EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2014/2723991690

MONITORING AKO CLOUDOVÁ SLUŽBA

Diplomová práca

2014 **Bc. Kristína Krásna**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

MONITORING AKO CLOUDOVÁ SLUŽBA

Diplomová práca

Študijný program: Manažérske rozhodovanie a informačné technológie

Študijný odbor: 6258 Kvantitatívne metódy v ekonómii

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: Ing. Jaroslav Kultan, PhD

2014 **Bc. Kristína Krásna**

ZADANIE S ORIGINÁLNYMI PODPISMI ZODPOVEDNÝCH – školiteľa, vedúceho katedry a prodekanky pre štúdium

Čestné vyhlásenie

Vyhlasujem, že som predkladanú diplomovú prácu “Monitoring ako cloudová služba” spracovala samostatne a pod odborným vedením školiteľa diplomovej práce. Použitú literatúru uvádzam v zozname použitej literatúry a zároveň osvedčujem použité citáty. Potvrdzujem tiež, že elektronická forma predkladanej diplomovej práce je 100% identická s tlačenou formou.

V Bratislave dňa ….....

….....

Vlastnoručný podpis

ABSTRAKT

Bc. KRÁSNA, Kristína: Monitoring ako cloudová služba. [Diplomová práca] – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. - školiteľ: Ing. Jaroslav Kultan, PhD – Stupeň odbornej kvalifikácie: Inžinier. - Bratislava: FHI EUBA, 2011. Počet s. 79.

V danej práci študent musí preukázať svoje znalosti z oblasti databázových systémov, počítačových sietí, operačných systémov a softvérového inžinierstva. Súčasťou práce je analýza existujúcich monitorovacích systémov overujúcich dostupnosť internetových služieb. Systém by mal dokázať merať a vyhodnocovať dostupnosť vybraných internetových služieb pomocou jednoduchých stavov (dostupné/nedostupné) a latenciu. Zároveň by mal dokázať overovať dostupnosť služby a latenciu pomocou upraviteľného workflow. Namerané hodnoty by mal systém ukladať do databázy. Používateľským rozhraním daného systému má byť webová stránka.

Kľúčové slová: Monitoring, cloud computing, dostupnosť, softvér ako služba

ABSTRACT

Bc. KRÁSNA, Kristína: Monitoring as a cloud service [Graduation Thesis] / Kristína Krásna. - University of Economics Bratislava. Faculty of Business Informatics – Economic Informatics; Department of Applied Informatics - supervisor Ing. Jaroslav Kultán PhD. Degree of professional qualification: Master. - Bratislava: FHI EUBA, 2011. 79 p.

In the present work the student must demonstrate his knowledge of database systems, computer networks, operating systems and software engineering. Part of this work is to analyze existing monitoring systems verifying the availability of Internet services. The system should be able to measure and evaluate the availability of certain web services using simple states (available / unavailable) and latency. It should prove to verify availability Services and latency using a customizable workflow. Results should be stored in the system database. User interface of the system should be a website.

Keywords: monitoring, cloud computing, availability, software as a service

PREDHOVOR

V rámci tejto diplomovej práce som mala možnosť vytvoriť systém na monitorovanie dostupnosti webových služieb. Práve to ma viedlo k výberu danej témy – mať možnosť vytvoriť niečo konkrétne, čo sa dá využiť v praxi a čo by mohlo byť prínosom pre túto spoločnosť.

“Vedecká práca je našou jedinou cestou k poznaniu okolitej reality.”

– *Siegmund Freud*

Touto cestou by som sa tiež rada poďakovala vedúcemu záverečnej práce Dr. Ing. Jaroslavovi Kultanovi, PhD za jeho odbornú pomoc, obetovaný čas a cenné rady, ktoré mi poskytol pri jej vypracovaní.

Motto :

“Monitoring pripravený na prevádzku skôr, než narátate do troch”

“Monitoring ready until you count to three”

Diplomová práca má 99076 znakov.

Obsah

ZOZNAM ILUSTRÁCIÍ

ZOZNAM TABULIEK

Slovník Pojmov

CRM	Customer Relationship Management	Manažérstvo vzťahov so zákazníkmi
SLA	Service level agreement	Zmluva o úrovni poskytovania služieb
MTA	Mail transfer agent	Softvér zabezpečujúci doručovanie emailov
SMTP	Simple Mail Transfer Protocol	Jednoduchý emailový protokol
TCP	Transmission Control Protocol	Kontrolný prevodný protokol
ICMP	Internet Control Message Protocol	Internetový protokol pre kontrolné hlášky
HTTP	Hypertext Transfer Protocol	Protokol definujúci požiadavky a odpovede medzi klientmi a servermi
IM	Instant Messaging	Okamžité správy
NAS	Network-attached storage	Sieťový dátový sklad
DRBD	Distributed Replicated Block Device	Distribuované replikované blokové zariadenie
NFS	Network File System	Sieťový súborový systém
SPOF	Single point of failure	Citlivá časť systému, na ktorej môže systém zlyhať
IPMI	Intelligent Platform Management Interface	Inteligentný modul riadenia platformy
JVM	Java Virtual Machine	Java virtuálny stroj
VPN	Virtual Private Network	Virtuálna uzavretá sieť
ATM	Asynchronous Transfer Mode	Asynchrónny prenosový režim
SNMP	Simple Network Management Protocol	Protokol jednoduchého sieťového riadenia

Úvod

V dnešnej dobe čím ďalej, tým viac firiem je závislých na webových službách poskytovaných tretími stranami. Medzi takéto služby môžeme zaradiť napríklad emaily a groupware, účtovnícky či fakturačný softvér, webhosting, hostované CRM alebo cloudový servicedesk.

Interné procesy firmy alebo výrobného podniku sú často závislé na jednej alebo viacerých takýchto služieb naraz a pri výpadku jednej z nich môže dojsť k obchodnej strate a zníženiu kredibility v očiach zákazníkov. Z tohto dôvodu sa spoločnosti pri rozhodovaní sa o prechode z tradičného modelu na službu poskytovanú v cloude zameriavajú na poskytovateľov zaručujúcich sa dodržiavať určité parametre kvality služby (definované v SLA). Mnohokrát však chýba akákoľvek externá kontrola a to, či služba poskytovateľa spĺňa v zmluve zadefinované podmienky nie je nijako externe overované a papier znesie všetko. Zmluva síce zvykne obsahovať aj penále v prípade prípade nedodržania dohodnutej úrovne služieb, ale pokiaľ chýba akákoľvek externá kontrola, klient sa tým oberá o možnosť vyžadovať penále a zároveň stráca prehľad o skutočnej kvalite poskytovania danej služby.

Štatistiky uverejňované poskytovateľom nie sú nestranné a nedá sa na ne spoľahnúť. Práve v tomto prípade si spoločnosti, ktoré sú klientmi cloudových služieb nechávajú merať dostupnosť daných služieb externým poskytovateľom monitoringu. Máloktorý z existujúcich monitoringov poskytuje možnosť komplexného merania mimo základné dva stavy dostupné – nedostupné. Samotná možnosť napr. vytvoriť TCP spojenie k serveru alebo fakt, že sa na monitorovanej webovej službe zobrazuje určitý textový reťazec ešte neznamená, že je služba aj reálne funkčná a teda pre klienta dostupná.

Peniaze a dostupnosť, to sú slová, ktoré v technických oboroch v poslednej dobe počujeme čoraz častejšie. Už dávno nežijeme v dobe, kedy by výpadok webovej služby nemal veľký vplyv na chod firmy či nadnárodnej korporácie. Každá minúta odstávky služby alebo kompletného výpadku môže stáť firmu nemalú stratu, často až v desiatkach či stovkách tisíc eur. Ak si napríklad predstavíme komplexný internetový obchod zabehnutej firmy, môže sa obrat v danom mesiaci pohybovať rádovo v stovkách tisíc eur, kedy každá neprijatá objednávka alebo stratený zákazník pri výpadku webhostingu môže firmu poškodiť a znamenať stratu. Takáto strata sa nemusí prejaviť hneď, ale až s postupom času. [1]

Túto tému som si vybrala z dôvodu jej komplexnosti a prepojenia poznatkov nadobudnutých zo štúdia z viacerých predmetov ako sú počítačové siete, správa databáz, programovanie a operačné

systémy. Keďže som pri príležitosti bakalárskej práce už nadobudla skúsenosti s vytvorením jednoduchej webovej aplikácie, rozhodla som sa prijať túto výzvu a vytvoriť niečo väčšie, robustnejšie a zároveň si vyskúšať programovanie inej funkcionality. V tejto práci sa zameriavam najmä na maximalizáciu dosiahnuteľnej dostupnosti a analýzu reálnych možností ako ju dosiahnuť.

Dôležitým aspektom cloudových služieb je tiež ochrana dát, scenáre v prípade postihnutia poskytovateľa živelnou katastrofou, zálohovanie,dôvera voči poskytovateľovi cloudovej služby a vysoká dostupnosť. Preto by malo byť v záujme každého poskytovateľa zachovať si dobré meno a nespoliehať sa len na jedno dátové centrum. Na dosiahnutie vysokej dostupnosti služby je teda optimálne využiť aj geografickú redundancia umiestnenia infraštruktúry, výberom nezávislých poskytovateľov pripojenia do siete Internet alebo poskytovateľov serverovej infraštruktúry a ich vzájomnému prepojeniu do jedného logického celku schopného vysporiadať sa aj s výpadkom na úrovni celého dátového centra alebo poskytovateľa. Pri jeho výbere je tiež dôležité zamyslieť sa nad tým, čo všetko je nutné monitorovať a akým spôsobom je možné výpadky webovej služby odhaliť.

1 Analýza súčasného stavu

V tejto časti sa venujem objasneniu pojmov monitoring a cloud computing a analýzou existujúcich riešení. Keďže poskytovateľov a produktov je nepreberné množstvo, vybrala som niekoľko najznámejších, s ktorými som už nadobudla osobnú skúsenosť.

1.1 Monitoring

Monitoring je termín zastrešujúci všetky typy priameho systematického zaznamenávania (logovania), pozorovania alebo sledovania procesu alebo postupu pomocou technických pomôcok alebo iných monitorovacích systémov. Opakované pravidelné vykonávanie je jeho kľúčovým prvkom s cieľom vyvodiť závery na základe porovnania výsledkov.

Jedna z funkcií monitorovania je zasiahnuť do riadenia na základe overovanej sekvencie alebo procesu, za predpokladu, že nebude dosahovať požadované parametre alebo určitú prahovú hodnotu. Monitoring je teda zvláštny typ protokolovania.

Na základe predmetu pozorovania monitorovaním možno rozlišovať nasledujúce 2 typy monitorovania [2]:

- **Monitoring systémových komponentov** - Tento typ monitoringu skúma vyťaženosť procesorov, obsadenosť pevných diskov, obsadenosť operačnej pamäte, prietoky sieťových rozhraní, počet aktívnych/pripojených používateľov.
- **Monitoring externej dostupnosti systému** – monitoring systému zvonka, testovaním interakcie s jednotlivými službami (HTTP, HTTPS, TCP, ICMP echo request a echo reply). Pre účel tejto práce sa budem zaoberať práve týmto typom monitoringu.

1.1.1 Popis existujúcich monitorovacích systémov externej dostupnosti

Tieto monitorovacie systémy skúmajú dostupnosť webových služieb zvonka, v závislosti od spôsobu, akým sú dodávané či poskytované

Lokálne inštalované systémy

Väčšina monitorovacích systémov je inštalovaná lokálne. Takéto systémy samé o sebe vyžadujú konštantnú starostlivosť (aktualizácie) a aspoň základnú kontrolu funkcionality. Na takéto riešenie je nevyhnutný dedikovaný hardvér a operačný systém, na ktorom je monitorovací systém nasadený.

Silné stránky: legitimita dát bezpečnosť dát flexibilita	**Slabé stránky:** nutnosť spravovania komplikovaná prvotná inštalácia potreba vlastného hardvéru
Príležitosti: možnosť doprogramovať funkcionalitu	**Hrozby:** hrozba výpadku monitorovacieho systému

Tabuľka 1: SWOT analýza v prípade lokálne inštalovaného riešenia

Nagios

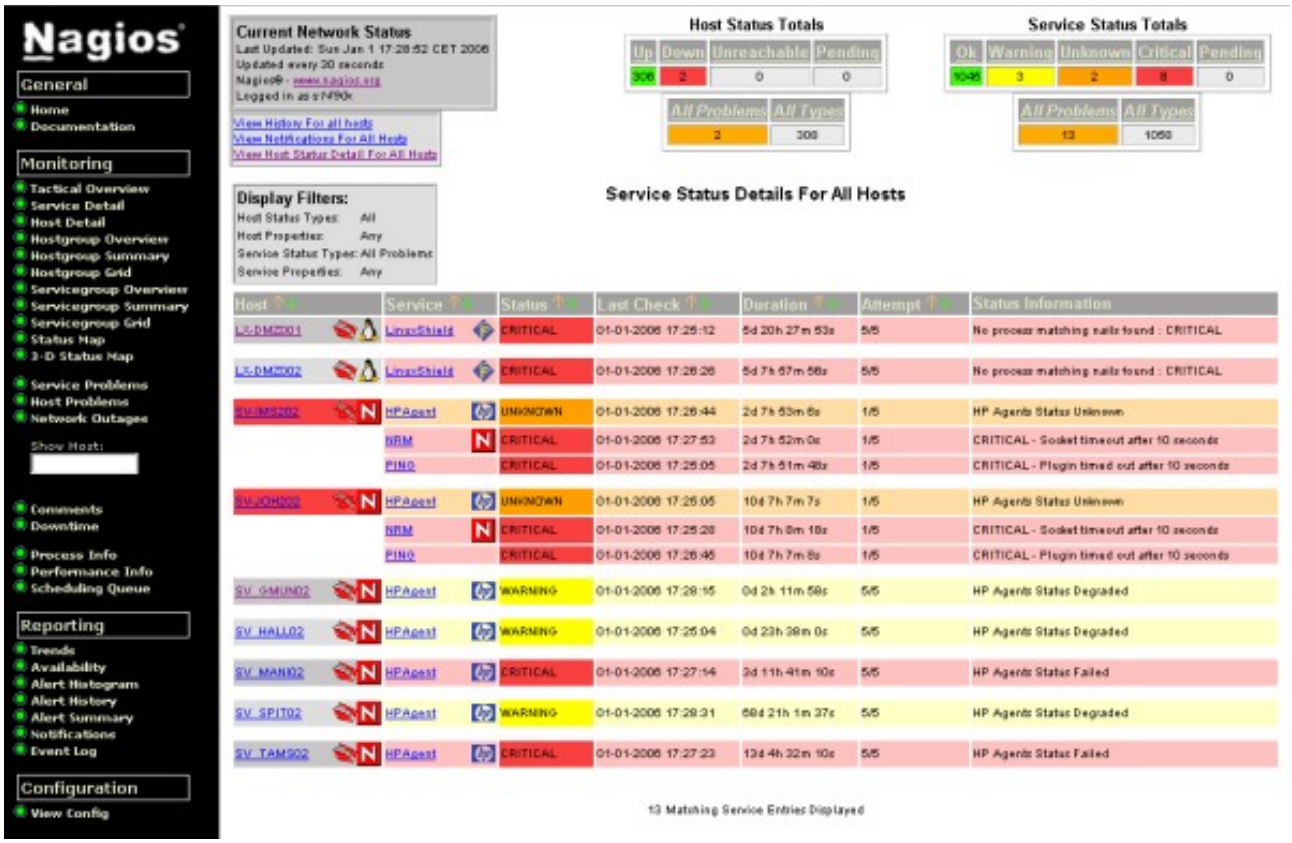
Nagios je monitorovací nástroj navrhnutý na okamžitú notifikáciu administrátora o problémoch so sieťou. Zväčša sa to deje ešte pred tým, ako to pocítia koncoví používatelia. Je určený na beh v OS Linux a rôznych variantoch OS Unix. Monitorovacia služba spúšťa nespojité kontroly špecifikovaných koncových staníc a služieb, použitím externých modulov, ktoré vracajú výsledky kontrol hlavnému modulu Nagiosu. Keď sú zistené problémy, služba je schopná poslať upozornenie na preddefinované kontakty pomocou rôznych typov komunikácie (email, SMS, alebo IM). Aktuálny stav, historické záznamy a reporty sú prístupné cez webové rozhranie. [10]

Výhody:

- monitoruje Windows, Unix aj Linux
- monitoruje ľubovoľné typy služieb
- je dobre testovateľný vo veľmi zložitých scenároch
- pružne prispôsobiteľný a rozšíriteľný
- má jasne usporiadané rozhranie prehliadača

Nevýhody:

- vyžaduje určený server s veľkým množstvom pamäte
- konfigurácia je manuálny proces
- nie je to jednoduchý softvérový balík na konfigurovanie
- má jednoduchú inštaláciu, ale vyžaduje si veľa času
- neexistuje žiadna dobrá metóda založená na webových stránkach pre vytváranie monitoringu



Obrázok 1: Monitorovací systém Nagios [2]

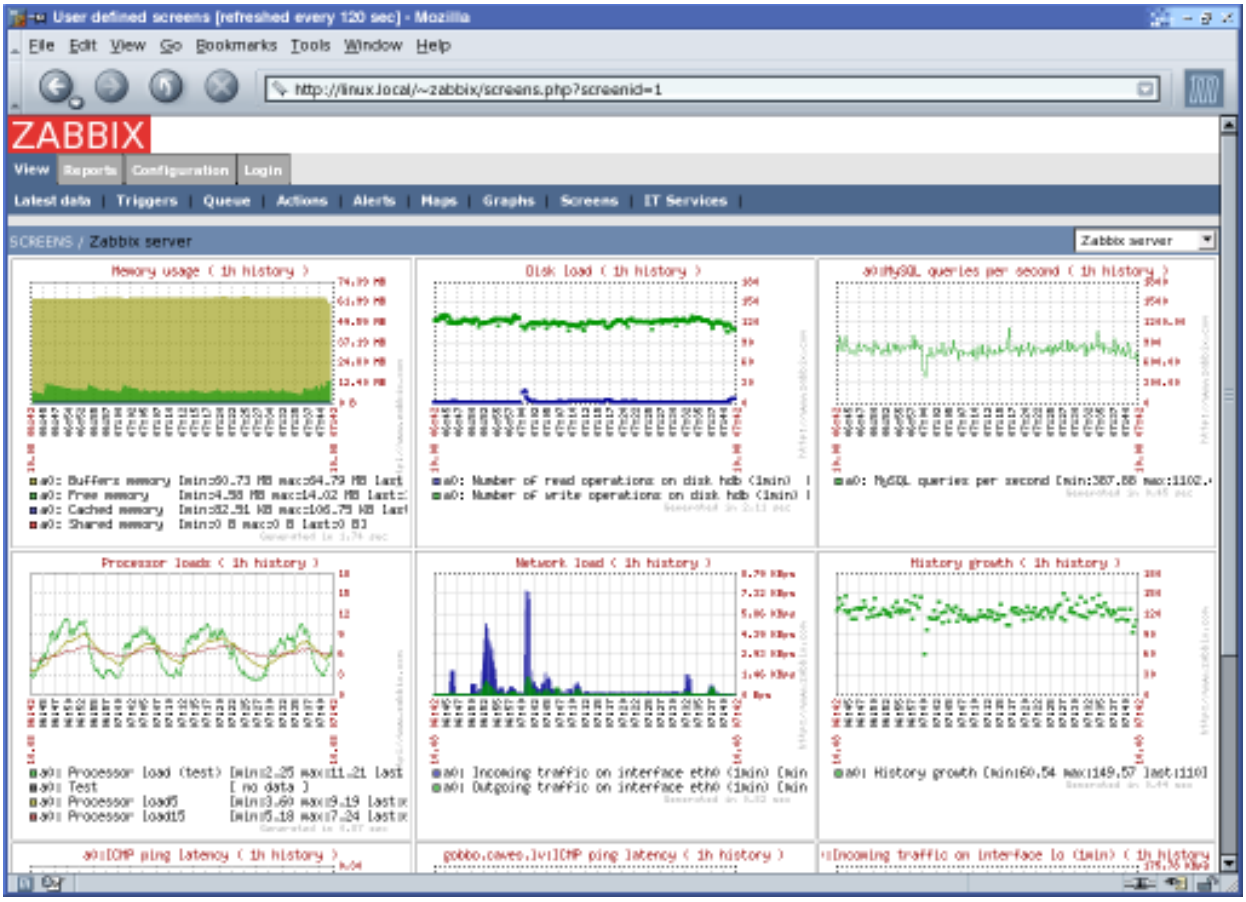
Zdroj: https://www.novell.com/coolsolutions/feature/16723.html

Zabbix

Zabbix je riešenie pre monitorovanie podnikových sietí a aplikácií, vytvorený Alexejom Vladishevom. Je určený na sledovanie a zaznamenávanie stavu rôznych sieťových služieb, serverov, a ďalšieho sieťového hardvéru. Zabbix používa MySQL, PostgreSQL, SQLite, Oracle alebo IBM DB2 pre ukladanie dát. Jeho backend je napísaný v C a web rozhranie je napísané v PHP. Zabbix ponúka niekoľko možností sledovania:

- Jednoduchými kontrolami je možné overiť dostupnosť a odozvu štandardných služieb, ako je SMTP alebo HTTP bez nutnosti inštalácie akéhokoľvek softvéru na sledovanom počítači. Zabbix agent môže byť inštalovaný na unixových aj windowsových platformách na monitorovanie štatistík ako napríklad vyťaženie procesora, využívanie kapacity siete či pevných diskov.
- Ako alternatívu k inštalácii agenta na počítačoch, Zabbix zahŕňa podporu pre monitorovanie cez SNMP, TCP a ICMP kontroly, rovnako ako nad IPMI, JMX, SSH, telnet a pomocou vlastných parametrov. Zabbix podporuje celý rad notifikačných mechanizmov v reálnom čase.

