazov sa zobrazia argumenty, ktoré je možné používať pri spúšťaní programu.

4.4.3 Dokumentácia

Užívateľská dokumentácia sa nachádza na stránke www.perfcake.org/docs/perfcake-user-guide.pdf. Skladá sa zo siedmich kapitol, v ktorých popisuje architektúru, vytváranie scenárov, generovanie záťaže, posielanie správ, generovanie výsledkov, validáciu a rozšírenia PerfCake. Pre každú komponentu testu obsahuje jednoduchý príklad.

4.4.4 Použiteľnosť vo vývojovom prostredí

Eclipse
PerfCake obsahuje doplnok PerfClipse pre vývojové prostredie Eclipse Kepler. Doplnok je možné nainštalovať cez Eclipse Marketplace v menu Help. Doplnok obsahuje aj jednoduché užívateľské prostredie, pomocou ktorého je možné vytvárať testy. PerfClipse vytvoril v rámci svojej záverečnej práce Jakub Knetl. Archív práce je k dispozícii na stránke: https://is.muni.cz/auth/th/396062/fi_b/?lang=cs.

IntelliJ IDEA
IntelliJ IDEA doplnok PerfCakeIDEA je dostupný na stránke https://github.com/PerfCake/PerfCakeIDEA. Pre inštaláciu doplnku je potrebné najskôr stiahnuť archív http://download.jetbrains.com/idea/ideaIU-14.0.2.zip do Maven repozitára, zostaviť z príkazovej riadky príkazom *mvn package* a nakoniec nainštalovať v programe IntelliJ IDEA v menu Súbor (File) → Nastavenia (Settings) → IDE Settings (Nastavenia IDE) → Doplnky (Plugins) → Nainštalovať doplnok z disku... (Install plugin from disk...)[8].

Apache Maven
Doplnok pre Maven a návod k jeho inštalácii je k dispozícii na stránke https://github.com/PerfCake/PerfCakeMavenPlugin. Maven doplnok nahrádza chýbajúce doplnky pre vývojové prostredia NetBeans a IntelliJ IDEA.

4.4.5 Generovanie výsledkov

V prípade nástroja PerfCake sú všetky informácie vypisované do príkazového riadku v časových intervaloch, ktoré sa nastavujú v testovacích scenároch. Výsledky som vytvoril presmerovaním výstupu z terminálu do súboru. Všetky informácie sú prehľadne umiestnené v jednom súbore. Okrem toho je možné PerfCake nastaviť tak, aby výpis generoval do súbora s príponou *.csv*.

Kapitola 5

Testy

5.1 Testovaná aplikácia

K vytvoreniu testovanej aplikácie som použil programovací jazyk Java, konkrétne JDK vo verzii 7 a ako vývojové prostredie NetBeans 8.0.1. Aplikácia sa skladá z dvoch častí: webových služieb a triedy obsluhujúcej Java Message Services (ďalej len JMS). Webové služby som naimplementoval pomocou Java aplikačného programovacieho rozhrania pre REST[1] webové služby, ktoré sa v skratke označuje ako JAX-RS. Aplikácia pozostáva zo siedmich webových služieb, ktoré sú volané pomocou HyperText Transfer Protocol (ďalej len HTTP). HTTP je protokol umožňujúci prenos dokumentov[9].

Druhá časť aplikácie implementuje JMS rozhranie pre posielanie správ medzi dvomi klientami. Testovacie nástroje posielajú správy do fronty a tieto správy sú preposielané do druhej fronty, odkiaľ správu vyberie testovací nástroj.

Webová služba *main* prijíma na vstupe reťazec znakov a za použitia zvyšných webových služieb vypíše na výstup pôvodný reťazec, otočený reťazec, dĺžku reťazca, počet samohlások, spoluhlások, číslic a nakoniec veľkosť pôvodného reťazca v bytoch. Pre účely testov som službu upravil tak, aby po zadaní reťazca "generate: x"vygenerovala reťazec o dĺžke x znakov, pričom každý vygenerovaný znak má veľkosť jeden bajt. Veľkosťou správy rozumieme veľkosť vygenerovaného reťazca. Tento spôsob tvorby som použil preto, aby som nasimuloval sťahovanie informácií zo stránky.

Služba *memoryLeak* sa od služby *main* líši len v tom, že simuluje únik pamäte. Simuláciu som naprogramoval pomocou mapy, ktorá nemá naimplementované metódy equals a hashCode. Správne naprogramovaná mapa musí mať tieto metódy korektne naimplementované, pretože pomocou nich rozhoduje, či sa vkladaná hodnota v mape už nachádza. Bez týchto metód moja mapa nedokáže detekovať duplicity a preto do mapy vkladá stále nové a nové hodnoty. Tým sa vytvorí únik pamäte. Službu používam len pri vytrvalostných testoch, ktoré sa skladajú z dvoch častí: Bez memory leaku a S memory leakom. Pri testovaní Bez memory leaku používajú testy službu *main* a pri testovaní S memory leakom volajú službu *memoryLeak*.

Poslednou priamo volanou službou je *echo*, ktorú používam pri testovaní teoretickej priepustnosti jednotlivých nástrojov. Táto služba vracia jednoduchý reťazec "echo", vďaka čomu dokáže server veľmi rýchlo vrátiť odpoveď a tým sa teoretické úzke

1. Representational state transfer je abstraktný architektúrny vzor pre distribuované prostredie.

hrdlo (Bottleneck) presunie od služby k testovaným nástrojom. Úzke hrdlo (Bottleneck) v softvérovom inžinierstve označuje tú časť systému, ktorá má najnižšiu priepustnosť zo všetkých častí prenosovej sústavy a tým výrazne limituje priepustnosť celého systému[10].

Zdrojové kódy aplikácie som priložil aj na CD. Aplikáciu je možné zostaviť spustením skriptu *build.sh*, ktorý sa nachádza v koreňovom adresári aplikácie. Výstupom zostavenia je súbor s príponou .war, ktorým sa aplikácia nasadzuje na server. Súbor sa nachádza v podadresári target.

5.2 Testovacie prostredie

K testovniu som použil dva samostatné počítače s identickým operačným systémom, procesorom a operačnou pamäťou. Na počítačoch je nainštalovaný 64bitový operačný systém Linux so štvorjadrovým procesorom Quad-Core AMD Opteron(tm) Processor 2350. Každé jadro procesora pracuje s frekvenciou 2 GHz. Operačná pamäť má veľkosť 8GB.

5.3 Testy

Pre potreby porovnania nástrojov som navrhol 7 typov testov. Medzi jednotlivými testami JBoss server reštartujem. Výsledky testov zaznamenávam pomocou generátorov výsledkov, ktoré ponúkaju jednotlivé nástroje. Medzi požiadavkami klientov nenastavujem oneskorenie.

1. **Základný test s jedným klientom**
   Prvý test má za úlohu simulovať maximálnu možnú záťaž systému jedným klientom, pričom je vykonávaný po dobu piatich minút. Na základe tohto testu je možné sledovať efektivitu generovania maximálnej záťaže, generovanie výsledkov a spotrebu zdrojov jedným klientom.

2. **Testy s rastúcim počtom klientov**
   K simulácii stresového typu testu slúži postupné zvyšovanie nárokov na systém. Test sa preto skladá zo série testov, pričom rastie množstvo klientov využívajúcich systém. Klienti zasielajú správy o veľkosti 100 znakov po dobu piatich minút. Je spustených päť druhov testov, z nich každý päťkrát, celkovo je spustených dvadsaťpäť testov pre každý nástroj. Množstvo klientov v testoch je: 10, 50, 100, 150 a 200. Test slúži k identifikácii optimálneho množstva klientov využívajúcich systém a monitorovaniu efektivity nástrojov pri rôznych počtoch klientov.

3. **Testy s rastúcou veľkosťou správ**
   Stresový typ testu je možné nasimulovať aj pomocou rastúcej veľkosti generovaných správ pri fixnom počte 50 klientov. Podobne ako v predchazajúcom prípade sa test skladá zo série testov, pričom sa postupne zväčšuje veľkosť správ. Dĺžka každého testu je päť minút. Zasielaných je päť druhov správ pre každý testovaný nástroj. Počet klientov v každom teste je 50 a veľkosti správ sú: 5 znakov, 1024 znakov (1 KiB), 5120 znakov (5 KiB), 51200 znakov (50 KiB) a 512000 znakov (500 KiB). Veľkosťou správy myslím dĺžku reťazca, ktorý generuje server.

4. **Základný JMS test so 100 klientmi**
   Počas JMS testu nástroje zasielajú správy do vstupnej JMS fronty na serveri. Server správy preposiela do výstupnej fronty, odkiaľ ich vyberajú nástroje. Test prebieha po dobu 5 minút. Hlavným účelom testu je overiť schopnosť nástrojov pracovať s JMS. Samozrejmosťou je sledovanie a porovnanie výkonu nástrojov.

5. **Základný test so 100 klientmi**
   Najčastejšie používaný typ testov výkonu je reprezentovaný pomocou základného testu so 100 klientmi. V teste používam 100 klientov a zasielam správy o dĺžke 5120 znakov, čo sú stredné hodnoty z druhého a tretieho typu testu.

6. **Vytrvalostný test**
   Dlhodobé monitorovanie systému je realizované pomocou vytrvalostných testov. Systém je vystavený záťaži generovanej sto klientmi po dobu jednej hodiny. Veľkosť správy je rovnaká ako v predchádzajúcom teste, t.j. 5120 znakov. Test sa skladá z dvoch častí. V prvej časti je testovaná služba použitá v predchádzajúcich testoch a v druhej časti upravená služba simulujúca únik pamäte. Druhá časť testu ukáže rastúcu spotrebu pamäte a v porovnaní s prvou časťou tak bude možné odhaliť vzniknutý únik pamäte. Pre tento test som musel nastaviť v nástrojoch monitorovanie operačnej pamäte na serveri.

