6:SHO Počítačový server

PROJEKT NA VUT FIT 2012/2013 DO IMS

Michal Lukáč (xlukac05), Jakub Sznapka (xsznap01)

1. Úvod

V této práci je řešena implementace SHO, který bude bude použit pro sestavení modelu počítačového serveru. Na základě modelu a simulačních experimentů bude ukázáno vytížení serveru a jeho jednotlivých komponent v různých konfiguracích. Model vychází z reálného serveru relaxhosting.cz. Statistiky přístupu uživatelů, které nebylo možné získat jsou doplněny informacemi ze serveru mujhost.net. Smyslem experimentu jezjistit, které komponenty se podílejí na výsledném výkonu v konkrétním případě nejvíce a vyplatilo by se je s ohledem na jejich cenu aktualizovat. Pro zpracování modelu bylo nutné získat podrobné statistiky o fungování doplněné konzultací se správcem serveru. Výsledky experimentů mohou být použity jako zdroj informací pro webhostingové servery tohoto rozsahu v případě, že je bude třeba aktualizovat.

1.1 Zůčastněné osoby a zdroje informací

Autory projektu jsou studenti FIT VUT Jakub Sznapka a Michal Lukáč.

Člověkem, který velkou měrou pomohl je Ing. Martin Dostál, správce serveru mujhost.net.Ten poskytnul obecné informace o fungování serveru a doplnil chybějící informace ke statistikám. Poskytnul také informace o konfiguraci serveru mujhost.net, ze které se vycházelo v případě nedostatečných informací.

Důležitým zdrojem informací pro sestavení modelu byly podrobné statistiky ze serveru relaxhosting.cz získáné pomocí nástroje Munin a administrátora tohoto serveru.

1.2 Validita modelu

Validita modelu byla ověřena pomocí výstupních statistik, které odpovídají realným modelům a chování realných systémů. Výstupní data byla v souladu s výsledky testů[1,2], které byly prováděny na reálných systémech. Některé principy byly také potvrzeny správci výše zmíněných serverů.

2. Rozbortématu a použitýchmetod/technologií

Systém je modelován jako systém hromadné obsluhy(SHO)[IMS, s. 139]. SHO, jak ji definoval D. G. Kendall má tři hlavní hlediska X,Y,c. Specifikace SHO má tedy tvar X/Y/c, kde X je typ stochastického procesu popisujícího příchod požadavků k obsluze. Y je zákon o rozložení délky obsluhy a c vyjadřuje počt dostupných obslužných linek. X,Y nabývají hodnot M,D,G,En,Kn,GI, které vyjadřují rozložení

příchodů v případě X a rozložení doby obsluhy v případě Y. c Je přirozeně velké číslo vyjadřující počet obslužných linek.

Model našeho počítačového serveru obsahuje 3 procesy, které simulují reálné skupiny procesů na serveru. Proces webových služeb zahrnuje příchod požadavku na server, zabrání obslužné linky a zpracování požadavku. Skupina systémových procesů zahrnuje cyklus 270 systémových procesů, které jsou podle potřeby aktivovány, jinak se nacházejí ve stavu IDLE. Skupina uživatelských procesů obsahuje proces vzniku, zpracování a zániku.

Systém obsahuje 3 linkytypu Facility a jednu linku typu Store[IMS, s. 149], které zpracovávají požadavky. Fronty typu Facility byly zvoleny u komponenty procesor, disk a databázový server, ke kterému může přistupovat pouze jeden proces. Procesor je modelován jako pole linek, což umožňuje realizovat multi-procesorový systém. Jako linka typu Store byl zvolen webový server, který takto může být realizován jako konkurentní.

Všechny procesy jsou řazeny do jediné sdílené fronty, kde čekají na uvolnění linek.Volba jedné fronty je zde zvolena, protože všechny procesy jsou si rovnocenné a nepotřebujeme uvažovat prioritu jednotlivých procesů.

Na webovém serveru byl použit systém memcached[3] v kombinaci s MySQL cachingem[4]. Tento způsob byl zvolen, abychom zjistilili, jaký vliv má na celkový výkon takto nastavený databázový server. Memcached je open-source high-performance caching multiplatformní systém, najčastěji používaný v dynamických webových stránkach s vetším počtem uživatelských a dotazů do databáze. Pomocí memcache je na rozdíl od databázové cache možno cachovat jakékoliv objekty. V praxi se tedy server nejprve dotazuje memcache, jestli obsahuje požadované data a v případě, že nejsou dostupné se teprve hledá na disku, popř. databázi. Tento systém používá například server, na kterém běží reddit[5], facebook[6]. Při implementaci jsme vycházeli z článku uveřejněném na serveru highscalability.com[4].

V systému bylo potřeba použít několik periferií. Po konzultacích jsme vybrali několik na serveru běžně používaných disků a pamětí. Parametry byly odvozeny z technických specifikací a veřejně uveřejněných testů, recenzí a veřejné opoře k předmětu IPZ[7, 8, 9].

2.1 Popis použitých postupů

Pro popis modelu jsme využili Petriho síť. Ta nám umožnňuje jednoduchým způsobem definovat veškeré operace nad modelem[IMS, s. 126]. Model je založen na systému hromadné obsluhy, který umožnňuje oblusu transakcí čekajících ve frontě. Jako implementační jazyk byl zvolen jazyk C++, jehož objektově orientovaný přístup umožňuje pohodlně vytvořit abstrakní model. V tomto modelu je tedy objekt abstrakcí/analogií realného objektu. Byla také využita simulační knihovna SIMLIB, která usnadnila práci s modelem.