Zabbix je vydaný v súlade s licenciou GNU/GPL verzie 2, takže sa radí medzi slobodný softvér s otvoreným zdrojovým kódom.



Obrázok 2: Monitorovací systém Zabbix

Zdroj: http://en.wikipedia.org/wiki/Zabbix

Výhody:

- jednoduchá inštalácia softvéru aj agenta
- konfigurácia v databáze an nie v súboroch
- jednoduchá konfigurácia cez webové rozhranie
- zo všetkých výsledkov automaticky generuje grafy

Nevýhody:

- nepodporuje detekciu prirýchlo sa meniaceho stavu
- v porovnaní s Nagiosom nie je možné vytvárať tak flexibilné checky

Monitoring ako služba

Má nižšie, prípadne skoro žiadne iniciálne náklady ani nevyžaduje toľkú starostlivosť, keďže zodpovednosť zaň a starostlivosť o jeho funkcionalitu nesie poskytovateľ danej monitorovacej služby. Po zriadení je takáto služba pre podnik takpovediac bezúdržbová. Ani toto riešenie však nie je dokonalé, nakoľko monitorovací systém poskytovateľa musí mať prístup do citlivých častí siete za firewallom. Toto však nie je technický problém (VPN), skôr je to otázka dôvery voči poskytovateľovi.

Silné stránky: komfort používania komfort zriadenia minimálna počiatočná investícia	**Slabé stránky:** strata kontroly obmedzené možnosti použitia nutná dôvera voči poskytovateľovi
Príležitosti: možnosť flexibilného rozšírenia/zúženia poskytovanej služby	**Hrozby:** nedodržanie SLA zo strany poskytovateľa

Tabuľka 2: SWOT analýza v prípade monitorovania pomocou externej služby

Pingdom

Pingdom je monitorovacia služba, ktorá sleduje funkčnosť alebo nefunkčnosť a výkon webových stránok. Založený bol vo Švédsku, dnes monitoruje webové stránky z viacerých geografických miest, takže je schopný rozlíšiť skutočnú nedostupnosť od problémov so smerovaním či konektivitou. Táto služba poskytuje notifikáciu aj prostredníctvom SMS a tiež štatistiky týkajúce sa nielen dostupnosti danej webovej služby, ale aj štatistiky súvisiace s jej návštevnosťou a štruktúrou návštevníkov.

Cena: 11.87 až 362.08 v závislosti od počtu testov a počtu používateľov [3]

Monitoring-serverov

Táto monitorovacia služba ponúka testovanie HTTP (head), HTTPS (get), FTP, FTP-SSL, POP3, IMAP, IMAPs, SMTP, SMTP (+EHLO), SMTPs, SMTPs (+EHLO), MySQL, PostgreSQL, Ping, LDAP, LDAPs, DNS server, Telnet (beta test) v zvolenom intervale každých 15 sekúnd až každých 10 minút. Systém podporuje emailovú aj sms notifikáciu a umožňuje zobrazovanie štatistík vykonaných testov.

Cena: 0,6 – 14 eur mesačne za 1 druh testu v závislosti od intervalu testovania [4]

1.2 Cloudové Služby

Cloud computing je poskytovanie výpočtovej techniky v tzv. cloude, v ktorom sú väčšie skupiny vzdialených serverov zosieťované tak, aby umožnili centralizované uloženie dát a online prístup k počítačovým službám či zdrojom. Cloudy môžu byť klasifikované ako verejné, súkromné a hybridné.

Kritika tohto spôsobu poskytovania výpočtových služieb je smerovaná najmä na jeho sociálnu implikáciu, pokiaľ je majiteľ vzdialených serverov iný ako používateľ, keď dochádza k rozpolteniu záujmov, napríklad ak používateľ (klient) si želá uchovávať spracúvané informácie ako súkromné, ale majitel vzdialených serverov môže naopak chcieť využiť tieto informácie na vlastnú podnikateľskú činnosť.

Cloud computing je založený na zdielaní výpočtových zdrojov pre dosiahnutie súdržnosti a úspory z rozsahu cez sieť. Pri tvorbe cloud computingu je širší koncept konvergovanej infraštruktúry a zdieľaných služieb.

Cloud computing, alebo v jednoduchšom skrátene len "cloud", sa tiež zameriava na maximalizáciu efektivity zdieľaných prostriedkov. Cloud prostriedky zvyčajne nie sú len fixne rozdelené medzi viacerými používateľmi, ale sú aj dynamicky prerozdelované na vyžiadanie. To môže fungovať pri prideľovaní zdrojov používateľom. Napríklad, cloudové zariadenia, ktoré slúžia európskym používateľom v európskych denných hodinách pre konkrétnu aplikáciu (napr. e-mail), môžu pridelovať totožné prostriedky, aby slúžili používateľom v Severnej Amerike počas tamojšej pracovnej doby s inou aplikáciou (napríklad webový server). Tento prístup by mal maximalizovať využitie výpočtového výkonu a tým znížiť škody na životnom prostredí, šetriť spotrebu energie, znižovať používanie klimatizácie, potreby rackového priestoru atď a to aj pri širokej škále funkcionality. Takto môže veľa používateľov pristupovať na jediný server a načítať či aktualizovať svoje dáta bez nutnosti nákupu vlastných licencií a inštalácie rôznych aplikácií.

Zástancovia cloudových služieb tvrdia, že cloud umožňuje spoločnostiam ušetriť vysoké počiatočné náklady a zamerať sa tak viac na svoje projekty a neriešiť popri tom náročnú úlohu zabezpečenia vlastnej podpornej infraštruktúry. Zástancovia tiež tvrdia, že cloud umožňuje spoločnostiam spustiť ich projekty podstatne rýchlejšie, s lepšou manažovateľnosťou, jednoduchšou údržbou a umožňuje takto IT oddeleniu podstatne rýchlejšie reagovať na kolísajúcu potrebu výpočtového výkonu a nepredvídateľné udalosti.

Poskytovatelia cloudovej infraštruktúry väčšinou používajú model „plať ako používaš“, akokoľvek je výhodný pri správnom nastavení služby, môže tento model spôsobiť aj obrovské náklady, ak sa administrátori nedostatočne pripravili na prechod na úplne iný (elastický) systém využívania IT infraštruktúry.

Aktuálna dostupnosť vysokokapacitných sietí, lacných výpočtových serverov a úložných riešení a taktiež veľké rozšírenie hardvérovej virtualizácie a servisne orientovanej architektúry viedlo k nezanedbateľnému nárastu využívania cloudových služieb. Poskytovatelia cloudu bežne dosahujú rast na úrovni 50% ročne. [5]

1.2.1 Pôvod pomenovania Cloud Computing

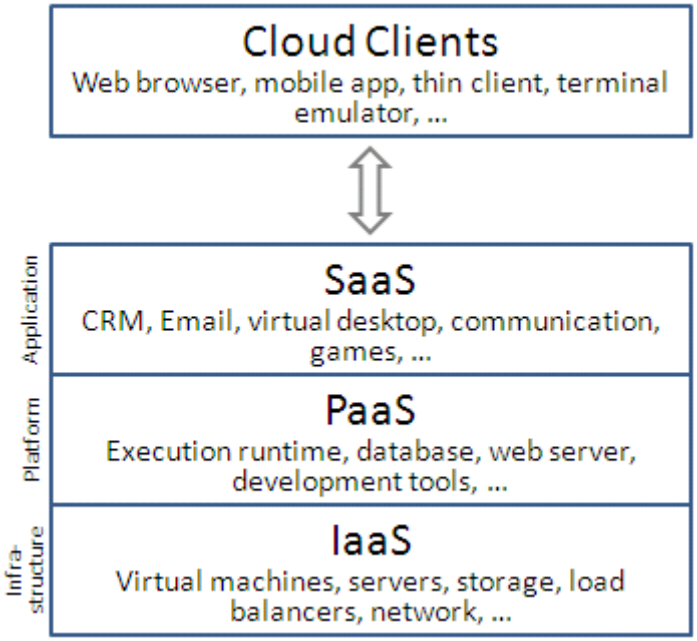
Pôvod pomenovania Cloud Computing nie je úplne jasný. Tento výraz sa bežne používal vo vedných odboroch ako substitúcia veľkého množstva objektov ktoré opticky z diaľky pôsobia ako oblak a ich konkrétne vlastnosti nebolo v danom momente nutné bližšie špecifikovať pre daný kontext.

Analogicky sa tento termín používal ako metafora reprezentujúca sieť internet a taktiež sa uchytil piktogram obláčikového tvaru. Takýto obláčik sa často používal na schémach v telekomunikáciách a neskôr v sieťových diagramoch na znázorňovanie veľkej nie úplne známej siete, najčastejšie siete Internet. Toto zjednodušenie v podstatne vyjadrovalo že nie je relevatné poznať vnútornú štruktúru siete pokiaľ sú známe koncové body a funkčnosť samotnej siete v zmysle čiernej skrinky. Prvá známa reprezentácie siete Internet zaznačená ako obláčik do ktorého sú pripojené servre je známa z roku 1994 na dokumentoch IBM.

Použitie výrazu cloud computing v dnešnom zmysle sa objavilo v roku 1996 na interných dokumentoch spoločnosti Compaq. O najvačšiu popularizáciu termínu cloud computing sa však postarala v roku 2006 spoločnosť Amazon.com predstavením služby Elastic Compute Cloud. [5]

1.2.2 Delenie Cloudových Služieb podľa druhu poskytovanej služby

V závislosti od rozsahu poskytovaných cloudových komponentov môžeme cloudové služby rozdeliť do niekoľkých skupín.



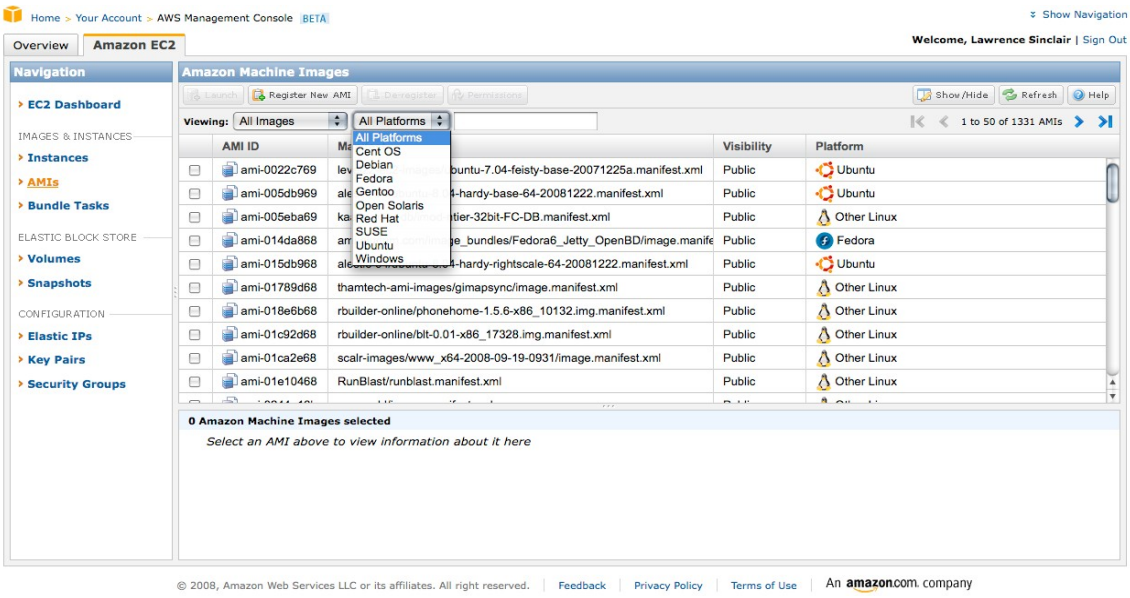
Obrázok 3: Rozdelenie cloudových služieb

Zdroj: http://en.wikipedia.org/wiki/Cloud_computing

IaaS – Infraštruktúra ako služba

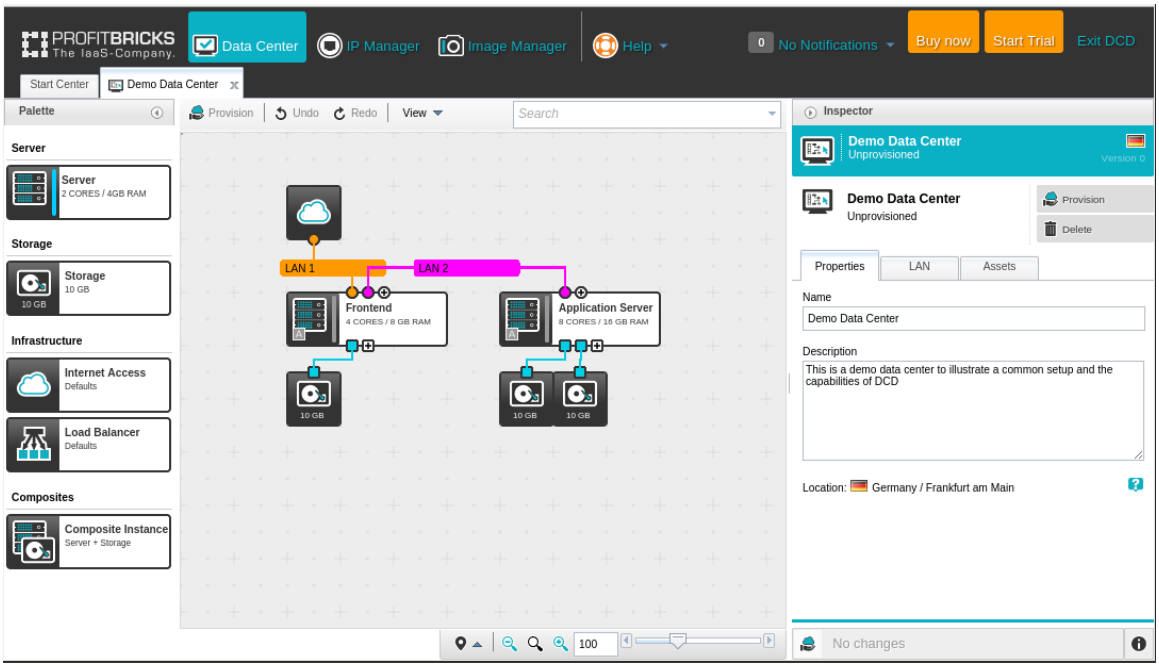
Infraštruktúra ako služba reprezentuje najnižšiu vrstvu cloudových služieb podľa modelu IETF (Internet Engineering Task Force). Poskytovatelia IaaS ponúkajú fyzické, alebo častejšie virtuálne servre a ostatné prostriedky (napríklad virtuálne aktívne prvky počítačovej siete). Pri poskytovaní virtuálnych serverov sa na tento účel používa hypervisor, (najčastejšie Xen, KVM, VirtualBox, Vmware ESX/ESXi, prípadne Microsoft Hyper-V) ten sa stará o beh jednotlivých virtuálnych strojov a zabezpečuje ich prístup k fyzickým prostriedkom servera (ako napríklad CPU, RAM, sieťová karta, …). Najčastejšie prevádzkujú poskytovatelia veľké množstvo fyzických serverov/hypervízorov a úložných riešení, prostriedky ktorých potom môžu efektívne rozdeľovať medzi veľké množstvo klientov s elastickými parametrami služby, možnosťou vo veľmi krátkom čase (rádovo minúty) zapínať a vypínať tisícky serverových inštancií a ostatných poskytovaných prostriedkov. Tento model je najbližšie pôvodnej prevádzke, nakoľko v podstate umožňuje premigrovať kompletnú existujúcu infraštruktúru do virtuálneho dátového centra. Poskytovatelia IaaS najčastejšie spoplatňujú svoje služby na základe reálne využitých prostriedkov (CPU čas, RAM, disk, prenesené dáta, …) a cena tohto riešenia v prípade správne navrhnutej aplikácie môže tiež odzrkadľovať reálne využité prostriedky (automatická migrácia prostriedkov, vypínanie/zapínanie nod podľa potreby, ...).

Príklad IaaS: Spoločnosti Profitbricks a Amazon sú jednými z mnohých poskytovateľov plnohodnotných virtuálnych dátových centier.



Obrázok 4: Ukážka ovládacieho panelu IaaS ponuky spoločnosti Amazon

Zdroj: http://coenraets.org/blog/2011/11/set-up-an-amazon-ec2-instance-with-tomcat-and-mysql-5-minutes-tutorial/



Obrázok 5: Ukážka ovládacieho panelu IaaS služby spoločnosti Profitbricks

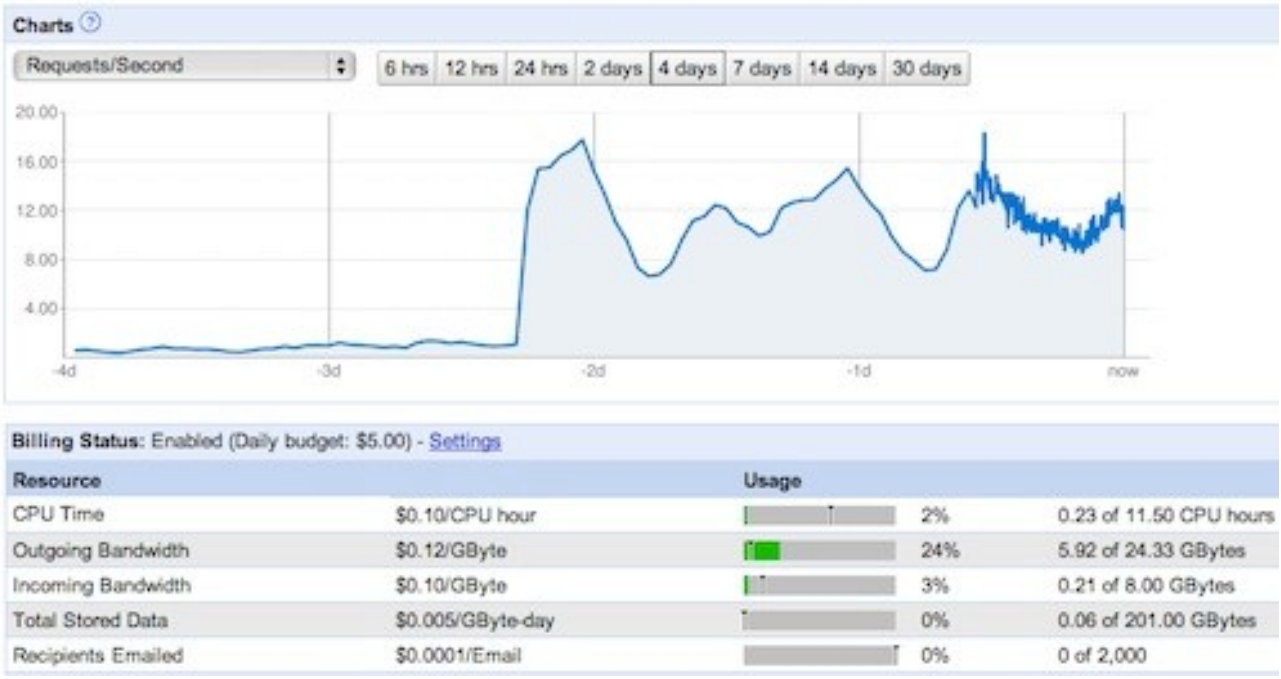
Zdroj: http://www.marketwired.com/press-release/profitbricks-launches-new-cloud-data-center-management-tool-1969459.htm

PaaS – Platforma ako Služba

Ako pomenovanie už napovedá, pri PaaS sa poskytuje už hotová platforma, na ktorej si môžu klienti napríklad spustiť svoju aplikáciu bez toho aby sa museli starať o middleware ako napríklad webserver, databazový server, či runtime environemnt programovacieho jazyka. Táto forma zásadne znižuje komplexicitu s ktorou sa musia programátori/firmy stýkať a umožňuje im sa ešte viac sústrediť na svoju skutočnú prácu. Aktuálne niektoré PaaS produkty na trhu (Napríklad Google App Engine, ale dokonca aj Microsoft Azure) poskytujú nie len samotnú platofrmu, ale dokážu dokonca automaticky a dynamicky škálovať využitie dostupných prostriedkov podľa momentálneho dopytu, čiže klient sa nemusí starať o alokáciu práve potrebného množstva prostriedkov sám.