7. **Test teoretickej priepustnosti nástroja**
   Teoretická priepustnosť nástroja určuje, ako rýchlo dokáže nástroj generovať záťaž. Vysoká priepustnosť nástroja umožňuje generovať vyššiu záťaž. Test prebieha generovaním maximálnej možnej záťaže aplikácie, ktorá ako odpoveď vracia vždy rovnaký reťazec. Server je schopný odpovedať na požiadavku v čase kratšom, ako je čas potrebný na vygenerovanie požiadavky. Celková priepustnosť systému je tak limitovaná samotným nástrojom. Na generovanie záťaže je použitých 100 klientov.

5.4 Automatizácia testov

Pre účely testu bola vytvorená architektúra obsahujúca zložená z klientskej a serverovej časti. Architektúra je vytvorená tak, aby bolo možné spúšťať testy v Linuxových operačných systémoch.

5.4.1 Klient

Klient obsahuje adresáre so spúšťacími skriptami a testami, symbolické linky adresárov, testovacie nástroje a skripty spúšťajúce testy a upravujúce testy. Názov adresára obsahuje číslo a popis spúšťaného testu. V každom adresári sú umiestnené podadresáre Apache JMeter, Faban, Gatling a PerfCake, ktoré obsahujú testy pre daný nástroj a skript, ktorý riadi spúštanie testu. V adresároch druhého, tretieho a šiesteho testu sa nachádzajú podadresáre obsahujúce modifikácie daného testu, tak ako je to uvedené v ich definícii. Tieto testy navyše obsahujú skript, ktorý spúšťa jednotlivé modifikácie daných testov. Nasleduje popis skriptov, ktoré sa nachádzajú v koreňovom adresári.

- ***change_runtime.sh***
  Slúži na nastavenie dĺžky testov. Prijíma dva prepínače: "-b" a "-t". Prepínač -t prijíma ako argument číslo, ktoré reprezentuje dĺžku trvania vytrvalostných testov v minútach a argument prepínača -b reprezentuje dĺžku trvania zvyšných testov v sekundách. Implicitná hodnota pre prepínač -t je 60, čiže jedna hodina a pre prepínač -b 300, teda 5 minút. Tieto hodnoty sa použijú v prípade, ak je skript spustený bez argumentov. Skript sa automaticky volá v skripte *run_all.sh*, ktorým sa spúšťajú všetky testy. Tým sa zabezpečí korektné nastavenie času v testoch.
  Použitie: *./change_runtime.sh -b 10 -t 120*

- ***change_server_and_port.sh***
  Skript nastavuje adresu servera, číslo portu pre webové služby a číslo JMS portu. Na vstupe berie tri prepínače: "-s", "-p" a "-m". Argumentom prepínača -s je adresa servera, ktorá môže byť reprezentovaná IP adresou. IP adresa môže byť reprezentovaná číslom, pomocou ktorého je možné jednoznačne identifikovať server v počítačovej sieti alebo aliasom, ktorý je k danej IP adrese pridelený. Prepínač -p prijíma ako argument číslo portu, pomocou ktorého je možné komunikovať s webovými službami aplikácie. A nakoniec prepínač -m, ktorý nastavuje port pre JMS. V prípade, že nie sú pri spustení skriptu použité argumenty, použije sa pre server implicitná hodnota "localhost", pre port webových služieb hodnota "8080" a port JMS sa nastaví na hodnotu "4447". Použité čísla portov sú východiskovými portmi JBoss AS servera pre dané komunikačné technológie. Podobne, ako v predchádzajúcom prípade sa skript automaticky spúšťa v skripte *run_all.sh*, ktorým sa spúšťajú všetky testy.
  Použitie: *./change_server_and_port.sh -s 192.168.0.2 -p 18080 -p 14447*

- ***open_scenarios.sh***
  Po spustení skriptu sa otvoria všetky testovacie scenáre v textovom editore gedit. Skript často používam pri kontrole správneho nastavenia scenárov.
  Použitie: *./open_scenarios.sh*

- ***remove_all_logs.sh***
  Tento skript slúži na vymazanie všetkých záznamov a logovacích súborov, ktoré vznikli pri testovaní. Vygenerované záznamy a logovacie súbory majú značnú veľkosť, čo spôsobuje problémy pri kopírovaní testov.
  Použitie: *remove_all_logs.sh*

- ***remove_big_logs.sh***
  Podobne ako v predchádzajúcom prípade odstraňuje skript súbory, ktoré vznikli pri testovaní. Na rozdiel od predchádzajúceho prípadu odstraňuje len najväčšie súbory. V prípade nástroja Apache JMeter sú to súbory s príponou ”.jtl” a v prípade nástroja Gatling súbory ”simulation.log”. Tieto súbory obsahujú záznam pre každú prijatú odpoveď, čo je užitočné v prípade výskytu chyby, pretože je možné presne identifikovať kedy chyba nastala. Súbory neobsahujú informácie dôležité pre vyhodnotenie a porovnanie testov, preto je žiadúce ich zmazanie po úspešnom ukončení testov.
  Použitie: *remove_big_logs.sh*

- ***restart_faban.sh***
  Nástroj Faban sa skladá z dvoch hlavných komponent: Faban Driver Framework a Faban Harness. Kým Faban Driver Framework slúži k vytváraniu testov a definovaniu testovacej záťaže, Faban Harness slúži k samotnému spúšťaniu a monitorovaniu testov. Komponenta beží na Apache serveri, ktorý je reštartovaný pred každým spustením Faban testu. Cieľom tohto reštartu je minimalizovanie vplyvu predchádzajúcich testov.
  Použitie: *restart_faban.sh*

- ***run_all.sh***
  K spusteniu všetkých testov slúži tento skript, ktorý prijíma 7 prepínačov: ”-s”, ”-p”, ”-m”, ”-t”, ”-b”, ”-u” a ”-d”. Pred samotným sputstením všetkých testov sú spustené skripty *change_runtime.sh* a *change_server_and_port.sh*, ktorým sú predané argumenty prepínačov pre nastavenie dĺžky trvania testov, adresy servera, čísla portu webových služieb a čísla portu pre JMS. Prepínač -u prijíma ako argument meno užívateľa, cez ktorého klient pomocou príkazu *ssh* reštartuje JBoss AS server, ktorý beží na inom stroji. Pokiaľ tento atribút nie je definovaný, skript skončí s chybou. Argument posledného prepínača reprezentuje adresár, v ktorom je umiestnený koreňový adresár obsahujúci server JBoss AS a skript potrebný na jeho obsluhu. V prípade, že adresár nie je definovaný, použije sa implicitná hodnota ”/home/user”, kde user je meno užívateľa nastavené prepínačom -u.

Použitie: *run_all.sh -s 192.168.0.2 -p 18080 -m 14447 -b 10 -t 120 -u tomas -d /home/tomas/server*

5.4.2 Server

Na strane servera sa nachádza adresár obsahujúci server JBoss AS, skript potrebný k jeho reštartovaniu a agenti Apache JMeter a PerfCake, ktorí slúžia k monitorovaniu využitia pamäte na strane servera.

- ***ssh_restart_jboss.sh***
  Skripty spúšťajúce testy na strane klienta reštartujú JBoss AS server pred každým spustením testu. K tomu dochádza spustením skriptu *ssh_restart_jboss.sh* príkazom ssh na strane klienta. Tento skript prijíma dva argumenty. Prvým argumentom je adresár, v ktorom je umiestnený koreňový adresár. Ako druhý argument je potrebné uviesť IP adresu servera. Adresa sa používa pri spustení servera a server pomocou nej komunikuje s klientom. Server sa vždy spúšťa na porte 8080.
  Použitie: *ssh user@192.168.0.2 "/user/dir/ssh_restart_jboss.sh /user/dir 192.168.0.2"*

5.5 Problémy pri vytváraní testov a testovaní

Táto sekcia popisuje problémy, s ktorými som sa stretol pri vytváraní testov a testovaní.

5.5.1 Apache JMeter

- **Výsledky testov**
  Apache JMeter vypisuje výsledky testov do konzoly v rôznych časových intervaloch. Z toho dôvodu som pre Apache JMeter nemohol zostaviť grafy zobrazujúce počet vykonaných požiadaviek počas testu. Asynchrónne vypisovanie výsledkov súvisí s optimalizáciou behu testu. Apache JMeter vypisuje výsledky v priemere každých 30 sekúnd. V dokumentácii sa mi nepodarilo nájsť spôsob, ktorým by som toto vypisovanie výsledkov mohol zmeniť na mnou požadované hodnoty.

5.5.2 Faban

- **Generovanie výsledkov testov**
  Problém vytvárania výsledkov spočíva v tom, že Faban ukladá výsledky do adresára output, ktorý sa nachádza v domovskom adresári nástroja. Dokumentácia nepopisuje spôsob, ktorým by som mohol zmeniť tento implicitný adresár pre ukladanie výslekov. Preto som po každom teste musel adresár pomocou skriptu presunúť do adresára daného testu a zmeniť jeho meno v súlade s číslom testu a iteráciou testu. Ďalší problém, ktorý som musel riešiť je spojený so spôsobom spúšťania testov. Faban testy ukladá do fronty a testy spúšťa postupne. Do skriptu, ktorý spúšťa testy som preto musel pridať príkaz, ktorý zastavil beh skriptu

na dobu potrebnú pre vykonanie testu a vygenerovanie výsledkov a logovacích súborov z testu. Až potom som mohol presunúť adresár s výsledkami a zmeniť jeho meno. Doba, počas ktorej skript čaká je nastavená na dĺžku testu plus 100 sekúnd, ktoré sú potrebné na naštartovanie testu a vygenerovanie výsledkov. Z tohto dôvodu musím dobu čakania upravovať pri každej zmene dĺžky trvania testu. Pri veľkom počte krátkych testov spôsobí toto čakanie značné predĺženie doby testovania.