2.2 Popis původu použitých metod

Model byl vytvořen jako SHO. Podklady pro vytvoření modelu byly získány ze statistik serveru relaxhosting.cz[11]. Chybějící infomrace byly dohledány na stránkách výrobců periferií, popř. ze statistik podobného serveru mujhost.net. V modelu byla využita metoda memcached, jejíž popis a chování jsme získali z oficiální stránek projektu[3].

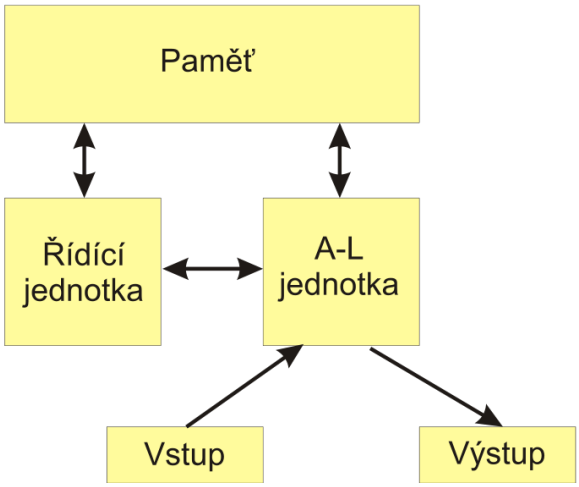
3. Koncepcemodelu

Náš model je založen na reálném systému. Model obsahuje 3 hlavní procesy, které vyjadřují webový a databázový server, dále systémové procesy a nakonec uživatelské procesy. Pro každou větev jsou samostatně generované procesy. V každé větvi uvažujeme nutnost práce s daty za účasti procesoru, disku a paměti. Ve větvi pro zpracování uživatelských požadavků na webový server je zakomponován systém memcached kombinovaný s MySQL cache systémem. To umožňuje zpracovávat požadavky za účasti jincýh komponent a při jiné rychlosti. Větev se systémovými procesy negeneruje procesy běžným způsobem, ale na začátku je vygenerováno 270procesů a ty dále nezanikají, ani nevznikají nové. Jsou pouze přepínány do IDLE módu obdobně, jako je to na reálném systému. Větev uživatelských procesů modeluje přístup uživatele na server a zpracování jeho požadavků.

Cílem modelu je podat spolehlivé informace o serveru a jeho výkonu v různých konfiguracích a nastavení jeho komponent. Pro naše potřeby tedy není potřeba vytvářet naprosto přesný model. Z serveru relaxhosting.cz se nám podařilo získat detailní statistiky o provozu tohoto serveru za poslední rok. Z těch jsme byli schopni vyčíst relevantní data. Bohužel se nám nepodařilo získat přesný seznam komponent, ze kterých se tento server skládá. Nicméně po konzultaci s Ing. Martinem Dostálem jsme použili seznam komponent obsažených na podobném serveru mujhost.net. Z tohoto serveru jsme také použili statistiky přístupu k webovému a databázovému serveru. Některé veličiny nebylo možné přesně změřit na serveru, ale došlo k jejich odhadu. Jedná se například o přenosové rychlosti a propustnost disků a operačních pamětí. V případě zmíněných periferií bylo využito dokumentací na stránkách výrobců, popř. nezávislých testů a recenzí[7, 8, 9]. K odhadům došlo také u měření počtu a době běhu uživatelských a systémových procesů. Některé data jsme byli schopni vyčíst ze statistiky, nicméně nebyly kompletní. K potřebnému odhadu jsme došli na základě konzultace, nebo hodnot naměřených na jiném, velmi podobném systému. Při tomto procesu jsme se snažili zachovat celkovou validitu modelu.Ta by neměla být ohrozena, neboť vycházíme z již známých a prozkoumaných faktů. Jsme tedy schopni ověřit, zda-li naše odhadnuté, nebo vypočítané hodnoty odpovídají reálným hodnotám. V našem případě jsme měli taktéž možnost, některé hodnoty naměřit na vlastních stanicích (například rychlost disků, pamětí).

3.1 Vyjádření modelu

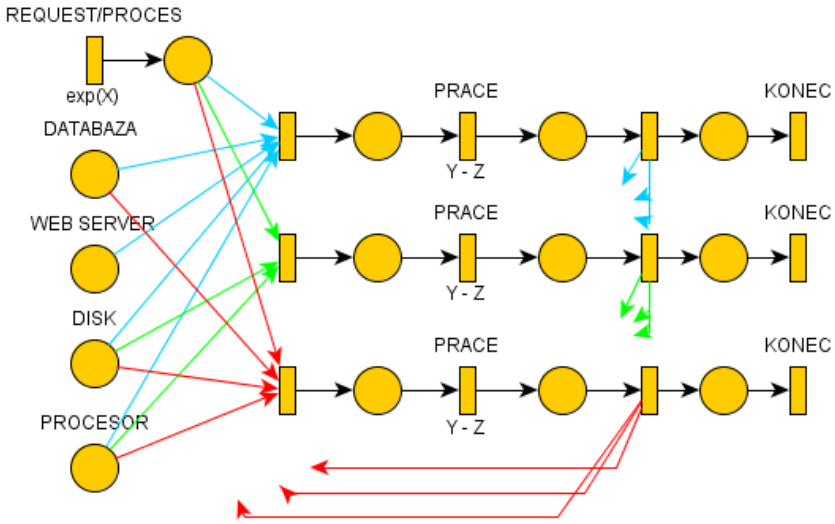
Konceptuální model byl sestaven na základě fungování klasického PC. Navíc má pouze nainstalován webový a databázový server. Můžeme tedy vycházet z klasické Von Neumannovy architektury, která je znázorněna na obrázku č.1.



