VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

UNIVERZÁLNÍ REZERVAČNÍ SYSTÉM V CLOUDU

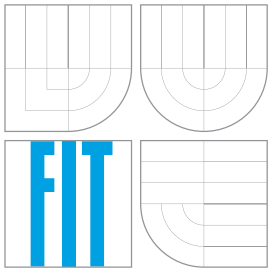
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN BULÍN

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

UNIVERZÁLNÍ REZERVAČNÍ SYSTÉM V CLOUDU

UNIVERSAL RESERVATION SYSTEM IN CLOUD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN BULÍN

VEDOUCÍ PRÁCE
SUPERVISOR

RNDr. PETR ŠKODA

BRNO 2015

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací univerzálního rezervačního systému v cloudu. Velkou část této práce tvoří přehled poskytovatelů cloudu a již existujících rezervačních systémů. Dále se práce zabývá návrhem a implementací rezervačního systému. Na tomto systému si ukážeme výhody cloudu. V závěru práce se zabývám škálovatelností cloudu a testováním vytvořeného systému v cloudu Amazon Web Services.

Abstract

This bachelor thesis describes design and implementation of universal reservation system in cloud. Main part of this thesis includes review of cloud providers and existing reservation systems. Furthermore this academic paper deals with proposal and implementation of the reservation system. This system can show us the benefit of the cloud. In the conclusion I deal with scalability of cloud and testing created system in Amazon Web Services cloud.

Klíčová slova

Cloud, Rezervační systém, Amazon Web Services, Elastic Compute Cloud, Relational Database Service, PHP, škálovatelnost

Keywords

Cloud, Reservation system, Amazon Web Services, Elastic Compute Cloud, Relational Database Service, PHP, scalability

Citace

Martin Bulín: Univerzální rezervační systém v Cloudu, bakalářská práce, Brno, FIT VUT v Brně, 2015

Univerzální rezervační systém v Cloudu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana RNDr. Petra Škody. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Bulín
19. května 2015

Poděkování

Děkuji vedoucímu práce RNDr. Petru Škodovi za trpělivost při vedení, konzultace a všechny přínosné rady, které mi pomohly při vypracování této práce.

Obsah

Kapitola 1

Úvod

S rozvojem počítačů a internetu dochází k velkému rozvoji informačních technologií. V posledních letech se stále častěji vyskytuje pojem cloud computing.

Cloud computing je všude okolo nás. Tento pojem se objevuje v technických časopisech, reklamách nebo na webech z oboru IT. Cloud computing jednoznačně a nezadržitelně míří k dominantnímu postavení ve světě IT a jeho potenciál je nezpochybnitelný. Důvody pro nasazení cloudu jsou lákavé. Můžeme se zbavit problémů s fyzickým hardwarem a infrastrukturou, čímž eliminujeme investiční výdaje za výpočetní kapacitu. Hostováním v cloudu můžeme i urychlit nasazení nového produktu na trh, kdy nám odpadnou investice za nákup, konfiguraci a správu hardwaru. Proto se cloud hodí pro nastartování začínajících projektů.

Cílem této bakalářské práce je představit pojem cloud computing, charakterizovat jej a shrnout jeho výhody a nevýhody. Poté navrhnout a implementovat univerzální rezervační systém v cloudu, který bude využívat flexibility a škálovatelnosti cloudu.

Ve druhé kapitole této práce se seznámíme s pojmem cloud computing a provedeme jeho charakteristiku. Představíme si historii cloudu a dále shrneme jeho výhody i nevýhody. Kapitola 3 představí poskytovatele cloudových služeb a shrne jejich vlastnosti. Kapitola 4 porovnává vlastnosti univerzálních rezervačních systémů a kapitola 5 představuje analýzu a návrh našeho vlastního rezervačního systému. Kapitola 6 obsahuje podrobný návod jak začít s EC2 a popíšeme si zde implementaci systému v cloudu. V kapitole 7 se pokusíme dokázat škálovatelnost našeho systému v cloudu. V poslední kapitole zhodnotíme dosavadní výsledky a škálovatelnost našeho řešení a dáme možnost návrhu dalšího rozšíření.

Kapitola 2

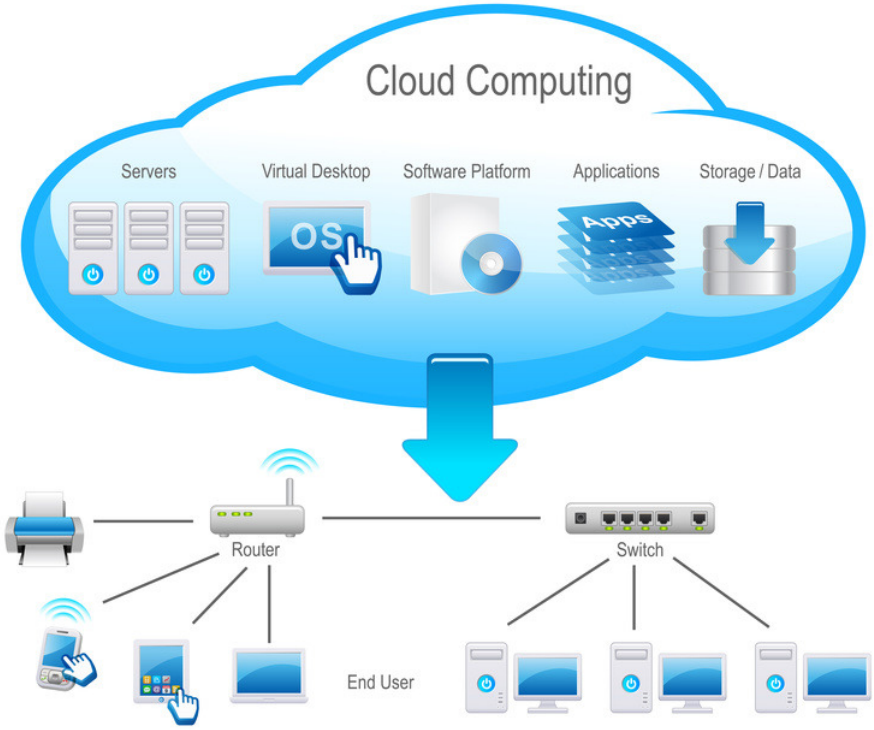
Cloud Computing

V této kapitole si představíme pojem Cloud computing.

2.1 Co je to Cloud?

Termín Cloud computing získal svůj název jako metafora Internetu. Cloud computing je sdílení hardwarových i softwarových prostředků pomocí sítě. Internet se obvykle v síťových schématech znázorňuje jako oblak (cloud). Ikona oblaku reprezentuje všechny komponenty internetu, díky kterým síť funguje. Obvykle také představuje oblast diagramu nebo řešení, za kterou zodpovídá někdo další, takže není potřeba jej konkretizovat. Tento princip se nejspíš nejlépe hodí pro popis koncepce cloud computingu. Cloud computing slibuje úsporu provozních a investičních nákladů. Ještě důležitější je, že oddělením IT umožňuje, aby se místo udržování datových center v chodu soustředila na strategické projekty[2].

Cloud computing představuje nastupující trend, který je podobně jako například WEB 2.0 založen na již existujících a ověřených technologiích. Měl by zpřístupňovat každý element IT infrastruktury jako službu na vyžádání: operační systémy, aplikace, uložiště, servery, zařízení a správu obchodních procesů. Cloud computing je vyvrcholením trendu využívání aplikací bez toho, abyste museli mít cokoliv nainstalované na svém počítači[18]. Obrázek 2.1 znázorňuje základní schéma, jak vypadá a co obsahuje Cloud computing.



Obrázek 2.1: Základní schéma – jak vypadá a co obsahuje Cloud computing [1].

2.2 Grid computing

Pojem grid computing se často zaměňuje s pojmem cloud computing. Grid computing je předchůdce cloud computingu. Grid computing můžeme definovat jako technologii využívání prostředků z více počítačů v síti, ve stejném čase a na stejném problému. Tato technologie obvykle slouží k řešení vědeckých nebo technických problémům. Klientovi, který se chtěl podílet na řešení problému, stačilo nainstalovat si příslušný program pro zpracování údajů a koordinaci zpracování. Klient poskytuje nevyužité cykly procesoru na svém počítači. Co mají koncepce grid computingu a cloud computingu společné?

- Velký projekt u grid computingu je rozložen mezi více počítačů, tak aby se lépe využily jejich prostředky.
- U cloud computingu je tomu naopak, dovoluje souběžné fungování více menších aplikací.[2]

2.3 Historie

Historie Cloud computingu se začala psát přibližně před 50. lety. Za zakladatele cloud computingu je považován počítačový vědec John McCarthy. Ten jako první přišel na myšlenku

[1] Z internetové stránky http://www.uc-solutions.net/index.php/solutions/cloud-computing.

sdílení počítačových technologií, ve stejné logice jako sdílení elektrické energie. Elektrickou energii potřebuje každá domácnost, avšak málokterá domácnost si kvůli tomu postaví vlastní elektrárnu. Nejčastější je model, kdy jednu elektrárnu sdílí více domácností. V dnešním světě si můžeme za elektrárnu představit datové centrum poskytovatele cloud computingu, elektrickou síť co by internet a domácí spotřebič, jako počítač. Navíc stejně jako u elektřiny platí zákazník, pouze přesnou hodnotu spotřebované energie.

Klíčovou roli ve vývoji cloud computingu sehrála společnost Amazon. Po modernizaci svých serverů totiž společnost využívala pouhých cca 10 % jejich výkonu a tak se rozhodla nabídnout nevyužitou kapacitu externím zákazníkům. Vlastní cloudová služba nazvaná Amazon Web Service byla spuštěna v roce 2006. Vznikla tak první komerční služba cloud computingu. O rok později se připojily společnosti jako Google, IBM a řada univerzit začala pracovat na vědeckých nebo komerčních programech založených na Cloud Computingu. Od roku 2009 je cloud computing vnímán jako klíčová budoucí technologie.[22]

2.4 Komponenty cloudu

V jednoduchém topologickém smyslu je řešení cloud computingu tvořeno několika prvky. Jedná se o klienty, datová centra a distribuované servery. Tyto komponenty představují tři součástí řešení cloud computingu. Každý prvek má svůj účel a hraje při poskytování funkční aplikace založené na cloudu nezastupitelnou roli.

- Klienti – Klienti jsou zařízení, se kterými koncoví uživatelé pracují při správě svých dat v cloudu. Klienty řadíme do několika kategorií:
  - Mobilní – Do této kategorie můžeme zařadit mobilní telefon.
  - Tencí – Patří sem klienti, kteří nemají interní pevné disky. Veškeré operace zajišťuje server a klient pouze zobrazuje informace.
  - Tlustí – Do této kategorie patří běžné počítače.
- Datová centra – Datové centrum je skupina serverů, které hostují různé aplikace.
- Distribuované servery – Servery jsou často geograficky rozptýleny. Z hlediska uživatele pracují tak, jako kdyby byly zapojeny jeden vedle druhého.[2]

2.5 Výhody

Cloud computing má bezesporu spousty výhod:

- Rychlé nasazení – cloud přináší koncepci centralizované platformy, která je kdykoliv připravená k použití, stačí si službu zřídit.
- Vysoká flexibilita – přístupové zdroje mají virtuální charakter, výsledný potenciál cloudu není limitován výkonností a kapacitou lokálních nebo vzdálených počítačů.
- Sdílení zdrojů – sdílení hardwarových prostředků umožňuje lépe distribuovat výkon mezi jednotlivé uživatele.[18]
- Eliminace nákladů na správu a údržbu – Je nutné vzít v úvahu hardware, který nemusíme kupovat. Aplikace hostuje jiná společnost, díky tomu mohou klesnout investiční náklady. Vzhledem k tomu, že naše aplikace hostuje někdo jiný, nepotřebujeme nakupovat servery ani platit za jejich napájení a chlazení.[2]

- Úspory v oblasti spotřeby energie – Lepší využití elektrické energie. Nemusíme platit za napájení a chlazení serverů.
- Jednoduchost – Klient nemusí znát princip fungování cloudu a nemusí znát princip funkce hardwaru a softwaru.
- Online řešení – Znamená pro nás, že máme možnost se připojit k poskytované službě téměř odkudkoliv a z jakéhokoli zařízení (notebooku, tabletu, mobilu), pokud máme internetovou přípojku a zařízení umožňující přístup na internet. Již nejsme omezeni firemní infrastrukturou.[27]
- Škálovatelnost, rozšiřitelnost – máme možnost si kdykoliv alokovat dodatečnou úložnou kapacitu nebo výpočetní zdroje podle aktuální potřeby a v potřebném objemu. Později se úložná kapacita i výpočetní zdroje mohou flexibilně vrátit. Cloud nám v případě potřeby umožní dokoupení úložné kapacity. Víme, že nejvíce bývá server zatížen pouze v odpoledních hodinách, proto by bylo finančně náročné využívat vlastní softwarovou i hardwarovou infrastrukturu. Cloud můžeme využít i v malých podnicích, kdy se může, z důvodů velkého zatížení serveru, dokoupit úložná kapacita.

2.6 Nevýhody

Jako všechny IT technologie i cloud má svoje nevýhody. Podíváme se na některé z nich:

- Výpadek spojení – Mohou nastat situace kdy dojde k výpadku služeb internetu. V dnešní době se výpadky internetu vyskytují pouze zřídka, ale když se vyskytnou, tak nám znemožní přístup a práci s aplikacemi v cloudu. Také může nastat situace, kdy nedojde k problémům na naší straně ale na straně poskytovatele, na kterém chceme pracovat.
- Závislost na poskytovateli – Uživatel je závislý primárně na jednom poskytovateli cloudového řešení. Musíme počítat s tím, že poskytovatel může zdražit služby, rozhodovat jaký software používat a nebo dokonce přestat komunikovat. Přenositelnost dat na jiného dodavatele je potom velice problematická. Může dojít i k tomu, že společnost provozující cloud, může zkrachovat. Proto je dobré vybrat si společnosti jako Google či Microsoft, kde je pravděpodobnost zrušení poskytovaných služeb téměř nulová.
- Uživatelské rozhraní – Cloudové řešení většinou poskytuje méně funkcí v porovnáni s desktopovým. Často jediným způsobem, jak pracovat s cloudovou aplikací, je internetový prohlížeč. Většinou je nutno disponovat nejnovější verzí prohlížeče z důvodu podpory nejnovějších technologií. V různých podnicích není ale třeba možné z důvodu zachování kompatibility s jinými aplikacemi instalovat jiné verze prohlížečů, či dělat jejich upgrade[16]
- Zabezpečení a poškození dat – Používání cloudu přes internet vyvolává otázky jestli je cloud bezpečný. Data se kterými uživatel pracuje, jsou fyzicky uloženy na cizích serverech poskytovatele cloudu. Tím pádem je nemáme pod kontolou. Poskytovatel se ovšem snaží data co nejlépe zabezpečit, pokud by totiž byla data uživatelů zneužita, poskytovatel by byl v očích uživatelů nespolehlivý. V případě nespolehlivosti poskytovatele by se mohlo stát, že by uživatelé poskytovatele opustili.