- **Faban Harness**
  K spúšťaniu testov je nutné najskôr spustiť Faban Harness. Táto komponenta Fabanu je reprezentovaná serverom, ktorý som musel pred každým testom reštartovať, aby som tak zabezpečil rovnaké vstupné podmienky pre každý test. K tomu som musel vytvoriť skript *restart_faban.sh*, ktorý som spúšťal pred každým testom.

- **Monitorovanie využitia operačnej pamäte**
  Faban implicitne monitoruje využitie procesora počas testu. Pre potreby Vytrvalostného testu som potreboval monitorovať využitie operačnej pamäte na serveri. Dokumentácia neobsahuje žiadne informácie o monitorovaní operačnej pamäte. Nepodarilo sa mi vyhľadať žiadne informácie, ako toto monitorovanie nastaviť a preto usudzujem, že to nie je možné.

5.5.3 Gatling

- **Monitorovanie využitia operačnej pamäte**
  Dokumentácia obsahuje návod, pomocou ktorého je možné nastaviť zaznamenávanie využitia operačnej pamäte. Napriek veľkej snahe sa mi podľa tohto návodu nepodarilo nastaviť zaznamenávanie využitia pamäte. Nastavenie vyžadovalo zmenu konfiguračného súboru Gatlingu a inštaláciu programov tretích strán. Po zmene konfiguračného súboru sa stal nástroj nepoužiteľným a nástroj som musel inštalovať odznova. V dokumentácii navyše chýbal návod k inštalácii programov tretích strán. Návod je k dispozícii na stránke: `http://gatling.io/docs/2.1.1/realtime_monitoring/index.html`.

5.6 Výsledky testov

5.6.1 Základný test s jedným klientom

- Apache JMeter

  Počas testu sa nevyskytli žiadne problémy. Apache dosiahol najlepšie výsledky

spomedzi všetkých nástrojov a taktiež najnižší rozdiel medzi najlepším a najhorším výsledkom iterácie testu.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	400242	1334,140
2. iterácia	401162	1337,207
3. iterácia	399109	1330,363
4. iterácia	399407	1331,357
5. iterácia	400173	1333,910
Priemer	400018,6	1333,395

Tabuľka 5.1: Apache JMeter Základný test s jedným klientom

- Faban

  Všetých päť iterácií prvého testu prebehlo bez problémov. Zaujímavá je spoľahlivosť výsledkov jednotlivých iterácií testu, keď medzi najlepším a najhorším výsledkom je rozdiel porovnateľný s rozdielom Apache Jmeter.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	166362	554,540
2. iterácia	167141	557,137
3. iterácia	167346	557,820
4. iterácia	166108	553,693
5. iterácia	168210	560,700
Priemer	167033,4	556,778

Tabuľka 5.2: Faban Základný test s jedným klientom

- Gatling

  Podobne ako v prípade Fabanu všetkých päť iterácií prvého testu prebehlo bez problémov. Gatling vykonal viac požiadaviek, ako Faban. Negatívom je najhoršia spoľahlivosť výsledkov, keď medzi najlepším a najhorším výsledkom je rozdiel 14870 vykonaných požiadaviek. Gatling má najslabší štart zo všetkých nástrojov, keď sa na svoje maximum dostal až 50 sekúnd po štarte, čo je pri 300 sekundovom teste veľmi dlhá doba štartu. Graf zobrazujúci priebeh vykonaných požiadaviek počas testu je k dispozícii v adresári Grafy v priloženom CD.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	302827	1009,423
2. iterácia	296602	988,673
3. iterácia	311472	1038,240
4. iterácia	303419	1011,397
5. iterácia	307912	1026,373
Priemer	304446,4	1014,821

Tabuľka 5.3: Gatling Základný test s jedným klientom

- PerfCake

  Ani počas testov PerfCake sa nevyskytli žiadne problémy. PerfCake dosiahol druhý najlepší výsledok, ale mal výrazne väčší rozdiel vo výsledkoch ako Apache JMeter a Faban.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	365398	1217,993
2. iterácia	372943	1243,143
3. iterácia	370583	1235,277
4. iterácia	372749	1242,497
5. iterácia	365319	1217,730
Priemer	369398,4	1231,328

Tabuľka 5.4: PerfCake Základný test s jedným klientom

- Vyhodnotenie

  Najviac vykonaných požiadaviek a zároveň najnižší rozdiel vo výsledkoch dosiahol Apache JMeter. Nízky rozdiel má aj Faban, ale tu je nutné zohľadniť nízky počet požiadaviek, ktoré vykonal. S vyššou rýchlosťou klesá aj spoľahlivosť výsledkov. Druhý najvyšší počet požiadaviek vykonal PerfCake, ale dosiahol pri tom takmer štyrikrát vyšší rozdiel medzi najlepším a najhorším výsledkom, ako Apache JMeter a Faban. Po ňom nasleduje Gatling a Faban. Za zmienku stojí veľký rozdiel vo výsledkoch Gatlingu. Jasným víťazom tohto testu je Apache JMeter nasledovaný PerfCake. Tretie miesto obsadil Faban a posledné Gatling kvôli veľmi vysokému rozdielu vo výsledkoch.

Nástroj	Priemerný počet požiadaviek	Rozdiel vo výsledkoch
Apache JMeter	400018,6	2053
Faban	167033,4	2102
Gatling	304446,4	14870
PerfCake	369398,4	7624

Tabuľka 5.5: Základný test s jedným klientom

5.6.2 Testy s rastúcim počtom klientov

- Apache JMeter

  Najlepší výsledok v druhom type testu dosiahol Apache JMeter hneď v prvej modifikácii s 10 klientmi. Zaujímavý je aj výsledok 5. iterácie testu, keď Apache JMeter dosiahol výrazne horší výsledok, ako v predchádzajúcich iteráciach. To spôsobilo aj výrazne najhorší rozdiel vo výsledkoch.

  Prvý test – 10 klientov

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	4060026	13533,420
2. iterácia	4052695	13508,983
3. iterácia	4020450	13401,500
4. iterácia	4048666	13495,553
5. iterácia	3549533	11831,777
Priemer	3946274	13154,247

Tabuľka 5.6: Apache JMeter Test s rastúcim počtom klientov – 10 klientov

  Druhý test – 50 klientov

  Kým zvyšné nástroje dosiahli maximum v teste s 50 klientmi, Apache JMeter zaznamenal o približne 22% horší výsledok, ako v prípade testu s 10 klientmi. Tak isto zaznamenal v jednej iterácii výrazný prepad výsledku a podobne, ako v prvom teste výrazne najhorší rozdiel vo výsledkoch.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3233657	10778,857
2. iterácia	3207578	10691,927
3. iterácia	2642832	8809,440
4. iterácia	3192770	10642,567
5. iterácia	3191952	10639,840
Priemer	3093757,8	10312,526

Tabuľka 5.7: Apache JMeter Test s rastúcim počtom klientov – 50 klientov

Tretí test – 100 klientov

Nebyť výrazného prepadu vo výsledkoch testu s 50 klientmi, bol by rozdiel vo výsledkoch týchto modifikácií vyšší. Tentokrát Apache JMeter nezaznamenal výrazný prepad výsledkov v žiadnej z iterácií.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3027911	10093,037
2. iterácia	3058845	10196,150
3. iterácia	3087140	10290,467
4. iterácia	3130614	10435,380
5. iterácia	3061439	10204,797
Priemer	3073189,8	10243,966

Tabuľka 5.8: Apache JMeter Test s rastúcim počtom klientov – 100 klientov

Štvrtý test – 150 klientov

Zaujímavým zistením je mierne zvýšenie výkonu Apache JMeter v teste so 150 klientmi. Maximum z testu s 10 klientmi však neprekonal. Prepad výsledkov sa nevyskytol v žiadnej iterácii testu.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3095486	10318,287
2. iterácia	3086050	10286,833
3. iterácia	3153720	10512,400
4. iterácia	3202832	10676,107
5. iterácia	3184660	10615,533
Priemer	3144549,6	10481,832

Tabuľka 5.9: Apache JMeter Test s rastúcim počtom klientov – 150 klientov

Piaty test – 200 klientov

Posledný druh testu dosiahol porovnateľné výsledky, ako test so 150 klientmi. Prepad výsledkov sa neprejavil ani v poslednom druhu testu.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3012575	10041,917
2. iterácia	3238759	10795,863
3. iterácia	3107115	10357,050
4. iterácia	3081815	10272,717
5. iterácia	3176557	10588,523
Priemer	3123364,2	10411,214

Tabuľka 5.10: Apache JMeter Test s rastúcim počtom klientov – 200 klientov

- Faban

Prvý test – 10 klientov

Najhorší výsledok spomedzi testov s 10 klientmi dosiahol Faban. Pozitívom je, že v teste s 10 klientmi dosiahol Faban najnižší rozdiel vo výsledkoch jednotlivých iterácií testu.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	1608896	5362,987
2. iterácia	1614361	5381,203
3. iterácia	1614381	5381,270
4. iterácia	1616067	5386,890
5. iterácia	1612970	5376,567
Priemer	1613335	5377,783

Tabuľka 5.11: Faban Test s rastúcim počtom klientov – 10 klientov

Druhý test – 50 klientov

Najlepší výsledok dosiahol Faban v teste s 50 klientmi. Zvýšenie výkonu oproti prvej modifikácii je vyše dvojnásobné, pričom si Faban opäť zachoval najvyššiu spoľahlivosť výsledkov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3300257	11000,857
2. iterácia	3304396	11014,653
3. iterácia	3286365	10954,550
4. iterácia	3280246	10934,153
5. iterácia	3303527	11011,757
Priemer	3294958,2	10983,194

Tabuľka 5.12: Faban Test s rastúcim počtom klientov – 50 klientov

Tretí test – 100 klientov

V 5. iterácii testu sa objavila chyba Error initializing driver object. com.sun.faban.dri-ver.ConfigurationException: Only single host:port currently supported. Chyba sa vyskytla len pri jednej iterácii. Chybu sa mi nepodarilo odstrániť, pretože k nej dochádza úplne náhodne a jej popis naznačuje, že sa jedná o chybu nástroja, ktorý nedokázal inicializovať jadro (tzv. Driver) testu.