obrázek č.1, Von Neumannova arhitektura

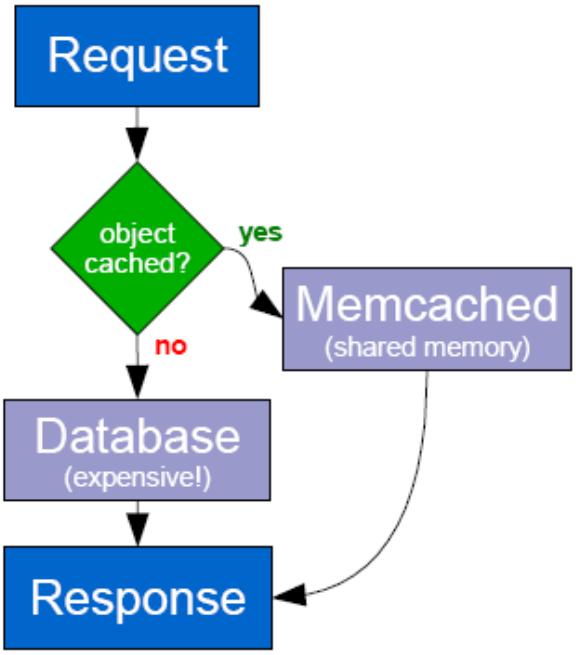
3.2 Forma konceptuálního modelu

Konceptuální model je vyjádřenPetriho sítí na obrázku č.2. Kompletní síť, že které jsme vycházeli je k nalezení v příloze[12].Síť zjednodušeně zachycuje všechny procesy a to, jak pracují s linkou.



Obrázek č.2, zjednodušený model systému

Obrázek č.3 zachycuje fungování metody memcache. Ta umožňuje cachování a tedy i rychlý přístup ke všem typům objektů. Na obrázku je zobrazeno, jak pracuje s normální MySQL cachí a tím pádem umožňuje rychlejší zpracování požadavků na straně databázového serveru, než v klasickém případě, kdy je využita jen SQL cache.



Obrázek č.3, Memcache[11]

4. Architekturasimulačníhomodelu/simulátoru

Simulační model pracuje jako konzolová aplikace. Model je implementován v jazyce C++ s využitím knihovny SIMLIB a je modelován podle abstraktního modelu vyjádřeného pomocí Petriho sítě. O simulačním a abstraktním modelu můžeme říct, že jsou izomorfními systémy[IMS, s. 27], neboť jsou ve vztahu 1:1. Chování simulace může být ovlivněno několika parametry.

-d	INT	Výběr disku(0=HDD 7200RPM,1=HDD 15000RPM, 2=SSD)
-p	INT	Počet procesorů
-s	INT	Počet systémových procesů
-ho	INT	Celkový simulační čas v hodinách
-m	INT	Kolik procent dotazů je realizováno pomocí memcache

4.1 Mapováníabstraktního modelu do simulačního modelu

Jak je vidět na obrázku č.2. Model obsahuje 3 procesy.

První proces odpovídá požadavku na webový server. Ten je realizován pomocí třídy `WebRequest`. Procesy tohoto typu jsou generovány pomocí expenoneciálního rozložení a třídy události `GeneratorWeb`. Podstatou této větve je zpracování uživatelského požadavku na webový server. Ten pokud jsou mu přiděleny všechny zdroje zpracuje požadavek v různých rychlostech a za pomocí různých komponent.

Druhý proces simuluje systémové procesy skrze třídu `SystemProcess`. Na počátku je vytvořeno 270procesů, které se nachází buďto v aktivním stavu, nebo ve stavu IDLE a podle toho je s nimi nakládáno.

Uživatelské procesy simuluje třetí proces, kterému odpovídá třída `UserProcess`. Procesy jsou zde generovány exponenciálním rozložením v třídě udalásotí `GeneratorUser`. Proces v případě, že dostane potřebné zdroje zpracovává uživatelské procesy a to buďto s využitím databázového serveru, nebo pracuje pouze s daty na disku.

Výstupem jsou statistiky, které jsou potřebné pro realizaci experimentů.

5. Podstatasimulačníchexperimentů a jejichprůběh

Experimenty se budeme snažit najít nejvýhodnější způsob, jak aktualizovat zkoumaný server. Nakoupení ruzných komponent a testování různých konfigurací by bylo jak časově, tak i finančně náročné. Vytvoření simulačního modelu se proto jeví jako nejvhodnější způsob. Ve výsledku se tedy budeme snažit sestavit skóre pro několik konfigurací. U těch bude bráno v potaz několik klíčových ukazatelů výkonu a také odhad ceny výsledné konfigurace. Pro výpočet orientační ceny bylo využit cen na eshopu alza.cz. Cena a typ jednotlivých komponent se nachází v příloze [13].