- Legislativa – Tyto problémy vyplývají z toho, že poskytovatel služby sídlí v různých zemích. V těchto zemích mohou být různé právní normy. Například společnost, která sídlí v USA je povinna podstoupit data klienta vládě. Tento problém by se dal vyřešit tím, že bychom přešli na evropského poskytovatele. Společnosti Google a Microsoft mají v podmínkách užívání napsáno: *Používáním služeb Google berete na vědomí a souhlasíte s tím, že společnost Google je oprávněna přistupovat k údajům o vašem účtu a k libovolnému obsahu, který s daným účtem souvisí, uchovávat je a zveřejňovat, vyžádá-li si to zákon, nebo v dobré víře, že takovýto přístup, uchování nebo zveřejnění je nezbytné pro: (a) splnění příslušného zákona, nařízení, zákonného procesu či vymahatelného vládního požadavku, (b) prosazení podmínek, včetně vyšetřování jejich možného porušení, (c) odhalení, prevenci nebo jiný způsobu řešení podvodů, bezpečnostních nebo technických problémů (mimo jiné filtrování spamu), nebo (d) ochranu proti hrozící újmě na právech, majetku nebo bezpečnosti společnosti Google, jejích uživatelů nebo veřejnosti, jak vyžaduje nebo povoluje zákon.*[12]Tyto podmínky mohou některé uživatele od cloudu odradit.

2.7 Modely nasazení cloud computingu

Model nasazení nám říká, jak je cloud poskytován. Máme několik modelů nasazení:

- Veřejný cloud – Cloud computing, který poskytovatel nabízí z vlastních sdílených prostředků jako službu zákazníkům z řad veřejnosti. Charakteristikou veřejného cloudu je schopnost poskytovat prostředky na vyžádání, elasticky a samoobslužně, síťový přístup a také měřitelnost spotřebované služby v rámci sdíleného fondu prostředků. Záleží jen na rozhodnutí daného poskytovatele, jaké prostředky zpřístupní a kterému zákazníkovi, proto může být služba zabezpečená i nezabezpečená.

- Komunitní cloud – Prostředí pro cloud computing, které vzniká sdružením prostředků vlastněných určitou skupinou členů (tzv. komunitou) a poskytuje tyto prostředky zpět formou služby stejné skupině členů.[7]

- Privátní cloud – Privátní cloud je provozovaný pro firmu nebo organizaci. Službu poskytuje firma sama sobě, přesněji IT oddělení ji poskytuje pro ostatní organizační složky firmy, případně ji pro konkrétní firmu nebo organizaci poskytuje třetí strana.

- Hybridní cloud – Hybridní cloud je kombinací propojených veřejných a privátních cloudů. Vůči okolí však vystupuje jako jeden cloud. [18]

2.8 Modely poskytování cloud computingu

Jednotlivé modely se liší v tom, jaké služby poskytují. Model se zabývá tím, co je v rámci služby nabízeno, obvykle software nebo hardware či jejich kombinace. Modely cloudu jsou Ifrastruktura jako služba (IaaS), Platforma jako služba (PaaS) a Software jako služba (SaaS). Cloud je nová technologie, která se neustále mění, a proto je potřeba brát v úvahu časté inovace modelů cloudu. Častá inovace může mít za následek i to, že rozdíly mezi IaaS, PaaS a SaaS jsou často nejasné. Nemusí být vždy zřejmé, zda jde o PaaS nebo IaaS. Rozdíl mezi nimi není zdaleka tak důležitý, jako pochopení konceptů těchto technologií a znalost, jak je správně využívat.[19]

2.8.1 Infrastruktura jako služba (IaaS)

Infrastruktura jako služba (IaaS) není nový koncept. Lidé sdíleli hardwarové prostředky v datacentrech již od začátku jejich vzniku. Na IaaS je výjimečné to, že se zákazníci zbaví starostí a investičních nákladů souvisejících s nákupem a provozem serverů a uložišť. IaaS poskytuje mechanismus pro lidi, kteří potřebují nahradit všechna svoje datacentra. Běžné služby IaaS zahrnují:

- datové úložiště;
- vyvažování zatížení;
- firewally;
- poskytnutí serveru;
- konektivita veřejných a privátních sítí.

Navíc, všechny závislosti pro tyto služby jsou také k dispozici. Závislosti zahrnují monitorování, napájení, chlazení, opravy, bezpečnost, sledování zdrojů a možná nejdůležitější lidské zdroje. Někteří poskytovatelé IaaS dokonce vynalezli pohodlné řešení pro diverzifikaci výpočetních zdrojů. Znamená to rozdělení strategie podnikání, která se snaží snižovat rizika tím, že se nespoléhá na jediný produkt.[26] Tohle vše je k dispozici za cenu, které není možné dosáhnout tradiční výpočetní technikou. Typické sazby pro hostitele se pohybují v řádech desítek haléřů za hodinu.

V praxi to znamená, že doba mezi okamžikem, kdy se někdo rozhodne pro hostování na serveru, a dobou přihlášení se do serveru, byla snížená na několik minut. Vývojáři nemusejí dávat společně velkou nabídku, která zahrnuje servery, úložiště, síť, instalaci, konfiguraci, atd. Celou koncepci je možné dokázat za minimální náklady. Není třeba čekat hodiny na administrátorskou práci a nemusíme čekat ani dny na dodání objednávky. Místo toho stačí IaaS, malá počáteční investice a pár minut na výběr serveru.[19]

2.8.2 Platforma jako služba (PaaS)

Narozdíl od IaaS je PaaS mnohem abstraktnější koncept. Tento servisní model nabízí komplexní hardwarovou a softwarovou platformu, označuje se proto někdy také jako cloudware. PaaS poskytuje zařízení a služby požadované na podporu úplného životního cyklu budování programu včetně návrhu, vývoje, testování a nasazení bez nutnosti instalace softwaru.[18] Uživatelé této služby se nemusí starat o investice ani budování infrastruktury pro vývoj a následný provoz svých aplikací. PaaS je obvykle založena na jazyku HTML a JavaScript. Platforma jako služba bývá často využívána současně mnohými uživateli, proto je důraz kladen na vytvoření prostředí, které poskytuje automatickou správu souběžného přístupu, odolnost proti selhání, zabezpečení a škálovatelnost.[17]

Jako příklad může sloužit aplikace v jazyce Python. Pro spuštění aplikace potřebujeme mít v systému nainstalované rozhraní pro vykonání kódu v jazyce Python a také rozhraní, které zobrazí výsledek tohoto kódu. Příkladem může být rozhraní, nazývané WSGI 1, které funguje jako modul do webového serveru Apache. Vývojáři nebo poskytovatelé PaaS potřebují rozhraní WSGI, někde kde kód může být načten a vystaven přes webovou adresu. Je to ale pouze jeden ze způsobů, jak spouštět aplikace napsané v jazyce Python. Představíme si Apache a Python na veřejné internetové adrese s úložným prostorem a vyrovnávačem zátěže (load balancer). Nesmíme zapomenout, že je zde celá řada dalších závislostí, které jsou

třeba k funkčnosti těchto základních prvků. Apache potřebuje, aby jeho operační systém byl nakonfigurován, udržován a kontrolován. V cloud computingu tento operační systém pracuje uvnitř virtuálních strojů. Toto je detail vrstvy IaaS, kterou jsme zmínili dříve. Výhoda PaaS je právě v tom, že poskytovatel PaaS poskytuje kompletní prostředky pro podporu celého životního cyklu aplikace a to od běhu Apache přes operační systém až po napájení a chlazení v datacentru.[19]

Model PaaS je ideální platformou pro zákazníky, kteří startují se svými projekty, odpadá starost o hardware. Výhody PaaS jsou:

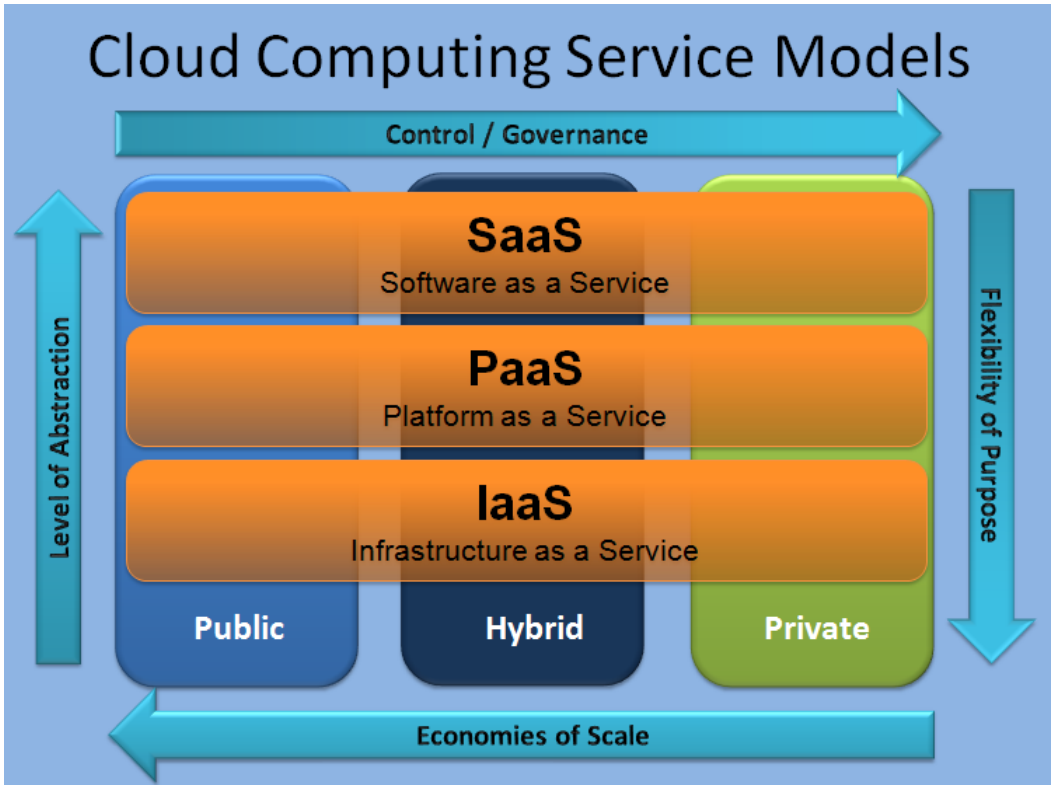
- ušetření nákladů na provoz IT;
- máme přehled o nákladech a náklady jsou předem dané;
- přenesení odpovědnosti na poskytovatele PaaS;
- přístup k aplikacím z více lokalit;
- škálovatelnost, flexibilita a zabezpečení.

Nevýhodou koncepce PaaS je snížená schopnost přenosu mezi různými poskytovateli a chybějící interoperabilita. To znamená, že není vždy možné přejít k jinému poskytovateli, pokud ano, je to často velmi finančně náročné. Další nevýhodou je ztráta veškerých dat a vytvořených aplikací v případě, že poskytovatel ukončí svůj provoz.[17]

2.8.3 Software jako služba (SaaS)

Myšlenka Software jako služba (Software as a service) není ničím nová, ale pojem SaaS už nový je. SaaS pouze odkazuje na software, který je pro použití k dispozici pouze na vyžádání. Tradičně, když někdo chtěl použít software, tak musel vejít do obchodu, koupit instalační disk, odnést si ho domů a nainstalovat program do počítače. SaaS stačí použít hostující software. U SaaS není žádná instalace, žádná aktualizace a žádné další starosti. Na SaaS není nic kouzelného. Každý kdo používá elektronickou poštu, již používá SaaS.

Ve vývoji SaaS došlo za poslední desetiletí ke značnému pokroku. Někteří moderní poskytovatelé SaaS učinili pro to, aby SaaS správně fungovalo mnoho efektivní práce. Můžeme si představit, co vše museli vývojáři Googlu vytvořit, aby Gmail vypadal tak, jak ho známe dnes. Kdyby se jeden z těchto vývojářů vrátil v čase a použil by Gmail, byl by určitě velmi ohromený tím jak vypadá, nicméně základní stavební kameny softwaru by byly podobné jako v době kdy se na něm pracovalo. Pokud by se stejný vývojář pokusil zprovoznit Gmail, byl by naprosto ztracený. Toto je běžné téma v cloud computingu. Některé aspekty jsou nám známé a některé zcela cizí ve srovnání s tradičním výpočetním prostředím.[19] Výhody SaaS jsou v nízkých pořizovacích nákladech, flexibilitě, průběžným financováním a žádnými starostmi s provozem softwaru.



Obrázek 2.2: Schéma základních typů cloud computingu [2].

[2] Z internetové stránky https://toyinogunmefun.wordpress.com/2011/06/16/effective-data-protection-for-cloud-computing-and-its-relevance-in-the-nigeria-economy/.

Kapitola 3

Nejvýznamnější poskytovatelé Cloud computingu

Cloud computing se výborně hodí pro uživatele, kteří s webovými službami teprve začínají. Dostali nápad na webovou aplikaci, ale potýkají se s nedostatkem prostředků a financí potřebných k realizování nápadu do praxe. V tomto ohledu je cloud computing nesmírně silný. Stačí, abychom si vybrali z pestré nabídky poskytovatelů cloudu. Nemusíme se starat o investice, ale jen o svoji aplikaci. Díky škálovatelnosti cloudu si v případě nečekaného zájmu o náš projekt můžeme dodatečnou kapacitu přikoupit, přitom platíme jen za to, co skutečně spotřebujeme. Na trhu existuje mnoho poskytovatelů cloud computingu. V další kapitole si některé z nich představíme.

3.1 Heroku

Heroku je aplikační platforma, která osvobodí uživatele od práce se servery, nasazením aplikace, nebo škálovatelností. Uživatel se tak může plně věnovat práci na své aplikaci a také se věnovat svému kódu.

Heroku podporuje jazyky jako Ruby, Node.js, Python, Java, and PHP. Heroku podporuje více než stovku doplňků (např. příkazový řádek nebo MySQL databáze). Doplňky jsou plně spravované služby integrované do platformy Heroku. Doplňky poskytují služby pro logování, mezipaměť, monitorování, vytrvalost a další. Dále Heroku obsahuje i sadu skriptů pro sestavování aplikací. Heroku také umožňuje správu konfigurace prostředí specificky oddělené od zdrojového kódu pro větší bezpečnost a přenositelnost. Tato data mohou být spravována pomocí konfiguračních Vars. Vars je jednoduché prostředí pro chod aplikace. Heroku podporuje Git repositáře pro správu kódu. Součástí Heroku je i velmi rozsáhlá dokumentace.[14]

Heroku vytvořila jednotku výpočetního výkonu nazvanou Dynos, tato jednotka poskytuje lehké a izolované kontejnery, na kterých běží naše aplikace. Dynos může provádět jakýkoliv typ procesu. Tyto procesy jsou navržené tak, aby běh a škála byly nezávislé. Webový Dynos spouští svůj kód a reaguje tak na požadavky HTTP. Dynos lze použít ke zvýšení výkonu HTTP. Pracovní Dynos pracuje na pozadí a typicky provádí náš kód a zpracovává úlohy z fronty. Heroku nám v nejnižší nabídce nabízí paměť 512 MB RAM, 1*CPU (mikroprocesor) a Dynos za 0.05 $ na hodinu. V nejvyšší nabídce nabízí Heroku paměť 6 GB RAM, kvalitní servis, nízké zpoždění mezi vydáním povelu paměťovému zařízení a výstupem prvních dat, vysokou propustnost a Dynos za 0.80 $ na hodinu.