Výsledky sa oproti predchádzajúcej modifikácii mierne zhoršili, ale Faban si zachoval svoju vysokú spoľahlivosť výsledkov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3208731	10695,770
2. iterácia	3232719	10775,730
3. iterácia	3220658	10735,527
4. iterácia	3225564	10751,880
5. iterácia	Chyba	Chyba
Priemer	3221918	10739,727

Tabuľka 5.13: Faban Test s rastúcim počtom klientov – 100 klientov

Štvrtý test – 150 klientov

Faban opäť zaznamenal mierny pokles vo výsledkoch, ale aj výrazné zlepšenie spoľahlivosti. Kým v predchádzajúcej modifikácii bol rozdiel vo výsledkoch 23988 požiadaviek, v tomto teste je to len 8046 požiadaviek.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3189387	10631,290
2. iterácia	3186480	10621,600
3. iterácia	3184917	10616,390
4. iterácia	3187227	10624,090
5. iterácia	3192963	10643,210
Priemer	3188194,8	10627,316

Tabuľka 5.14: Faban Test s rastúcim počtom klientov – 150 klientov

Piaty test – 200 klientov

V poslednej modifikácii sa znova objavila chyba z 5. iterácie testu so 100 klientmi. Faban tiež v tejto modifikácii zaznamenal najväčší rozdiel vo výsledkoch spomedzi všetkých modifikácií testu s rastúcim množstvom klientov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3186355	10621,183
2. iterácia	3206037	10686,790
3. iterácia	Chyba, viď. test 2.3.5	Chyba, viď. test 2.3.5
4. iterácia	3187980	10626,600
5. iterácia	3217781	10725,937
Priemer	3199538,25	10665,128

Tabuľka 5.15: Faban Test s rastúcim počtom klientov – 200 klientov

- Gatling

  V teste s 10 klientmi dosiahol Gatling len o trochu lepší výsledok, ako Faban. Na druhej strane mal výrazne nižšiu spoľahlivosť výsledkov.

  Prvý test – 10 klientov

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	1831121	6103,737
2. iterácia	1818769	6062,563
3. iterácia	1835757	6119,190
4. iterácia	1812333	6041,110
5. iterácia	1827221	6090,737
Priemer	1822361	6074,537

Tabuľka 5.16: Gatling Test s rastúcim počtom klientov – 10 klientov

Druhý test – 50 klientov

Najhorší výsledok v teste s 50 klientmi zaznamenal Gatling. Na rozdiel od ostatných nástrojov sa oproti predchádzajúcej modifikácii zlepšil len mierne. Rozdiel vo výsledkoch sa však zdvojnásobil.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	1918599	6395,330
2. iterácia	1952562	6508,540
3. iterácia	1913446	6378,153
4. iterácia	1935923	6453,077
5. iterácia	1905032	6350,107
Priemer	1928029	6426,763

Tabuľka 5.17: Gatling Test s rastúcim počtom klientov – 50 klientov

Tretí test – 100 klientov

Pomalý rast výkonu pokračoval aj v teste so 100 klientmi. Zároveň však došlo k zhoršeniu spoľahlivosti výsledkov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	2136292	7120,973
2. iterácia	2167595	7225,317
3. iterácia	2128423	7094,743
4. iterácia	2108157	7027,190
5. iterácia	2148207	7160,690
Priemer	2140062,75	7133,543

Tabuľka 5.18: Gatling Test s rastúcim počtom klientov – 100 klientov

Štvrtý test – 150 klientov

V zlepšovaní výsledkov Gatling pokračoval aj v teste so 150 klientmi. Pozitívom je zlepšenie spoľahlivosti výsledkov, keď sa rozdiel vo výsledkoch najlepšej a najhoršej iterácie klesol zmenšil na polovicu. S prihliadnutím na zvýšenie výkonu je to veľmi pozitívna správa, pretože rozdiel vo výsledkoch rastie s rastúcim počtom vykonaných požiadaviek.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	2323968	7746,560
2. iterácia	2322686	7742,287
3. iterácia	2336059	7786,863
4. iterácia	2334234	7780,780
5. iterácia	2352479	7841,597
Priemer	2333341,75	7777,806

Tabuľka 5.19: Gatling Test s rastúcim počtom klientov – 150 klientov

Piaty test – 200 klientov

Aj v poslednej modifikácii testu s rastúcim počtom klientov dosiahol Gatling zlepšenie oproti predchádzajúcej modifikácii. Rozdiel vo výsledkoch sa zvýšil na hodnotu z testu so 100 klientmi. Pri zvýšení počtu vykonaných požiadaviek to ale nie je tak výrazné zhoršenie spoľahlivosti výsledkov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	2508025	8360,083
2. iterácia	2511472	8371,573
3. iterácia	2490477	8301,590
4. iterácia	2536607	8455,357
5. iterácia	2550135	8500,450
Priemer	2526559,75	8421,866

Tabuľka 5.20: Gatling Test s rastúcim počtom klientov – 200 klientov

- PerfCake

  Najlepší výsledok testu s 10 klientmi dosiahol PerfCake a rozdiel vo výsledkoch bol tiež veľmi malý.

  Prvý test – 10 klientov

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3951517	13171,723
2. iterácia	3963717	13212,390
3. iterácia	3991775	13305,917
4. iterácia	3957599	13191,997
5. iterácia	3984467	13281,557
Priemer	3969815	13232,717

Tabuľka 5.21: PerfCake Test s rastúcim počtom klientov – 10 klientov

Druhý test – 50 klientov

Suverénne najlepší výsledok dosiahol v teste s 50 klientmi PerfCake, pričom sa nezhoršila ani spoľahlivosť výsledkov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	8436852	28122,840
2. iterácia	8468862	28229,540
3. iterácia	8364056	27880,187
4. iterácia	8378125	27927,083
5. iterácia	8457090	28190,300
Priemer	8420997	28069,990

Tabuľka 5.22: PerfCake Test s rastúcim počtom klientov – 50 klientov

Tretí test – 100 klientov

Oproti predchádzajúcej modifikácii dosiahol PerfCake výrazný prepad vo výsledkoch, ako aj v spoľahlivosti výsledkov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	7382610	24608,700
2. iterácia	6638997	22129,990
3. iterácia	7469201	24897,337
4. iterácia	6891892	22972,973
5. iterácia	7257491	24191,637
Priemer	7128038,2	23760,127

Tabuľka 5.23: PerfCake Test s rastúcim počtom klientov – 100 klientov

Štvrtý test – 150 klientov

V štvrtej modifikácii PerfCake zaznamenal mierne zníženie výkonu oproti predchádzajúcej modifikácii, ale zároveň došlo k výraznému zlepšeniu spoľahlivosti výsledkov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	6824064	22746,880
2. iterácia	7221323	24071,077
3. iterácia	7325744	24419,147
4. iterácia	6845474	22818,247
5. iterácia	7289434	24298,113
Priemer	7101207,8	23670,693

Tabuľka 5.24: PerfCake Test s rastúcim počtom klientov – 150 klientov

Piaty test – 200 klientov

Posledná modifikácia opäť zaznamenala mierne zníženie výsledkov. Väčším negatívom je, že sa zároveň znížila aj spoľahlivosť výsledkov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	7011158	23370,527
2. iterácia	6667422	22224,740
3. iterácia	6807714	22692,380
4. iterácia	7142392	23807,973
5. iterácia	7256323	24187,743
Priemer	6977001,8	23256,673

Tabuľka 5.25: PerfCake Test s rastúcim počtom klientov – 200 klientov

- Vyhodnotenie

  Vykonané požiadavky
  Z pohľadu počtu vykonaných požiadaviek je jasným víťazom PerfCake. V teste s 50 klientmi, kde dosiahli najlepší výkon všetky nástroje okrem Gatling, dokázal PerfCake vykonať takmer trikrát viac požiadaviek, ako Apache JMeter a Faban. Najhorší výsledok dosiahol Gatling, ktorý ale ako jediný dokázal zvyšovať počet vykonaných požiadaviek so zvyšujúcim sa počtom klientov. Apache JMeter a Faban dosiahli takmer rovnaké výsledky. Faban výrazne zaostal v teste s 10 klientmi, no v ďalších modifikáciach už mal navrch.

Test	Apache JMeter	Faban	Gatling	PerfCake
10 klientov	3946274	1613335	1825040	3969815
50 klientov	3093758	3294958	1925112	8420997
100 klientov	3073190	3221918	2137735	7128038
150 klientov	3144550	3188195	2333885	7101208
200 klientov	3123364	3199538	2519343	6977002

Tabuľka 5.26: Test s rastúcim počtom klientov – priemerný počet vykonaných požiadaviek

Rozdiel vo výsledkoch
Najspoľahlivejšie výsledky poskytoval počas všetkých testov Faban. Neohrozil ho ani druhý Gatling, ktorého rozdiel vo výsledkoch nie je o mnoho väčší, pretože dosiahol nižší počet vykonaných požiadaviek. Ako vidieť z výsledkov, s rastúcim počtom vykonaných požiadaviek rastie rozdiel vo výsledkoch. Pre porovnanie Faban a Gatling najlepšie poslúži rvý test, v ktorom dosiahli podobné výsledky, ale Gatling mal viac ako trikrát väčší rozdiel vo výsledkoch. Najhoršie dopadol Apache JMeter, ktorého výsledky boli konzistentné, ale niektoré iterácie zaznamenali výrazný prepad vo výsledkoch. Z pohľadu absolútnych čísel dosiahol veľký rozdiel aj PerfCake, no v jeho prospech hrá výrazne vyšší počet vykonaných požiadaviek.