5.1 Postup experimentování

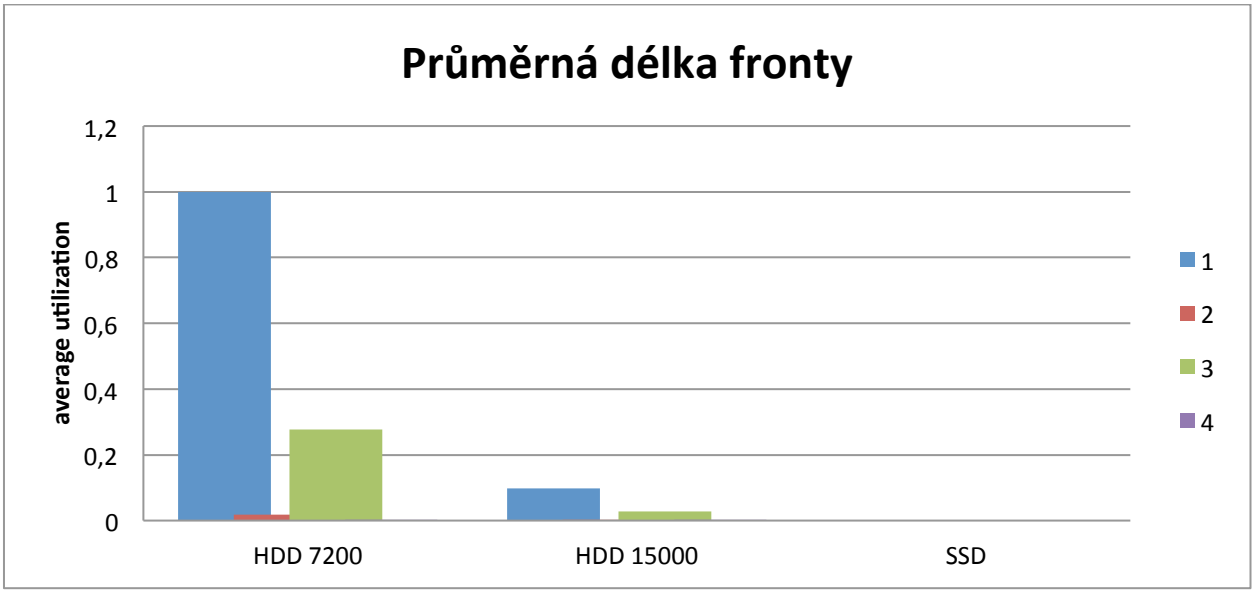
Bylo vytvořeno několik experimentálních konfigurací serveru (viz. tabulka č.1). V grafu u každého experimentu je vždy zaznačeno číslo konfigurace 1-4, které odpovídá konfiguracím 1-4,5-8,9-12 vždy pouze s jiným diskem. Na každé konfiguraci bude na 24 hodin spuštěn provoz a budeme měřit některé statistiky, které nám mohou sloužit jako ukazatele výkonu. Pro každou konfiguraci spustíme experiment 3krát a spočítáme průměrnou hodnotu. Každou z těchto specifikací převedeme do intervalu 0-1 podle nejnižší a nejvyšší naměřené hodnoty. Pro každou konfiguraci tedy budeme mít hodnotu 0-1 vyjadřující skóre. Pro každou konfiguraci poté spočteme celkové skóre a sestavíme finální skóre podle všech statistik. U každého grafu platí, že nižší skóre je lepší.

No.	Typ disku	počet CPU	% dotazůpřesMemcache	Cena [CZK]
1.	HDD 7200 rpm – 123 MB/s	4	30	41160
8.	HDD 7200 rpm – 123 MB/s	8	30	65156
3.	HDD 7200 rpm – 123 MB/s	4	80	64340
4.	HDD 7200 rpm – 123 MB/s	8	80	88336
5.	HDD 15000 rpm – 172 MB/s	4	30	79561
6.	HDD 15000 rpm – 172 MB/s	8	30	103557
7.	HDD 15000 rpm – 172 MB/s	4	80	102741
8.	HDD 15000 rpm – 172 MB/s	8	80	126737
9.	SSD – 497 MB/s	4	30	180526
10.	SSD – 497 MB/s	8	30	204522
11.	SSD – 497 MB/s	4	80	203706
12.	SSD – 497 MB/s	8	80	227702