PaaS je v podstate medzimodel medzi IaaS a SaaS, poskytuje infraštruktúru, operačný systém, middleware pod kontrolou poskytovateľa, ale naďalej si klient spúšťa vlastný softvér.

Príklad: Spoločnosť Google a produkt app engine a spoločnosť Microsoft s produktom Azure sú poskytovatelia elastických platforiem ako služba.



Obrázok 6: Ukážka ovládacieho panelu platformy ako služby spoločnosti Google

Zdroj: http://gamesfromwithin.com/life-in-the-top-100-and-the-google-app-engine

Obrázok 7: Ukážka ovládacieho panelu platformy ako služby spoločnosti Microsoft

Zdroj: http://blogs.msdn.com/b/briankel/archive/2014/04/10/building-your-dream-devops-dashboard-with-the-new-azure-preview-portal.aspx

SaaS – softvér ako služba

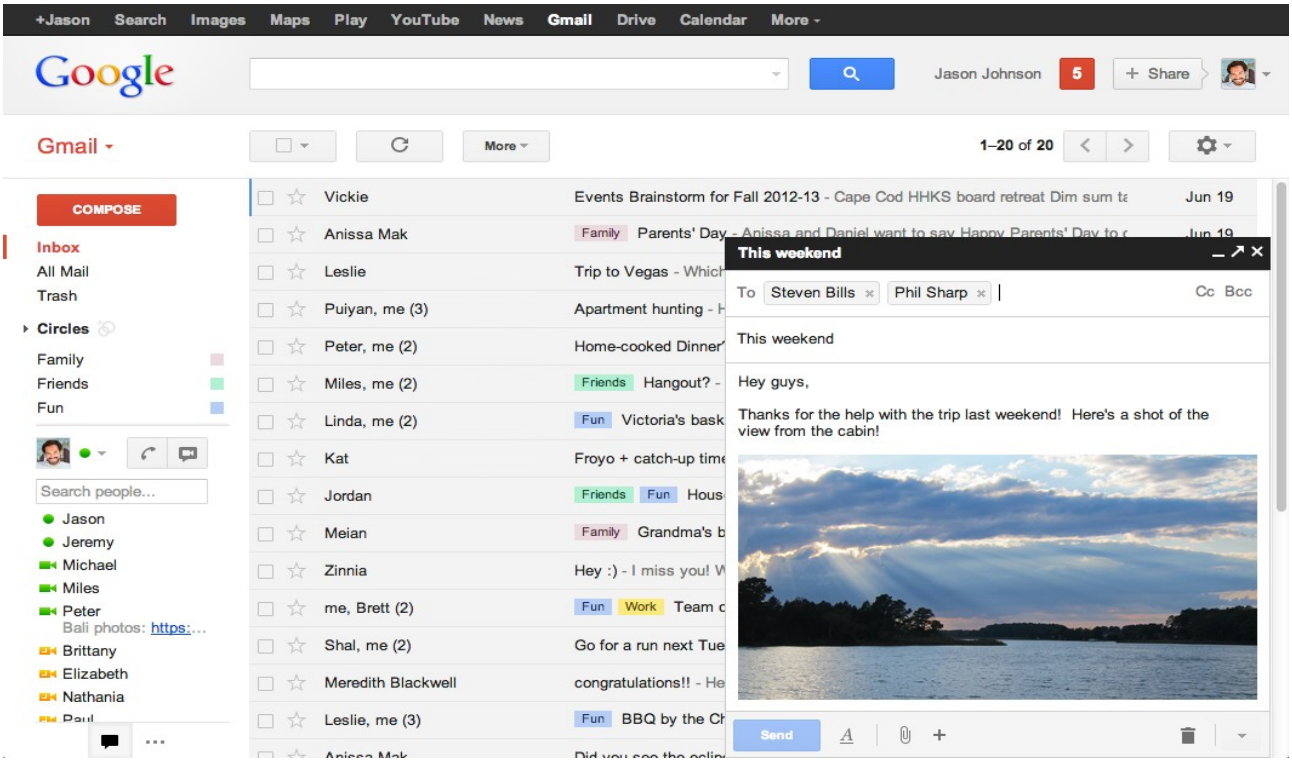
Pri obchodnom modeli softvér ako služba je používateľom väčšinou poskytnutý prístup až k samotnej aplikácií, klient sa vôbec nestará o infraštruktúru ani platformu na ktorej daná aplikácia beží. SaaS sa občas nazýva aj ako „softvér na požiadavku“, nie však v zmysle že si klient objedná tvorbu určitého softvéru, ale v zmysle, že za daný softvér platí podľa aktuálnych požiadaviek. Pri softvére ako službe sa najčastejšie uplatňuje cenová politika, pri ktorej je cena odvedená od počtu používateľov, alebo určitého množstva využitej funkcionality (ako napríklad počet prenesených SOAP správ medzi dvoma IT systémami).

SaaS aplikácie sa najčastejšie konzumujú cez webový prehliadač klientov a teda nevyžadujú okrem webového prehliadača žiaden ďalší lokálne inštalovaný software. Tieto SaaS webaplikácie tak získavajú aj veľkú výhodu vo forme interoperability medzi rôznymi operačnými systémami a rôznymi zariadeniami (čiže aj rôznymi architektúrami).

V prípade SaaS je tiež kompletne celá starosť so škálovateľnosťou, dostupnosťou a zálohami služby na pleciach jej poskytovateľa a ten ako expert v tejto oblasti väčšinou dokáže poskytovať služby v nepozorovateľne vyššej kvalite ako by to bol schopný klient sám v prípade použitia nižšie postaveného cloudového modelu IaaS či PaaS. Dobrým príkladom je služba Google apps for business, konkrétne napríklad Gmail na vlastnej doméne. Služba Gmail v júni 2012 obsluhovala 425 miliónov používateľských účtov. Takéto množstvo používateľov a teda aj dát vyžaduje nemalú infraštruktúru s veľkým množstvom serverov, napriek tomu sa táto služba v očiach používateľa tvári ako jeden spojitý systém.

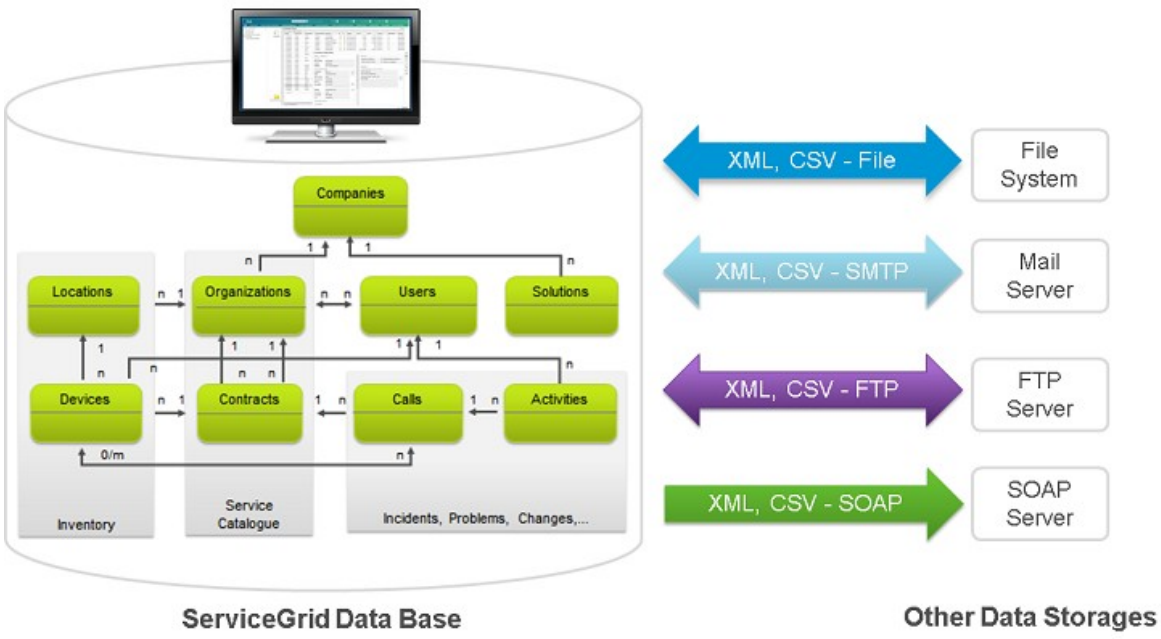
Zástancovia SaaS tvrdia, že dokáže spomedzi všetkých cloudových modelov najviac zredukovať potrebu vlastnej infraštruktúry a IT personálu a má teda potenciál najviac redukovať prevádzkové náklady. Medzi ďalšie výhody takto u poskytovateľa centralizovaného softvéru patrí napríklad aj nahrávanie nových verzií poskytovateľom, pri ktorom odpadá nutnosť inštalovať updatey na klientských staniciach.

Medzi veľké nevýhody SaaS patrí umiestnenie čistých dát priamo u poskytovateľa. Kým pri ostatných formách cloudových služieb si môže klient aspoň z časti chrániť svoje dáta (od šifrovanie pri IaaS, cez spracúvanie len surových dát a prezentácie len výsledku pri PaaS), pri SaaS sú všetky dáta a často aj business procesy definované na systémoch poskytovateľa. V prípade prieniku a získaniu týchto dát neautorizovanými osobami sa o tom klient nemusí ani dozvedieť, alebo sa to dozvie až neskoro, keď už reakcia nemusí byť dosť účinná a môže tak dôjsť k veľkým priamym či nepriamym (strata dobrého mena u zákazníkov klienta SaaS služby) finančným stratám.



Obrázok 8: Ukážka softvéru ako služby spoločnosti Google

Zdroj: https://www.google.com/work/apps/business/



Obrázok 9: Ukážka Service Gridu spoločnosti Cisco

Zdroj: http://www.cisco.com/web/services/portfolio/operations-management/servicegrid/index.html

1.2.3 Delenie Cloudových Služieb podľa umiestnenia

Privátny Cloud

Privátny cloud je infraštruktúra cloudového charakteru prevádzkovaná interne alebo cez externého poskytovateľa pre jedinú organizáciu/klienta. Podujať sa na migráciu, alebo vytvorenie privátneho cloudu je projekt vyžadujúci nemalé množstvo úsilia a znalostí z odboru. Samotná migrácia podnikových procesov do cloudu vyžaduje od organizácie veľké množstvo ťažkých rozhodnutí, ktoré sú v prípade privátneho cloudu ešte náročnejšie. Pokiaľ sa spraví správne, môže podstatne zlepšiť fungovanie podnikových procesov, ale každý krok projektu vyvoláva obavy o bezpečnosť, ktorým musí byť venované nemalé úsilie na zabránenie vzniku vážnych zraniteľností. Vlastná prevádzka dátových centier býva tiež veľmi finančne náročná, má veľký dopad na priestory, veľké nároky na podpornú infraštruktúru (chladenie, fyzická bezpečnosť), hardware a ďalšie firemné procesy. Navyše všetky tieto spomenuté náklady musia byť periodicky obnovované, čo vedie k ďalšej finančnej náročnosti. Skutočné privátne cloudy majú v očiach vedúcich IT osobností (CIO spoločností) väčšinou viac nevýhod ako výhod, nakoľko si spoločnosti „naďalej musia kupovať, postaviť a spravovať infraštruktúru sami“ a teda nezískavajú výhodu vo forme nižších praktických nákladov a celému riešeniu chýba ekonomický zmysel a inak lákavý koncept elasticity cloudov.

Verejný Cloud

Cloud sa nazýva verejným, keď sú jeho služby priamo dostupné cez verejnú sieť Internet, respektíve pokiaľ sú jeho služby poskytované veľkému množstvu zákazníkov na zdielanej infraštruktúre poskytovateľa. V podstate najsignifikantnejší rozdiel medzi privátnym a verejným cloudom je v bezpečnosti na ňom uložených dát a logiky firemných procesov. Všetky takto uložené dáta sa nachádzajú na systémoch poskytovateľa a vo všeobecnosti sa dá povedať že väčšina cloudových poskytovateľov poskytuje svoje služby cez nezabezpečenú sieť Internet. Aspoň malé zlepšenie predstavujú VPN tunely prípadne dedikované linky, ak ich poskytovateľ poskytuje (Amazon AWS má Direct Connect a Microsof Azure svoj ExpressRoute, spoje prenajateľné až do peeringového centra).

Komunitný Cloud

Ako komunitný cloud je väčšinou označovaná služba zdielaná niekoľkými organizáciami alebo individuálmi z určitej komunity. Ako SaaS príklad môžeme spomenúť špecifické portály pre geodetov, právnikov, alebo iné uzavreté skupiny spoločností a užívateľov. Ako IaaS/PaaS príklad by mohli poslúžiť rôzne univerzitné výpočtové centrá poskytujúce svoje služby cloudovou formou výhradne pre svojich študentov, alebo celú akademickú sféru.

Hybridný Cloud

Hybridný cloud predstavuje spojenie dvoch alebo viacerých modelov cloudov (privátnych, verejných alebo komunitných), ktoré sú spojené do jednej služby. Tento termín sa používa tiež v prípade, že spoločnosť si napríklad citlivé dáta a business logiku uchováva výhradne na vlastnom privátnom cloude, ale náročné výpočty, alebo iné prostriedky si zabezpečuje z ponuky verejných cloudov.

Gartner, Inc. definuje hybridný cloud ako akékoľvek spojenie cloudov od rôznych poskytovateľov pri ktorom už celok nie je možné jasne zaradiť do žiadnej kategórie privátnych, verejných alebo komunitných cloudov.

Časté využitie hybridných cloudov je napríklad „Cloud Bursting“. Táto forma sa spolieha primárne na privátny cloud, ale v prípade náhlej potreby vyššej kapacity prostriedkov aplikácia „burstuje“ na verejný cloud. Najväčšia výhoda „Cloud Burstingu“ je, že spoločnosť nemusí predimenzovať svoj privátny cloud, a napriek tomu vie vykryť aj nečakané špičky s veľmi malou dodatočnou investíciou, nakoľko v prípade verejných cloudov zaplatí len za spotrebované prostriedky, respektíve za skutočný čas behu serverov.

Distribuovaný Cloud (Grid Computing)

Za distribuovaný cloud computing, respektíve grid computing sa označuje forma, pri ktorej veľké množstvo serverov na rôznych miestach prispieva svojim výkonom/prostriedkami do spoločného cloudu. Medzi najznámejšie príklady patrí platforma BOINC a Folding@Home. Zaujímavý počin v tomto ohľade bol projekt Cloud@Home, snažiaci sa vytvoriť verejný cloud z dobrovoľne zdielaných prostriedkov jeho používateľov.

Multicloud

Posledná forma je Multicloud – simultánne využívanie viacerých cloudov od rôznych poskytovateľov bez závislosti na jednom poskytovateľovi v jednotnej heterogénnej aplikácii. Hlavná odlišnosť tohto modelu oproti hybridnému cloudu je spoliehanie sa na rozličných poskytovateľov (Napríklad Amazon AWS a PorfitBricks) na rozdiel od rozličných umiestnení (Privátny, Verejný, Komunitný, ...). Správne navrhnutá architektúra typu Multicloud je tak odolná voči výpadku hoci aj všetkých služieb jedného poskytovateľa a dosahuje veľkú flexibilitu možnosťou za behu migrovať prostriedky od jedného poskytovateľa k druhému.

Cloudové systémy

Má nižšie, prípadne skoro žiadne iniciálne náklady ani nevyžaduje toľkú starostlivosť, keďže zodpovednosť a starostlivosť za jeho funkcionalitu nesie poskytovateľ danej monitorovacej služby. Po zriadení je takáto služba pre podnik takpovediac bezúdržbová. Ani toto riešenie však nie je dokonalé, nakoľko monitorovací systém poskytovateľa musí mať prístup do citlivých častí siete za firewallom. Toto však nie je technický problém (VPN), skôr je to otázkou dôvery voči poskytovateľovi.

1.3 Záver analýzy súčasného stavu

Síce existuje niekoľko zahraničných poskytovateľov cloudovej služby monitoringu webových služieb, ale zo slovenských spoločností túto službu neposkytuje ani jedna. Čo je škoda, keďže využívaním zahraničných softvérových riešení nepodporujeme slovenské podniky a slovenskú ekonomiku, ale tie zahraničné. Ani zahraničných poskytovateľov však nie je veľa a väčšina nespĺňa požiadavku na redundanciu služby. Z tohto dôvodu som sa rozhodla implementovať prototyp takejto služby sama.

Po preskúmaní cloudových možností som prišla k záveru, že poskytovanie monitoringu webových služieb ako cloudovú službu možno zaradiť do kategórie softvér ako služba (SaaS) a popri tom ešte využiť ďalšie aspekty cloudu, napríklad infraštruktúru ako službu (IaaS) ako prostriedok na dosiahnutie želaných parametrov služby. V tejto časti vykonaná analýza napomôže využitiu kladných stránok uvedených riešení a vyhnutiu sa záporným stránkam.

2 Cieľ

Cieľom tejto práce je analýza aktuálne dostupných riešení a postavenie základov vysokodostupného monitorovacieho systému dostupnosti webových služieb v sieti Internet, ktorý by bol rozšíriteľný, dostatočne univerzálny a využiteľný pre široké spektrum webových služieb. Systém by mal vykonávať štandardné monitorovanie – overovať dostupnosť systémov cez ICMP Ping, overovať či sú otvorené TCP porty a dajú sa otvárať TCP spojenia a overovať či daná služba vracia preddefinované hodnoty alebo predurčené reťazce, merať rýchlosť odoziev a následne by mal tento monitorovací systém vytvárať a uchovávať reporty aj za uplynulé obdobia. Dôraz bude kladený na čo najvyššiu dostupnosť a decentralizovanosť služby.

Na základe analýzy súčasného stavu a nutnosti vytvoriť taký monitorovací systém, ktorý bude vysoko dostupný som tento cieľ rozdelila na niekoľko podcieľov.

Analýza nástrojov na tvorbu – v tejto časti som popísala potenciálne prostriedky na vytvorenie navrhovaného riešenia v závere kapitoly je možné nájsť vyhodnotenie tých najvhodnejších prostriedkov.

Návrh informačného systému – v tejto časti sa nachádza návrh infraštruktúry, návrh monitorovacieho systému a jeho častí

Implementácia – v tejto záverečnej fáze som vyhodnotila a popísala realizáciu návrhu. Jej výsledkom by mal byť monitorovací systém alebo jeho časť bežiaca na funkčnej geograficky redundantnej infraštruktúre.

Globálnym cieľom je zavedenie vytvoreného riešenia na trh a poskytnutie vysoko dostupného a spoľahlivého monitorovacieho systému dostupnosti webových služieb, ktorý bude rozšíriteľný o nové moduly a druhy testov a vytvorenie určitej autority na trhu ako nestranné a spoľahlivé meradlo dostupnosti. S týmto cieľom je samozrejme spojená aj nutnosť nemalého marketingového úsilia a vytvorenie značky, ktoré však už sú mimo rozsahu tejto práce.

3 Metodológia Riešenia

V tejto časti sa zaoberám analýzou skúmaných protokolov, dostupnými prostriedkami, ktoré by som na prácu mohla využiť a ich parametrami, ktoré by mali napomôcť k rozhodnutiu pre konkrétny model realizácie.