Cena Heroku se odvíjí i od výběru variant Postgres databáze. Můžeme mít nejzákladnější za maximálně 9 $ až po nejvyšší nabídku, která se pohybuje od 15 000–30 000 $ za měsíc.[15]

3.2 Openshift

Openshift je veřejná a hybridní cloudová služba od společnosti Red Hat. Stejně jako ostatní poskytovatelé cloudových služeb i Openshift podporuje všechny charakteristiky cloudu, jako vývoj aplikací, škálovatelnost, hostování aplikací atd. Openshift nabízí 3 typy služeb:

- Openshift Online – aplikace veřejného cloudu a hosting platforma, která automatizuje zajišťování, řízení a škálování aplikací. Můžeme si vybrat z široké nabídky jazyků jako jsou Java, Ruby, PHP, Node.js, Python nebo Perl. Kromě pestrého výběru programovacích jazyků má Openshift Online další výhodu a tou je rychlost, kdy se snižuje čas potřebný k poskytování a správě infrastruktury, tím pádem se můžeme plně soustředit na naši aplikaci. Openshift Online je open source, tím pádem se nám usnadňuje přenositelnost aplikace. Openshift online se snadno používá, obsahuje integrované vývojové nástroje a intuitivní rozhraní.

- Openshift Enterprise – Openshift Enterprise je podobný Openshift online. Rozdíl je v tom, že si můžeme vybrat jestli budeme chtít veřejný, soukromý nebo hybridní cloud. Podporu poskytuje sám Red Hat. Openshift Enterprise může běžet na našich serverech nebo na našem privátním cloudu. Tato služba se hodí spíše pro firmy, než pro jednotlivce.

- Openshift Origin – jedná se o open source aplikační hosting platformu vyvinutou společností Red Hat. Openshift zahrnuje podporu Java EE6, Ruby, PHP, Python, Perl, MongoDB, MySQL a PostgreSQL. Openshift Origin se spouští přes virtuální stroj [21]

Openshift podporuje repositáře Git pro správu kódu. Ceník Openshift je založen na gears, které používáme. Nejnižší cena je 0.02 $ za hodinu a nejvyšší 0.10 $ za hodinu. Velikost úložného prostoru záleží na počtu gears, kdy za jeden můžeme dostat od 1 GB–6 GB podle typu služby. Platí se zde 1 $ za GB za měsíc. Za zmínku stojí i počáteční bonus, kdy v základní verzi dostaneme 3 malé gears a 1GB prostoru za jeden gear zdarma. Ve zkušební verzi Openshift nabízí: 3 malé gears a 1 GB datového úložiště na jeden gear jsou zdarma [20]

3.3 Engine Yard

Jeden z prvních cloudových poskytovatelů s dlouholetou tradicí, zaměřující se převážně na PaaS. Platforma Engine Yard využívá služeb Amazon Web Services. Engine Yard je tudíž zákazníkem Amazon Web Services. V Engine Yard můžeme pracovat s jazyky Ruby, PHP, NODE.JS, MySQL. Engine Yard obsahuje všechny vlastnosti cloudu. Nemusíme se starat o infrastrukturu a máme čas na vývoj naší aplikace. Oproti ostatním poskytovatelům nám umožní plný přístup superuživatele (root) na virtuální servery a pružnost vlastního řešení konfigurace serverů. Dále poskytuje soběstačné prostředí s vyčleněnými prostředky, které je izolováno od ostatních. Engine Yard nezavádí žádné mezivrstvy mezi naší aplikací a infrastrukturou. Samozřejmostí je také vysoká úroveň bezpečnosti a škálovatelnosti.[10]

Cena platformy a podpory se pohybuje od 0.04 $ do 0.10 $ za hodinu. Nejnižší cena za infrastrukturu se pohybuje od 0.010 $ za hodinu za poskytnutí 1 GB RAM a 1 CPU. Naopak nejvyšší cena je 2.10 $ za hodinu za poskytnutí 244 GB RAM a 32 CPU. Engine Yard poskytuje široké spektrum infrastruktury počítače.

Jako bonus při registraci získáme 25% slevu při použití Amazon Web Services (AWS) s Engine Yard[11]

3.4 Windows Azure

Windows Azure dává vývojářům prostředí pro ukládání a zpracování dat na vyžádání. V tomto prostředí mohou hostovat, škálovat a spravovat webové aplikace prostřednictvím datových center společnosti Microsoft. Naše aplikace je také výborně pokryta a to 99,95 % měsíčně SLA. Windows Azure nám umožňuje pracovat s jazyky .NET, Java, Node.js, PHP, Python nebo Ruby. Pro vytváření aplikací nám Microsoft Azure umožňuje pracovat s Microsoft Visual Studio. Aplikaci si před nasazením do cloudu můžeme otestovat pomocí Azure emulátoru. Vývojáři se nemusí starat o chybu v hardwaru nebo v síti, vše je navrženo tak, aby mohla aplikace fungovat i v případě havárie a poruchy. Windows Azure nabízí i SQL databáze. Dále se u Windows Azure můžeme spolehnout na automatické aktualizace, díky kterým je naše aplikace stále v bezpečí.[5]

Pro nejnižší instance se cena se pohybuje od 0.0211 $ do 5.1671 $ za hodinu. Podle výběru instance se nám mění počet CPU (0 – 16) a RAM paměti (0.75 GB – 112 GB). Výhodou u Windows Azure je možnost vyzkoušet verzi zdarma, ke které dostaneme 220 $. Díky tomu si můžeme pořídit nejrůznější služby, které Windows Azure poskytuje.[6]

3.5 Google App Engine

Jedním z nejvýznamnějších poskytovatelů cloudových služeb je Google App Engine. Google App Engine se řadí mezi PaaS poskytovatele. Aplikace Google App Engine je snadné vytvořit, udržovat, ukládat data a škálovat provoz aplikace. Jako u všech cloudů, tak i zde nepotřebujeme servery a další počítačovou architekturu. Staráme se jen o tvorbu naší vlastní aplikace. Google App Engine podporuje Python, PHP, Go a Java. Google App Engine umožňuje snadno vytvářet a nasazovat aplikace, která běží spolehlivě i při velkém zatížení a s velkým množstvím dat. Dalšími vlastnostmi jsou automatická škálovatelnost, plánované úkoly pro spuštění akce ve stanovené době, integrace s jinými cloudovými službami a rozhraní API Google. Aplikace běží v bezpečném prostředí. Google App Engine nabízí Software Development Kits (SDK) pro všechny podporované jazyky. Google nám pro začátek nabízí 1 GB pro ukládání dat a provoz zdarma, jenž může být zvýšen. Google poskytuje dva databázové systémy. Klasický MySQL a speciální No–SQL Database, sloužící pro ukládání velkého množství dat. No–SQL Database má jednoduchý design a podporuje horizontální i vertikální škálovatelnost.[9]

Google App Engine má širokou nabídku služeb a rozsáhlý ceník. Cena nejzákladnější nabídky se pohybuje od 0.049 $ za hodinu. K této službě máme k dispozici 1 virtuální jádro a 3.75 GB paměti. Nejzákladnější balíček databází, které Google App Engine nabízí, se nazývá D0. Databázi poté můžeme mít za 0.36 $ na den. Tento typ databáze má paměť 0.125 GB a můžeme do ní uložit až 0.5 GB dat.[8]

Google App Engine poskytuje i zkušební verzi zdarma. Tato verze nabízí 28 hodin běhu instancí zdarma. Dále pak datové úložiště s kapacitou 5 GB a 1 GB příchozího a odchozího

síťového provozu za den. Pokud překročíme dané kvóty, aplikace poběží dál ale uživatel si bude muset sáhnout hluboko do peněženky.

3.6 AppFog

Mezi další významné poskytovatele cloudových služeb, kterého si představíme je AppFog. AppFog patří mezi poskytovatele PaaS. AppFog podporuje velké množství jazyků jako Java, Ruby, PHP, Python, Node.js, Scala a Erlang a nabízí MySQL, PostgreSQL, REDIS a RabbitMQ spolu s třetí stranou, doplňkové služby. AppFog je založen na open source platformě Cloud Foundry a podporuje Git, SVN a Mercurial pro správu kódu. Není zde žádná další konfigurace serverů, firewallů, bezpečnosti nebo instalace frameworků.

Na základní úrovni služby existují poplatky za překročení kvót. Úroveň služeb je založena na paměti, namísto CPU. Cenové plány se pohybují od 20 dolarů za měsíc (2 GB RAM), 720 $ za měsíc (32 GB RAM). Appfog nabízí 30 denní free verzi, která je limitována množstvím aplikací do 2 GB RAM, 100 MB trvalé úložiště (MySQL nebo PostgreSQL) a až 8 instancí služby.[3]

Nevýhodou AppFog je, že v současné době nemá trvalý systému souborů a další úložné systémy mohou být integrovány až za další příplatek.

3.7 Amazon Web Services

Amazon Web Services je pravděpodobně první poskytovatel cloudových služeb, který nás ve spojitosti s cloudem napadne. Amazon patřil mezi první společnosti, které poskytly služby cloud computingu veřejnosti.

I když je Amazon Web Services především IaaS, mnoho dostupných služeb v AWS jsou srovnatelné s nabídkami, které poskytují jiní poskytovatelé cloudu. Můžeme využít možnosti služeb platformy, jenž jsou k dispozici v AWS, aniž by bylo nutné vytvořit nebo udržovat si své vlastní aplikační servery. Amazon Web Services podporuje velký výběr jazyků, jako Javu, Python, Ruby, Perl atd. Z databáze nabízí Oracle, MySQL a SQL Server a také webovou službu RDS, která eliminuje ruční správu databáze. Vývojáři mohou využít Amazon Elastic Beanstalk pro automatické vyvažování zátěže, auto-škálování a sledování stavu aplikace. Vzhledem k tomu, že Amazon Web Services je z velké části IaaS, není prakticky žádný limit na programovací jazyky, databáze nebo serverové technologie, které si můžeme nainstalovat a spustit. Jako nevýhodu vidím u služby Amazon Web Services to, že může vyžadovat větší režijní náklady na správu, než ostatní poskytovatelé cloudu. Cena se odvíjí na základě instancí, úložiště a aplikačních služeb, nicméně Amazon nabízí měsíční kalkulačku díky které, můžeme odhadnout náklady. Navíc noví uživatelé mohou získat 750 hodin, 30 GB dat pro ukládání a 15GB šířku pásma zdarma.[25]

Amazon Web Services poskytuje několik typů služeb, představíme si nejdůležitější z nich:

- Amazon Elastic Compute Cloud(EC2) – nabízí virtualizovaný výpočetní výkon v podobě celého virtuálního serveru a dodatečné procesorové cykly.
- Amazon Simple Storage Service(S3) – dovoluje do služby virtuálního cloudového úložiště ukládat datové položky velké až 5GB. Datové položky mohou být přístupné přes protokol HTTP nebo torrent.

- Amazon Elastic Block Store – Jedná se o perzistentní úložiště souboru pro využití společně s EC2. Na tomto úložišti jsou v případě zastavení nebo poruchy EC2 k dispozici naše soubory.
- Simple Queue Service(SQS) – umožňuje firemním počítačům komunikovat spolu navzájem, pomocí API na zasílání zpráv.
- SimpleDB – webová služba pro spouštění dotazů na strukturovaná data v reálném čase. SimpleDB spolupracuje se službami S3 a EC2, které dohromady umožňují ukládat, zpracovávat a dotazovat se na data uložená v cloudu.
- Amazon Relational Database Service (Amazon RDS) – RDS umožňuje snadno nastavit a provozovat relační databáze v cloudu. RDS Poskytuje efektivní náklady a možnost změny velikosti kapacity.
- Amazon CloudWatch – Amazon CloudWatch je služba, jenž umožňuje prohlížet a zpracovávat data o vytíženosti zakoupených instancí Amazon EC2 v reálném čase. Poskytované API nabízí širokou škálu využití pro analytické nástroje a systémy optimalizující zátěž. Služba dále umožňuje nastavit tzv. alarmy, což jsou automatická upozornění na výkyvy ve výkonu.

3.8 Shrnutí poskytovatelů cloud computingu

Jak můžeme vidět, na trhu je spousta poskytovatelů cloudových služeb. V této kapitole jsme si představili pouze ty nejdůležitější. Poskytoval cloud computingu, který mě nejvíce zaujal je Amazon Web Services. Amazon Web Services nám umožňuje vyzkoušet si verzi zdarma za velmi rozumných podmínek. Dává nám na výběr z mnoha jazyků. Patří k prvním společnostem, které začaly cloud nabízet veřejnosti. Tím pádem má bohaté zkušenosti s cloudem a má rozsáhlou uživatelskou podporu. Nehrozí ani situace, že by Amazon zbankrotoval a my museli přenášet naši aplikaci k jinému poskytovateli.

Kapitola 4

Univerzální rezervační systémy

V této kapitole nejdříve vysvětlíme, co to vlastně je univerzální rezervační systém a představíme si nejvýznamnější rezervační systémy na trhu.

4.1 Univerzální rezervační systém

Rezervační systém je druh informačního systému, jehož primárním účelem je přesně evidovat rezervace a dostupnost libovolných služeb v reálném čase. Rezervační systémy pomáhají šetřit čas a energii. Rezervační systémy mají tu výhodu, že jsou on-line a uživatelé mají vždy přehled nad svými rezervacemi. Na trhu existuje spousta druhů rezervačních systémů. Existují rezervační systémy pro hotely, sportoviště nebo služby.

V dnešní době se rezervační systémy stávají skutečným hnacím motorem mnohých sportovních center nebo dalších služeb. Přináší klientům možnosti, které plně vyhovují jejich požadavkům a mají velký vliv na jejich spokojenost. Rezervační systém nepřináší výhody pouze klientům, ale i různým uživatelům, kteří služby nabízejí. Díky rezervačnímu systému totiž mohou klienti své rezervační aktivity řídit z pohodlí domova, práce nebo i z mobilního telefonu. Stačí tak jen párkrát kliknout a vše se přizpůsobí. Komfort rezervačního systému potěší každého a vaše podnikání to může posunout vpřed.[23]

V mé bakalářské práci bychom se zaměřil na univerzální rezervační systém, který by se dal použít pro všechny druhy rezervací.

4.2 SuperSaas

SuperSaas dokáže přizpůsobit vzhled naší webové stránce a o důležitých informacích nás informuje prostřednictvím SMS. SuperSaas je dostupný v 26 jazycích, podporuje 26 měn a všechny světové zóny. Flexibilní potvrzování a připomínky, kde se nastaví denně počet rezervací a emaily nebo sms zprávy, se dokáží přizpůsobit tomu, aby zákazník na rezervaci nezapomněl. U SuperSaas je nutná registrace, spustí se průvodce, který nás vším provede. SuperSaas má přívětivé uživatelské rozhraní, které práci s ním zjednodušují.

- Integrované platby – Podpora platby přes PayPal. Můžeme prodávat kredity nebo vouchery, které můžou sloužit ke slevám. Možnost získání slev na čase, nebo dalších podmínkách.
- Rezervace – Rozvrhy rezervací mohou být integrovány do jakéhokoliv webu, nebo mohou fungovat samostatně, pokud má zákazník k dispozici vlastní web. Rezervace

jsou zobrazovány v externím kalendáři. Do rezervačního systému můžeme automaticky importovat údaje o volných nebo přihlášených termínech z Google kalendáře a můžeme je v kalendáři dokonce i zobrazit.

- Správa účtů – Jeden účet může spravovat více uživatelů, stačí jim k tomu přidání oprávnění (superuživatel). Možnost blokování a odblokování uživatelů. Všechna data mohou být vždy vyexportována. Vyexportovaná data poté můžeme použít i v excelu. SuperSaas sází na bezpečnost a spolehlivost.