tabulka č.1, konfigurace serveru

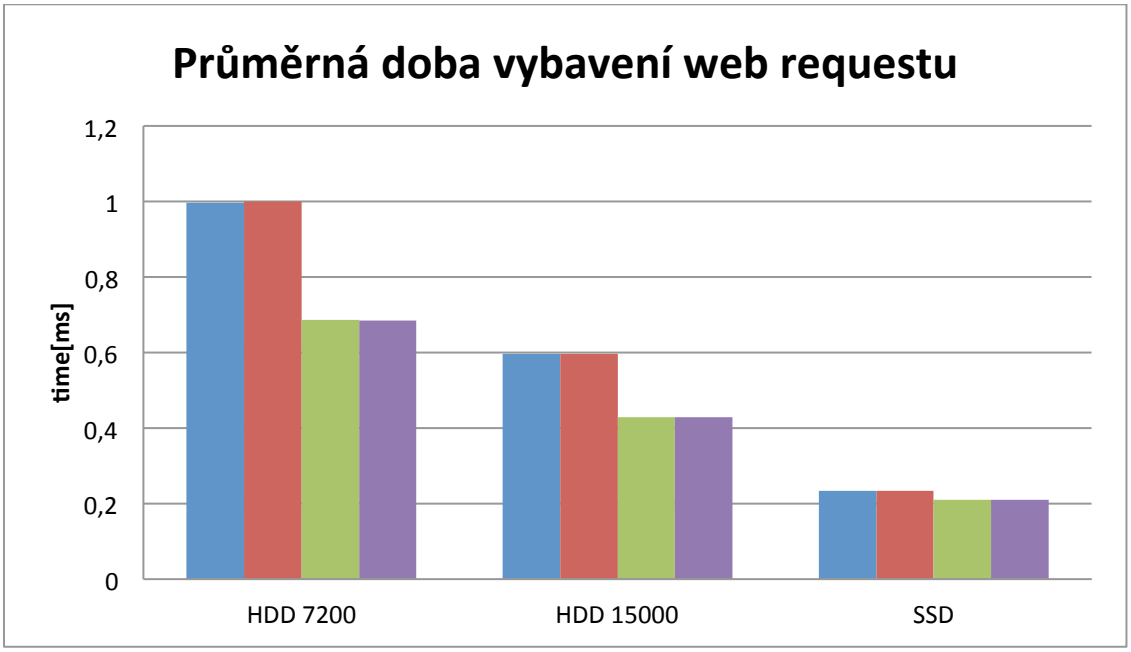
5.2 Prováděné experimenty

Experiment č.1 byl zaměřen na průměrnou délku fronty. Ta nám může být ukazatelem, jak moc je celkově systém vytížen a jak moc tedy procesy čekají na přidělení procesoru. Z experimentu vyplývá, že navýšením počtu procesorů dosáhneme lepších výsledku. Stejně tak navýšením rychlosti disků zkrátíme průměrnou délku fronty. Výsledky jsou shrnuty v grafu č.1.



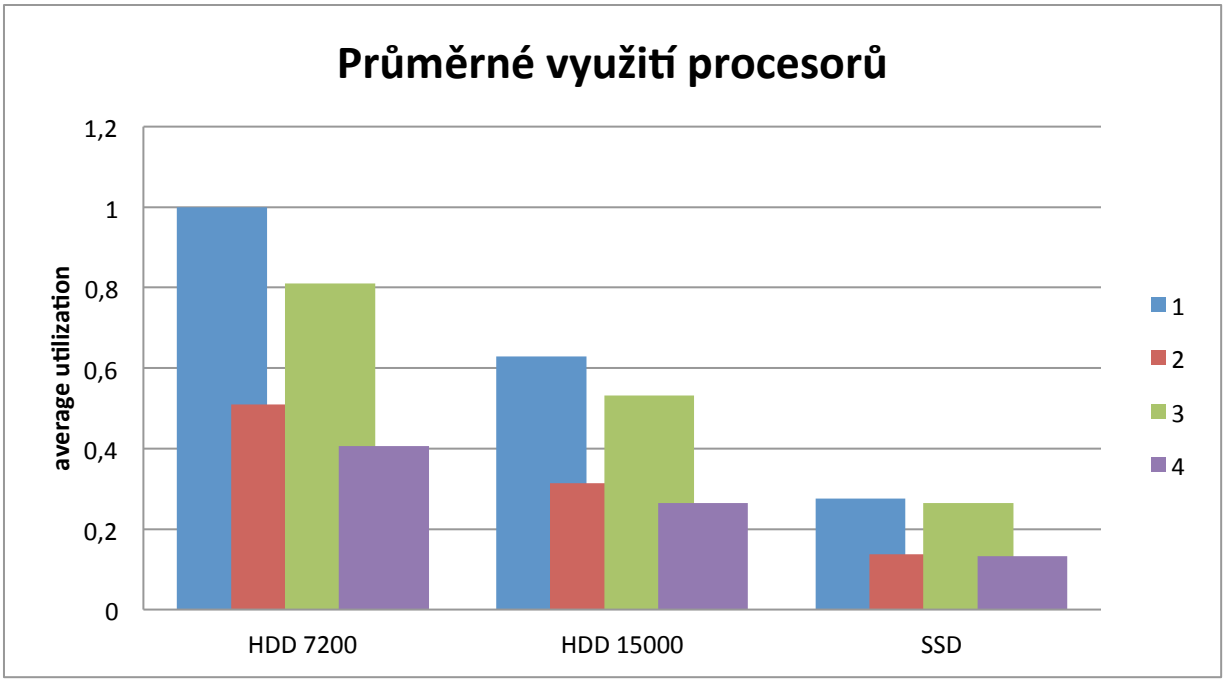
graf č.1, průměrná délka fronty

Experiment č.2 byl zaměřen na průměrnou dobu obsluhy web requestu. Tímto experimentem jsme se snažili zjistit, jak metoda memcaching ovlivňuje rychlost zpracování požadavků. Z výsledku experimentů dokážame vyčíst, že rychlost se s navýšením velikosti memcache zvyšuje, ale není tak výrazná, jak jsme očekávali. U rychlejších disků rozdíl mizí. Výsledek je nejspíše ovlivněn i tím, že na server nepřícházelo tolik požadavků. Pokud bychom generovali více požadavků, rozdíl mezi výsledky s různě velkou memcachí by byl razantnější. Výsledky jsou shrnuty v grafu č.2.