3.1 Špecifikácia Požiadaviek

Monitorovanie webových služieb

- HTTP, HTTPS
  - vyhodnocovanie otvorenosti TCP portov
  - Metoda HTTP get
  - vyhodnocovanie dostupnosti prostredníctvom response kódu a počtu prenesených bytov
  - vyhľadávanie reťazca
- SMTP
  - testovanie odozvy služby (či je TCP port dostupný)
- ICMP PING
  - pingovanie destinácie pomocou doménového mena alebo IP adresy
  - vyhodnocovanie časovej dostupnosti a priemernej latencie

Používateľské prostredie

- ovládanie prostredníctvom webového rozhrania

User management

- Používatelia sa do systému prihlasujú zadaním mena a hesla
- Administrátor môže pridávať používateľov a počiatočné heslo
- Používateľ si môže zmeniť svoje heslo

Zabezpečenie - aplikácia nesmie umožňovať

- SQL Injection
- Cross Side Request Forgery
- XSS

3.2 Forma – Cloudová Služba

Zákazníkovi odpadajú starosti a náklady týkajúce sa administrácie, prevádzky a údržby servera a aplikácie samotnej, proces nasadenia produktu do prevádzky je výrazne kratší a predovšetkým, zákazník sa môže v plnej miere ďalej plne sústrediť na podstatu svojho podnikania. Takýto softvér sa dá pred nasadením do produkcie plne prispôsobiť procesom a ich atribútom klientského podniku.

Takáto forma riešenia je výhodnejšia aj pre zákazníka, keďže sa nemusí o systém starať, nepotrebuje drahé zariadenia na prevádzkovanie systému. Spoplatnenie takéhoto softvéru je férové voči malým podnikom, ktoré zväčša nie sú schopné alebo nemajú potrebu využívať všetky možnosti a moduly monitorovacieho systému a pri tejto forme platia len za to, čo aktívne využívajú.

3.3 Nástroje na tvorbu

Všetok softvér použitý na vývoj alebo ako súčasť produktu musí byť dostupný s otvoreným zdrojovým kódom a uverejnený pod submisívnou licenciou umožňujúcou modifikáciu, redistribúciu a aj predaj bez obmedzení.

Ako použiteľné licencie som vyhodnotila licencie kompatibilné s copyleft filozofiou a BSD kompatibilné.

3.3.1 Databázové Systémy

Na ukladanie používateľských nastavení, ukladanie výsledkov testov a ukladanie reportov z testov potrebujeme databázu. Nakoľko sa budú ukladať veľmi odlišné dáta ktoré však spolu súvisia, rozhodla som sa pre tradičné riešenie vo forme relačnej databázy SQL. Produktov ponúkajúcich SQL databázový server je na trhu veľa, keďže však je môj výsledný produkt decentralizovaný, vyžadujem zabudovanú podporu klastrovania bez nutnosti inštalácie ďalších balíčkov tretích strán.

PostreSQL

PostgreSQL, respektíve zjednodušene Postgres, je objektovo-relačný databázový riadiaci systém ORDBMS (object-relational database management system) so zameraním na rozšíriteľnosť a dodržiavanie štandardov. Ako databázový server, je jeho primárna funkcia bezpečne ukladať dáta a tieto ďalej poskytovať ďalšiemu softvéru, či už bežiacemu lokálne, alebo hoci aj na druhej strane zeme pripojenému cez sieť Internet. Postgres sa dokáže vyrovnať so záťažou typickou pre jednu jednoduchú lokálnu aplikáciu až po záťaž generovanú tisíckami naraz pripojených užívateľov modernej Internetovej aplikácie. Od verzie 9.0 obsahuje zabudovanú podporu replikácie na dosiahnutie vysokej dostupnosti a lepšej škálovateľnosti výkonu. PostreSQL implementuje väčšinu štadardu SQL:2011, je ACID kompatibilný a transakčný (vrátane väčšiny DDL definícií), netrpí problémami s uzamykaním vďaka využívaniu multiversion concurrency control (MVCC) techniky, je imúnny voči nečistému čítaniu (napríklad načítanie nesprávnej hodnoty čítacou transakciou po tom čo sa predošlá modifikujúca vrátila pomocou rollback), ponúka plnú serializovatelnosť, zvláda zložité SQL dotazy s mnohými indexami, ktoré nie su prítomné v iných databázových severoch, ma upraviteľné náhľady, triggery, vzdialené kľúče, podporuje funkcie a uložené procedúry a umožňuje ďalšie výrazné rozširovanie cez produkty tretích strán. Vďaka veľmi širokej podpore štandardov a ponuky kvalitných migračných nástrojov, umožňuje bezproblémový prechod z iných databáz, ako napríklad Oracle. PostgreSQL je multiplatformový a štandardne podporuje systémy Linux, FreeBSD, Solaris, Microsoft Windows a Mac OS X. Začinajúc verziou OS X 10.7 Lion je PostgreSQL štandardnou databázou v edícií server a klientská časť nástrojov je prítomná aj na štandardnej desktopovej verzií. PosgreSQL je vyvíjaný skupinou označujúcou sa ako PostgreSQL Global Development Group, pozostávajúcu z rôznych firiem a individuálov. PostgreSQL je slobodný software s otvoreným zdrojovým kódom.

MySQL

MySQL je slobodný a otvorený multivláknový, viacužívateľský SQL relačný databázový server. MySQL je podporovaný na viacerých platformách (ako Linux, Windows či Solaris). Databázový systém je relačný typu DBMS (database management system). Každá databáza je v MySQL tvorená z jednej alebo z viacerých tabuliek, ktoré majú riadky a stĺpce. V riadkoch sa rozoznávajú jednotlivé záznamy, stĺpce udávajú dátový typ jednotlivých záznamov, pracuje sa s nimi ako s poľami. Práca s MySQL databázou je vykonávaná pomocou takzvaných dotazov, ktoré vychádzajú z programovacieho jazyka SQL (Structured Query Language).

Oracle

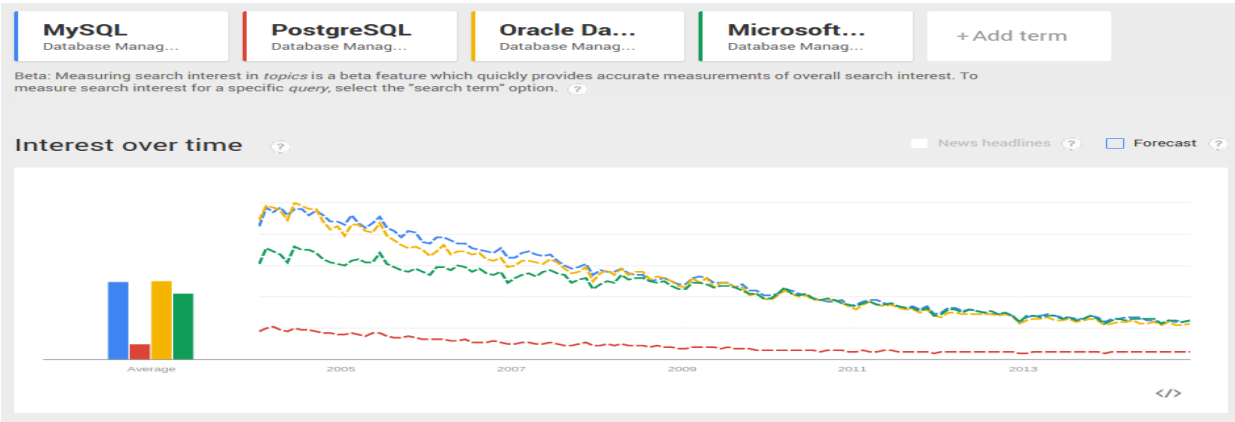
Oracle je systém riadenia bázy dát (*Oracle database management system* – DBMS), moderný multiplatformový databázový systém s veľmi pokročilými možnosťami spracovania dát, vysokým výkonom a jednoduchou škálovateľnosťou. Je vyvíjaný spoločnosťou Oracle Corporation.

Aktuálna verzia je Oracle Database 11g. Podporuje nielen štandardný relačný dopytovací jazyk SQL podľa normy SQL92, ale aj proprietárne firemné rozšírenie Oracle, napríklad na hierarchické dopyty, imperatívny programovací jazyk PL/SQL rozširujúci možnosti vlastného SQL. V tomto jazyku je možné tvoriť uložené procedúry, používateľské funkcie, programové balíky a spúšťať triggery. Podporuje i objektové databázy a databázy uložené v hierarchickom modeli dát, ako napríklad XML databázy a jazyk XSQL.

Microsoft SQL

Microsoft SQL Server je relačný databázový systém programovaný firmou Microsoft. Jeho hlavné dotazovacie jazyky sú T-SQL a ANSI SQL.

Je treba zdôrazniť, že databázové systémy PostgreSQL a MySQL sú na rozdiel od Microsoft SQL voľne šíriteľné a bezplatné. Snaha o minimalizovanie nákladov na realizáciu tohto projektu sa musí odzrkadliť na výbere najvhodnejšieho databázového systému. Ďalším faktorom pri rozhodovaní je jednoduchosť tvorenia databáz – aby sa prvá verzia dala vytvoriť jednoducho a rýchlo a aby sa v nej dali flexibilne vykonávať zmeny. Cieľom teda nie je vybrať čo najprofesionálnejší a najkomplexnejší databázový systém, ale práve naopak – zvoliť databázový systém, ktorý je na vytvorenie riešenia postačujúci a zároveň čo najmenej zložitý.



Obrázok 10: Analýza trendu používania vybraných databázových systémov

Zdroj: http://www.google.com/trends/

Ako môžeme na obrázku 10 vidieť, MySQL patrí ešte stále medzi najvyužívanejšie databázové systémy a dopyt po MySQL databázovom systéme je približne rovnaký ako po komerčnom databázovom systéme od spoločnosti Oracle.

3.3.2 Programovacie Prostriedky

Detailným popisom jednotlivých programovacích jazykov som sa zaoberala v mojej bakalárskej práci [9], preto ďalej uvádzam len skrátený prehľad už spomínaných programovacích prostriedkov a aktuálny trend používanosti vytvorený pomocou funkcie Trends od spoločnosti Google na obrázku č.11.

PHP

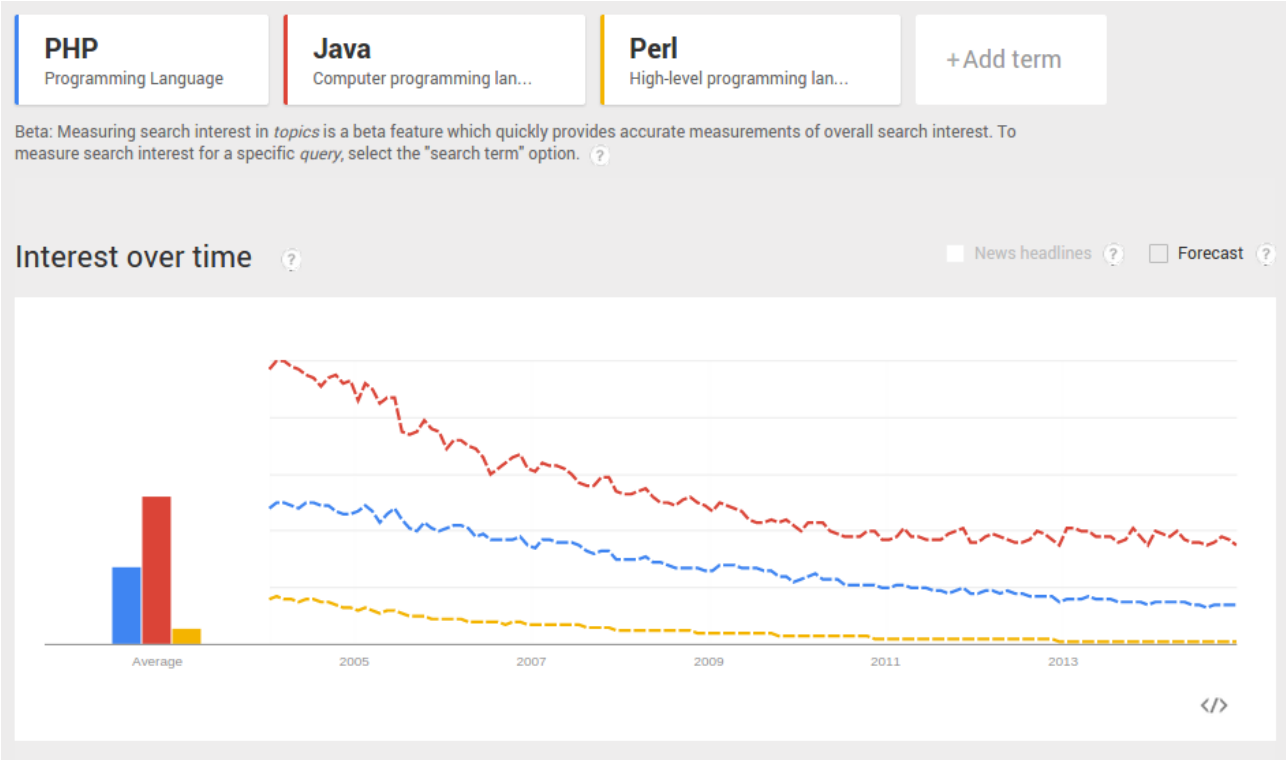
PHP (PHP Hypertext procesor) je populárny open source skriptovací programovací jazyk, ktorý sa používa najmä na programovanie klient-server aplikácií na strane servera a pre vývoj dynamických webových stránok.

Java

Java je všeobecne použiteľný objektovo orientovaný programovací jazyk navrhnutý tak, aby mal čo najmenšie možné implementačné závislosti. Medzi jeho základnú filozofiu patrí „Napíš raz, spusti všade“ (WORA, Write once, run everywhere), kód napísaný pre jednu platformu je spustiteľný na inej bez rekomiplácie. Java aplikácie sa štandardne kompilujú do bytekódu ktorý sa spúšťa v JVM a je teda nezávislý na architektúre CPU či platforme a operačnom systéme. Java je považovaná (2014) za jeden z najpopulárnejších programovacích jazykov na tvorbu klient-server web aplikácii s približným počtom deväť miliónov aktívnych vývojárov.
Pôvodne bola Java vyvinutá v roku 1995 Jamesom Goslingom v Sun Microsystems (ktoré je medzičasom Oracle Corporation) a jej syntax je odvodená od C a C++.

Perl

Perl (Practical Extraction and Report Language) je programovací jazyk primárne určený na prácu s textom.



Obrázok 11: Analýza trendu používania programovacích jazykov

Zdroj: http://www.google.com/trends/

Síce sa ešte stále medzi najpoužívanejšími jazykmi drží Java, trend používania tohto jazyka je výrazne klesajúci. Pri výbere programovacieho jazyka však zohľadňujem aj osobné skúsenosti s daným jazykom, preto sa prikláňam k implementácií v jazyku PHP.

3.3.3 Operačné Systémy

Okrem všetkých hore spomenutých všeobecných podmienok musí operačný systém spĺňať aj rad daľších požiadaviek. Medzi najdôležitejšie patrí kompatibilita s procesorovou architektúrou x86_64, plný TCP/IP stack, podpora štandardného serverového hardvéru, plná podpora virtualizácie ako klienstkého systému a aktuálny aktívny vývoj.

Nakoľko bude môj monitorovací systém fungovať ako cloudová služba na mnou ovládaných

serveroch, je výber vhodného operačného systému dôležitým rozhodnutím. Ako operačný systém pre monitorovacie nody boli evaluované tri rodiny operačných systémov:

- Microsoft Windows
- BSD
- GNU/Linux

Ostatné exotickejšie operačné systémy ako Android Linux, Mac OS X a rôzne predajcami špecifikované systémy ako HP UX, AIX, Hurd, QNX, … boli hneď zo začiatku vylúčené z dôvodu nedostatočnej HW podpory a nedostatočnej podpory virtualizácie.

MICROSOFT WINDOWS

Do porovnania som si zvolila verziu Windows Server 2012 64 Bit Standard Edition, ktorá umožňuje beh dvoch virtuálnych inštancií na jednom jednoprocesorovom serveri. Licencia stojí 517,50 Eura bez dane (August 2014, BA-Computer GmbH). Microsoft Windows je postavený na modulárnej architektúre, jej najnižšia vrstva predstavuje hardvérovú abstrakčnú vrstvu HAL. Nad ňou pracuje samotný kernel (hybridný kernel) a ostatné subsystémy. Samotný kernel sa stará o prideľovanie operačnej pamäte a procesorového času, nad kernelom zase pracujú ďalšie subsystémy ako napríklad Win32, ktorý sa stará o stavbu grafického rozhrania a spracúva signály zo vstupných zariadení.

Microsoft Windows je najrozšírenejší platený operačný systém na osobných počítačoch kompatibilných s architektúrou x86 a x86_64. Aj z dôvodu jeho veľkej rozšírenosti, má veľmi dobrú podporu hardvéru a spoločnosť Microsoft sa stará o funkčnosť a bezpečnosť updatemi v pravidelných intervaloch.

BSD (BERKELY SOFTWARE DISTRIBUTION)

Berkeley Software Distribution (BSD) je variant operačného systému UNIX, ktorý vznikal od roku 1977 na Kalifornskej Univerzite v Berkeley. Väčšina aktuálnych BSD operačných systémov je licencovaných pod BSD licenciou (slobodný a otvorený zdrojový kód) a sú voľne dostupné na stiahnutie. BSD systémy používajú buď monolitický kernel, alebo v prípade DragonFly BSD hybridný kernel. Vyvojári BSD Systémov (jadra) sú väčšinou aj vývojármi userspace programov a zdrojový kód býva teda centrálne spravovaný.

Software pod BSD licenciou je možné slobodne využívať, je povolené ho duplikovať, meniť a ďalej šíriť. Jediná podmienka je zachovanie aj pôvodného copyright textu predošlého autora, z tohto dôvodu sa software pod BSD licenciou hodí aj ako základ pre komerčný software.

Pre potrebu tohto projektu som si vybrala tri systémy z tejto rodiny na ďalšiu evaluáciu.

DragonFly BSD

DragonFly BSD cieli na jednoduchosť a zrozumiteľnosť používania a je vyvíjaný hlavne pre použitie s viacjadrovými/viacprocesorovými systémami. DragonFly BSD vzniklo ako odnož FreeBSD 4.8 s cieľom radikálne zmeniť kernel na dosiahnutie lepšej škálovateľnosti a spoľahlivosti.

OpenBSD

OpenBSD je zamerané na bezpečnosť a spoľahlivosť, jeho filozofia je “štandardne bezpečne” (Secure by default). OpenBSD podporuje skoro všetky dnes dostupné architektúry (alpha, amd64, armish, armv7, aviion, hp300, hppa, i386, landisk, loongson, luna88k, macppc, mvme68k, mvme88k, octeon, sgi, socppc, sparc, sparc64, vax und zaurus) a je teda skoro univerzálne nasaditeľné.

FreeBSD

FreeBSD je jednoducho ovládateľný, univerzálny viacúčelový operačný systém a dosahuje výbornú škálovateľnosť na serveroch. Podporuje však len bežnejšie procesorové architektúry (x86, x86_64, ia64, powerpc, powerpc64 a sparc64).

Všetky tri operačné systémy z rodiny BSD by boli vhodné ako operačný systém pre moje monitorovacie servery, avšak z dôvodu najaktívnejšej komunity a vďaka najväčšej popularite som sa do ďalšieho kola rozhodla posunúť FreeBSD.