Test	Apache JMeter	Faban	Gatling	PerfCake
10 klientov	510493	7171	23424	40258
50 klientov	590825	24150	47530	104806
100 klientov	102703	23988	59438	830204
150 klientov	116782	8046	29793	501680
200 klientov	226184	31426	59658	588901

Tabuľka 5.27: Test s rastúcim počtom klientov – rozdiel vo výsledkoch

5.6.3 Testy s rastúcou veľkosťou správ

- Apache JMeter

  Najvyšší počet vykonaných požiadaviek v prvej modifikácii testu s rastúcou veľkosťou správy dosiahol Apache JMeter. Vzhľadom k vysokému počtu požiadaviek mal aj dobrý výsledok rozdielu medzi najlepšou a najhoršou iteráciou.

  Prvý test – 5 znaková správa

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	10930840	36436,133
2. iterácia	11432411	38108,037
3. iterácia	11602715	38675,717
4. iterácia	11039125	36797,083
5. iterácia	11279402	37598,007
Priemer	11256898,6	37522,995

Tabuľka 5.28: Apache JMeter Test s rastúcou veľkosťou správy – 5 znakov

Druhý test – 1024 znaková správa (1 KiB)

S nárastom veľkosti správy sa logicky znížilo aj množstvo vykonaných požiadaviek. Zároveň sa ale objavil náhly výkyv počtu požiadaviek u jednej iterácie, ako tomu bolo aj v prípade prvých dvoch modifikácii testu s rastúcim počtom klientov. V tomto prípade nedošlo k zníženiu výkonu, ale naopak k radikálnemu zýšeniu počtu vykonaných požiadaviek v 3. iterácii testu.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	335654	1118,847
2. iterácia	336515	1121,717
3. iterácia	485819	1619,397
4. iterácia	335579	1118,597
5. iterácia	336568	1121,893
Priemer	366027	1220,090

Tabuľka 5.29: Apache JMeter Test s rastúcou veľkosťou správy – 1024 znakov

Tretí test – 5120 znaková správa (5 KiB)

Tretia modifikácia zaznamenala v prípade Apache JMeter v priemere päťnásobné zníženie počtu vykonaných požiadaviek (bez započítania 3. iterácie testu s 1024 znakovou správou). To signalizuje, že sa väčšina záťaže presunula na server alebo prenosovú sústavu, t.j. mimo nástroj Apache JMeter. Rozdiel vo výsledkoch oproti predchádzajúcemu testu výrazne klesol.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	67304	224,347
2. iterácia	67570	225,233
3. iterácia	68036	226,787
4. iterácia	67756	225,853
5. iterácia	67915	226,383
Priemer	67716,2	225,721

Tabuľka 5.30: Apache JMeter Test s rastúcou veľkosťou správy – 5120 znakov

Štvrtý test – 51200 znaková správa (50 KiB)

Desaťnásobné zväčšenie správy spôsobilo približne desaťnásobné zníženie počtu vykonaných požiadaviek. Rozdiel vo výsledkoch sa znova znížil, ale už nie takým výrazným pomerom, ako v prípade 2. a 3. iterácie.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	6805	22,683
2. iterácia	6855	22,850
3. iterácia	6813	22,710
4. iterácia	6768	22,560
5. iterácia	6828	22,760
Priemer	6813,8	22,713

Tabuľka 5.31: Apache JMeter Test s rastúcou veľkosťou správy – 51200 znakov

Piaty test – 512000 znaková správa (500 KiB)

Ďalšie desaťnásobné zväčšenie veľkosti správy opäť znamenalo zníženie počtu prenesených správ rovnakým pomerom. Rozdiel v správach opäť klesol.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	691	2,303
2. iterácia	692	2,307
3. iterácia	693	2,310
4. iterácia	693	2,310
5. iterácia	704	2,347
Priemer	694,6	2,315

Tabuľka 5.32: Apache JMeter Test s rastúcou veľkosťou správy – 512000 znakov

- Faban

  Prvý test – 5 znaková správa

  Faban si rovnako ako v predchádzajúcich testoch drží najvyššiu spoľahlivosť výsledkov. Počet vykonaných požiadaviek už v prvej modifikácii nie je najhorší, ako tomu bolo v predchádzajúcich testoch.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	6807453	22691,510
2. iterácia	6802899	22676,330
3. iterácia	6857271	22857,570
4. iterácia	6831598	22771,993
5. iterácia	6837934	22793,113
Priemer	6827431	22758,103

Tabuľka 5.33: Faban Test s rastúcou veľkosťou správy – 5 znakov

Druhý test – 1024 znaková správa (1 KiB)

Druhá modifikácia bola poznačená dvomi zlyhaniami testu. V 2. a 4. iterácii sa vyskytla rovnaká chyba, ako v teste 2.3.5.

Počet vykonaných požiadaviek klesol. Rovnako klesol aj rozdiel vo výsledkoch. Spoľahlivosť výsledkov výrazne narušili dve zlyhania testov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	339326	1131,087
2. iterácia	Chyba, viď. test 2.3.5	Chyba, viď. test 2.3.5
3. iterácia	339615	1132,050
4. iterácia	Chyba, viď. test 2.3.5	Chyba, viď. test 2.3.5
5. iterácia	336776	1122,587
Priemer	338572,3	1128,574

Tabuľka 5.34: Faban Test s rastúcou veľkosťou správy – 1024 znakov

Tretí test – 5120 znaková správa (5 KiB)

Podobne, ako v predchádzajúcej modifikácii, aj teraz sa vyskytli chyby v 2. a 4. iterácii testu. Počet vykonaných požiadaviek, ako aj rozdiel vo výsledkoch klesol. Dôveryhodnosť výsledkov je znížená dvomi zlyhaniami testov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	68625	228,750
2. iterácia	Chyba, viď. test 2.3.5	Chyba, viď. test 2.3.5
3. iterácia	68748	229,160
4. iterácia	Chyba, viď. test 2.3.5	Chyba, viď. test 2.3.5
5. iterácia	68932	229,773
Priemer	68768,3	229,228

Tabuľka 5.35: Faban Test s rastúcou veľkosťou správy – 5120 znakov

Štvrtý test – 51200 znaková správa (50 KiB)

Štvrtá modifikácia zaznamenala len jednu chybu v 4. iterácii testu. Počet vykonaných požiadaviek spolu s rozdielom vo výsledkoch klesol.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	6794	22,647
2. iterácia	6743	22,477
3. iterácia	6752	22,507
4. iterácia	Chyba, viď. test 2.3.5	Chyba, viď. test 2.3.5
5. iterácia	6772	22,573
Priemer	6765,3	22,551

Tabuľka 5.36: Faban Test s rastúcou veľkosťou správy – 51200 znakov

Piaty test – 512000 znaková správa (500 KiB)

Aj posledná modifikácia testu zaznamenala jednu chybu, tentokrát v prvej iterácii testu. Počet požiadaviek a rozdiel vo výsledkoch opäť klesol.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	Chyba, viď. test 2.3.5	Chyba, viď. test 2.3.5
2. iterácia	610	2,033
3. iterácia	612	2,040
4. iterácia	613	2,043
5. iterácia	615	2,050
Priemer	612,5	2,042

Tabuľka 5.37: Faban Test s rastúcou veľkosťou správy – 512000 znakov

- Gatling

  Najhorší výsledok dosiahol v prvej modifikácii testu s rastúcou veľkosťou správy Gatling. Rozdiel vo výsledkoch je vyšší, ako v prípade poslednej modifikácie testu s rastúcim počtom klientov, ktorý mal rovnaký počet vykonaných požiadaviek.

  Prvý test – 5 znaková správa

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	2578507	8595,023
2. iterácia	2606448	8688,160
3. iterácia	2609597	8698,657
4. iterácia	2516099	8386,997
5. iterácia	2586725	8622,417
Priemer	2571944,75	8573,149

Tabuľka 5.38: Gatling Test s rastúcou veľkosťou správy – 5 znakov

  Druhý test – 1024 znaková správa (1 KiB)

  Počet vykonaných požiadaviek klesol podobne ako u zvyšných nástrojov. Výrazne klesol rozdiel vo výsledkoch, keď sa dokonca dostal na úroveň nástroja Faban.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	334502	1115,007
2. iterácia	337257	1124,190
3. iterácia	337315	1124,383
4. iterácia	334833	1116,110
5. iterácia	336393	1121,310
Priemer	335746,25	1119,154

Tabuľka 5.39: Gatling Test s rastúcou veľkosťou správy – 1024 znakov

  Tretí test – 5120 znaková správa (5 KiB)

  Znova klesol počet vykonaných požiadaviek, ako aj rozdiel vo výsledkoch.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	68448	228,160
2. iterácia	68762	229,207
3. iterácia	68519	228,397
4. iterácia	68745	229,150
5. iterácia	68579	228,597
Priemer	68633,5	228,778

Tabuľka 5.40: Gatling Test s rastúcou veľkosťou správy – 5120 znakov

Štvrtý test – 51200 znaková správa (50 KiB)

Štvrtá modifikácia testu opäť zaznamenala pokles vykonaných požiadaviek, ako aj zodpovedajúci pokles rozdielu vo výsledkoch.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	6829	22,763
2. iterácia	6832	22,773
3. iterácia	6818	22,727
4. iterácia	6806	22,687
5. iterácia	6855	22,850
Priemer	6830,5	22,768

Tabuľka 5.41: Gatling Test s rastúcou veľkosťou správy – 51200 znakov

Piaty test – 512000 znaková správa (500 KiB)