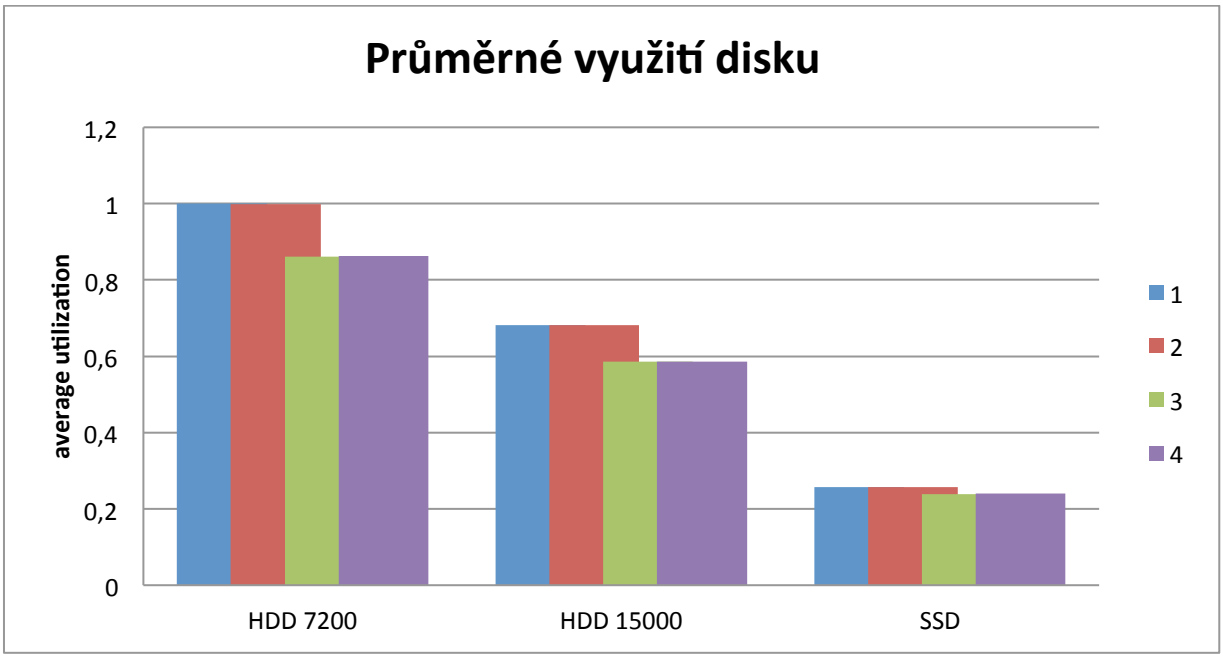
graf č.2, průměrná doba vybavení webrequestu

Experiment č.3 bylzaměřen na průměrné vytížení procesorů. Jelikož jsme používali pouze konfigurace s multi – procesorovým řešením, je průměrné vytížení aritmetickým průměrem vytížení všech procesorů. Ve výsledcích můžeme vidět, že i když se požadavky zpracovávají rychleji, tak pořád většina procesů čeká na disk. Můžeme také vidět, že memchaching nemá velký vliv na vytížení procesoru. Zejména potom u rychlejších disků. Výsledky jsou shrnuty v grafu č.3.