Obrázok 12: Analýza trendu používanosti operačných systémov typu BSD

Zdroj: http://www.google.com/trends/

GNU/LINUX

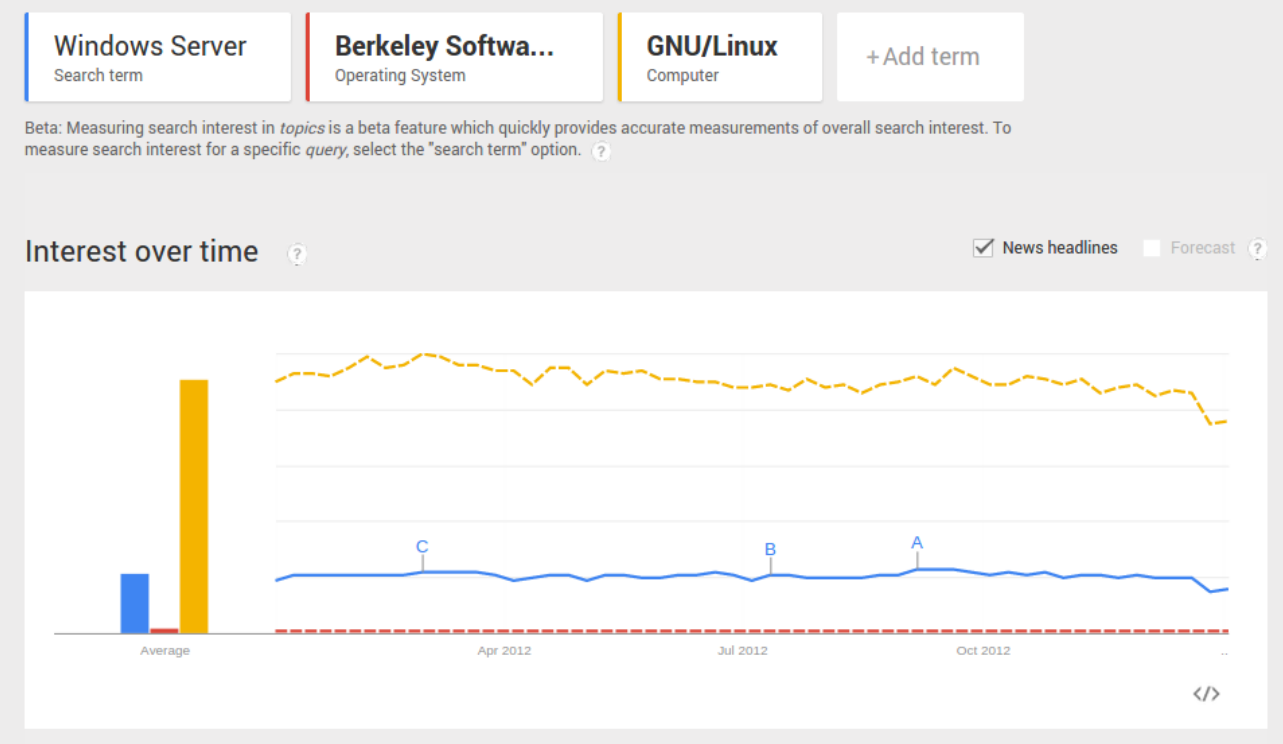
Ako Linux alebo GNU/Linux sa z pravidlá označujú viacužívateľské unixoidné operačné systémy postavené na jadre operačného systému Linux a GNU softvéru. Za svoju dnešnú enormnú popularitu vďačí aj slobodnej licencii GNU GPL a tým, že sa vyskytol vo vhodnom čase, keď GNU operačnému systému chýbal dobre fungujúci kernel.

Tento modulárny operačný systém je vyvíjaný programátormi na celom svete pracujúci na rozličných projektoch. Na vývoji sa podieľajú komerčné spoločnosti, neziskový sektor a obrovské množstvo individuálnych dobrovoľníkov. Tento operačný systém sa používa väčšinou vo forme Linuxovej distribúcie, ktorá obsahuje samotné Linuxové jadro a ďalšie programy, ktoré dokopy tvoria operačný systém. Operačné systémy postavené na jadre Linux sú najrozšírenejšie operačné systémy na svete a to aj napriek tomu, že nemalá časť počítačovo znalej verejnosti toto prvenstvo dodnes pripisuje operačným systémom z dielne spoločnosti Microsoft. Od smart chladničiek, pračiek a vysávačov cez televízory, mobilné telefóny, set top boxy až po superpočítače. Linux môže byť taktiež nasadený na systémy s veľkým dôrazom na bezpečnosť (SELinux, GrSecurity), dostupnosť a vysokú škálovateľnosť (HPC). Verejných linuxových distribúcii existuje približne tisíc, medzi najznámejšie patria komerčné Red Hat Enterprise Linux, SuSe Enterprise Linux, komunitné Debian a jeho deriváty, SlackWare, polokomerčné Ubuntu a jeho deriváty.

Pre môj projekt som sa rozhodla do finále posunúť Debian GNU/Linux pre jeho obrovskú komunitu a stálosť. Debian je najstaršia (od 1993) stále komunitne vyvíjaná Linuxová distribúcia a na jej základoch bolo postavené nespočetné množstvo verejných aj privátnych tzv. “Debian Based” distribúcií, ako napríklad aj Ubuntu. Medzi známe črty tejto distribúcie patri aj balíčkovací systém DPKG a jeho frontend APT. S týmto balíčkovacím systémom je možné systém updateovať, staré verzie Debianu nahradiť aktuálnymi alebo inštalovať nové softvérové balíky. Tie sú taktiež

zodpovedné za vyriešenie všetkých programom vyžadovaných závislostí. To nám zaručuje pohodlnú údržbu aktuálnosti softvéru.

Debian podporuje množstvo rôznych hardvérových architektúr: x86, x86_64, ia64, sparc, armel, armhf, powerpc, mips, mipsel, s390, s390x



Obrázok 13: Analýza trendu používanosti serverových operačných systémov
Zdroj: http://www.google.com/trends/

Na obrázku 13 jasne vidíme markantne vyššiu popularitu GNU/Linux operačných systémov pre serverové nasadenie oproti ostatným menovaným operačným systémom.

3.3.4 Clusteringové riešenia

Pre operačný systém Debian GNU/Linux mame na výber niekoľko softvérových balíčkov pomáhajúcich dosiahnuť vysokú dostupnosť. Medzi najviac využívané patrí heartbeat, jeho nástupca heartbeat2 v spojení s pacemaker, corosync. Naskytuje sa tiež možnosť napísať si jednoduchého správcu prostriedkov špeciálne pre tento projekt.

3.3.5 HTTP(S) Servre

Apache

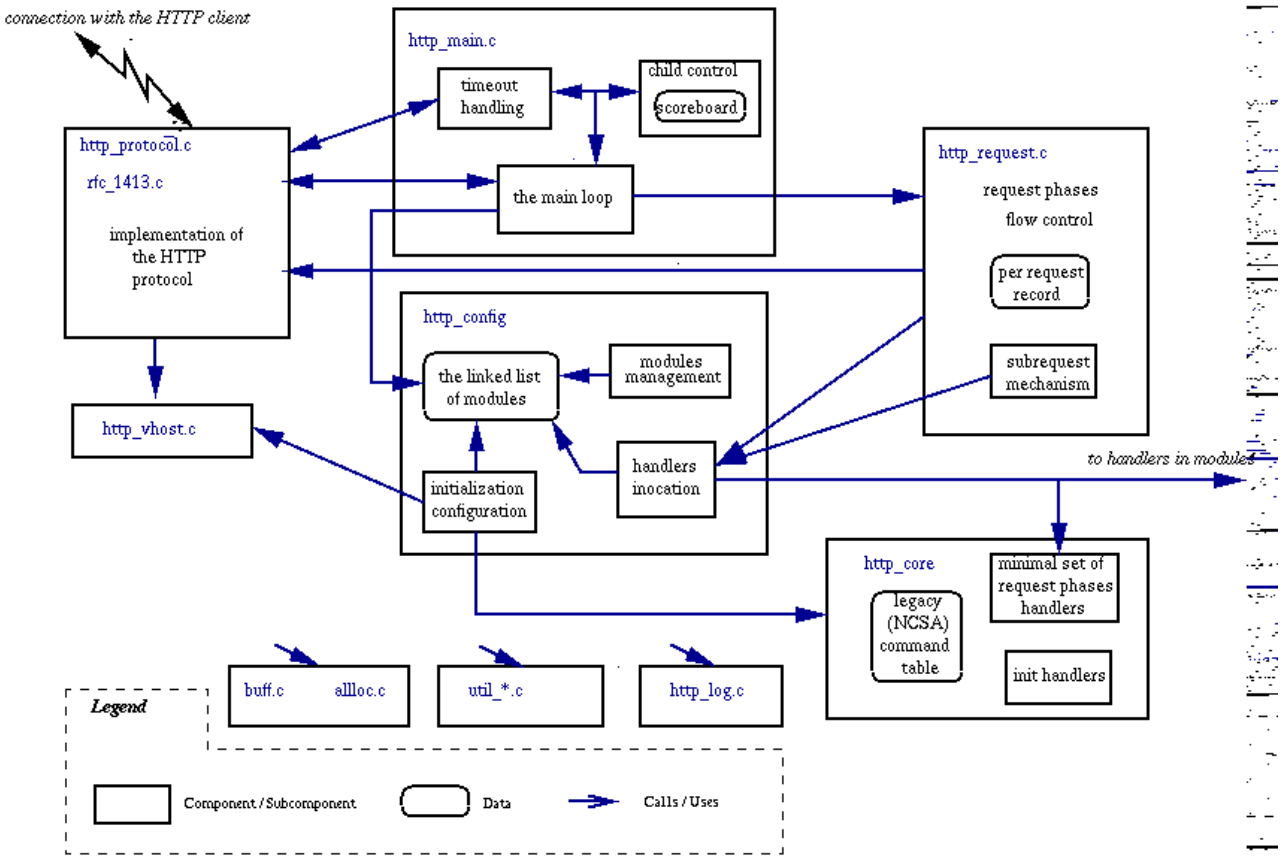
Apache je dodnes celosvetovo najpoužívanejší webový server. Pôvodne bol postavený na NSCA HTTPd, jeho vývoj začal v roku 1995, keď bol ukončený vývoj NSCA HTTPd servera. Apache hral kľúčovú úlohu v počiatočnom raste World Wide Webu siete Internet, nakoľko veľmi rýchlo prebral rolu dominantného servera a v tejto zostal od roku 1996. V roku 2009 sa stal prvým web serverom poskytujúcim viac ako 100 miliónov webových stránok.

Apache je komunitne vyvíjaný ako slobodný software s otvoreným zdrojovým kódom pod správou Apache Software Foundation licencovaný pod takzvanou Apache licenciou. Najčastejšie je používaný na UNIXoidných operačných systémoch, funguje však aj na exotickejších systémoch ako napríklad Microsoft Windows, Novell NetWare, OS/2, TFP, OpenVMS a eComStation. V júni 2013 obsluhoval Apache podľa odhadov 54,2% všetkých webových stránok.

Apache podporuje rôzne funkcie, veľa z nich je vyvinutých a skompilovaných ako moduly, ktoré rozširujú funkcionalitu samotného jadra webového servera. Tieto moduly poskytujú funkcionalitu od podpory rôznych skriptovacích jazykov (Perl, PHP, Python), podpory rôznych autentifikačných schém (mod_access, mod_auth, mod_digest, mod_auth_digest), podpory šifrovania SSL a TLS (mod_ssl), podpory proxy funkcionality (mod_proxy), podpory prepisovania URL (mod_rewrite), podpory vlastného logovania (mod_log_config), podpory filtrovania (mod_include a mod_ext_filter) a mnoho ďalších.

Apache narozdiel od iných neimplementuje jednu architektúru obsluhovania požiadaviek, ale ponúka na výber niekoľko rôznych metód cez MultiProcessing Moduly (MPM). Tie ponúkajú štandardnú proces na každú požiadavku metódu, hybridnú metódu (procesy a vlákna) a tiež udalostno-hybridnú metódu. Výber správnej metódy je dôležitý pre každý druh nasadenia nakoľko sa každá ináč správa po funčnej aj po výkonovej stránke. Ukážku vnútornej architektúry môžeme vidieť na obrázku 14.

Pri poskytovaní statických stránok bol Apache do verzie 2.2 považovaný za neporovnateľne pomalší ako Nginx. Na riešenie tohto problému bola do verzie Apache 2.4 zabudovaná nová architektúra kombinujúca niekoľko procesov a vlákien nad nimi, táto by mala podľa vyjadrenia prezidenta Apache Foundation Jima Jagielského poskytovať rovnaký alebo o niečo vyšší výkon ako webservery postavené na udalostnej architektúre. Podľa nezávislých testov je však Nginx ešte stále pri bežných podmienkach až dva krát tak výkonný.



Obrázok 14: Vnútorná architektúra webového servera Apache

Zdroj: http://ladywoodi.wordpress.com/2012/06/07/everything-about-apache/

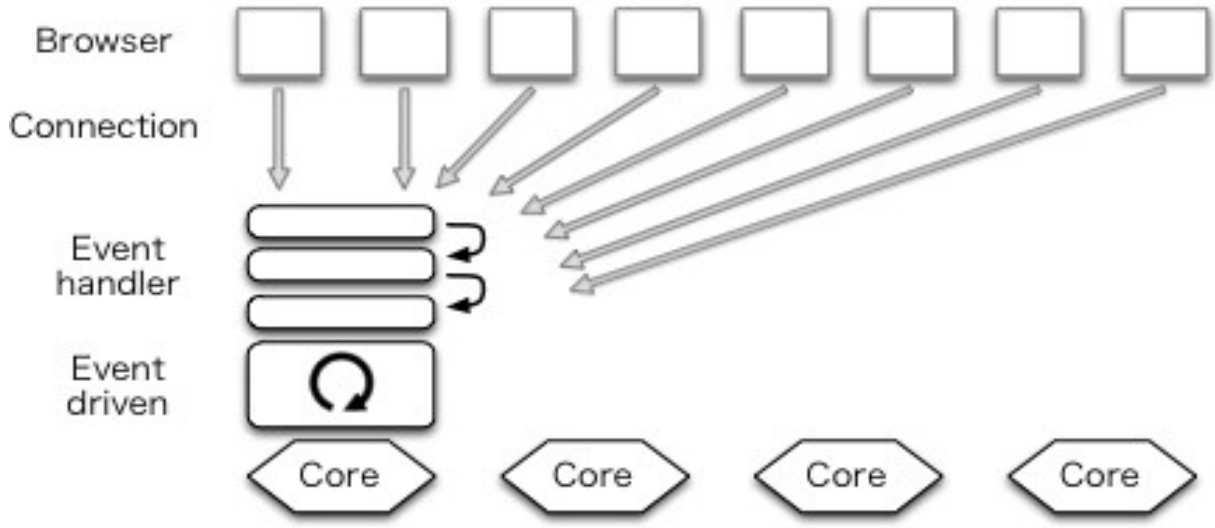
Lighttpd

LigHTTPd je slobodný software s otvoreným zdrojovým kódom, optimalizovaný pre nasadenie v situáciach pri ktorých je rýchlosť poskytovania kritická, zostáva však kompatibilný so štandardami, je bezpečný a flexibilný. Pôvodne začal s vývojom Jan Kneshke ako koncept riešenia problému c10k – Ako obsluhovať 10000 paralelných spojení na jednom serveri, LigHTTPd sa však rýchlo stal celosvetovo populárny.

Malé využitie operačnej pamäte v porovnaní s inými webservermi (Apache), malé zaťaženie CPU a pritom dosahované vysoké rýchlosti poskytovania požiadaviek robia ligHTTPd vhodným pre nasadenie na serveroch s vysokým zaťažením, alebo na poskytovanie statických častí webu pomimo dynamického obsahu. LigHTTPd je distribuovaný pod BSD licenciou a funguje natívne na UNIXoidných operačných systémoch, ale aj na Microsoft Windows. Ukážku jeho architektúry môžeme vidieť na obrázku 15.

LigHTTPd podporuje FastCGI, SGGI a CGI rozhranie k externým aplikáciam umožnujúce v podstate beh dynamickej aplikácie napísanej v akomkoľvek jazyku s dostupným jedným z týchto rozhraní. PHP je v tejto forme najčastejšie používané a teda FastCGI rozhranie servera LigHTTPd je konfigurovateľné na kompletnú a efektívnu podporu PHP aj s opcode cache funkcionalitou (Napríklad APC). Okrem PHP je LigHTTPd veľmi obľúbené v komunitách vývojárov v Pythone, Perle, Ruby a Lua. LigHTTPs tiež podporuje WebDNA databázu a je populárne ako web server pre Catalyst a Ruby on Rails web frameworky.

LigHTTPd podporuje okrem iného aj rozkladanie záťaže, dokáže fungovať ako reverzné proxy, podporuje udalostné mechanizmy select(), poll() a epoll(), podporuje efektívnejšie notifikácie udalostí (kqueue a epoll), podporuje podmienené URL prepisovanie (mod_rewrite), podporuje SSL a TLS aj s SNI cez OpenSSL knižnicu, podporuje autentifikáciu cez LDAP, AJP a mnoho ďalšieho.



Obrázok 15: Vnútorná architektúra webového servera LigHTTPd

Zdroj: http://www.yesodweb.com/assets/warp-posa/2.png

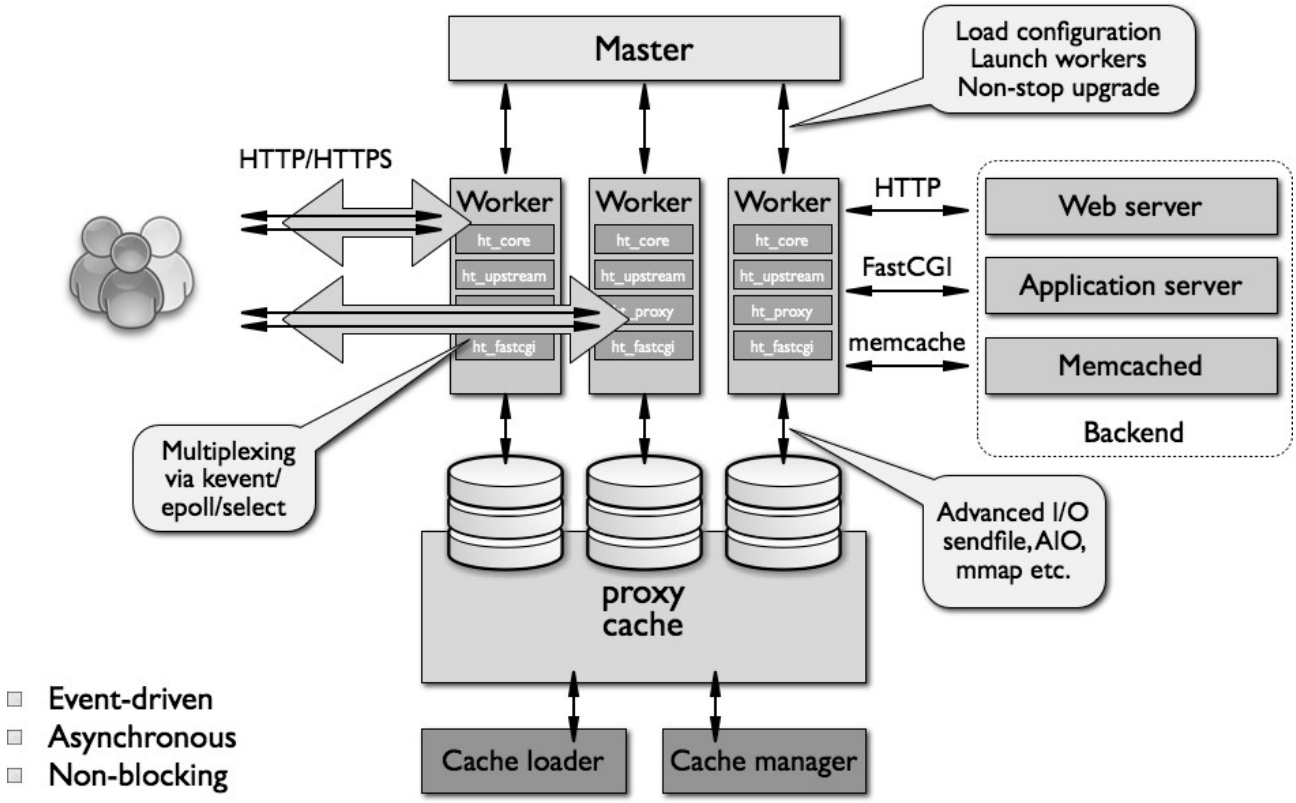
NGiNX

Nginx je slobodný software s otvoreným zdrojovým kódom fungujúci ako reverzný proxy server protokolov HTTP, HTTPS, SMTP, POP3 a IMAP, rozdelovač záťaže, HTTP cache a web server (HTTP/HTTPS). Nginx bol od začiatku vyvíjaný so zreteľom na mnohovláknovosť, vysoký výkon a malé využívané operačnej pamäte. Licencovaný je pod BSD licenciou a funguje na operačných systémoch Linux, BSD, Mac OS X, Solaris, AIX, HP-UX a ďaľších unixoidých

operačných systémoch. [6]

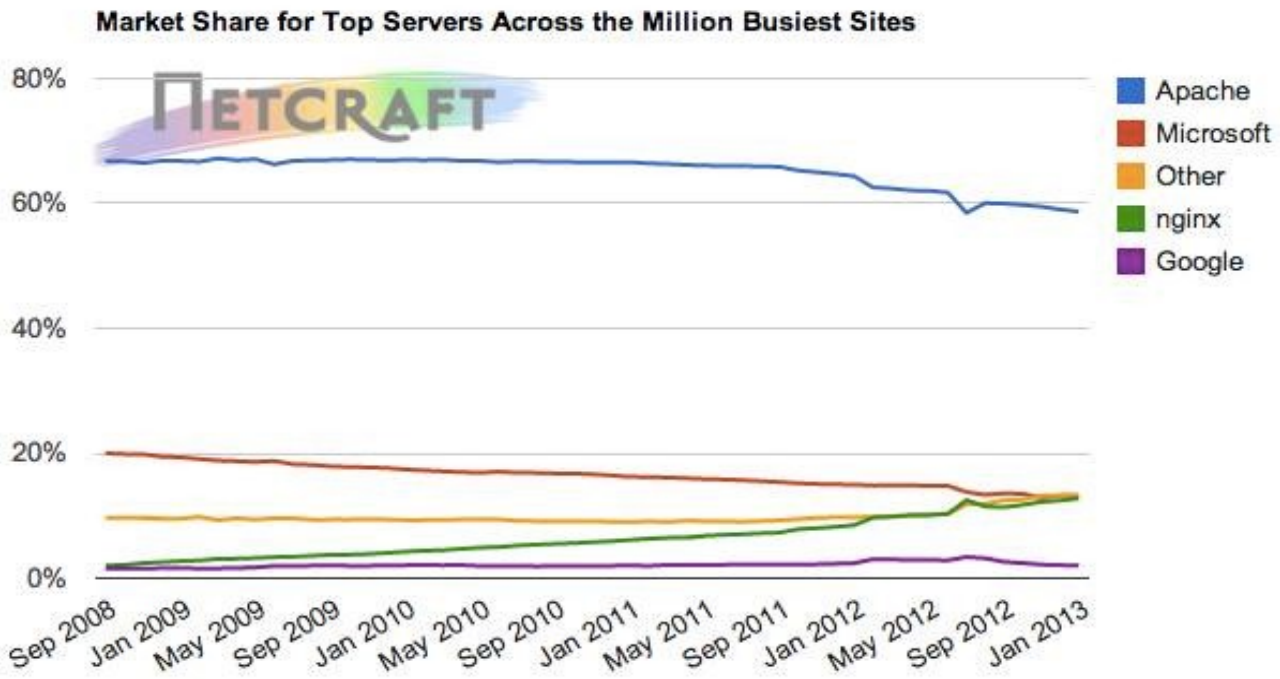
Nginx dokáže sprístupňovať dynamický HTTP obsah pomocou FastCGI rozhrania, SCGI handelorov, WSGI aplikačných serverov, alebo cez PP modul. Jeho vývoj začal v roku 2002 Igorom Sysoevom, v júli 2011 bola založená Nginx, Inc., ktorá od roku 2012 poskytuje komerčnú podporu.