Aj poslednej modifikácii klesol počet vykonaných požiadaviek spolu s rozdielom vo výsledkoch.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	692	2,307
2. iterácia	692	2,307
3. iterácia	689	2,297
4. iterácia	693	2,310
5. iterácia	689	2,297
Priemer	691,5	2,305

Tabuľka 5.42: Gatling Test s rastúcou veľkosťou správy – 512000 znakov

- PerfCake

  Prvý test – 5 znaková správa

  Druhý najlepší výsledok v prvej modifikácii testu s rastúcou veľkosťou správy dosiahol PerfCake. Negatívom je veľký rozdiel vo výsledkoch.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	8297885	27659,617
2. iterácia	7668593	25561,977
3. iterácia	8223258	27410,860
4. iterácia	8250351	27501,170
5. iterácia	8072231	26907,437
Priemer	8102463,6	27008,212

Tabuľka 5.43: PerfCake Test s rastúcou veľkosťou správy – 5 znakov

  Druhý test – 1024 znaková správa (1 KiB)

  PerfCake ako jediný nástroj dosiahol podobné výsledky aj v druhej modifikácii. Rozdiel vo výsledkoch klesol zhruba o polovicu.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	8345874	27819,580
2. iterácia	8192962	27309,873
3. iterácia	8294026	27646,753
4. iterácia	8527301	28424,337
5. iterácia	8436495	28121,650
Priemer	8359331,6	27864,439

Tabuľka 5.44: PerfCake Test s rastúcou veľkosťou správy – 1024 znakov

  Tretí test – 5120 znaková správa (5 KiB)

  Aj v tretej modifikácii dosiahol PerfCake podobné výsledky, ako v predchádzajúcich modifikáciach. Rozdiel vo výsledkoch znova stúpol na hodnotu z prvej modifikácie.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	8502858	28342,860
2. iterácia	8294405	27648,017
3. iterácia	7853568	26178,560
4. iterácia	8094159	26980,530
5. iterácia	8406732	28022,440
Priemer	8230344,4	27434,481

Tabuľka 5.45: PerfCake Test s rastúcou veľkosťou správy – 5120 znakov

Štvrtý test – 51200 znaková správa (50 KiB)

Počet vykonaných požiadaviek sa aj v štvrtej modifikácii pohyboval na úrovni predchádzajúcich modifikácií. Rozdiel vo výsledkoch znovu klesol na polovičnú hodnotu.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	8521986	28406,620
2. iterácia	8226342	27421,140
3. iterácia	8220155	27400,517
4. iterácia	8199410	27331,367
5. iterácia	8248134	27493,780
Priemer	8283205,4	27610,685

Tabuľka 5.46: PerfCake Test s rastúcou veľkosťou správy – 51200 znakov

Piaty test – 512000 znaková správa (500 KiB)

Aj posledná modifikácia dosiahla podobné výsledky, ako predchádzajúce modifikácie. Rozdiel vo výsledkoch sa znova zvýšil.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	8206422	27354,740
2. iterácia	8495780	28319,267
3. iterácia	8382870	27942,900
4. iterácia	7852399	26174,663
5. iterácia	8251700	27505,667
Priemer	8237834,2	27459,447

Tabuľka 5.47: PerfCake Test s rastúcou veľkosťou správy – 512000 znakov

- Vyhodnotenie

Vykonané požiadavky
Najzaujímavejším zistením bol nulový pokles v počte vykonaných požiadaviek nástrojom PerfCake. To je spôsobené minimalizmom PerfCake, ktorý nespracováva odpovede od požiadaviek, ak od nich návratový kód HTTP menší, ako 400. Testovaná aplikácia explicitne nenastavuje HTTP návratový kód služieb. Problém sa prejaví pri veľkých správach, ktoré majú dlhý čas návratu odpovede. V tomto teste neberiem výsledky PerfCake do úvahy.

Najväčšie rozdiely medzi nástrojmi sú v prípade správy s veľkosťou 5 znakov. Takáto správa je dostatočne malá na to, aby ju server dokázal spracovať veľmi rýchlo a do výsledkov sa prenesie výkon nástroja. V prvej modifikácii je možné porovnať PerfCake spolu s ostatnými nástrojmi.

Nasledujúce modifikácie ukazujú značný pokles počtu vykonaných požiadaviek v všetkých nástrojov, okrem PerfCake. Z porovnania tohto testu vychádza ako víťaz Apache JMeter, ktorého výkon je o takmer 10% lepší, ako výkon Faban a Gatling. Nasledujúce modifikácie o výkone nástrojov veľa informácií neprinášajú, pretože prenášané správy sú príliš veľké a hlavnú záťaž znáša server a prenosová sústava.

Test	Apache JMeter	Faban	Gatling	PerfCake
5 znakov	11256899	6827431	2579475	8102464
1KiB správa	366027	338572	336060	8359332
5KiB správa	67716	68768	68611	8230344
50KiB správa	6814	6765	6828	8283205
500KiB správa	695	613	691	8237834

Tabuľka 5.48: Test s rastúcou veľkosťou správ – priemerný počet vykonaných požiadaviek

Rozdiel vo výsledkoch
Z výsledkov vidieť, že hoci Faban vykonal najmenej požiadaviek v prvej modifikácii, drží si spolu s Gatling najnižší rozdiel vo výsledkoch zvyšných modifikácií, ktoré mali porovnateľné množstvo vykonaných požiadaviek. Za zmienku opäť stojí obrovský rozdiel vo výsledkoch Apache JMeter v druhej modifikácii testu. Najlepšie výsledky dosiahol Faban nasledovaný Gatling a Apache JMeter. Výsledky PerfCake opäť nie sú relevantné pre účely porovnávania.

Test	Apache JMeter	Faban	Gatling	PerfCake
5 znakov	671875	54372	93498	629292
1KiB správa	150240	2839	2813	334339
5KiB správa	732	307	314	649290
50KiB správa	87	51	49	322576
500KiB správa	13	5	4	643381

Tabuľka 5.49: Test s rastúcou veľkosťou správ – rozdiel vo výsledkoch

5.6.4 Základný JMS test so 100 klientmi

- Apache JMeter

  Výsledky Apache JMeter obsahovali chybu: JVM should have exitted but did not. Chybu som pri nastavovaní testov a spúšťaní v grafickom prostredí nespozoroval, objavila sa len v automatizovaných testoch spúšťaných cez príkazový riadok. Chyba spôsobila zlyhanie približne 97% požiadaviek.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	113729	379,097
2. iterácia	116433	388,110
3. iterácia	113632	378,773
4. iterácia	114521	381,737
5. iterácia	114332	381,107
Priemer	114529,4	381,765

Tabuľka 5.50: Apache JMeter Základný JMS test so 100 klientmi

- Faban

  Dokumentácia Fabanu neobsahovala žiadne informácie o testovaní JMS. S vynaložením nemalého úsilia sa mi podarilo nájsť na internete blog, ktorý popisoval nastavenie Fabanu, tak aby s ním bolo možné testovať JMS. Článok neobsahoval informácie o nastavení atribútov potrebných pre testovanie JMS. Nadpis článku naznačoval, že existuje pokračovanie, ktoré sa mi nepodarilo nájsť. Z toho usudzujem, že Faban nepodporuje testovanie JMS. Článok bol publikovaný 21.4.2009[11].

- Gatling

  Gatling dokumentácia obsahuje nastavenie testovacieho scenára pre testovanie JMS. Po nastavení všetkých parametrov testu sa pri testovaní objavuje výnimka "User NULL" naznačujúca, že nie je možné identifikovať užívateľa, pomocou ktorého je možné pristupovať k JMS frontám JBoss AS servera. Meno a heslo

užívateľa som nastavil v súlade s návodom a tieto údaje sú rovnaké, ako pri nástrojoch Apache JMeter a PerfCake, u ktorých testy fungujú. Návod je k dispozícii na stránke http://gatling.io/docs/2.1.1/jms.html.

- PerfCake

  Všetých päť iterácií JMS testu prebehlo bez problémov, počas testovania sa nevyskytli žiadne chyby.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	113714	394,867
2. iterácia	112162	413,784
3. iterácia	112527	388,142
4. iterácia	115062	397,234
5. iterácia	115012	377,893
Priemer	113695,4	378,985

Tabuľka 5.51: PerfCake Základný JMS test so 100 klientmi

- Vyhodnotenie

  JMS testu sa zúčastnili len Apache JMeter a Perfcake. Oba nástroje dosiahli podobné výsledky v počte vykonaných požiadaviek a porovnateľný je aj rozdiel vo výsledkoch. V neprospech Apache JMeter hovorí len chyba, ktorá sa objavila počas testovania.

Nástroj	Priemerný počet požiadaviek	Rozdiel vo výsledkoch
Apache JMeter	114529,4	2801
Faban	–	–
Gatling	–	–
PerfCake	113695,4	2900

Tabuľka 5.52: Základný JMS test so 100 klientmi

5.6.5 Základný test so 100 klientmi

- Apache JMeter

  Základný test so 100 klientmi Apache JMeter prebehol bez problémov.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	68390	227,967
2. iterácia	67537	225,123
3. iterácia	68908	229,693
4. iterácia	68753	229,177
5. iterácia	67863	226,210
Priemer	68290,2	227,634

Tabuľka 5.53: Apache JMeter Základný test so 100 klientmi

- Faban

  V 3. iterácii sa objavila rovnaká chyba, ako v 5. iterácii Faban testu so 100 klientmi.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	68286	227,620
2. iterácia	68590	228,633
3. iterácia	Chyba, viď. test 2.3.5	Chyba, viď. test 2.3.5
4. iterácia	67874	226,247
5. iterácia	68567	228,557
Priemer	68329,25	227,764

Tabuľka 5.54: Faban Základný test so 100 klientmi

- Gatling

  Počas testu sa nevyskytli žiadne problémy.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	68775	229,250
2. iterácia	68452	228,173
3. iterácia	68495	228,317
4. iterácia	68273	227,577
5. iterácia	68805	229,350
Priemer	68560	228,533

Tabuľka 5.55: Gatling Základný test so 100 klientmi

- PerfCake

Pri testovaní sa opäť objavila rovnaký problém, ako pri testovaní veľkých správ. V teste bola použitá správa s veľkosťou 5 KiB, čo postačuje na prejavenie sa problému a získanie veľkého množstva vykonaných požiadviek. Výsledky testu neberiem do úvahy.