- Verze rezervačního systému – SuperSaas poskytuje až 6 verzí svého rezervačního systému. První verze je verze Zdarma. Zde je nám umožněno si SuperSaas vyzkoušet. Tato verze je určena pouze pro soukromé a nekomerční účely. V této verzi je zobrazována reklama a je zde oproti ostatním verzím několik omezení. Verze obsahuje pouze 50 budoucích rezervací pro maximálně 50 uživatelů. Také se zde nenachází podpora synchronizace s Google kalendářem. Počet starých rezervací, které můžeme uchovat je maximálně 500. Nejlevnější varianta u SuperSaas je varianta A, která stojí 150 Kč. Tato varianta neobsahuje žádné reklamy a poskytuje automatickou synchronizaci s Google kalendářem. Počet starých rezervací, které můžeme uchovat je oproti předchozí verzi zvýšen na 1000. Zvýšil se i budoucí počet rezervací z 50 na 100 a maximální počet uživatelů má varianta A neomezeně.

- Ostatní verze – Zbytek variant se od varianty A liší vždy v ceně, maximálním počtu budoucích rezervací a v počtu starých rezervací. Nejvyšší varianta se nazývá varianta E, kde se zvedl počet budoucích rezervací až na 1500 a můžeme uchovávat 15 000 starých rezervací. Cena za tuto variantu je 750 Kč.

4.3 Reservanto

Reservanto poskytuje jednoduché, intuitivní, přehledné uživatelské rozhraní. Služba je vhodná pro všechny poskytovatele služeb a je zde možnost optimalizace pro konkrétní obor podnikání. Zákazníci mohou využívat platbu online buď kartou nebo převodem. Reservanto je optimalizováno i pro mobilní zařízení. Přístup je možný odkudkoliv z počítače, tabletu nebo mobilu. Nezáleží na tom, kde se zrovna zákazník nachází. Pohodlná a přehledná cesta zákazníků k rezervaci služeb. Reservanto obsahuje i prvky CRM. Tím pádem umožňuje přistupovat ke každému zákazníkovi individuálně. Díky Reservantu můžeme se zákazníky pohodlně komunikovat přes email nebo sms.

Pro zákazníky jsou připraveny 3 varianty:

- Varianta Basic – Varianta Basic je zdarma, má možnost neomezeně rezervací a zákazníků. Potvrzování zákazníků musíme provádět ručně. Můžeme využívat sms upomínek a služeb v neomezeném rozsahu. V případě problému se můžeme spolehnout na zákaznickou podporu.

- Varianta Professional – Obsahuje vše co varianta Basic, ale i něco navíc. Můžeme měnit vlastní vzhled, můžeme využít Reservanto pro sportoviště, budeme dostávat emailové reporty a upomínky. V této variantě můžeme využít správu účtu zákazníka a využívat rezervační systém jako mobilní aplikace, nebo si ho necháme přesunout na náš web. Varianta professional je založena na moderním a přirozeném designu. Systém je dočasně v beta verzi. Cena dosahuje 599 Kč za měsíc, existuje i bonus, kdy při roční platbě dostaneme 2 měsíce zdarma.

- Varianta Premium – Poslední varianta rozšiřuje variantu Professional o pokročilé statistiky, klubové karty, kreditní systém, slevy, poukazy, využívání náhradníků a import nebo export z účtu Googlu. Cena je 1499 Kč za měsíc a jako bonus můžeme ušetřit 2 měsíce při roční platbě.

4.4 Reservio

Rezervačný systém Reservio obsahuje 4 varianty:

1. Varianta Free – Varianta Free je zcela zdarma, bohužel má k dispozici pouze 40 rezervací za měsíc. Omezen je i počet registrovaných klientů. V této variantě máme možnost sledovat historii klientů 3 měsíce nazpět.

2. Varianta Starter – Počet rezervací za měsíc je zvýšen na 200 a počet registrací klientů na neomezený. Neomezeně můžeme sledovat historii klienta. Navíc tato varianta obsahuje možnost synchronizace naší rezervace s Google kalendářem, Outlookem nebo Apple iCalem. Exportované klienty máme možnost převést do formátů CSV nebo PDF a nechat vytisknout i s vlastní agendou. Varianta Starter nám umožní sledovat statistiky klientů jako je jejich chování, nejpopulárnější služby nebo nejpopulárnější zaměstnance. K dispozici máme denní agendu, do které můžeme psát vlastní poznámky a poté vytisknout. Poslední rozšíření nám umožňuje rozesílat připomínky o rezervacích přímo pomocí sms zpráv. Cena je 199 Kč za měsíc.

3. Varianta Standard – Další varianta v pořadí, cena se zvýší na 399 Kč za měsíc. Oproti minulé variantě se liší v počtu rezervací za měsíc, která se zvětšila o 300 na 500 rezervací za měsíc. Dále byly přidány i některé další funkce, jako notifikace zaměstnanců, díky tomu můžeme zasílat svým zaměstnancům emaily o nových rezervacích. Další novou funkcí je možnost správy časových, kreditních a vstupních permanentek.

4. Varianta Pro – Poslední z variant je varianta Pro za 799 Kč za měsíc. Od předchozích variant se liší tím, že rezervace za měsíc jsou neomezené a navíc můžeme logo Reservio z rezervačního formuláře, profilových stránek a emailových upozornění úplně odstranit.

4.5 NetNet

Jako další rezervační systém se zaměřím na rezervační systém od firmy Net Net s.r.o. Nabízejí podobné služby jako předchozí rezervační systémy, jediné v čem se liší je cena. Cena se podle poskytovaných služeb pohybuje v rozmezí 5–15 000 Kč za měsíc. Cena se odvíjí od počtu a velikosti provozoven, které bude rezervační systém používat. Jednou z výhod je i poskytnutí úvodního školení nebo použití zákaznické linky, bohužel jsou tyto služby také zpoplatněné.

4.6 Erezervuj.cz

Rezervační systém Erezervuj.cz je rozdělený na několik modulů, jako jsou Chalupy, soutěže, sportoviště a cvičení. Každý z modulů je rozdělen na Start a Full verze. Verze se od sebe liší v ceně, která se u Full verzí pohybuje od 99 – 249 Kč. Dalším rozdílem mezi verzemi

je v počtu rezervací za měsíc, kde ve Start verzi je to pouze 100, naopak u Full verze neomezené. U start verze chybí posílání informačních sms a také denním zálohování, které se naopak u Full verze nacházejí.

Kapitola 5

Analýza a návrh rezervačního systému

V této kapitole si vytvoříme analýzu a návrh rezervačního systému.

5.1 Požadavky na rezervační systém

Jedná se o univerzální rezervační systém, kam se uživatelé i zákazníci registrují. Uživatelé i zákazníci budou mít vlastní profil s možností úpravy údajů. Zákazník se zaregistruje a vybere si rezervaci. Uživatel nejdříve zákazníka potvrdí, a až poté se bude moci rezervovat. Zákazník vyhledá služby, které by si chtěl rezervovat a v kalendáři si zjistí volné termíny. Pokud už byla na zadaný čas událost potvrzena, uživatel se bude muset přihlásit na jiný čas, systém zabrání kolizi termínů. Zákazník se bude moci přihlašovat na termíny a aby se předešlo nešvarům například od konkurence, tak bude muset uživatel nabízející dané služby zákazníka potvrdit. Nicméně zákazník vidí, že už na dané události čeká nějaký uživatel na rezervaci a tak, když by se přihlásil na stejnou událost, dostal by se do fronty.

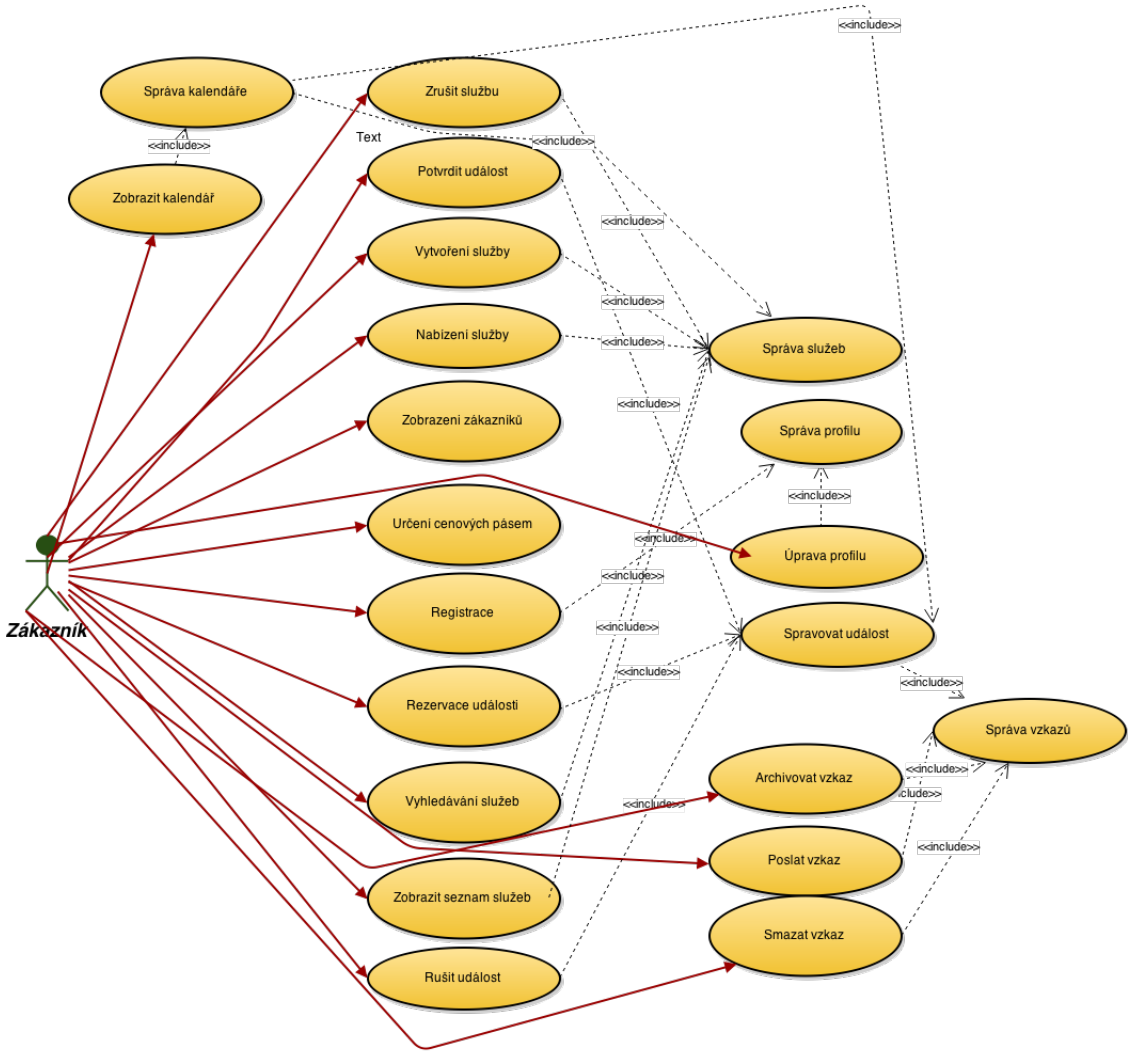
Uživatel bude moct spravovat svoje zákazníky a označovat je jako pasivní nebo aktivní. Zákazník se bude moci přihlašovat k více uživatelům. Uživatel si volí dny a hodiny, kdy chce pracovat. Uživatel si přidává služby, které chce nabízet a zákazník si služby vybírá. Zákazník i uživatel budou mít svůj kalendář se službami a rezervacemi. Zákazník se bude moci odhlásit z termínu do 24 hodin a nebo do doby než bude jeho rezervace potvrzena. Další požadavky na systém jsou zvýraznění kalendáře podle cenových pásem. Systém by měl dále umožňovat posílání vzkazů mezi uživatelem a zákazníkem. Dále se uživatel formou zprávy dozví o nové zarezervované události nebo o zrušení události.

5.2 Návrh rezervačního systému

Po analýze našeho informačního systému musí přijít na řadu návrh. Návrh bychom si měli vytvořit před začátkem každého, nejen informačního systému. Návrh je důležitý pro představu, jak bude náš rezervační systém vypadat a také jak bude fungovat. Existuje mnoho užitečných nástrojů pro návrh informačního systému. Ve své práci jsem pro návrh využil jazyk Unified Modeling Language (zk. UML). UML je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových nástrojů. Pro návrh rezervačního systému jsem využil diagram případu užití (tzv. Use Case diagram),

diagram komponent (component diagram), entity relationship diagram (zk. ERD) s UML notací a také jsem vytvořil schéma zobrazující architekturu služeb od Amazon Web Services.

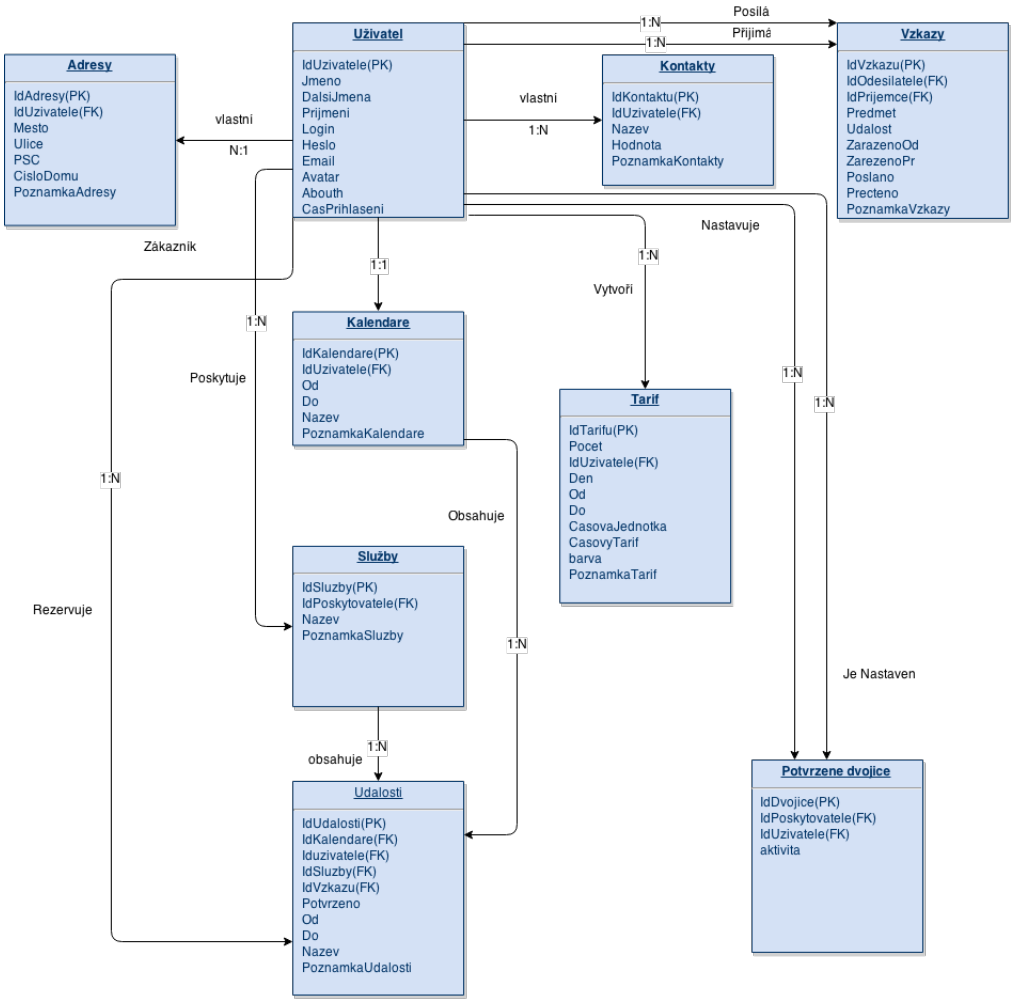
Use Case diagram zobrazuje chování systému tak, jak ho vidí uživatel. Účelem diagramu je popsat funkcionalitu systému, tedy co od něj klient nebo my můžeme očekávat. Diagram vypovídá o tom, co má systém umět, ale neříká jak to bude dělat. Tento diagram se většinou vytváří jako první, neboť je důležité nejprve zjistit, co bude systém umět. Poté má smysl se ptát, jak systém budeme vytvářet.[4] Na obrázku 5.1 můžeme vidět vytvořený Use Case diagram.