Nginx používa asynchrónny udalostný model obhospodarovania požiadaviek, narozdiel od Apache HTTP servera, ktorý štandardne využíva vláknový alebo procesný model ako je vidieť na obrázku 16. Model obhospodarovania požiadaviek Nginxu produkuje predpovedateľnejší výkon pri vysokom vyťažení. Vývoj servera Nginx bol začatý pre projekty ako Rambler, pre ktorý obhospodaroval 500 miliónov požiadaviek denne v septembri 2008.



Obrázok 16: Vnútorná architektúra webového servera NGiNX

Zdroj: http://www.aosabook.org/en/nginx.html



Developer	December 2012	Percent	January 2013	Percent	Change
Apache	586,594	59.04%	583,143	58.69%	-0.34
Microsoft	131,344	13.22%	131,830	13.27%	0.05
nginx	123,593	12.44%	126,909	12.77%	0.33
Google	20,700	2.08%	19,879	2.00%	-0.08

Obrázok 17: Analýza trendu používanosti webových serverov podľa spoločnosti Netcraft
Zdroj: http://news.netcraft.com/archives/2014/04/02/april-2014-web-server-survey.html

Na obrázku 17 je jasne viditeľný stúpajúci trend používania webového servera NGiNX oproti klesajúcemu trendu webového servera Apache alebo Microsoft, ktorý hovorí jednoznačne v prospech webového servera NGiNX.

3.3.6 Metóda rozdelenia záťaže Round-robin DNS

Jedná sa o metódu rozdelenia záťaže, ktorá nevyžaduje dedikovaný softvér ani hardvér. Spočíva v tom, že s jedným doménovým menom je asociovaných viacero IP adries, klient (prehliadač) si vyberá, na ktorú sa pripojí.

3.4 Relevantné sieťové protokoly

3.4.1 IP

Tento protokol je implementovaný na sieťovej vrstve, je stavový a nespojitý. Aj keď počíta so sto percentným doručením paketov, nekladie žiadne nároky na druh spojenia. To sa odzrkadluje na schopnosti komunikovať týmto protokolom cez rôzne typy sietí (s možnosťou prepnutia trasy). Aktuálne existujú 2 verzie tohto protokolu, ktoré môžu súčasne bežať na jednom počítači (dual-stack), nie sú ale navzájom kompatibilné.

Medzi nevýhody patrí najmä to, ak nejaká prenosová technológia poskytuje niečo naviac, protokol IP to nedokáže využiť. Napríklad technológia ATM dokáže podporovať takzvanú kvalitu služieb a rôznym druhom prevádzok garantovať požadované parametre prenosu. Protokol IP však s ničím takým nepočíta a ak je implementovaný nad technológiou ATM, jej prednosti nedokáže využiť. Medzi výhody takejto minimalistickej koncepcie protokolu IP patrí skutočnosť, že je možné ho implementovať takmer nad každou fyzickou prenosovou technológiou (alebo prenosovou technológiou spadajúcou do vrstvy sieťového rozhrania)

IPv4

Prvá verzia IP. Dnes stále prevažujúci štandard. Adresa zaberá 32 bitov. Skladá sa zo štvorice čísel separovanými bodkou (tzv. IP adresa). Každé toto číslo môže nadobúdať hodnoru od 0 do 255. Pri svojom vzniku sa to javilo ako postačujúce, masovým rastom Internetu sa však ukázalo, že to bol mylný predpoklad. Pred zavedením vyššej verzie IP sa ošetrilo vyčerpanie IPv4 nasledovnými troma sposobmi:

- Beztriedne smerovanie medzi doménami
- Podsiete s premenlivou dĺžkou masky
- Maskovanie podsietí

IPv6

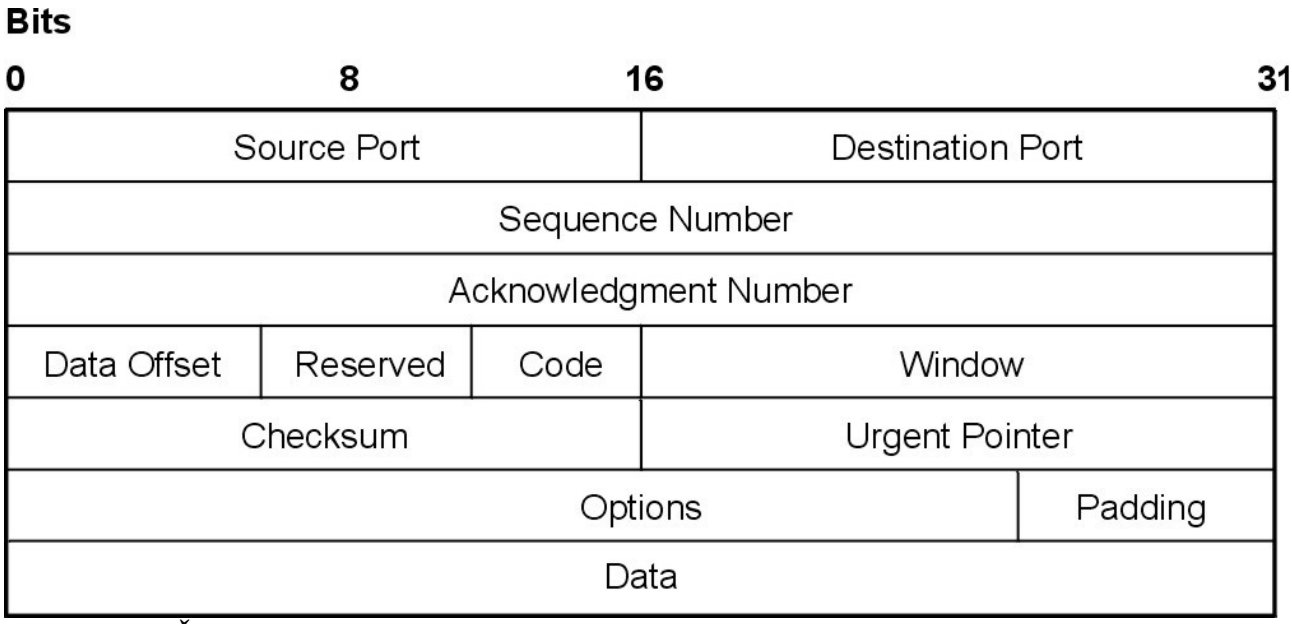
Tento protokol nerieši len problém nedostatku adresného priestoru. Má automatickú konfiguráciu (funkcia NDP – Neighbor Discovery Protocol), zdokonalené smerovanie (obsahuje

funkcie na zaistenie úrovne služieb QoS – Quality of Service) a vylepšené zabezpečenie. Snaha autorov bola, aby väčšina protokolov vyšších vrstiev fungovali bezo zmien rovnako ako aj s IPv4. Adresný priestor má 128 bitov. Záhlavie má fixnú veľkosť.

3.4.2 TCP

Ide o najpoužívanejší transportný protokol dnešnej doby. Vychádza zo štandardu RFC 793. Riadiaci mechanizmus protokolu zabezpečuje, že dáta sa prenesú nepoškodené a zostavia sa v správnom poradí. Tu je niekoľko riadiacich príkazov ovplyvňujúcich jednak množstvá dát v paketoch, ale aj vysielaciu frekvenciu týchto paketov. Tento protokol je početne využívaný pri prehliadaní webových stránok, programami na prenos súborov či doručenie elektronickej pošty.

Autori tohto protokolu mali na pamäti nespoľahlivosť siete a na úkor výkonnosti (priveľa režijných úkonov) dbali na to, aby túto nespoľahlivosť eliminovali. Štruktúra paketu TCP môže vyzerať nasledovne:



Obrázok 18: Štruktúra TCP paketu

Zdroj: http://www.diablotin.com/librairie/networking/firewall/ch06_03.htm

Princíp

Hlavným princípom je vytvorenie virtuálneho spojenia medzi dvoma bodmi (systémami a sieťovými uzlami), pričom tieto body sú stále, ale cesta sa môže meniť. Body sú definované IP

adresou a číslom portu. Pred zahájením každej transakcie musí prebehnúť tzv. „potrasenie rukami“ alebo trojcestné zahájenie spojenia (three – way handshake):

1. Zahájiteľ spojenia pošle druhému požiadavku na synchronizáciu (príznak SYN)
2. Príjemca odpovie príznakom SYN a potvrdzovacím príznakom ACK. Zároveň dojde k výmene čísla sekvencie ISN. Toto číslo je pre každé spojenie iné a dochádza k jeho zmene.
3. Zahájiteľ tohto spojenia odpovie správou iba s ACK príznakom. Týmto je potvrdené spojenie a oba koncové body nazývame sockety. Po tomto „potrasení rukami“ už prebieha výmena dát. Takéto spojenie definuje IP adresa príjemcu, port príjemcu, IP adresa odosielateľa, port odosielateľa.

3.4.3 UDP

Protokol zo skupiny internetových protokolov, ktorý je založený na bezstavovom spojení. Koncové body tohto spojenia se nazývajú datagramové sockety a dáta, čo putujú medzi nimi, sa nazývajú datagramy. Virtuálne spojenie je založené na využívaní portov a umožňuje tak multiplexovánie, čiže súbežné posielanie dátových tokov paralelnými procesmi. Prijaté datagramy sa ihneď extrahujú, bez ohľadu na poradie, alebo či sa nejaké datagramy stratili. Vďaka tomu je tento protokol omnoho rýchlejší než TCP, ale nie je zase vhodný pri potrebe stoprocentného doručenia paketov.

UDP protokol sa používa pri kratších a väčšinou viacsmerových správach (napr. DNS, DHCP, RIP, SNMP, hlasové a audiovizuálne aplikácie, atď.). Je to logické, nakoľko strata jedného obrazového rámčeka nám zo subjektívneho hľadiska veľmi nevadí.

Na obrázku môžme vidieť jednoduchý formát datagramu UDP. Pri porovnaní s paketom TCP je vidieť úsporu v réžii už na štruktúre.

+	bity 0 – 15	16 – 31
0	zdrojový port	cieľový port
32	dĺžka	kontrolný súčet
64	data	

Obrázok 19: Formát datagramu UDP

Zdroj: http://sk.wikipedia.org/wiki/Pou%C5%BE%C3%ADvate%C4%BEsk%C3%BD_datagramov%C3%BD_protokol

Porty

Transportné protokoly TCP aj UDP používajú abstraktnú veličinu nazývanú port, ktorý slúži na založenie internetového socketu na oboch koncových bodoch komunikácie. Keď dáta dorazia do svojho ciela, preskúma sa ich zdrojová adresa, zdrojový port, cieľová adresa a cieľový port. Existuje dohoda definuúca priradenie čísel portov rôznym typom datovej UTB komunikácie. Spravuje ju organizácia s názvom IANA (Internet Assigned Numbers Authority, teda autorita pre priraďovanie čísel v Internete). [8]

Táto organizácia má na svojich webových stránkach zoznam všetkých portov. Samostatný a nezávislý správca vlastných portov je počítač sám. Existujú špeciálne procesy pre monitorovanie a vedenie záznamov o portoch.

Porty sú rozdelené do 3 skupín [7]:

Názov	**Použitie**	**Rozsah**
Dobre známe porty	Pre najbežnejšie služby	0 - 1023
Registrované porty	Registruje sa v ICANN	1024 - 49151
Dynamické a súkromné porty	Na dynamické prideľovanie a súkromné využitie	49152 - 65535

Tabuľka 3: Rozdelenie portov do 3 skupín

3.4.4 ICMP

ICMP je protokol definujúci správy, ktoré si systémy v sieti IP zasielajú ako požiadavky alebo reakcie na udalosti pri prenose dát. Je velmi dôležitý pre riadenie prevádzky a zahltenie siete, slúži na signalizáciu korektného doručenia paketov aj požadavok na ich opätovné zaslanie a kontrolu smerovania. Expirácia parametra životnosti (TTL – Time To Live) IP paketov je jednou udalosťou, ktorá má za následok zaslanie chybovej správy ICMP. Pre správne fungovanie protokolu IP musia systémy podporovať správy ICMP. Korektné správanie ICMP je nutnou podmienkou pre bezchybnú komunikáciu v sieťach IP. Existujú pritom dve verzie tohoto protokolu, jedna určená pre IPv4 a druhá pre IPv6. [8]

ICMP je nespoľahlivý formát prenosu, pretože ICMP správy sa vždy zmestia do jedného datagramu a nie je teda nutné overovať doručenie. IANA na svojom webe taktiež rozpisuje jednotlivé typy odoziev. Pre ukážku zobrazím vybrané odozvy aj rozdiely medzi verziami (ICMPv4 a ICMPv6):

Type	Code	Description
0 – Echo Reply	0	Echo reply (used to ping)
1 and 2		Reserved
3 – Destination Unreachable	0	Destination network unreachable
	1	Destination host unreachable
	2	Destination protocol unreachable
	3	Destination port unreachable
	4	Fragmentation required, and DF flag set
	5	Source route failed
	6	Destination network unknown
	7	Destination host unknown
	8	Source host isolated
	9	Network administratively prohibited
	10	Host administratively prohibited
	11	Network unreachable for TOS
	12	Host unreachable for TOS
	13	Communication administratively prohibited
	14	Host Precedence Violation
	15	Precedence cutoff in effect
4 – Source Quench	0	Source quench (congestion control)
5 – Redirect Message	0	Redirect Datagram for the Network
	1	Redirect Datagram for the Host
	2	Redirect Datagram for the TOS & network
	3	Redirect Datagram for the TOS & host
6		Alternate Host Address
7		Reserved
8 – Echo Request	0	Echo request (used to ping)
9 – Router Advertisement	0	Router Advertisement
10 – Router Solicitation	0	Router discovery/selection/solicitation
11 – Time Exceeded	0	TTL expired in transit
	1	Fragment reassembly time exceeded
12 – Parameter Problem: Bad IP header	0	Pointer indicates the error
	1	Missing a required option
	2	Bad length
13 – Timestamp	0	Timestamp
14 – Timestamp Reply	0	Timestamp reply

15 – Information Request	0	Information Request
16 – Information Reply	0	Information Reply
17 – Address Mask Request	0	Address Mask Request
18 – Address Mask Reply	0	Address Mask Reply
19		Reserved for security
20 through 29		Reserved for robustness experiment
30 – Traceroute	0	Information Request
31		Datagram Conversion Error
32		Mobile Host Redirect
33		Where-Are-You (originally meant for IPv6)
34		Here-I-Am (originally meant for IPv6)
35		Mobile Registration Request
36		Mobile Registration Reply
37		Domain Name Request
38		Domain Name Reply
39		SKIP Algorithm Discovery Protocol, Simple Key-Management for Internet Protocol
40		Photuris, Security failures
41		ICMP for experimental mobility protocols such as Seamoby [RFC4065]
42 through 255		Reserved

Tabuľka 4: Kontrolné odozvy a ich porovnanie

3.4.5 SNMP

Spolu s neustále sa zvyšujúcou komplexnosťou sietí naberá potreba evidovať, administrovať a riadiť zariadenia v sieti na dôležitosti. Pre splnenie týchto potrieb bol v rámci organizácie IETF (International Engineering Task Force) navrhnutý protokol SNMP (Simple Network Management Protocol). Jedná sa o protokol aplikačnej (siedmej) vrstvy, ktorý sa postupem času stal najobľúbenejším nástrojom pre správu sieťových systémov. [8]

Chápať sa dá ako systém SNMP, ktorý má týchto päť prvkov:

- Sieťový protokol SNMP – Protokol umožňuje komunikáciu medzi zariadením a špeciálnym softvérom a to prostredníctvom SNMP v TCI/IP sieťach.
- Riadené objekty – Tie objekty, ktoré sú spravované. Napr.: routery, prepínače, tlačiarne atď.

- Agenti – Softvérová časť bežiaca na riadených objektoch. Tá zhromažďuje dáta o objektu samotnom a o sieťovej komunikácií. Tieto dáta poskytujú pomocou dotazov protokolu SNMP.
- Báza MIB (Management Information Base) – Kompletné informácie o riadených objektoch vo forme databázy. Väčšina dát je určená iba na čítanie, ale existujú aj dáta, ktoré je možné meniť (premenné riadiaceho objektu). Dôraz je kladený hlavne na jednoduchú rozšíriteľnosť databázy. Obsah jednotlivých objektov je daný typom tohto objektu. SNMP neurčuje žiadny povinný typ atribútov, aké musí každý prvok obsahovať. Dokonca ani neurčuje, ktoré atribúty musia byť premenné. SNMP určuje spôsob uloženia (súbory MIB) a definuje spôsob poskytnutia pre používateľa.
- Konzola – V tomto softvéri sa tvoria dotazy a zbierajú dáta. Komunikuje prostredníctvom SNMP protokolu.

Verzie protokolu sú:

- **SNMPv1** – Prvotná verzia protokolu. Slabá bezpečnosť.
- **SNMPv2c** – Vychádza z prvej verzie. Sú tu pridané dalšie datové typy a štruktúry. Nie je kompatibilná s verziou č. 1 (rozdielny formát správ a operacií). Pridaná kontrola doručenia správy (ošetrenie stratovosti paketu pomocou UDP protokolu).
- **SNMPv3** – Rozšírená o zabezpečenie (šifrovanie) a vzdialenú konfiguráciu.

Princíp

Základom sú SNMP príkazy odoslané smerom k riadeným objektom (respektíve uzlom). Konzola tieto príkazy posiela a zbiera odpovede od agentov. Odpovede ukladá, aby boli neskôr k nahliadnutiu používateľovi. Používateľ môže riadené objekty konfigurovať cez konzolu a to pomocí SNMP príkazov.