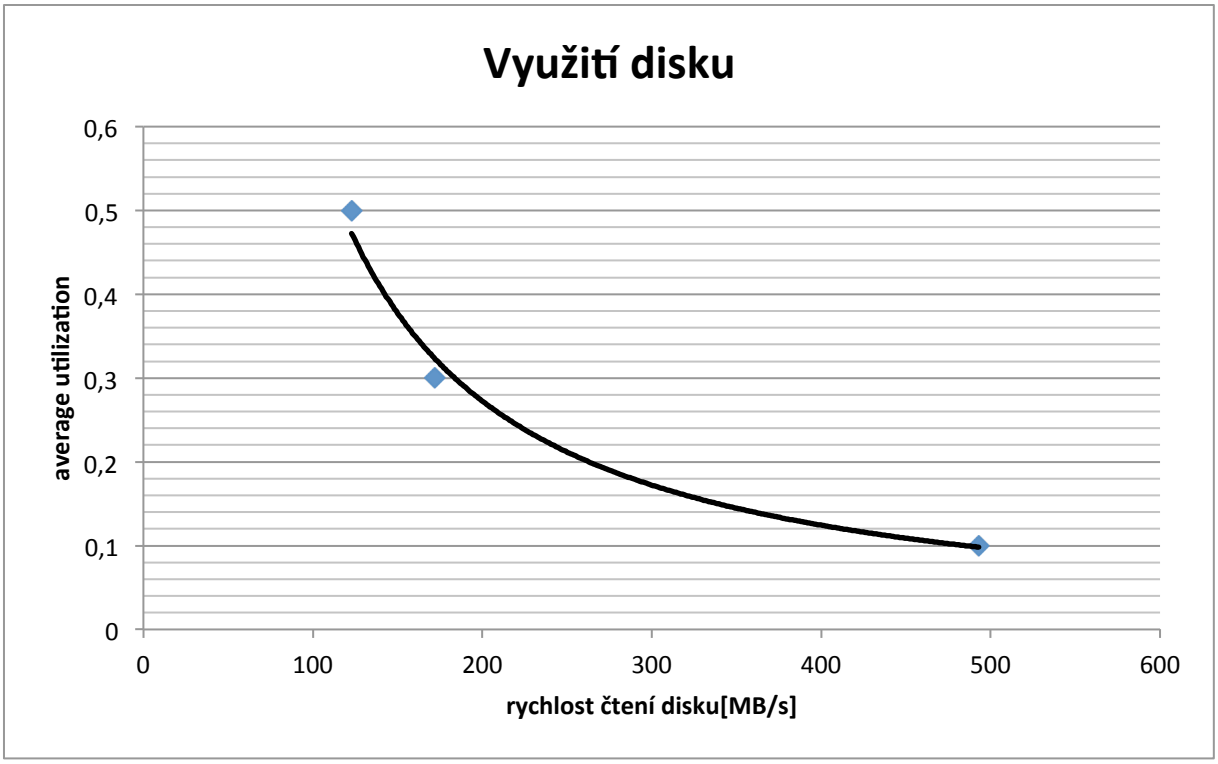


graf č.3, průměrné využití procesoru

Experiment č.4byl zaměřen na průměrné vytížení disku.Z tohoto experimentu můžeme vidět, jak ostatní komponenty ovlivňují zatížení disku. Podle všech předpokladů by k největšímu zpomalení systému mělo docházet právě u disků. Ve výsledcích můžeme vidět, že ostatní komponenty průměrné vytížení neovlivňují nějakým zásadním způsobem. Z toho můžeme usoudit, že právě disk je nejvytěžovanějším zařízením a je tedy i největší brzdou systému. Výsledky jsou shrnuty v grafu č.4.V grafu č.5 můžeme vidět závislost rychlosti disku na průměrném vytížení. V grafu můžeme vidět, že vytížení je tím menší, čím je větší rychlost disku.

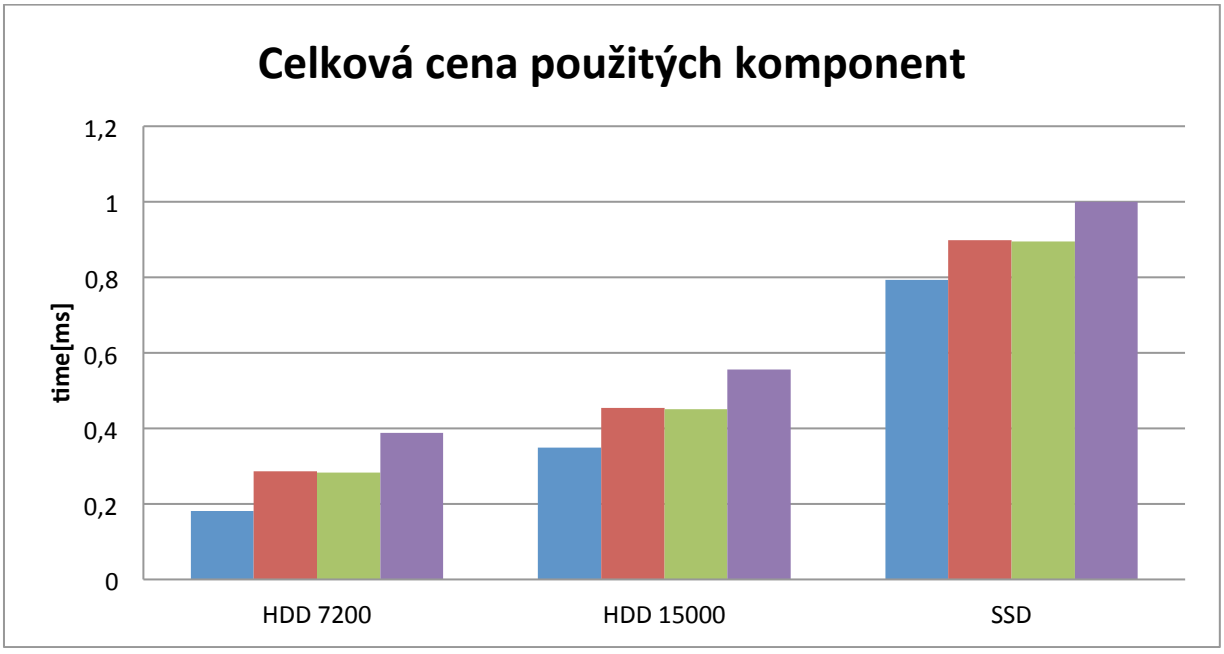


graf č.4, průměrné využití disku



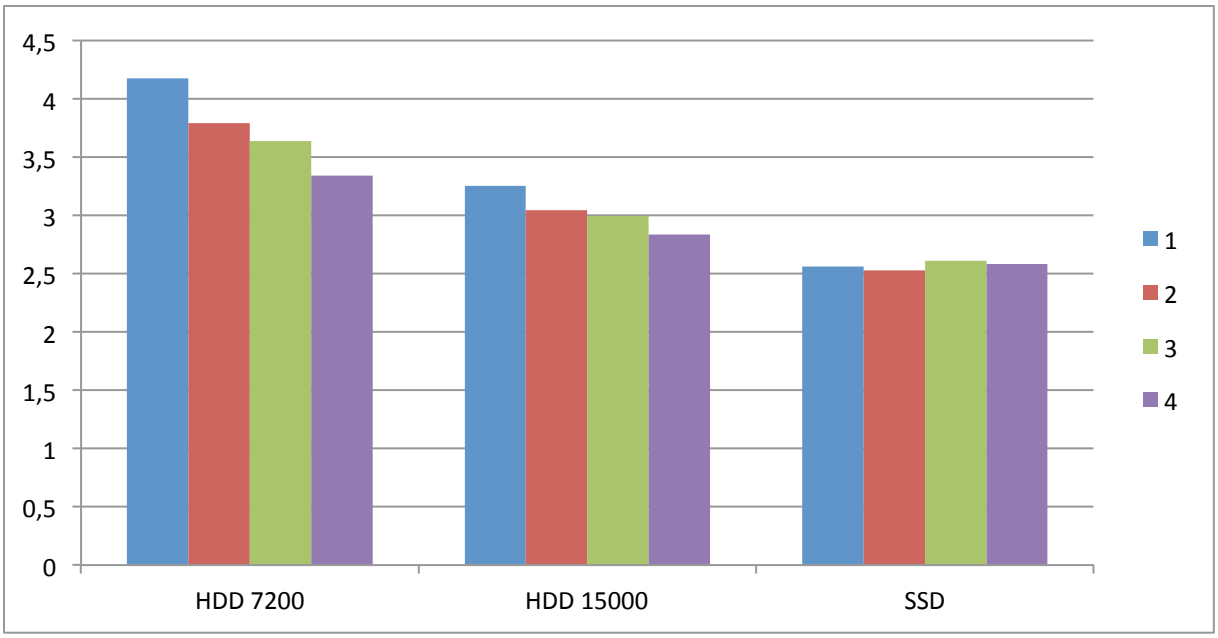
graf č.5, průměrné využití disku ve vztahu s rychlostí disku