Obrázek 5.1: Use Case diagram

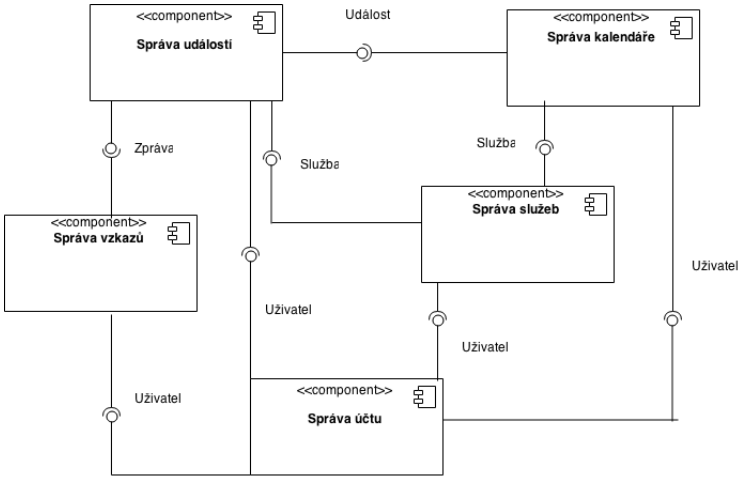
E–R digram reprezentuje data v rámci domény. Skládá se z entit a reprezentuje vztahy mezi jednotlivými entitami. Entita je objekt reálného světa rozlišitelný od jiných objektů. V našem E–R diagramu se nachází několik entit. Jedná se o entity: uživatel, adresy, kontakty, služby, potvrzené dvojice, tarif, kalendáře, události, vzkazy. Na následujícím obrázku, budou zobrazeny všechny entity. Dále jsme u všech entit zobrazili i jejich atributy a vztahy mezi jednotlivými entitami. Každá entita musí vlastnit primární klíč a může obsahovat i cizí klíče. Cizí klíče slouží k propojování tabulek. E–R diagram si můžeme prohlédnout na obrázku

5.2.



Obrázek 5.2: ER diagram

Diagram Komponent zobrazuje komponenty, jenž tvoří náš systém a závislosti mezi nimi. Jednotlivé komponenty reprezentují modulární součástí systému se zapouzdřeným obsahem, které je možné samostatně užívat i aktualizovat. Komponenty mohou obsahovat atributy a metody a mohou být vnitřně strukturovány. Diagram komponent je znázorněn na obrázku 5.3.



Obrázek 5.3: Diagram Komponent

5.3 HyperText Markup Language(HTML) a Cascading Style Sheets (CSS)

HTML je značkovací jazyk pro tvorbu webových stránek, které jsou propojeny hypertextovými odkazy. HTML je jazyk pro vytváření stránek v systému Worl Wide Web, který umožňuje publikaci dokumentů na internetu. Stejně jako každý jazyk má svá slova, tak i jazyk HTML obsahuje svoje slova, kterým se říká tagy (značky).

CSS vznikly někdy v roce 1996 jako reakce na chaoticky se vyvíjející HTML jazyk. Obecně řečeno, CSS je nějaký zápis, který určuje vzhled HTML dokumentu. CSS slouží pro oddělení definice vzhledu dokumentu od jeho obsahu. Soubor CSS je sám o sobě k ničemu, jeho funkčnost se projeví až po propojení s HTML dokumentem. HTML dokumentu můžeme různými CSS soubory zajistit různé vhledy a naopak různým HTML souborům můžeme pomocí různých CSS souborů zajistit stejný vzhled.

V mé práci jsem využil HTML a CSS pro vytvoření designu webové stránky a dále potom formuláře pro jazyk PHP. Dále jsem využil PHP a CSS pro tvorbu kalendáře.

5.4 PHP5: Hypertext Preprocessor

PHP je skriptovací jazyk pro tvorbu dynamického webu a jeho počátky spadají do roku 1994. PHP je programovací jazyk, který pracuje na straně serveru. PHP umí ukládat, měnit a mazat data. Vše se odehrává na webovém serveru. PHP skript se nejprve provede na serveru a potom odešle prohlížeči pouze výsledek. PHP je nejrozšířenějším skriptovacím jazykem pro web, v listopadu 2014 byl podíl serverů pod PHP 82 %. Proč je PHP tolik oblíbený? Jelikož je spousta webů napsána pod PHP, existuje všude spousta návodů. PHP je otevřený projekt a podporuje širokou řadu souvisejících technologií, formátů a standartů. PHP si výborně rozumí s webovým serverem Apache a snadno komunikuje s databázemi, jako MySQL, PostreSQL atd.. PHP je multiplatformní a lze jej provozovat s většinou webových serverů a na většině operačních systémů. Navíc PHP podporuje spousta poskytovatelů webhostingových služeb. Mezi nevýhody PHP se řadí například to, že PHP je interpretovaný ne kompilovaný jazyk. To znamená, že při každém spuštění PHP scriptu musí server znova kompilovat. Kdokoli má přímý přístup k serveru, může nahlédnout do našich PHP

skriptů. Dále podpora objektového programování není v PHP na dobré úrovni. Nicméně v PHP5 se situace zlepšila. PHP je aktivně vyvíjen, proto se může stát, že se některé funkce mohou změnit a nebo chovat jinak než doposud.

5.5 Javascript

Javascript je jazyk, který je součástí webových prohlížečů. JavaScript lze použít pouze v internetovském prohlížeči, ovšem zato na různých místech. Základní použití je v HTML dokumentech. Kód JavaScriptu se provádí ihned v průběhu načítání stránky do prohlížeče. Dokument může obsahovat několik párů značek SCRIPT. Důležité je, že všechny části kódu roztroušené po dokumentu dávají dohromady jeden program.[13]

5.6 Spojení mezi PHP a databázovým serverem

Databázi jsem se rozhodl tvořit v MySQL. MySQL je jeden z nejrozšířenějších databázových jazyků, proto má rozsáhlou uživatelskou podporu a rozsáhlý manuál. MySQL podle mého názoru výborně spolupracuje s jazykem PHP. Pro lepší způsob práce s databázemi jsem využil rozhraní PDO. PDO je abstraktní databázová vrstva, která slouží k oddělení logiky aplikace od logiky komunikace s databází. Aplikace poté může za určitých předpokladů používat jedno z několika databázových řešení, které je podporováno. PDO také definuje mnoho metod, které se pro každou databázi jmenují stejně a jsou také přehlednější. Pomocí funkce phpinfo() ověříme, zda je PDO k dispozici a pro jaké aplikace.

Kapitola 6

Amazon Web Services (AWS)

V této kapitole se zaměřím na práci s AWS, jak zprovoznit a dále pracovat s aplikací pod Amazonem tak, aby vše pochopil i začátečník. AWS nabízí jak Iaas tak PaaS. Klíčovým rozlišovacím znakem mezi IaaS a PaaS je to, jaký typ služby je nabízen. Zákazníci v IaaS typicky pracují s virtuálními stroji, které si konfigurují sami podle potřeby. V PaaS zákazníci pracují s již vytvořenými službami a udržovanými PaaS poskytovatelem.

Pro uživatele, kteří s vývojem rezervačního nebo jakéhokoliv jiného informačního systému začínají, si představíme návod, jak vůbec zprovoznit služby od Amazonu. Popíšeme si vše potřebné pro nainstalování a zprovoznění služby, pod kterou budeme informační systém vyvíjet. Cílem mé práce je ukázat výhody cloud computingu oproti běžným webovým hostingům. To se může hodit převážně uživatelům začínajícím s tvorbou nového webu a nemají ponětí o tom, jak začít. Na internetu je bezpochyby spousta návodů jak pracovat na informačním systému. Náš rezervační systém bude ukázkou běžného webu psaného stejně jako dříve bez různých frameworků. Ukážeme si dále jak takový systém přenést do cloudu, a jak se dá takový systém škálovat.

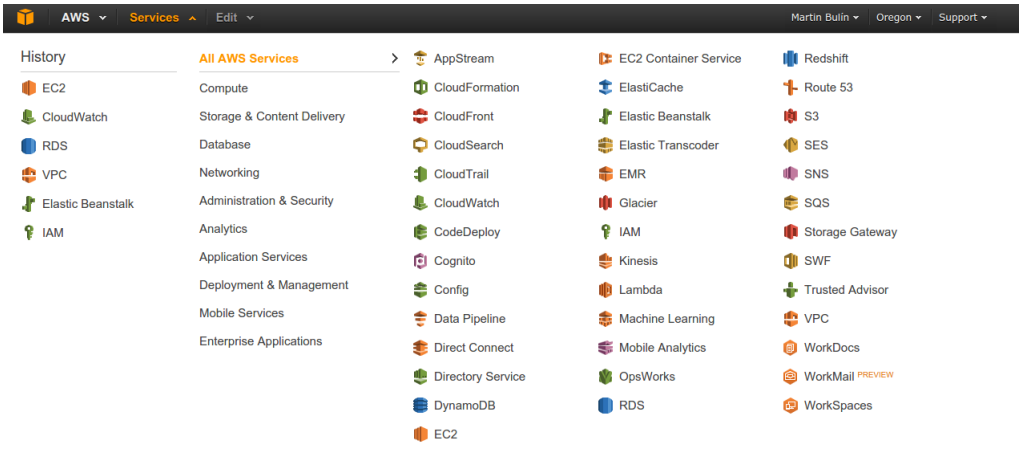
Pro většinu začínajících uživatelů by byla tvorba webových projektů pod službou S3 velice složitá, proto jsem se rozhodl poskytnout návod na službu EC2 a ukázat, jak pomocí této služby dokázat výhody cloudu. Pro začínajícího uživatele nabízí EC2 prvky a doplňkové služby, díky kterým Amazon Web Services provede kvalitní škálovatelnost, i když se řadí pod IaaS. Amazon EC2 je služba, která je navržena tak, aby škálovatelnost byla pro uživatele co nejjednodušší. Amazon EC2 umožňuje zvýšit nebo snížit kapacitu během několika minut. Amazon EC2 funguje také ve spojení s Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon SimpleDB a Amazon Simple Queue Service (Amazon SQS). Amazon EC2 nabízí vysoce spolehlivé a bezpečné prostředí. Nejdůležitější na službě Amazon EC2 je to, že je služba velice jednoduchá na pochopení. Na obrázku 6.2 je vyobrazen diagram architektury služeb AWS, který zobrazuje, jak jsou mezi sebou služby propojené.

6.1 Amazon Elastic Compute Cloud (EC2)

Pro svou práci jsem si vybral službu Amazon Elastic Compute Cloud (EC2). Zde si ukážeme, jak nainstalovat a spustit službu EC2 Instance s Apache, PHP a MySQL.

Ze všeho nejdříve se musíme zaregistrovat. Pokud už jsme registrovaní, můžeme se přihlásit. Automaticky nás Amazon přesměruje na stránku AWS Management Console. Console Home je jednou z nejdůležitějších stránek našeho online účtu v AWS, kde máme

možnost měnit nastavení všech služeb AWS, které využíváme. Vybereme si službu EC2 a vytvoříme si virtuální server v cloudu. Na další stránce máme přehled všech našich zdrojů, také se zde nachází i všechny naše běžící instance. My si vytvoříme novou instanci. Klikneme na tlačítko spustit instanci (Launch Instance). Po kliknutí se dostaneme ke kroku, kde si vybereme Amazon Machine Image (AMI). AMI je šablona, která obsahuje nastavení softwaru (např. operační systém nebo aplikační server). Na výběr máme Amazon Linux AMI, Red Hat Enterprise, Ubuntu Server a další. AMI jsou buď zcela zdarma, a nebo se zde nachází verze premium za příplatek. Na většinu projektů postačí verze zdarma. Pokud bychom ovšem počítali s rozsáhlým projektem s mnoha uživateli, doporučoval bych premium verzi. Nejlepší z nabídky AMI se kterým se bude nejlépe pracovat, je podle mého názoru Ubuntu, jelikož v tomto systému pracuji a mám sním bohaté zkušenosti.



Obrázek 6.1: AWS Management Console

V dalším kroku přijde na řadu výběr instance. Amazon EC2 poskytuje široký výběr typů instancí, které jsou optimalizovány tak, aby se dokázaly přizpůsobit různým případům použití. Instance jsou virtuální servery, na kterých lze spustit aplikace. Jednotlivé instance se od sebe liší v různých kombinacích CPU, paměti, úložištích, síťových kapacitách a dají nám možnost zvolit si vhodnou kombinaci zdrojů pro naše aplikace.

Pro svůj projekt jsem si vybral instanci t2.micro, která je zdarma. T2.micro nabízí 1*CPU, 1 GiB paměti, pouze EBS úložiště a nízkou až střední síťovou kapacitu. Pokud jsme si vybrali instanci, klikneme na Next:Configure Instance Details a přesuneme se na další krok. Zde si nastavíme konfiguraci instance tak, aby vyhovovala našim požadavkům. Můžeme dokonce spustit více instancí ze stejného AMI, toho můžeme využít k nižším cenám a dalším kombinacím. Kliknutím na Next:Add Storage se přesuneme dále.

Přidáme si úložiště. Naše instance zahájí následující nastavení paměťového zařízení. Na instanci můžeme připojit dodatečné jednotky EBS, jednotky instance úložiště, nebo upravit nastavení svazku root. Můžeme také připojit dodatečné jednotky EBS po spuštění instance, ale ne pomocí instancí svazků úložiště.

EBS (Amazon Elastic Block Store) poskytuje trvalé svazky na úrovni bloku úložiště pro použití s Amazon EC2 v AWS Cloud. Každý svazek EBS je automaticky replikován do dostupné zóny, kterou chrání před selháním komponent, nabízející dostupnost a trvanlivost. Jednotky EBS nabízí konzistentní nebo nízkou latenci, a také výkon potřebný ke spuštění naší pracovní zátěže. S EBS se může měřítko využití pohybovat nahoru nebo dolů během

několika minut.

V dalším kroku zadáme Tag a přejdeme k nastavení bezpečnostní skupiny. Bezpečnostní skupina je soubor pravidel brány firewall, které řídí provoz instancí. Můžeme zde vytvořit novou bezpečnostní skupinu, a nebo si vybrat ze skupiny existujících bezpečnostních skupin. Také zde můžeme přidat pravidla tak, aby konkrétní provoz dosáhl dané instance. Například můžeme přidat pravidla, která umožní neomezený přístup k HTTP a HTTPS portům. Pro mou práci jsem využil protokoly SSH s portem 22, HTTP s portem 80. Na poslední stránce máme rekapitulaci nastavení naší instance, kde můžeme popřípadě změnit naše nastavení. Pokud máme vše nastaveno, můžeme spustit naši instanci.