3.5 Výber Prostriedkov

Na základe predchádzajúcej analýzy potenciálne vhodných prostriedkov na tvorbu monitorovacieho systému dostupnosti webových služieb poskytovaného ako cloudová služba som sa rozhodla pre nižšie uvedené prostriedky a technológie. V tejto kapitole uvádzam dôvody môjho rozhodnutia a výhody mnou zvolených prostriedkov.

3.5.1 Výhody jazyka PHP

- Výkon
- Rozhranie pre mnoho druhov databáz
- Nízke náklady – PHP je k dispozícií zadarmo
- Jednoduché učenie a používanie
- Zabudované knižnice pre najčastejšie implementované úlohy

3.5.2 Výhody databázového systému MySQL

- Jednoduché použitie
- Prenesiteľnosť
- Nízka cena – MySQL je k dispozícií zadarmo

3.5.3 Výhody operačného systému GNU/Linux

Kandidáti na operačný systém boli operačné systémy z rodiny UNIX, konkrétne z rodiny BSD a distribúcie GNU/Linux. Z nich boli vybrané FreeBSD a Debian GNU/Linux na evaluáciu, následne zvíťazil Debian GNU/Linux z dôvodu početnosti skúseností s Debian based distribúciami a jeho perfektná podpora zo strany komunity.

GNU/Linux vyhodnocujem ako najvhodnejší operačný systém pre daný projekt. Rozhodujúci vplyv na rozhodnutie oproti operačným systémom od spoločnosti Microsoft bola plnohodnotná nezávislosť od výrobcu a náklady na licenciu. Rozhodnutie medzi BSD a GNU/Linux som uskutočnila na základe preskúmania detailov komfortu používania a použiteľnosti už nadobudnutých skúseností.

Distribúcia Debian GNU/Linux nielen spĺňa všetky na projekt stanovené kritéria, mám s ním ešte aj najviac skúseností. Všetky softvérové komponenty, ktoré som vybrala v analýze má Debian k dispozícií priamo ako balíček z centrálnych distribučných repozitárov.

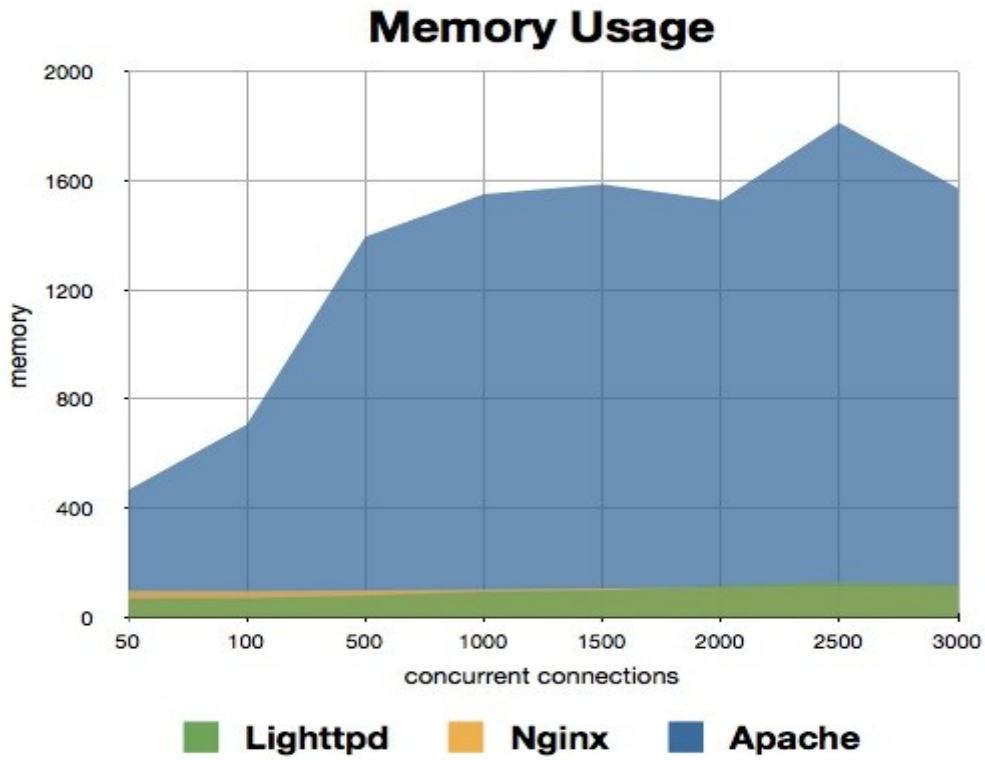
3.5.4 Výhody Heartbeat

Pre daný projekt som sa rozhodla použiť heartbeat2 bez dodatočného správcu prostriedkov, nakoľko sa veľmi jednoducho konfiguruje, je to rokmi overený stabilný softvér schopnosťami prevyšujúci potreby tejto práce.

3.5.5 Výhody NGiNX

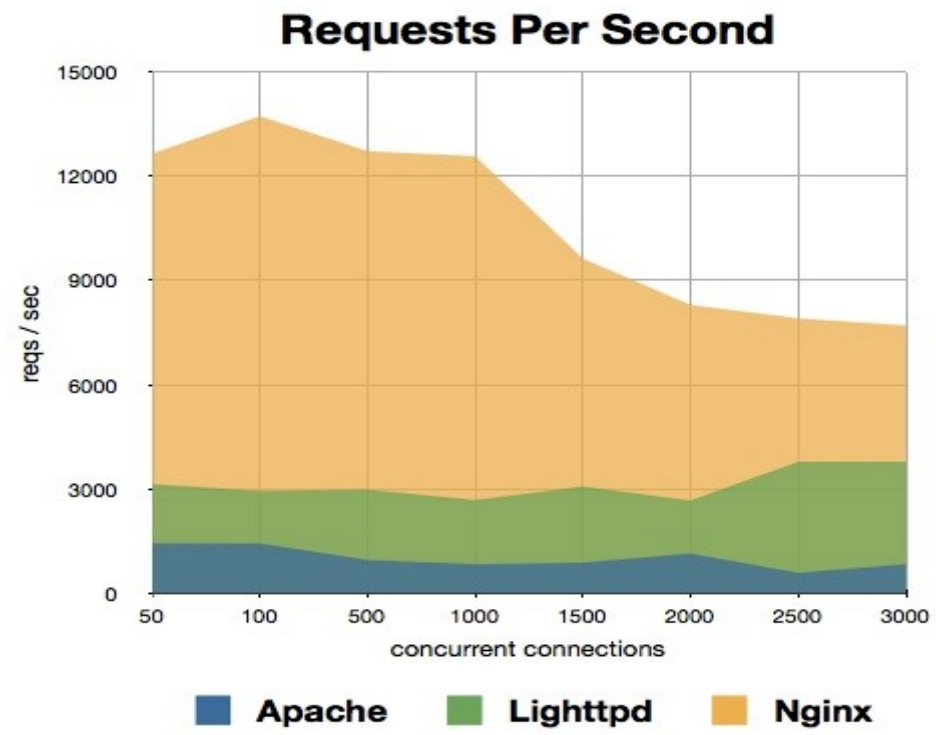
- Stabilita
- bezpečnosť
- jednoduchá konfigurácia
- vyšší výkon a efektivita ako Apache

NGiNX by mal byť schopný vykonať viac požiadaviek s menším vyťažením výpočtových zdrojov vďaka svojej architektúre. Pozostáva z nadradeného procesu, ktorý deleguje prácu jednému alebo viacerým bežiacim pracovným procesom. Každý „pracovný“ proces obsluhuje viacero požiadaviek príležitostným a asynchrónnym spôsobom využívajúc špeciálnu funkcionalitu z linuxového kernelu (epoll/select/poll). To umožňuje Ngnixu obsluhovať veľký počet konkurenčných požiadaviek rýchlo a nízkoréžijne. Aj Apache sa môže konfigurovať na podobný spôsob spracovania, jeho využitie pamäte a vyťažovanie procesora je pri rovnakom počte požiadaviek neporovnateľne vyššie ako Nginx. Tento webový server sa osvedčil aj pri obrovských projektoch ako WordPress a Wikipedia.



Obrázok 20: Analýza využívania operačnej pamäte v závislosti od počtu požiadaviek

Zdroj: http://wiki.dreamhost.com/Web_Server_Performance_Comparison



Obrázok 21: Analýza výkonu v závislosti od počtu požiadaviek

Zdroj: http://wiki.dreamhost.com/Web_Server_Performance_Comparison

3.5.6 Výhody Round Robin DNS

Najväčšou výhodou tohto servera je vo zvyšovaní výpočtovej kapacity distribúciou záťaže medzi viaceré potenciálne geograficky rozptýlené uzly. Nie je však najvhodnejším opatrením na zabezpečenie vysokej dostupnosti, pretože zabezpečuje len distribúciu záťaže tak, že pri výpadku jedného z uzlov neobslúži približne 50% požiadaviek (percento priradené vypadnutému uzlu) oproti centralizovanému systému, ktorý by pri výpadku uzla neobslúžil 100% požiadaviek, takže vysokú dostupnosť je nutné zabezpečiť aj inými mechanizmami.

3.5.7 Výhody DRBD a NFS

Kombinácia štandardného linuxového EXT súborového systému nad DRBD a nad ním NFS sa už roky úspešne používajú v produkcií (Cisco ServiceGrid Team dokumentácia), bez problémov a v prípade ak by aj problémy nastali, administrátor si vystačí so štandardnými linuxovými nástrojmi. Na rozdiel od tohto riešenia fungujú skutočné cluster súborové systémy väčšinou v userspace režime a ich výkon aj spoľahlivosť sú nižšie.

3.5.8 Výhody IaaS riešenia

IaaS spomedzi všetkých cloudových mi poskytuje najvyššiu mieru flexibility pri vývoji mojej aplikácie, nakoľko mám pod kontrolou operačný systém a sčasti aj sieť. Za minimálne náklady, s cenou odzrkadľujúcou reálne spotrebované množstvo prostriedkov (prenesené dáta, využitý CPU čas, využitá operačná pamäť a využitý diskový priestor) si môžem dovoliť spustiť niekoľko nezávislých virtuálnych serverov rozmiestnených v rôznych krajinách u rôznych poskytovateľov.

Náklady na vybudovanie takejto infraštruktúry tradičným spôsobom (prenajatie housingového priestoru v niekoľkých dátových centrách vo svete a zakúpenie dostatočnej serverovej a sieťovej infraštruktúry) by boli enormné a úplne mimo rozpočtu tohto projektu. Z tohto dôvodu som sa rozhodla pre využitie cloudu aj z mojej strany.

4 Implementácia

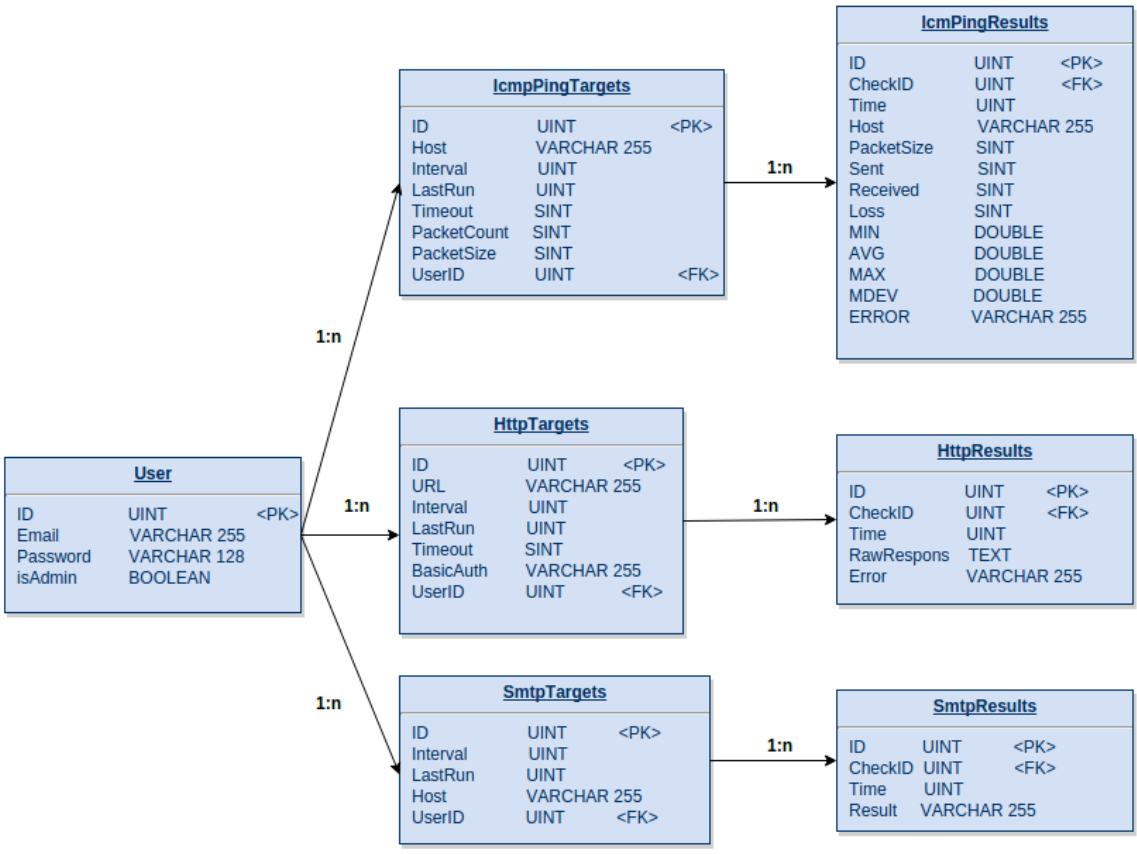
V tejto záverečnej fáze som vyhodnotila a popísala realizáciu návrhu. Jej výsledkom by mal byť monitorovací systém alebo jeho časť bežiaca na funkčnej geograficky redundantnej infraštruktúre

4.1 Návrh jednotlivých častí cloudovej služby

Mnou navrhnutá cloudová služba sa skladá v prvom rade z podpornej infraštruktúry (serverov, pripojenia do siete internet), middleware (webový server, aplikačné prostredie, databázový server...) a samotnej aplikácie.

4.1.1 Návrh databázy

Databázu som navrhla s nasledovnými tabuľkami podľa obrázku 22, aby sa mi s ňou dobre pracovalo na aplikačnej úrovni.



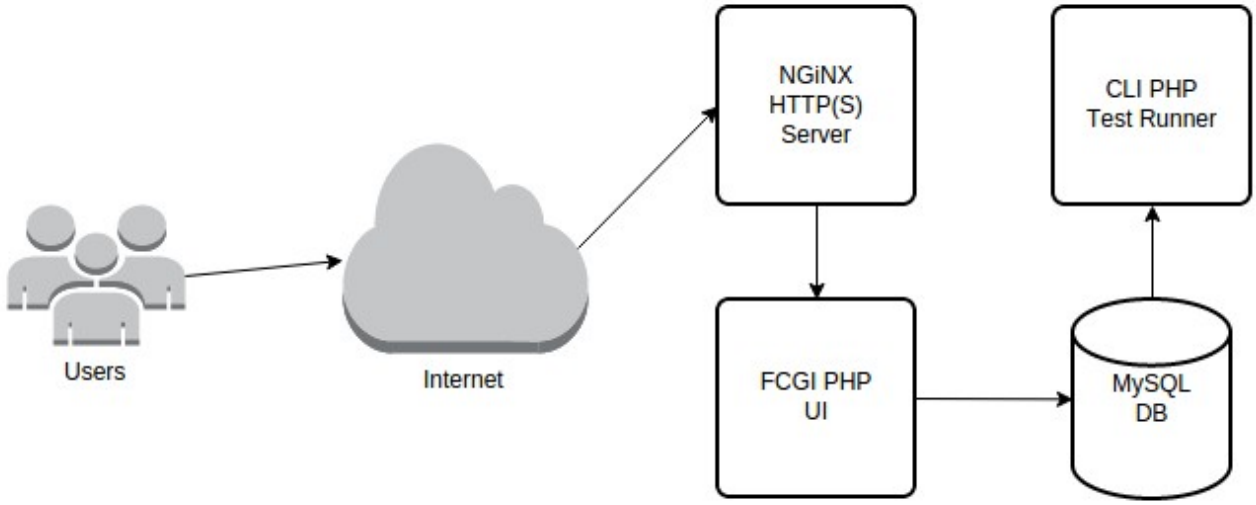
Obrázok 22: Entito-relačný diagram databázy

4.1.2 Návrh jednotlivých aplikačných častí

Samotná aplikácia bude rozdelená na niekoľko funkčných celkov:

- FrontEnd, ďalej deliaci sa na aplikáciu písanú v PHP a NGiNX webserver.
- BackEnd, ďalej deliaci sa na MySQL databázu a PHP program beziaci v Command Line Interface režime.

Moja prvotná predstava riešenia pred fázou návrhu bola nasledovná:



Obrázok 23: Prvotná predstava aplikačných častí

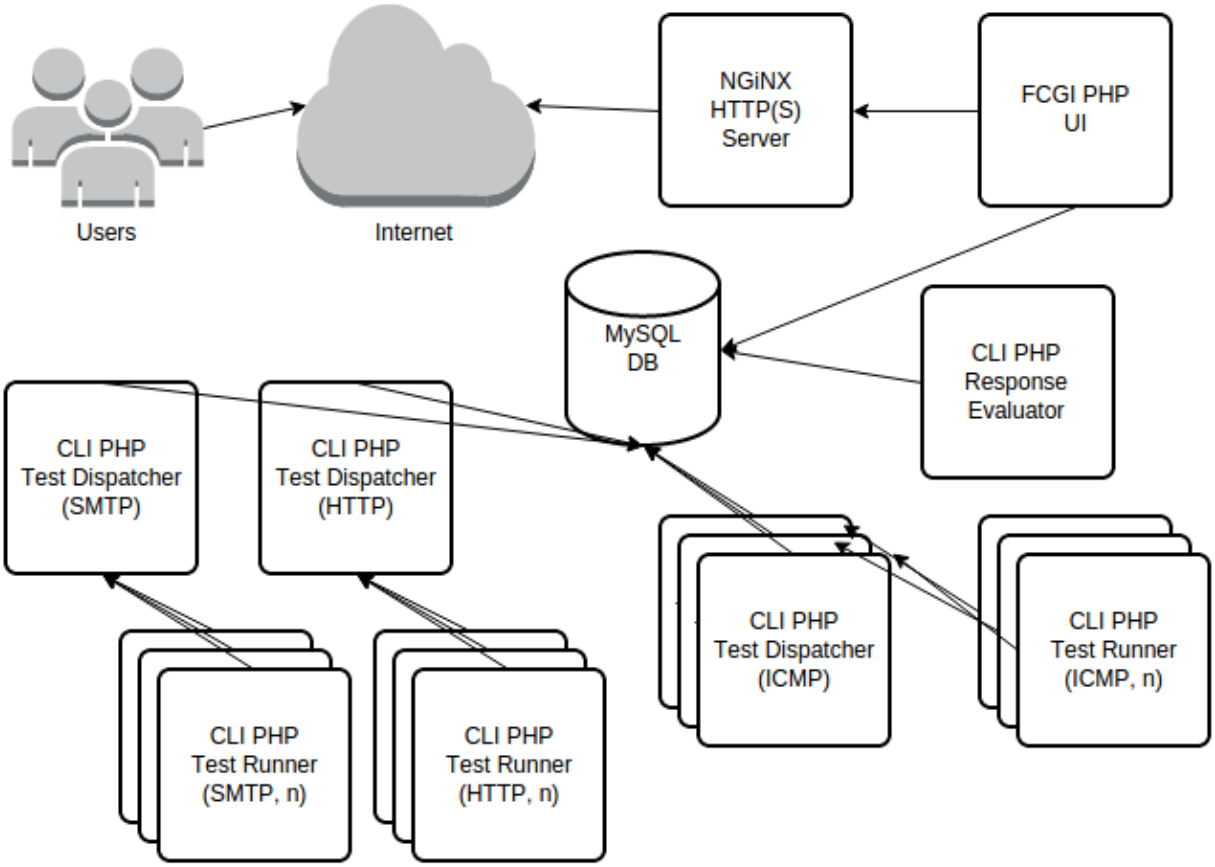
Zaťažiteľnosť v praxi

Pred samotnou implementáciou programového kódu som sa ešte zamyslela nad zaťažiteľnosťou prvotného návrhu v praxi. Na to najlepšie poslúžil modelový príklad:

2400 registrovaných používateľov s priemerne 5 testami v priemerne 5 minútových intervaloch vytvára záťaž vo forme 40 testov za sekundu, respektíve 2400 testov za minútu a 144000 testov za hodinu. Kaď zoberieme do úvahy blocking architektúru vybraného programovacieho jazyka PHP a nutnosť čakať až po definovaný timeout, nie je možné požadovaný výkon obhospodarovať jedným skriptom. Z tohto dôvodu som sa rozhodla danú aplikáciu rozdeliť na niekoľko paralelne bežiacich jednoúčelových procesov a prenechať takto komplikovanosť softvérového paralelizmu na multitaskingové schopnosti operačného systému.