Experiment č.5 bylzaměřen na celkovou orientační cenu použitých komponent. Tato data nebyla naměřena pomocí simulačního modelu, ale pomohou nám objektivně systém zhodnotit nejen podle hrubého výpočetního výkonu. Výsledky jsou shrnuty v grafu č.6.



graf č.6, celková cena komponent použitých v konfiguraci

Záverečným experimentembylosesumování skóre z předešlých experimentů. Konfigurace s nějnižším celkovým hodnocením dosáhla celkového nejlepšího výsledku. Je to tedy sestava s SSD diskem 8x CPU a 30% podílem memcache hitů. Výsledky v grafu č.7.



graf č.6, celková cena komponent použitých v konfiguraci

5.3 Poznatky získané z experimentů

Bylo provedeno celkem 5 experimentů, z toho 4 na simulačním modelu. V průběhu experimentování byly vhodně opraveny některé parametry serveru, abychom docílili co možná nejlepších výsledků. Naměřené hodnoty byly změřeny na výsledných konfiguracích uvedených v tabulce č.1. Z experimentů můžeme odvodit chování systémů s obdobnými parametry. Dalšími experimenty bychom nezískali již další významné a směrodatné data a proto není provádění dalších experimentů potřeba.

Závěr

Z výsledků experimentů vyplývá, že jako nejvýhodnější konfigurace se jeví konfigurace č.10, která používá 8 procesorů, SSD disk a 30% memcache hit. Výsledky experimentů můžeme považovat za validní, neboť odpovídají výsledkům naměřených při jiných typech experimentů. Můžeme tedy říci, že v současnosti největším problémem serverů jsou pevné disky. Z experimentů také vyplývá, že řešením můžou být například rychlé SSD disky u kterých je ale nutné počítat s vysokými pořizovacími náklady. Dále můžeme vypozorovat, že navyšovaní počtu procesorů nezískáme až takový výkon, jak by se mohlo zdát. Tento jev je dán zejména pomalou obsluhou dalších periferií. V našich experimentech jsme také zjistili, že metoda memcachingu neovlivňuje až tak zásadním způsobem, jak jsme očekávali. Tento jev je nejspíše způsoben malým provozem na straně webového server. V případě většího zatížení bychom mohli vidět větší rozdíly.

Literatura a zdroje

[1] Traditional Hard Drives VS Solid-State Drives. [online]. [cit. 2012-12-07]. Dostupné z: http://www.vaultnetworks.com/dedicated-servers/hard-disk-vs-solid-state.html

[2] Increasing productivity by selecting a multi-processor workstation. [online]. [cit. 2012-12-07]. Dostupné z: http://h20331.www2.hp.com/Hpsub/downloads/Multi-Processor_WhitePaper_090611.pdf

[3] Memcached. [online]. [cit. 2012-12-07]. Dostupné z: http://memcached.org

[4] A Bunch Of Great Strategies For Using Memcached And MySQL Better Together. [online]. [cit. 2012-12-07]. Dostupné z: http://highscalability.com/bunch-great-strategies-using-memcached-and-mysql-better-together

[5] Reddit. [online]. [cit. 2012-12-07]. Dostupné z: http://www.reddit.com

[6] Facebook. [online]. [cit. 2012-12-07]. Dostupné z: https://www.facebook.com

[7] Barracuda 7200.12 Serial ATA. [online]. [cit. 2012-12-07]. Dostupné z: http://www.seagate.com/staticfiles/support/disc/manuals/desktop/Barracuda%207200.12/100529369b.pdf

[8] Memory speeds and compatibility. [online]. [cit. 2012-12-07]. Dostupné z: http://www.crucial.com/support/memory_speeds.aspx

[9] IPZ HDD. [online]. [cit. 2012-12-07]. Dostupné z: https://www.fit.vutbr.cz/study/courses/IPZ/public/texty/rok2010/hdd10/hdd00.pdf

[10] Using Memcached as an object cache. [online]. [cit. 2012-12-07]. Dostupné z: http://cerberusweb.com/book/latest/admin_guide/performance/memcache.html

Příloha

[11] http://goo.gl/zpajl

[12] http://goo.gl/s0Ti5

[13] http://goo.gl/wtJ1x