6.2 Nastavení služby Amazon EC2

V další části si nastavíme dvojici klíčů. Dvojice klíčů se skládá z veřejného klíče AWS a soukromého klíče, které ukládáme. Oba klíče společně nám umožní bezpečně se připojit k instanci. Můžeme si vybrat z již existujících dvojic klíčů, anebo si vytvořit vlastní dvojici klíčů. Pokud si vytvoříme novou dvojici, musíme si ji vhodně pojmenovat a poté si klíče stáhneme. Klíč si bezpečně uložíme, protože pokud o něj přijdeme, tak si budeme muset vytvořit novou instanci.

Poté spustíme naši instanci. Objeví se nám zpráva ohledně spuštění naší instance. Klikneme na název naší právě vytvořené instance a objeví se nám tabulka, ve které můžeme vidět všechny naše vytvořené instance. Vidíme v jakém stavu se mohou naše instance nacházet. Mohou být pozastavené, v běhu nebo nevyřízené. Pokud je naše instance v nevyřízeném stavu, musíme počkat pár minut, než přejde do běhu.

Pokud vše proběhlo v pořádku, přesuneme se do složky, ve které se nachází naše klíče stažené před chvíli. Náš klíč si pojmenuji například testTutorial. Pro naše další nastavení budeme potřebovat příkazový řádek. Nejdříve musíme našemu klíči nastavit práva (např. Users) začneme příkazy:

```
chown: Users testTutorial.pem
chmod 400 testTutorial.pem
```

Těmito příkazy změníme v náš prospěch vlastnictví souboru. Dalším příkazem se pomocí protokolu SSH připojíme k naší instanci. Nejdříve ale budeme muset zjistit IP adresu. Tu zjistíme na stránce, kde se zobrazoval stav naší instance. Ve sloupci Public IP se nám zobrazí naše IP adresa (například: 54.84.151.125). Poslední věc, kterou budeme k připojení potřebovat, je název našeho virtuálního stroje. Pro Ubuntu je název ubuntu, pro Amazon je to ec2user a pro Red Hat je název root. Celý příkaz pro připojení k naší instanci je tedy:

```
ssh -i testTutorial.pem ubuntu@54.84.151.125
```

Pokud vše proběhlo bez problému, podařilo se nám připojit k našemu virtuálnímu stroji.

K dokončení našeho nastavení si budeme muset nainstalovat webový server. Rozhodl jsem se pro webový server Apache2. Před začátkem další práce provedeme aktualizaci. Aktualizaci provedeme příkazem:

```
sudo apt-get update
```

Nainstalování webového serveru a všech dalších potřebných knihoven provedeme příkazem:

```
sudo apt-get install apache2 libapache2-mod-php5 mysql-server
php5-mysql php5
```

Rozebereme si jednotlivé položky v příkazu. Apache2, jak už jsem zmínil před chvílí, je webový server. Libapache2-mod-php5 je knihovna, která obsahuje modul pro Apache2. Musíme vzít na vědomí i to, že tato knihovna funguje pouze s Apachem. Mysql-server nastaví MySQL databázi a php5-mysql je modul pro PHP5. Poslední příkaz php5 udává náš skriptovací jazyk. Pokračujeme dále, stiskneme enter, poté na všechny otázky odpovíme ano a otevře se nám stránka, kde se po nás bude chtít nastavit konfiguraci balíčku. Budeme muset zadat heslo a poté ho ještě potvrdit. Po potvrzení se nám vše potřebné doinstaluje. Po doinstalaci budeme muset restartovat server Apache, toho docílíme příkazem:

```
sudo service apache2 restart
```

6.3 Nastavení MySQL

Dále spustíme příkaz s názvem

```
sudo mysql_secure_installation
```

Spustí se průvodce, který nás provede přes postupy, odstraňující přednastavené výchozí hodnoty, jenž mohou být při našem dalším použití nebezpečné. Po zadání příkazu budeme nejprve vyzváni k zadání hesla uživatele root. Ihned poté budeme dotázáni na řadu otázek, počínaje otázkou změny hesla uživatele root. Toto je jedinečná příležitost, pokud jsme tak již neučinili, změnit heslo na co nejbezpečnější. Počínaje touto otázkou odpovíme na všechny zbývající otázky ano. Nyní je naše instalace konečně hotová. Ještě jednou restartujeme webový server Apache. Příkazem:

```
mysql -u root -p
```

si můžeme vyzkoušet zda naše databáze funguje. Pro další pokračování se budeme muset vrátit na webovou stránku, s výpisem našich instancí, kde si v kolonce Public DNS zkopírujeme adresu DNS (například: http://ec2-52-24-133-30.us-west-2.compute.amazonaws.com/), tím získáme naši internetovou adresu, na kterou se mohou ostatní uživatelé internetu připojit. Pokud chceme přidat soubory nakopírujeme je do složky /www.

6.4 Amazon Relational Database Service (RDS)

V této kapitole si představíme, jak nainstalovat a nastavit MySQL ve službě Amazon Relational Database Service (RDS). RDS umožňuje snadné nastavení a provozování relační databáze v cloudu. RDS je také důležité měřítko relační databáze v cloudu. RDS poskytuje nákladově efektivní možnost změnit velikost kapacity, zatímco správa úkolů pro správu databáze je časově náročná, uvolní nás tím a my se můžeme soustředit na vývoj naší aplikace. Amazon RDS poskytuje on-line přístup k relačním databázovým systémům jako jsou MySQL, Oracle, Microsoft SQL Server nebo PostgreSQL.

Tím, že Amazon RDS podporuje nejčastěji používané relační databázové systémy pro nás má jednu velkou výhodu. Výhodou je, že náš kód, aplikace a nástroje, které již dnes používáme s naší existující databází, můžeme použít v Amazon RDS. Podporované relační databáze jsou jednoduché na pochopení i méně zkušeným uživatelům a navíc mají širokou internetovou podporu. Amazon RDS automaticky opravuje databázový software a zálohuje databáze, dále provádí ukládání záloh na dobu, kterou si může uživatel sám nadefinovat a také umožňuje tzv. point-in-time zotavení. Můžeme také u Amazonu RDS využít flexibility

tím, že bude měřítko výpočetních prostředků nebo skladovací kapacity spojené s databázovými stupni prostřednictvím volání jediné API.

Amazon RDS umožňuje škálování nad kapacitu nasazené databáze. Stejně jako u všech služeb Amazon Web Services, nejsou u Amazon RDS žádné vstupní investice zapotřebí a platíme pouze za zdroje, které skutečně využíváme.[1]

6.5 Nastavení služby Relational Database Service (RDS)

Nyní si ukážeme, jak Amazon RDS zprovoznit a nastavit. Nejprve se přihlásíme pomocí našeho účtu do AWS. Na stránce AWS Management Console, si vybereme službu RDS. Dostaneme se na stránku, kde máme přehled o všech našich spuštěných instancích RDS. V sekci Create Instance založíme naši instanci. Objeví se nám seznam všech databázových systémů, které Amazon RDS nabízí. Na výběr máme MySQL, Oracle, Microsoft SQL Server nebo PostgreSQL. Já osobně jsem se ve svém projektu rozhodl pro MySQL. Dále vybereme odpověď ne, tato instance je určena pro externí použití výtvoru nebo v rámci RDS volné použití třídy. Pokud se někdy rozhodneme pro změnu, můžeme se později vrátit a označit druhou odpověď.

V dalším kroku nastavíme specifické detaily databáze. Zkontrolujeme, zda je vše v pořádku. V DB Engine bychom měli mít vybraný název našeho databázového systému. V našem případě *mysql*. Verzi databáze si můžeme nastavit dle libosti, nicméně doporučuji zadat nejvyšší možnou verzi. Může se stát, že v nějaké starší verzi se objeví chyba, která už je již ve vyšší verzi opravena. V DB Instance Class vybereme třídu db.t2.micro, která bude pro náš projekt plně postačující a navíc je zdarma. Pokud bychom samozřejmě chtěli databázi s vyšším výkonem, můžeme si vybrat z široké nabídky databázových instančních tříd.

Nyní přijde na řadu Multi-AZ Deployment, kde se rozhodneme, zda bude Amazon RDS udržovat synchronní pohotovostní repliku v jiném obsazenosti zóny než instance DB. Každý AZ je sestrojen tak, aby byl izolován od chyb jiného AZ. Při založení instance v odděleném AZ, můžeme ochránit naše aplikace od chyb v jednoduché lokaci. Amazon RDS dokáže udržovat synchronní pohotovostní repliku v jiné dostupné zóně než instance DB. Amazon RDS se automaticky převede do pohotovostního režimu v případě plánované či neplánované odstávky primárního serveru. Proto zvolíme ano, čímž dosáhne naše RDS vysoké dostupnosti. V Auto Minor Version Upgrade zaškrtneme ano. Pokud bychom zadali negativní odpověď, dosáhli bychom toho, že by se nám automaticky nevylepšovala verze naší databázové instance, když by byla dostupná. Nemusíme mít obavy, když bychom se rozhodli pro změnu, tak později budeme mít možnost tento krok změnit.

Dalším důležitým krokem je vybrat paměťový prostor (Storage Type). Na výběr máme z General Purpose (SSD), který je vhodný pro širokou škálu databázových zatížení. Další možností je zvolit IOPS (SSD), který je výhodný pro minimální počet vstupních a výstupních operací za sekundu a také intenzivního databázového zatížení. A jako poslední možnost zvolit si paměťový prostor jako magnetický disk, který bych nedoporučoval, jelikož je vhodný hlavně pro malé databázové zátěže, kde nejsou data často přístupná.

V sekci Allocated Storage si naalokujeme prostor v Gigabytech. Platíme pouze za tolik prostoru, který použijeme a v případě potřeby si prostor můžeme zvýšit nebo snížit. Minimum alokovaného prostoru je 5 GB a maximum se může vyšplhat až na 3072 GB. Nakonec se dostaneme k té zábavnější části, kde pojmenujeme naší instanci a vytvoříme Master uživatele. Do DB Instance Identifier vložíme unikátní klíč, díky kterému budeme moci dokázat identifikaci naší databázové instance. Tento unikátní klíč musí být jedinečný pro všechny instance databází našeho účtu AWS v aktuální oblasti. V dalším kroku zvolíme tedy jméno

Master uživatele například awsuser a jako poslední krok zadáme Master password například mypassword. Heslo, které jsme zadali poté ještě jednou potvrdíme a můžeme se posunout na stránku, kde si nastavíme pokročilá nastavení.

Zde v kolonce VPS nastavíme default VPC a v Subnet group nastavíme default. Pokud v Publicly Accessible zvolíme ano, říkáme tím, že chceme aby EC2 instance a zařízení bylo mimo VPC hostitele databázové instance a připojení k instanci databáze. Zvolíme-li ne, Amazon RDS nám nepřiřadí veřejnou IP adresu databázové instance, a také se žádná instance EC2 nebo zařízení mimo VPC nebude moci připojit. V availability Zone necháme no preference. Dále budeme muset vybrat jednu nebo více skupin zabezpečení VPC, můžeme nechat i základní nastavení. Tato nastavení určují, které EC2 instance a zařízení se může připojit k instanci databáze. Dalším nejdůležitějším nastavením je jméno naší databáze (například: users). Dále musíme ještě vyplnit port databáze. Jedná se o specifický TCP/IP port, který databázová instance použije pro připojení k naší aplikaci. Port necháme implicitně na 3306.

Zbytek nastavení můžeme nechat v základním nastavení. Za zmínku ještě stojí zvolení, za jaký počet dnů dojde k automatickému zálohování instance databáze. Můžeme si také vybrat časový rozsah, kdy se automatické zálohování vytvoří. Zmínil bych se ještě o nastavení automatické aktualizace, kde si můžeme vybrat nejen čas, kdy budeme chtít aktualizaci provést, ale také si můžeme vybrat, v jaký den.

Jestliže máme vše potřebné nastavené, můžeme se přesunout k založení naší instance databáze. Přesměruje nás to na jinou stránku. Na této stránce je napsáno, že je naše databázová instance úspěšně vytvořená. Dále se podíváme na shrnutí nastavení naší instance databáze kliknutím na view your DB instances. Objeví se nám stránky, na kterých je vypsáno naše dosavadní nastavení databázové instance. Pokud se nám změní status z creating na available, tak máme vytvořenou databázi v Amazon RDS.

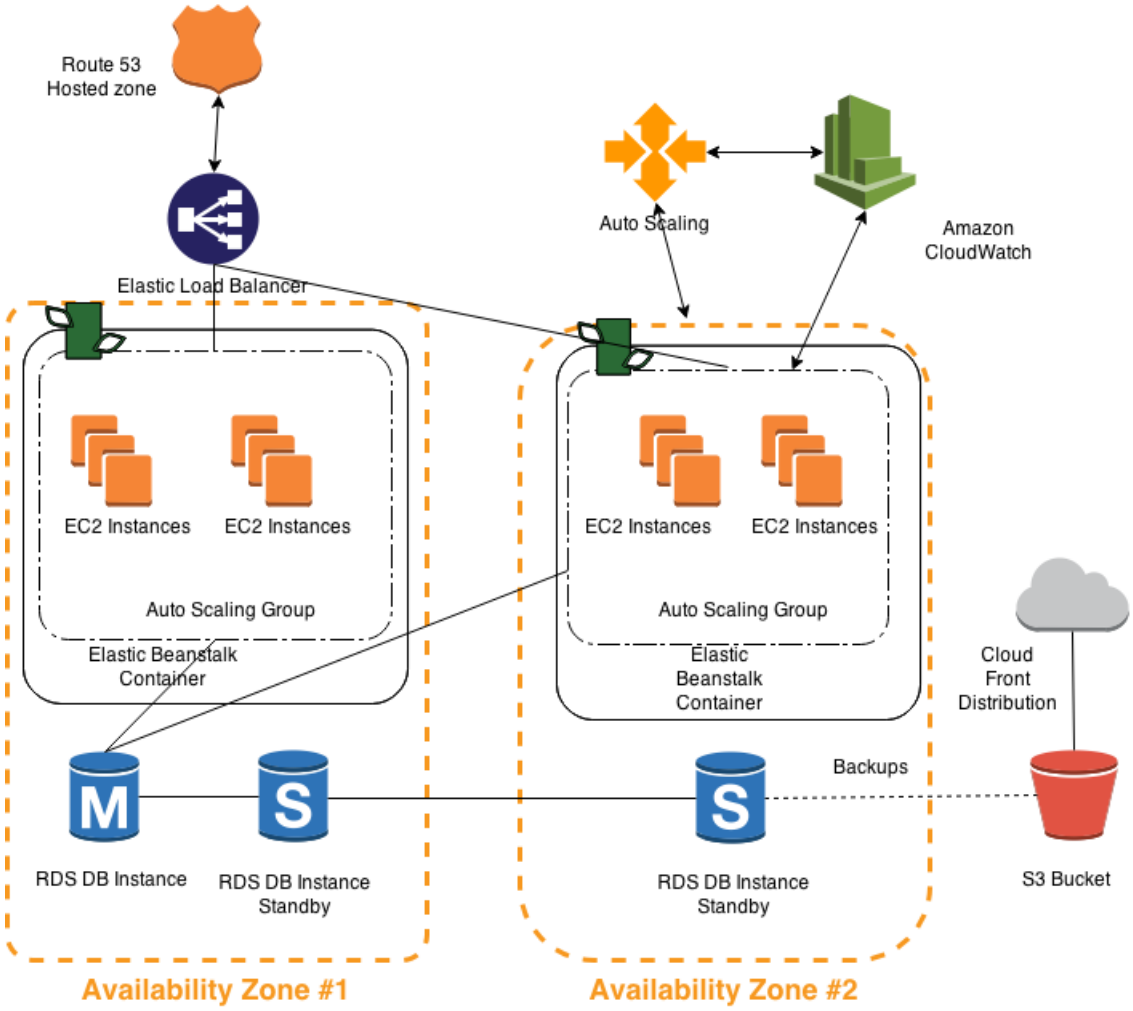
Nyní přijde na řadu zábavnější část, a to je připojení se k naší nově vytvořené instanci pomocí příkazového řádku. Připojíme se k naší instanci stejně tak, jak jsme popisovali v dřívější podkapitole a to např. příkazem:

```
ssh -i testTutorial.pem ubuntu@54.84.151.125
```

Po úspěšném připojení k naší instanci se připojíme do naší databáze. Příkaz pro připojení do databáze může být:

```
mysql -uawsuser -pmypassword
-hmydbinstance.cm2ze6rovzg0.us-west-2.rds.amazonaws.com
```

Údaj awsuser je naše uživatelské jméno, které jsme použili při vytváření instance databáze. Můžeme ho také najít v kolonce Username v detailech databáze. Mypassword je heslo, které jsme vytvořili při zakládání naší databáze. Nakonec nám zbývá ještě údaj mydbinstance.cm2ze6rovzg0.us-west-2.rds.amazonaws.com. Tento údaj najdeme v detailech databázové instance v kolonce Endpoint. Tento údaj zkopírujeme celý až po dvojtečku a číslo portu. Pokud tento příkaz použijeme, dostaneme se do databáze. Pomocí jednoduchých MySQL příkazů si můžeme vyzkoušet, zda databáze funguje správně. Později, až si vytvoříme naší vlastní databázi se zde pomocí různých příkazů můžeme podívat, jak vypadá a zkontrolovat si všechny zadané údaje.