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	7344119	24480,397
2. iterácia	7536481	25121,603
3. iterácia	7415389	24717,963
4. iterácia	7164312	23881,040
5. iterácia	7952241	26507,470
Priemer	7482508,4	24941,695

Tabuľka 5.56: PerfCake Základný test so 100 klientmi

- Vyhodnotenie

  Vo vyhodnotení nezohľadňujem výsledok nástroja PerfCake. Všetky porovnávané nástroje dosiahli podobné výsledky, ktoré sa líšia len rozdielom vo výsledkoch. Spôsobené to bolo príliš veľkou správou, ktorá zaťažila server a prenosovú sústavu oveľa viac, ako nástroje. Najlepší výsledok dosiahol Gatling, ktorý mal najnižší rozdiel vo výsledkoch. Najvyšší rozdiel vo výsledkoch dosiahol Apache JMeter.

Nástroj	Priemerný počet požiadaviek	Rozdiel vo výsledkoch
Apache JMeter	68290	1371
Faban	68329	716
Gatling	68560	532
PerfCake	7482508	787929

Tabuľka 5.57: Základný test so 100 klientmi

5.6.6 Vytrvalostný test

- Apache JMeter

  Monitorovanie pamäte zlyhalo, pretože nástroj monitoroval spotrebu pamäte na strane klienta a nie na strane servera. Výsledky oboch testov prebehli v poriadku, avšak sú takmer rovnaké a bez znalosti spotreby pamäte ich nie je možné porovnať.

  Bez úniku pamäte

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	820011	227,781
2. iterácia	821934	228,315
3. iterácia	824035	228,899
Priemer	821993,33	228,331

Tabuľka 5.58: Faban Vytrvalostný test bez úniku pamäte

S únikom pamäte

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	813733	226,037
2. iterácia	821163	228,101
3. iterácia	823443	228,734
Priemer	819446,33	227,624

Tabuľka 5.59: Faban Vytrvalostný test s únikom pamäte

- Faban

  Počas Faban vytrvalostného testu spotreba pamäte nebola monitorovaná, lebo sa mi nepodarilo toto monitorovanie nastaviť. Podrobný popis problému sa nachádza v sekcii problémy pri vytváraní testov a testovaní. Výkon nástroja z oboch modifikácií testu je porovnateľne veľký.

  Bez úniku pamäte

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	818265	227,296
2. iterácia	823862	228,851
3. iterácia	818863	227,462
Priemer	820330	227,869

Tabuľka 5.60: Faban Vytrvalostný test bez úniku pamäte

S únikom pamäte

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	822346	228,429
2. iterácia	817627	227,119
3. iterácia	818486	227,357
Priemer	819486,3	227,635

Tabuľka 5.61: Faban Vytrvalostný test s únikom pamäte

- Gatling

  Podobne ako v prípade nástroja Faban, ani v nástroji Gatling sa mi nepodarilo nastaviť monitorovanie využitia operačnej pamäte na serveri. Podrobný popis problému sa nachádza v sekcii problémy pri vytváraní testov a testovaní. Výsledky prvej a druhej iterácie testu sú porovnateľné.

  Bez úniku pamäte

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	816525	226,813
2. iterácia	824327	228,980
3. iterácia	820827	228,008
Priemer	820560	227,933

Tabuľka 5.62: Gatling Vytrvalostný test bez úniku pamäte

  S únikom pamäte

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	817001	226,945
2. iterácia	820303	227,862
3. iterácia	818141	227,261
Priemer	818482	227,356

Tabuľka 5.63: Gatling Vytrvalostný test s únikom pamäte

- PerfCake

  V prípade testu PerfCake bolo monitorovanie pamäte nastavené správne a získal som aj korektné výsledky využitia pamäte na serveri. Problémom ale je, že PerfCake opäť ako v prípade ostatných veľkých správ nečakal na odpoveď servera a služba, ktorá mala spôsobiť únik pamäte sa nevykonala. Spotreba pamäte sa v

oboch prípadoch líšila len v rádoch kilobajtov. Výsledky oboch testov sú porovnateľné.

Bez úniku pamäte

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	96115245	26698,679
2. iterácia	97714275	27142,854
3. iterácia	97478553	27077,376
Priemer	97102691	26972,970

Tabuľka 5.64: PerfCake Vytrvalostný test bez úniku pamäte

S únikom pamäte

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	91105106	25306,974
2. iterácia	93325088	25923,636
3. iterácia	93084772	25856,881
Priemer	92504988,7	25695,830

Tabuľka 5.65: PerfCake Vytrvalostný test s únikom pamäte

- Vyhodnotenie

  Pôvodný zámer ukázať únik pamäte nevyšiel. V dvoch prípadoch sa mi nepodarilo nastaviť monitorovanie pamäte v nástrojoch. Apache JMeter meral využitie pamäte na strane klienta a nie na strane servera. PerfCake monitorovanie pamäte zvládol, ale keďže jeho požiadavky neboli serverom spracované, neprejavil sa únik pamäte. Výsledky testov vsetkých nástrojov, okrem PerfCake sú porovnateľné.

  Vykonané požiadavky

Test	Apache JMeter	Faban	Gatling	PerfCake
Bez úniku pamäte	821993	820330	820560	97102691
S únikom pamäte	819446	819486	818482	92504989

Tabuľka 5.66: Vytrvalostný test – priemerný počet vykonaných požiadaviek

Rozdiel vo výsledkoch

Test	Apache JMeter	Faban	Gatling	PerfCake
Bez úniku pamäte	4024	5597	7802	1599030
S únikom pamäte	9710	4719	3302	2219982

Tabuľka 5.67: Vytrvalostný test – rozdiel vo výsledkoch

5.6.7 Test teoretickej priepustnosti nástroja

- Apache JMeter

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	13329690	44432,300
2. iterácia	13256854	44189,513
3. iterácia	13256973	44189,910
4. iterácia	13576708	45255,693
5. iterácia	13300559	44335,197
Priemer	13344156,8	44480,523

Tabuľka 5.68: Apache JMeter Test teoretickej priepustnosti

- Faban

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	8385444	27951,480
2. iterácia	8670415	28901,383
3. iterácia	8810458	29368,193
4. iterácia	8775981	29253,270
5. iterácia	8762981	29209,937
Priemer	8681055,8	28936,853

Tabuľka 5.69: Faban Test teoretickej priepustnosti

- Gatling

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	3044179	10147,263
2. iterácia	2846062	9486,873
3. iterácia	2989634	9965,447
4. iterácia	3034939	10116,463
5. iterácia	2979783	9932,610
Priemer	2978919,4	9929,731

Tabuľka 5.70: Gatling Test teoretickej priepustnosti

- PerfCake

Iterácia testu	Počet vykonaných požiadaviek	Počet požiadaviek za sekundu
1. iterácia	7700715	25669,050
2. iterácia	7952500	26508,333
3. iterácia	7922331	26407,770
4. iterácia	7539454	25131,513
5. iterácia	7714608	25715,360
Priemer	7765921,6	25886,405

Tabuľka 5.71: Perfcake Test teoretickej priepustnosti

- Vyhodnotenie

  Test teoretickej priepustnosti najlepšie zvládol Apache JMeter nasledovaný nástrojom Faban a PerfCake. Ako posledný skončil s veľkým odstupom nástroj Gatling. Apache JMeter dosiahol s ohľadom na počet požiadaviek najlepší výsledok aj pri porovnaní rozdielu vo výsledkoch. Nasleduje Faban, PerfCake a Gatling.

Nástroj	Priemerný počet požiadaviek	Rozdiel vo výsledkoch
Apache JMeter	13344157	319854
Faban	8681056	425014
Gatling	2978919	198117
PerfCake	7765922	413046

Tabuľka 5.72: Test teoretickej priepustnosti

Kapitola 6

Záver

6.1 Zhrnutie

Cieľom práce bolo porovnanie nástrojov, ich vlastností a porovnanie výsledkov testov. Vďaka tomu si čitateľ môže vybrať jemu vyhovujúci nástroj. Pre zjednodušenie výberu na záver poskytujem stručné zhrnutie a porovnanie všetkých hodnotených kategórií. Súčasťou hodnotenia je aj tabuľka, ktorá pre každú kategóriu obsahuje známku od 1 do 4 reprezentujúcu umiestnenie nástroja v danej kategórii. Nižšie číslo znamená lepšie umiestnenie a ak nástroje dosiahli rovnaké výsledky, hodnotím ich rovnakou známkou.

- **Inštalácia**
  V prvej kategórii hodnotím Apache JMeter a PerfCake známkou 1 a Gatling s Fabanom známkou 2. Všetky nástroje sa inštalujú veľmi jednoducho, stačí ich stiahnuť a rozbaliť. Gatling a Faban majú nižšiu známku, lebo požadujú JDK a nie JRE ako v prípade Apache JMeter a PerfCake.

- **Používanie**
  Víťazmi kategórie sa stávajú Apache JMeter, Gatling a PerfCake. Práca s nimi je jednoduchá a rýchla. Vytvorené testy sa spúšťajú jedným príkazom. Faban vyžaduje pred spustením testov naštartovanie servera.

- **Dokumentácia**
  Nebyť problému s JMS testami, víťazom by bol Gatling. Jeho dokumentácia je najprehľadnejšia a najlepšie sa v nej orientuje. Problém s nastavením JMS Gatling posunul za Apache JMeter a PerfCake, ktoré sa umiestnili na spoločnom prvom mieste. Dokumentácia oboch nástrojov dostačovala k riešeniu všetkých problémov a orientovalo sa v nej rovnako dobre. Posledný skončil Faban. Jeho dokumentácia je najmenej prehľadná.