Obrázek 6.2: Diagram architektury služeb AWS

6.6 Implementace rezervačního systému

Po analýze, návrhu systému a vybrání vhodné služby se přesuneme k implementaci. Vytvořili jsme si webovou službu rezervačního systému tak, abychom mohli vyzkoušet a otestovat vlastnosti služby v cloudu. Nejdříve vytvoříme design stránky pomocí jazyků HTML, CSS a případně JavaScriptu. Poté bychom si měli vytvořit databázi pomocí MySQL/MySQLi a připojit se k ní. K úspěšnému připojení budeme potřebovat adresu serveru, uživatelské jméno, heslo a název databáze. Databáze bude tvořena tabulkami a na řádcích každé tabulky se definují jednotlivé sloupce, jejich datový typ i počet znaků. U každé tabulky bychom neměli zapomenout také na primární a cizí klíče. Jádro webové služby se skládá z PHP.

Začínající uživatel bez zkušeností s PHP si může snadno vytvořit svůj vlastní web i bez různých frameworků. Web je psán jednoduše pro cloud, chceme využít škálovatelnosti webu pomocí interpretu jazyka PHP. Jelikož si web píšeme sami, můžeme si naprogramovat pouze to, co potřebujeme a tudíž je náš web rychlejší. Kód v PHP je i lépe přenositelný, skoro každý webový hosting nebo webový hosting v cloudu nám umožní vložit PHP kód, neboť většina hostingů podporuje jazyk PHP. Kalendář pro náš web jsme také vytvářeli čistě v PHP. Ačkoliv je na internetu spousta již vytvořených kalendářů, nenašel jsem ten,

který by nám vyhovoval. Open source kalendáře nejsou moc kvalitně vytvořené, navíc je celkem obtížné vyznat se v cizím kódu. Pokud se objeví kvalitní kalendář, tak stojí i několik desítek dolarů. Proto jsme si vytvořili vlastní kalendář, jenž splňuje naše podmínky. Dále si uživatelé mezi sebou mohou posílat vzkazy a také přijde upozornění, že někdo ruší nebo založil novou rezervaci. Samozřejmě jsme nezapomněli na registrace uživatelů s možností změny údajů.

Kapitola 7

Testování

V této kapitole si ukážeme, jak nastavit v AWS škálovatelnost naší aplikace a jak ji použít. Otestujeme si naši aplikaci na škálovatelnost a ukážeme si výhody cloudu.

7.1 Škálovatelnost

Jednou z motivací provozu webové aplikace v cloudu je škálovatelnost. Pojem škálovatelnost v počítačovém prostředí může být chápán jako schopnost systému pružně reagovat na vzrůstající nebo klesající nároky uživatelů na výkon systému. Díky škálovatelnosti můžeme plynule měnit množství alokovaných zdrojů, tím pádem dosáhneme efektivnosti a flexibilnosti aplikace. Škálovatelnost nám ušetří finanční prostředky, které můžeme využít pro zlepšení naší aplikace. Vyhneme se situacím, kdyby náš web nebyl dostupný kvůli přetížení služby při enormním nárůstu počtu uživatelů. Existují dva typy škálovatelnosti: vertikální a horizontální. Při vertikální škálovatelnosti dojde ke zvýšení objemu stávájících přidělených zdrojů (např. přidáním procesorů, paměti nebo disků). Oproti tomu horizontální škálovatelnost neupravuje stávající prostředky, ale rovnou přidává nové (např. přidáme celý nový server, beze změny stávajícího hardwaru). AWS poskytuje obě varianty škálovatelnosti. Jelikož bychom chtěli neomezený růst našeho webu, budeme se věnovat horizontálnímu škálování.[24]

Horizontální škálování poskytuje Amazon podporu ve službě Amazon EC2 Auto Scalling Groups. Auto Scalling nám zajistí správný počet dostupných instancí, které zvládnou jakoukoliv zátěž naší aplikace v EC2. Můžeme specifikovat minimum instancí v každé Auto Scalling group a Auto Scalling nám zajistí, že nikdy neklesneme pod tuto velikost. Naopak můžeme nastavit maximum instancí se zárukou, že se nikdy nedostaneme nad tuto úroveň. Auto Scalling nám bude tedy přidávat nebo odebírat instance podle potřeby. Kolik instancí budeme potřebovat záleží na objemu dat a složitosti aplikace. Pro náš web nám bude stačit jedna t1.micro.instance, nejmenší a nejlevnější možnost Amazonu EC2.

Ke sledování výkonnosti EC2 serveru budeme využívat službu Amazon CloudWatch. Díky této službě můžeme sledovat různé výkonnostní metriky serveru v reálném čase. Jedná se například o:

- využití CPU (CPU Utilization) (%);
- využití paměti (Memory Utilization) (%);
- využití sítě (Network Out Utilization) (MB);

- využití paměti (Memory Used) (MB);
- dostupná paměť (Memory Available) (MB).

Záleží jen na nás, co budeme chtít sledovat. Nejdůležitější metriky jsou využití procesoru, paměti nebo sítě. Podle těchto metrik budeme přidávat nebo odebírat instance. Pro nastavení Auto Scallingu musíme vybrat AMI, nastavit a spustit Elastic Load Balancing (ELB). ELB je služba, která kontroluje příchozí HTTP/HTTPS požadavky a dle potřeby je rozděluje mezi jednotlivé servery ve skupině Auto Scalling groups. U ELB je možné nastavit minimální a maximální množství serverů. ELB automaticky přidává nebo odebírá servery dle aktuální potřeby.

Poté se můžeme pustit do nastavení podmínek pro zvyšování a snižování počtu potřebných serverů (scalling policies) v Auto Scalling group. Zde máme možnost nastavit jméno, alarm a akci, která se provede a nakonec čas čekání, než se znova začne testovat. Můžeme vybrat existující alarm nebo přidat nový alarm.

Vytvoření nového alarmu U nově vytvářeného alarmu lze nastavovat zasílání upozornění na spuštěný alarm. Nastavíme podmínku, kdy se alarm vykoná. Lze vybírat z několika možností (např. minimum, maximum, průměr ...) a z různých metrik serveru (využití CPU, disku a sítě). Poté nastavíme limit metriky pro spuštění alarmu. Nakonec se nastaví délka periody, po kterou musí být nastavená podmínka pro spuštění alarmu.

Nastavení akce Pokud máme nastavený alarm, pokračujeme nastavením akce, kde máme 3 možnosti práce s instancemi (počet instancí nebo procentuální vyjádření z celkového počtu instancí ve skupině):

- přidat (add);
- odstranit (remove);
- nastavit (set to).

7.2 Testování s nástrojem Siege

Pro testování využijeme nástroj Siege, který dokáže vyvolat tisíce HTTP dotazů na webovou stránku ve velice krátkém čase. Díky tomu můžeme sledovat, co se stane s naším webem a zda funguje Auto Scalling. Pokud je vše správně nastavené, tak by se měly přidat další instance. Siege nainstalujeme v Linuxu jednoduchým příkazem:

```
sudo apt--get install siege
```

Siege je velice jednoduchý na používání. Spustí se příkazem:

```
siege --c25 --t10M www.example.com
```

Díky tomuto nástroji se nám zvýšil provoz v našem systému, což vedlo ke zvýšení počtu instancí. Po skončení testování se provoz snížil a instance byly automaticky odebírány. Ilustrační obrázky viz příloha D.

Po správném nastavení Auto Scallingu jsem dosáhl toho, že při velkém objemu dat se zvyšoval počet instancí, naopak při malém objemu dat došlo ke snížení počtu instancí. Při testování jsem došel k závěru, že nejslabším článkem škálovatelného systému je databáze.

Důvodem je to, že databáze by musela být na všech instancích aktuální. Škálovatelnost databáze by se dala použít v případě, že by se jednalo o čtení z databáze nebo lze využít databázi přímo v paměti (in–memory database), která je u Amazonu dostupná pouze u placené verze. Pokud se nám zvýší počet instancí, databáze se spustí na nejvýkonnější instanci.

Kapitola 8

Závěr

Cílem této práce bylo vytvořit univerzální rezervační systém využívající výhody cloud computingu. Na začátku práce je vysvětlen pojem cloud computing a jsou uvedeny výhody i nevýhody použití cloudu v praxi. Dále jsem vybral nejvýznamnější poskytovatele cloudu, kde jsem uvedl jejich základní parametry. Z širokého spektra poskytovatelů jsem se rozhodl pro Amazon Web Services, protože svoje služby poskytuje Amazon již mnoho let, má bohaté zkušenosti a poskytuje základní služby zdarma. Z nabídky služeb od Amazonu jsem nejvíce využil služby Amazon EC2 a Amazon RDS a popsal jejich nastavení.

V další části jsem popsal existující rezervační systémy. Podle zjištěných poznatků jsem analyzoval a navrhl vlastní rezervační systém. Rezervační systém je implementován pomocí jazyků PHP, HTML, CSS a Javascript a využívá databázi MySQL.

V poslední části práce jsem testoval škálovatelnost vytvořeného systému, což je hlavní předností systému v cloudu oproti systému na web hostingu. Pomocí nástroje Siege jsem simuloval velké množství přístupů uživatelů z několika počítačů v krátkém čase. Po správném nastavení Auto Scallingu jsem dosáhl toho, že při velkém objemu dat se zvyšoval počet instancí, naopak při malém objemu dat došlo ke snížení počtu instancí. Při testování jsem došel k závěru, že nejslabším článkem škálovatelného systému je databáze. Důvodem je to, že databáze by musela být na všech instancích aktuální. Škálovatelnost databáze by se dala použít v případě, že by se jednalo o čtení z databáze nebo lze využít databázi přímo v paměti (in–memory database), která je u Amazonu dostupná pouze u placené verze. Pokud se nám zvýší počet instancí, databáze se spustí na nejvýkonnější instanci.

Závěrem lze tedy doporučit aplikaci v cloudu všem, kdo předpokládají dynamický růst svých aplikací. Cloud oproti obyčejnému hostingu, umožňuje dynamickou změnu nastavení, která je podmíněna pouze cenou, kterou je uživatel ochotný za poskytované služby zaplatit. Pro statické stránky nebo jednoduché systémy můžu doporučit obyčejný web hosting, pro náročnější aplikace to již nemusí být ovšem dostačující.

Jako vhodné pokračování práce bych se věnoval dalším funkcím v rezervačním systému. Například přidáním více kalendářů jednomu uživateli, přidávání osob do seznamu stálých zákazníků, hodnocení služeb, systém slev pro stálé zákazníky, seznam svátků, upozornění na nadcházející události emailem, propojenost s externím kalendářem (např. Google kalendář) nebo vytvořit z našeho systému přednastavené šablony. Dále by bylo zajímavé vyzkoušet i jiné poskytovatele cloudových služeb a porovnat schopnost škálovatelnosti.

Literatura

[1] amazon.com: What Is Amazon Relational Database Service (Amazon RDS)? 2015 [cit. 2015-04-11].
URL http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html

[2] Anthony T. Velte, Toby J. Velte, Robert Elsenpeter: *Cloud computing Praktický průvodce*. Computer Press, a.s., 2011, ISBN 978-80-251-3333-0.

[3] appfog.com: Pricing. 2015 [cit. 2015-01-21].
URL https://www.appfog.com/pricing/

[4] ARLOW, J. a. I. N.: *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, a.s., 2007, ISBN 978-80-251-1503-9.

[5] azure.microsoft.com: Cloud Services. 2015 [cit. 2015-01-20].
URL http://azure.microsoft.com/en-us/services/cloud-services/

[6] azure.microsoft.com: Pricing. 2015 [cit. 2015-01-20].
URL http://azure.microsoft.com/en-us/pricing/details/cloud-services/

[7] cloud.cz: Cloud computing: Co ty pojmy znamenají? 2015 [cit. 2015-01-19].
URL http:
//www.cloud.cz/cloud/158-cloud-computingco-ty-pojmy-znamenaji.html

[8] cloud.google.com: Pricing. 2015 [cit. 2015-01-21].
URL https://cloud.google.com/pricing/#eu

[9] cloud.google.com: What is Google App Engine. 2015 [cit. 2015-01-21].
URL https://cloud.google.com/appengine/docs/whatisgoogleappengine

[10] engineyard.com: Features. 2015 [cit. 2015-01-20].
URL https://www.engineyard.com/features

[11] engineyard.com: Pricing. 2015 [cit. 2015-01-20].
URL https://www.engineyard.com/pricing

[12] Google.com: Smluvní podmínky společnosti Google. 2015 [cit. 2015-01-19].
URL http://www.google.com/apps/intl/cs/terms/user_terms.html

[13] Hanuš, P.: JavaScript: pravidla hry. 2015 [cit. 2015-04-11].
URL http://mat.fsv.cvut.cz/hanus/html/javascript.html

[14] heroku.com: Features. 2015 [cit. 2015-01-20].
URL https://www.heroku.com/features

[15] heroku.com: Pricing. 2015 [cit. 2015-01-20].
URL https://www.heroku.com/pricing

[16] Jonák, S.: Výhody Cloud computingu a překážky v jeho prosazení. 2014-02-19 [cit. 2015-01-19].
URL http://www.middleware.cz/cloud-computing/3-vyhody-cloud-computingu-a-prekazky-v-jeho-prosazeni

[17] Klímková, K.: PaaS. 2015 [cit. 2015-01-19].
URL http://wiki.knihovna.cz/index.php/PaaS

[18] Ľuboslav Lacko: *Osobní cloud pro domácí podnikání a malé firmy*. Computer Press, a.s., 2012, ISBN 978-80-251-3744-4.

[19] McGrath, M. P.: *Understanding PaaS*. O'Reilly, 2012 [cit. 2015-01-20], ISBN 978-1-449-32342-4.

[20] openshift.com: Pricing. 2015 [cit. 2015-01-20].
URL https://www.openshift.com/products/pricing

[21] openshift.com: Products. 2015 [cit. 2015-01-20].
URL https://www.openshift.com/products

[22] Petr Mácha: Cloud computing-historie a budoucnost. 2014-09-03 [cit. 2015-01-19].
URL http://www.ddconnect.cz/brezen-2012/datova-centra.html

[23] e rezervace.cz: Rezervační systém jako dobrý přítel a skvělý rádce. 2015 [cit. 2015-04-10].
URL http://www.e-rezervace.cz/rezervacni-system

[24] Smith, B.: Understanding Horizontal and Vertical Scaling. 2015 [cit. 2015-05-19].
URL http://blakesmith.me/2012/12/08/understanding-horizontal-and-vertical-scaling.html

[25] Sullivan, D.: PaaS Providers List: Comparison And Guide. 2015-01-31 [cit. 2015-01-28].
URL http://www.tomsitpro.com/articles/paas-providers,1-1517.html

[26] Wikipedia.org: Diverzifikace. 2015 [cit. 2015-01-19].
URL http://cs.wikipedia.org/wiki/Diverzifikace

[27] Zítko, J.: Co znamená Cloud computing? 2014-09-03 [cit. 2015-01-19].
URL http://google-apps.cz/co-je-cloud-computing-jeho-vyhody-a-nevyhody-2/

Příloha A

Obsah CD

1. images/ – Složka obsahující zdrojové soubory k vytvořeným obrázkům.
2. screenshots/ – Složka obsahující screenshoty z nastavování a testování
3. src/ – Složka obsahující zdrojové soubory.
4. tex/ – Složka obsahující zdrojové soubory bakalářské práce v LaTeXu.
5. tex/fig – Složka obsahující obrázky použité v bakalářské práci.

Příloha B

Zdrojové kódy

Zdrojové kódy jsou dostupné na přiloženém CD. Zdrojové kódy nalezneme ve složce src. U všech zdrojových souborů se v hlavičce nachází jméno autora s popisem souboru.

1. src/ – Složka obsahující zdrojové soubory.
2. src/css – Složka obsahující zdrojové soubory Kaskádového stylu.
3. src/images – Složka obsahující obrázky k webu.
4. src/js – Složka obsahující javascriptovou knihovnu jQuery.
5. src/jscolor – Složka obsahující zdrojové soubory k API JSColor.
6. src/web – Složka obsahující zdrojové soubory našeho webu.

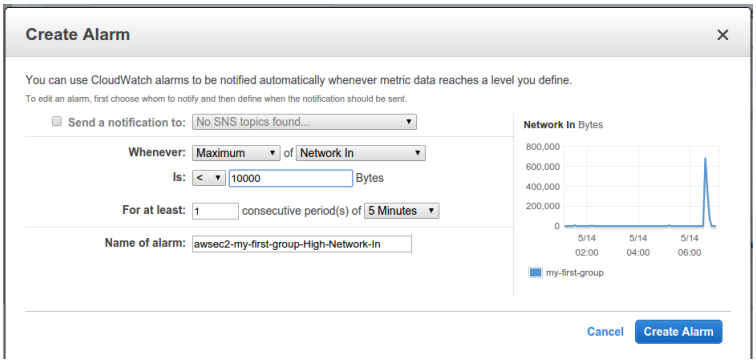
Příloha C

Spuštění rezervačního systému

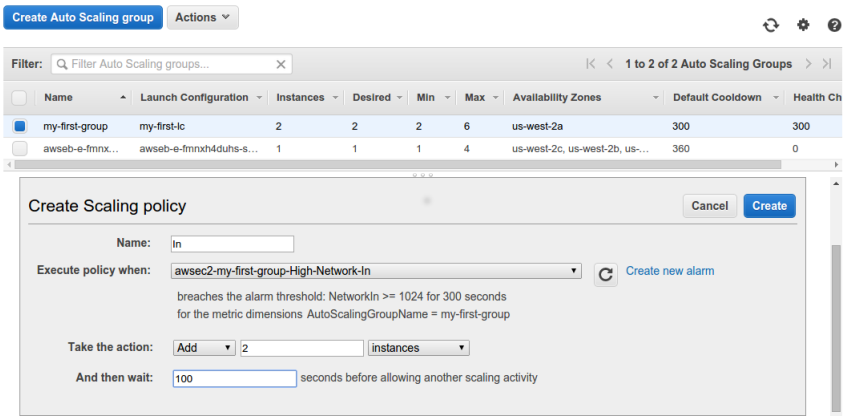
Aplikaci je možné spustit pod službou Amazon EC2 nebo v u jiného poskytovatele podporující jazyk PHP. Aplikace je dostupná na adrese: http://ec2-52-24-133-30.us-west-2.compute.amazonaws.com

Příloha D

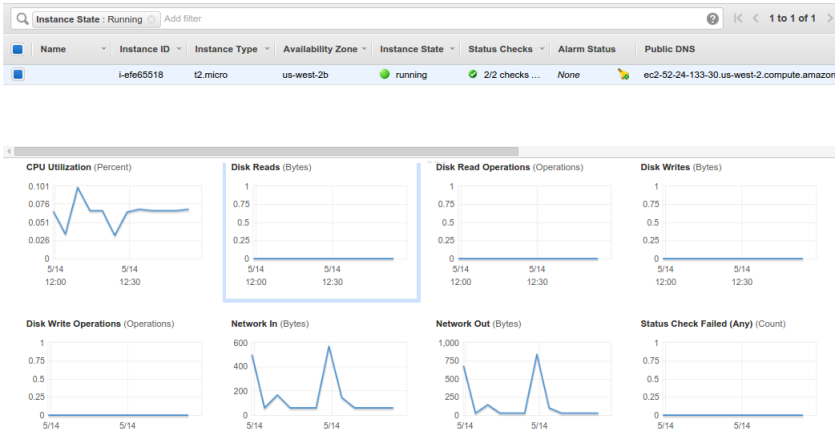
Testování pomocí Siege



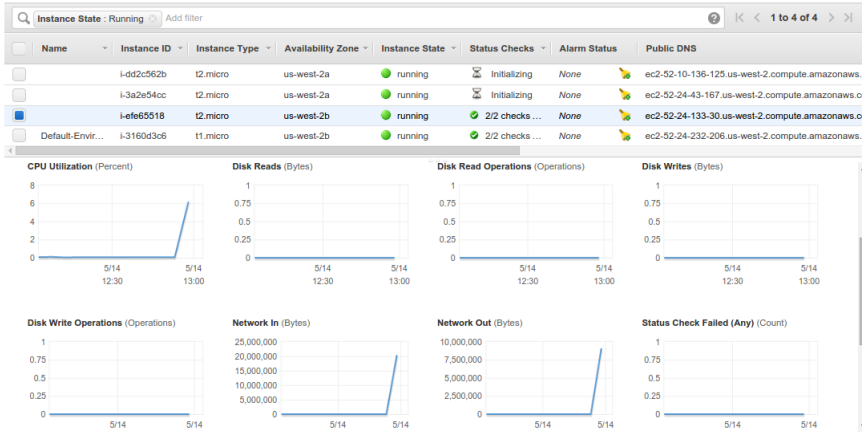
Obrázek D.1: Vytváření alarmu



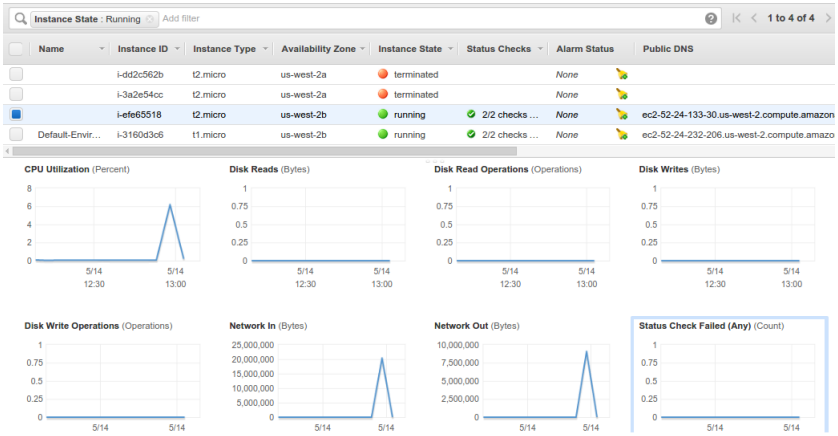
Obrázek D.2: Vytváření pravidla pro zvýšení počtu instancí



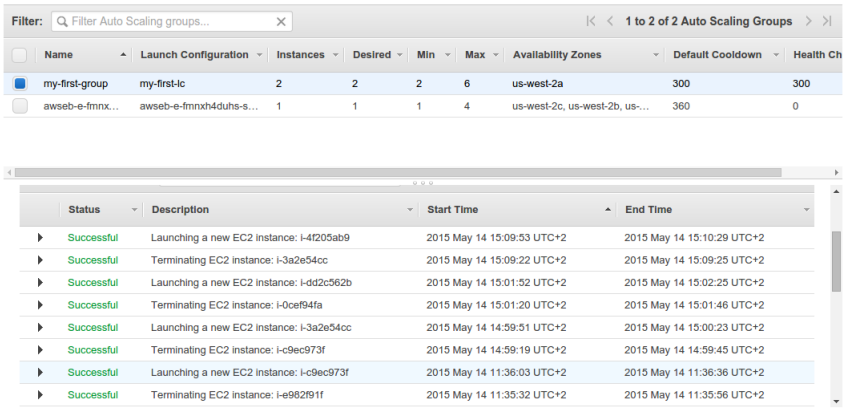
Obrázek D.3: Stav instancí před spuštěním testu



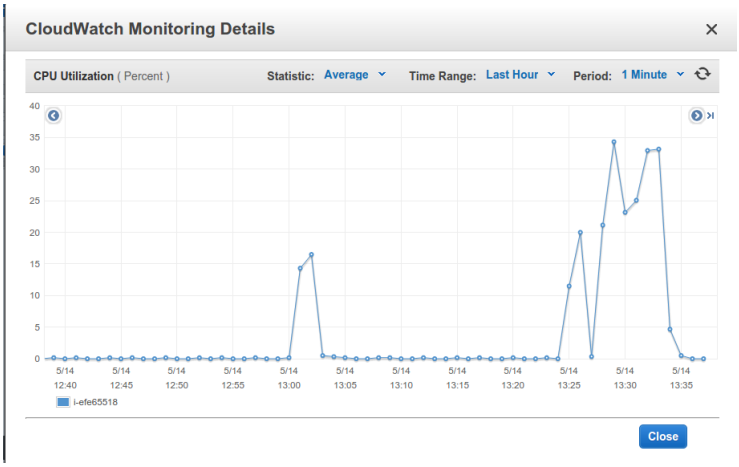
Obrázek D.4: Stav instancí během testu



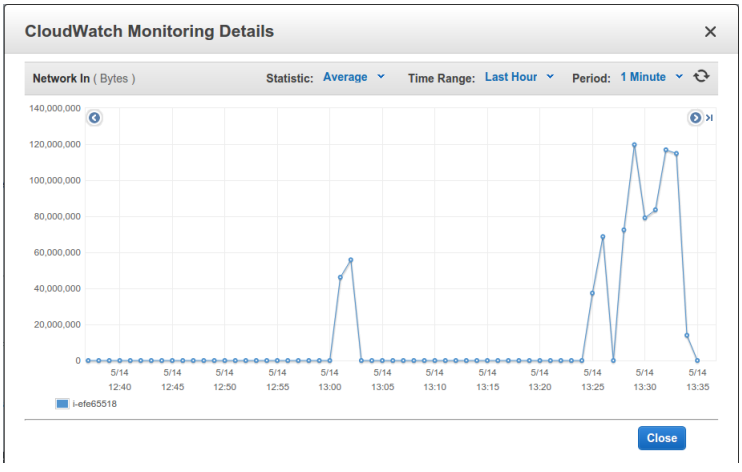
Obrázek D.5: Stav instancí po konci testu



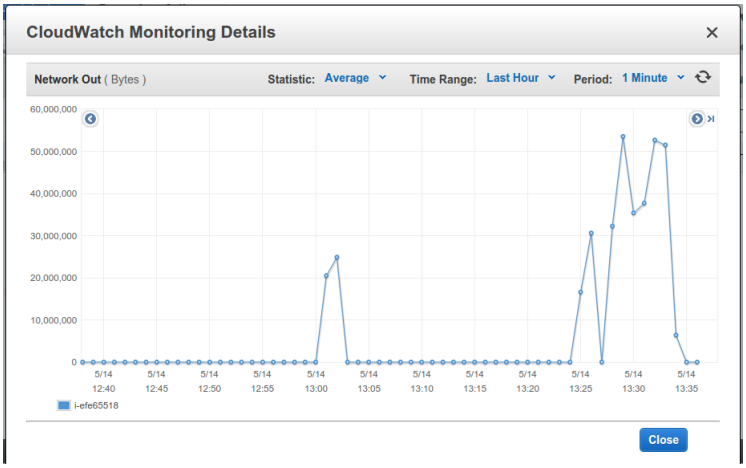
Obrázek D.6: Historie Auto Scaling group



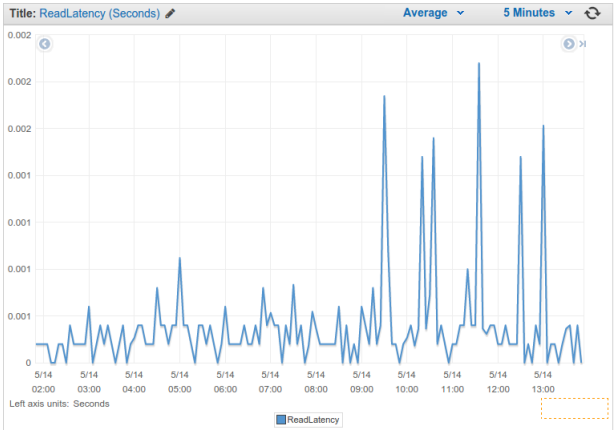
Obrázek D.7: Stav CPU při testu 600 uživatelů během 4 minut s přibližně 23 tisíci dotazy



Obrázek D.8: Počet příchozích bytů při testu 600 uživatelů během 4 minut s přibližně 23 tisíci dotazy



Obrázek D.9: Počet odchozích bytů při testu 600 uživatelů během 4 minut s přibližně 23 tisíci dotazy



Obrázek D.10: Latence databáze při testu 600 uživatelů během 4 minut s přibližně 23 tisíci dotazy

```
HTTP/1.1 200   2.45 secs:    3218 bytes ==> GET  /sluzby.php?id=5
HTTP/1.1 200   1.85 secs:    3218 bytes ==> GET  /sluzby.php?id=5
HTTP/1.1 200   1.87 secs:    3218 bytes ==> GET  /sluzby.php?id=5
HTTP/1.1 200   1.84 secs:    3218 bytes ==> GET  /sluzby.php?id=5
HTTP/1.1 200   1.89 secs:    3218 bytes ==> GET  /sluzby.php?id=5

Lifting the server siege...      done.

Transactions:                5605 hits
Availability:               99.89 %
Elapsed time:               59.82 secs
Data transferred:           17.21 MB
Response time:               1.58 secs
Transaction rate:           93.70 trans/sec
Throughput:                  0.29 MB/sec
Concurrency:               147.79
Successful transactions:     5605
Failed transactions:            6
Longest transaction:        12.22
Shortest transaction:        0.58
```

Obrázek D.11: Ukázka výstupu programu Siege