- **Vývojové prostredia**
  Kategóriu použiteľnosti vo vývojových prostrediach vyhráva Apache JMeter spolu s PerfCake. Apache JMeter poskytuje doplnok pre NetBeans, IntelliJ IDEA a Maven. PerfCake má podporou pre Eclipse Kepler, IntelliJ IDEA a Maven. Nasleduje Gatling s podporou pre Maven a nakoniec Faban neposkytujúci žiadnu podporu pre vývojové prostredia.

- **Generovanie výsledkov**
  Generovanie výsledkov Gatlingu a PerfCake prebieha v scenároch, je jednoduché a rýchle. JMeter výpis výsledkov je nutné nastaviť v grafickom užívateľskom rozhraní a dokumentácia nepopisuje nastavenie v scenároch, čo je jednoduchšia, ale hlavne rýchlejšia cesta. Najhoršie skončil Faban, v ktorom nie je možné zmeniť východiskový adresár pre ukladanie výsledkov a tým vzniká problém, ktorý som musel riešiť pomocou skriptov.

- **Tvorba testov**
  PerfCake testy sa tvoria najjednoduchšie a najrýchlejšie. Nasleduje Gatling s jazykom Scala, na ktorý som si musel chvíľu zvykať. Apache JMeter testy som musel vytvárať v grafickom užívateľskom prostredí, ktoré nie je také rýchle, ako písanie a upravovanie testov v textovom editore. Posledným je Faban, pre ktorý som musel vytvoriť Driver, profil testu a nakoniec samotný scenár.

- **JMS testy**
  V JMS testoch najlepšie obstál PerfCake, ktorý vykonal približne rovnaké množstvo požiadaviek, ako Apache JMeter, ale počas testu sa neobjavili problémy, ako v prípade nástroja Apache JMeter. Najhoršiu možnú známku obdržali Gatling a Faban, pretože u nich nebolo možné nastaviť JMS testy.

- **Monitorovanie pamäte**
  Víťazmi tejto kategórie sa stávajú Apache JMeter a PerfCake, pretože u nich bolo možné nastaviť monitorovanie pamäte a počas testov monitorovanie aj prebiehalo. V prípade Apache JMeter bola síce monitorovaná pamäť na klientskom stroji, ale to súvisí so zlým nastavením monitorovania. Faban a Gatling hodnotím najhoršou známkou, pretože u nich nebolo možné nastaviť monitorovanie pamäte.

- **Výsledky testov**
  Žiadny nástroj nedosiahol počas testov bezchybné výsledky. Spomedzi všetkých nástrojov najlepšie výsledky dosahoval Apache JMeter. Ako druhý najlepší hodnotím nástroj PerfCake, ktorý síce kvôli zlému nastaveniu neposkytol relevantné výsledky pri testoch s veľkými správami, ale vo všetkých zvyšných testoch dosahoval veľmi dobré výsledky. Na druhom mieste mohol skončiť aj Faban, nebyť zlyhania niektorých iterácií testov. V počte vykonaných požiadaviek výrazne zaostával len v testoch s nízkym počtom klientov. Na poslednom mieste sa umiestnil Gatling, ktorý mal vo väčšine testov najmenej vykonaných požiadaviek.

- **Spoľahlivosť výsledkov**
  Spoľahlivosť výsledkov nie je možné zhodnotiť jednoznačne. Výsledky testov ukázali, že rozdiel vo výsledkoch nerastie s množstvom vykonaných požiadaviek lineárne, ale skôr polynomiálne. Napriek tomu usudzujem, že najvyššiu spoľahlivosť výsledkov dosiahol Faban nasledovaný Apache JMeter a na treťom mieste PerfCake. Oba nástroje zaznamenali počas testov výrazné

výkyvy v niektorých iteráciách, ale v absolútnych číslach mal Apache JMeter lepšiu spoľahlivosť výsledkov ako PerfCake. S prihliadnutím na nízky počet vykonaných požiadaviek na posledné miesto posúvam Gatling.

- **Teoretická priepustnosť**
  Test teoretickej priepustnosti suverénne vyhral Apache JMeter nasledovaný nástrojom Faban a Perfcake. Na poslednom mieste skončil s výrazným odstupom Gatling.

- **Priemer**
  Priemer je vypočítaný ako súčet všetkých známok vydelený počtom hodnotiacich kategórií. Tým získam priemernú známku a na jej základe určujem najlepší nástroj.

Kategória	**Apache JMeter**	**Faban**	**Gatling**	**PerfCake**
Inštalácia	1	2	2	1
Používanie	1	2	1	1
Dokumentácia	1	4	3	1
Vývojové prostredia	1	4	2	1
Generovanie výsledkov	2	3	1	1
Tvorba testov	3	4	2	1
JMS testy	2	4	4	1
Monitorovanie pamäte	1	4	4	1
Výsledky testov	1	3	4	2
Spoľahlivosť výsledkov	2	1	4	3
Teoretická priepustnosť	1	2	4	3
Priemer	1,45	3	2,82	1,45

Tabuľka 6.1: Hodnotiaca tabuľka

Z hodnotiacej tabuľky vyplýva, že najlepšími nástrojmi sú Apache JMeter a PerfCake, nasleduje Gatling a posledný je Faban. Napriek tomu hodnotím ako najlepší nástroj Apache JMeter. Perfcake získal veľmi dobré hodnotenie v kategóriách, ktoré sa priamo nedotýkali testov. V samotných testoch sa potom umiestnil až na treťom mieste za prvým Apache JMeter a druhým Faban. Kategóriám, ktoré sa týkajú testov prikladám vyššiu váhu, ako zvyšným kategóriám. Na druhej strane výsledky testov s veľkými správami ukázali, že na výbere nástroja nezáleží pokiaľ je výrazne zaťažený server a prenosová sústava. Konečný výber nástroja závisí od preferencií užívateľa.

Literatúra

[1] W. Garside. Computing power: capacity on demand. www.computerweekly.com/feature/Computing-power-capacity-on-demand. Navštívené: 26.12.2014.

[2] J. Sochor. ÚVT MU zpravodaj. www.ics.muni.cz/bulletin/articles/61.html. Navštívené: 22.12.2014.

[3] Ian Molyneaux. *The Art of Application Performance Testing*. O'Reilly Media, Inc., Sebastopol, first edition edition, 2009. ISBN: 9780596520663.

[4] B.M. Subraya. *Integrated Approach to Web Performance Testing: A Practitioneres Guide*. IRM Press, Hershey, 2006. ISBN: 9781591407850.

[5] Apache Software Foundation. Apache JMeter. http://jmeter.apache.org/index.html. Navštívené: 19.12.2014.

[6] Faban. http://faban.org/about.html. Navštívené: 19.12.2014.

[7] Gatling. http://gatling.io/docs/2.1.1/. Navštívené: 19.12.2014.

[8] M. Michalko. PerfCakeIDEA. https://github.com/PerfCake/PerfCakeIDEA. Navštívené: 19.12.2014.

[9] R. Fielding. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. http://tools.ietf.org/html/rfc7230. Navštívené: 13.12.2014.

[10] Bob Wescott. *The Every Computer Performance Book: How to Avoid and Solve Performance Problems on The Computers You Work With*. CreateSpace Independent Publishing Platform, first edition edition, 2013. ISBN: 1482657759.

[11] Kim LiChong. How to write a JMS Faban driver (Part One). https://blogs.oracle.com/klc/entry/using_faban_as_an_asynchronous. Navštívené: 22.12.2014.

Dodatok A

Obsah priloženého CD

Súčasťou tejto práce je aj CD obsahujúce všetky materiály, ktoré boli vytvorené v rámci písania práce. Klientská a serverová časť je pre jednoduchšiu manipuláciu a prenos medzi počítačmi skomprimovaná do archívu. Adresárová štruktúra CD vyzerá nasledovne:

```
/
├── Aplikácia ..... Adresár obsahujúci zdrojové kódy testovanej Aplikácie
│   └── build.sh ..... Skript na zostavenie aplikácie
├── Bakalárska práca ..... Adresár s bakalárskou prácou
│   ├── bibliografie.bib ..... Bibliografická databáza práce
│   ├── bp.tex ..... Zdrojový kód práce
│   └── bp.pdf ..... Hotová práca
├── Grafy ..... Adresár obsahujúci grafy s výsledkami testov
├── klient.zip ..... Archív obsahujúci klientskú časť testov
│   ├── 1, 2, 3, 4, 5, 6, 7 ..... Adresáre s testami
│   └── Nastroje ..... Adresár obsahujúci nástroje
│       ├── apache-jmeter-2.11 ..... Nástroj Apache JMeter
│       ├── faban ..... Nástroj Faban
│       ├── gatling-charts-highcharts-2.0.1 ..... Nástroj Gatling
│       ├── change_runtime.sh ..... Skript meniaci čas testov
│       ├── change_server_and_port.sh ..... Skript meniaci server a port
│       ├── open_scenarios.sh ..... Skript otvárajúci scenáre
│       ├── remove_all_logs.sh ..... Skript odstraňujúci všetky logy a výsledky
│       ├── remove_big_logs.sh ..... Skript odstraňujúci velké logy a výsledky
│       ├── restart_faban.sh ..... Skript reštartujúci Faban server
│       └── run_all.sh ..... Skript spúšťajúci testy
└── server.zip ..... Archív obsahujúci serverovú časť testov
    ├── Nastroje ..... Adresár obsahujúci JBoss AS server
    │   └── jboss-as-7.1.1.Final ..... JBoss AS server
    └── ssh_restart_jboss.sh ..... Skript reštartujúci JBoss AS server
```