Aktuálne nasadené riešenie využíva jeden proces ako dispečer bežiaci v nekonečnej slučke pre každý druh testu (ICMP, HTTP, SMTP, …), tento dispečerský proces ďalej vytvorí pre každý test samostatný proces ktorého jedinou úlohou je vykonať samotný test a zapísať výsledok do databázy. V pozadí ešte beží ďalší samostatný proces vyhodnocujúci zapísané výsledky a zapisujúci výsledky do databázy.

Diagram vylepšenej architektúry:

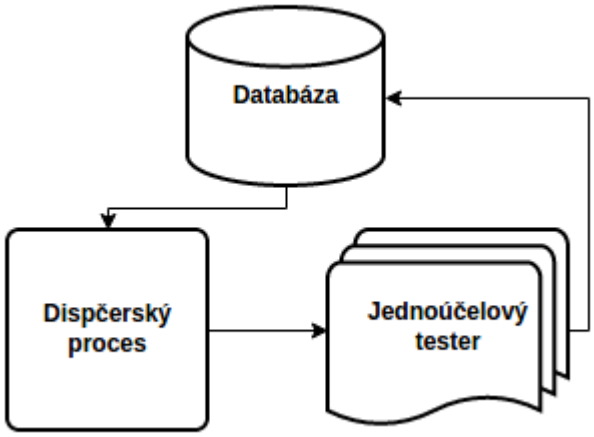


Obrázok 24: Diagram vylepšenej architektúry aplikačných častí

Táto architektúra umožňuje nie len vertikálny rast výkonu, ale aj veľmi jednoduchý horizontálny rast výkonu v prípade potreby. Ako úzky profil by sa v budúcnosti mohli prejaviť dispečerský proces a databáza, ale oba problémy sú relatívne jednoducho riešiteľné zdieľaním databázy a to až do miery rozdelenia tabuliek a tým pádom ďalším až exponenciálnym rastom možných procesov.

Dispečer

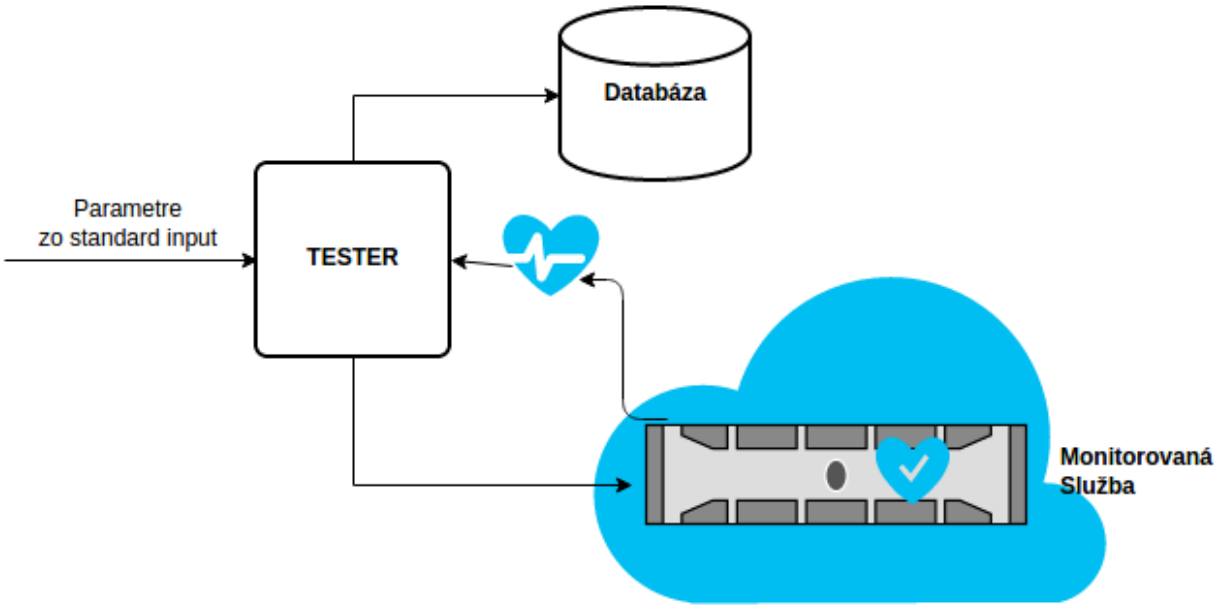
Dispečerský proces číta z tabulky <DruhTestu> Targets (Napríklad IcmpPingTargets) všetky záznamy ktorých posledný test bol vykonaný dávnejšie ako je aktuálny čas a hodnota poľa Interval. Pre každý vrátený riadok spustí tento proces ďalší nezávislý PHP program, ktorý vykoná samotný test a zapíše výsledok spať do databázy. Samotný dispečerský proces beží v nekonečnej slučke.



Obrázok 25: Diagram funkcie procesu dispečer

Tester

Tester je proces spustený dispečerom ako nový proces s parametrami na štandardnom vstupe. Tester má dve úlohy, vykonať test podľa parametrov s ktorými bol spustený dispečerom a následne výsledky testu zapísať do databázy.

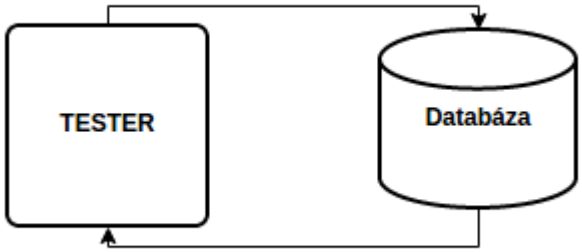


Obrázok 26: Diagram funkcie procesu tester

Vyhodnocovač

Aby sa testovacie procesy čo najrýchlejšie ukončili a nezaťažovali operačný systém enormným množstvom zbytočne bežiacich procesov, zapisujú do databázy surové namerané výsledky, respektíve celú odpoveď v prípade HTTP testu. Tieto výsledky spracúva na pozadí samostatný PHP proces, ResponseEvaluator.

ResponseEvaluator načíta všetky nespracované záznamy, spracuje ich a už vyhodnotené uloží spať do databázy.



Obrázok 27: Diagram funkcie procesu vyhodnocovač

4.1.3 Redundancia Aplikácie

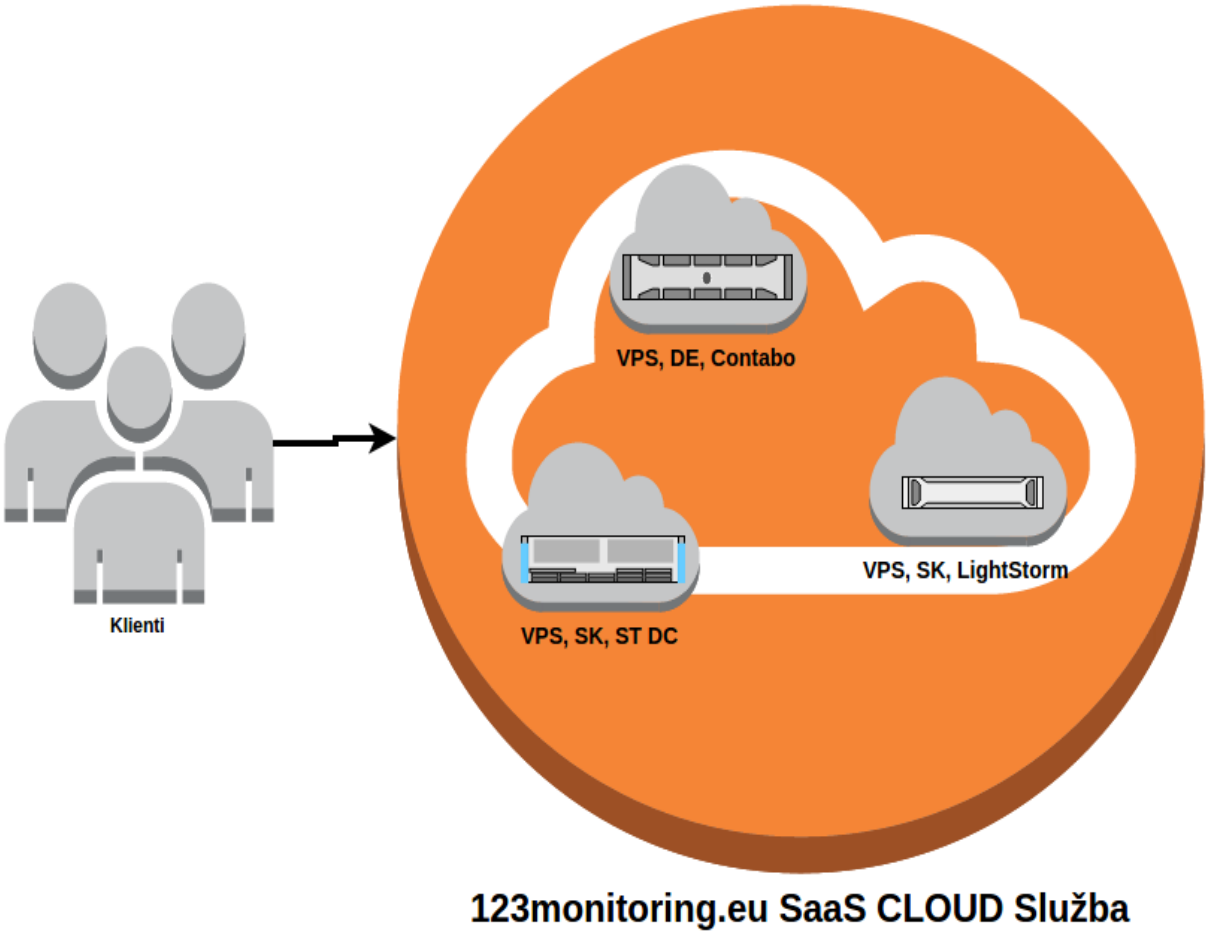
Na dosiahnutie redundancie aplikácie som sa rozhodla rozložiť všetky komponenty samotnej aplikácie na dva nezávislé servery. Keďže každá časť môže simultánne bežať iba raz, použila som nástroj heartbeat na spustenie relevantných častí aplikácie na inom serveri v prípade výpadku.

Vysoko dostupná infraštruktúra

Dosiahnutie vysoko dostupnej infraštruktúry je možné použitím redundancie všetkých komponentov a úplného vyhnutiu sa SPOF. Od dátového centra a jeho pripojenia do siete Internet až po každý jeden komponent servera musí byť aspoň zdvojený a jeden na druhom nezávislý pre dosiahnutie maximálnej možnej dostupnosti celého systému. Najslabší článok celého systému technicky určuje dostupnosť celého riešenia. V prípade, že by napríklad bolo plne redundantné riešenie zo strany serverov a dostupnosť siete umiestnenej v jednom dátovom centre spĺňajúcom štandard TIER III, bola by maximálna možná dostupnosť celého riešenia, s ktorou možeme počítať

na úrovni až do 99.982%.

Aby sme teda dosiahli teoretickú maximálnu možnú dostupnosť riešenia blížiacu sa k 100%, musíme nielen vytvoriť plne redundantný systém zo strany serverov, ale aj celé riešenie decentralizovať do niekoľkých dátových centier s odlišnými poskytovateľmi pripojenia do globálnej siete Internet.

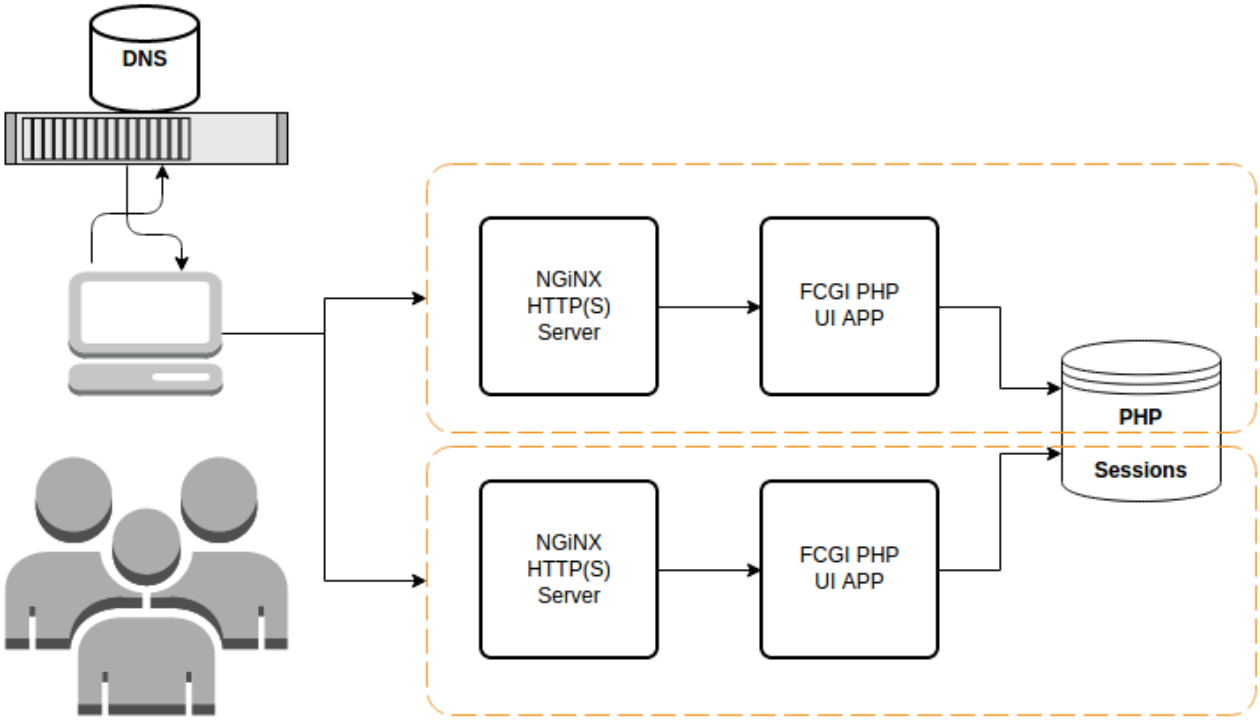


Obrázok 28: Diagram abstrakcie cloudovej služby od infraštruktúry

Redundancia webového rozhrania

Väčšina komponentov môjho riešenia je typu aktívny/pasívny a teda jednotlivé komponenty sú aktívne buď na jednom alebo na druhom virtuálnom serveri. V prípade webového rozhrania však nehrozí žiadna kolízia. Aby boli používatelia rozpoznaní aj pri prechode z jedného VPS na druhý, sú PHP sessions na zdielanom diskovom priestore. Takto sú všetky informácie o sessions prístupné na oboch serveroch.

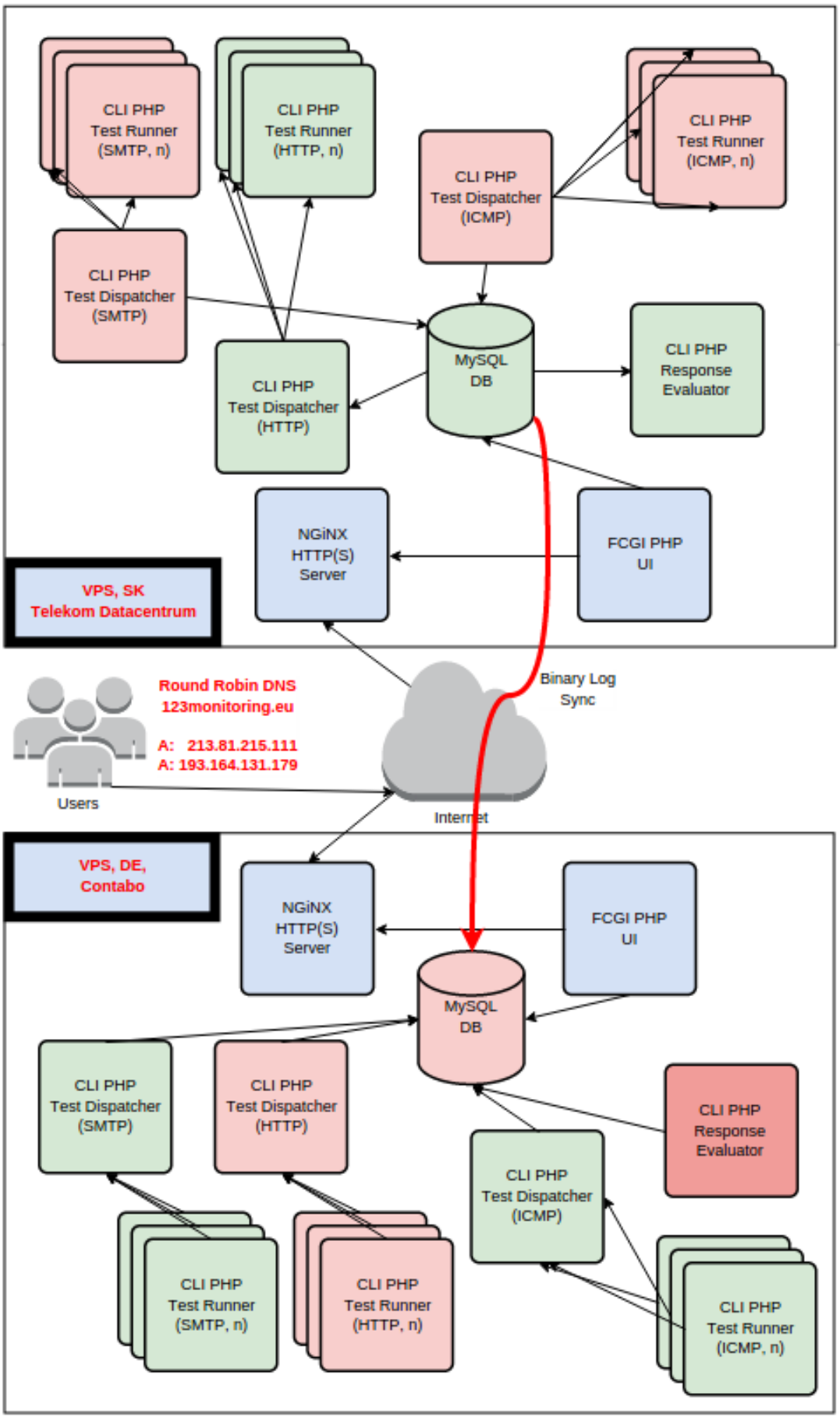
Aby sa dosiahla ešte vačšia dostupnosť služby bez špecifických nárokov na používateľov vracia DNS server pre internetovú adresu www.123monitoring.com tri záznamy s rozdielnymi IP adresami, dve smerujúce na servre na Slovensku a jednu na server v Nemecku. V prípade výpadku jedného virtuálneho servera, alebo aj celého dátového centra je maximálny možný výpadok konfiguračného rozhrania pozorovateľný používateľom na úrovni nenačítanie webovej strány na prvý pokus.



Obrázok 29: Diagram rozdelenia záťaže webového rozhrania

Aktuálne nasadené riešenie využíva jeden proces ako dispečer bežiaci v nekonečnej slučke pre každý druh testu (ICMP, HTTP, SMTP, …), tento dispečerský proces ďalej vytvorí pre každý test samostatný proces ktorého jedinou úlohou je vykonať samotný test a zapísať výsledok do databázy. V pozadí ešte beží ďalší samostatný proces vyhodnocujúci zapísané výsledky a zapisujúci výsledky do databázy.

Diagram aktuálnej architektúry popisujem na obrázku č. 30.



Obrázok 30: Diagram výslednej architektúry clustrovanej časti aplikácie

Takáto architektúra umožňuje nie len vertikálny rast výkonu, ale aj veľmi jednoduchý horizontálny rast výkonu v prípade potreby. Ako úzky profil by sa v budúcnosti mohli prejaviť dispečerský proces a databáza, ale oba problémy sú relatívne jednoducho riešiteľné zdieľaním databázy a to až do miery rozdelenia tabuliek a tým pádom ďalším až exponenciálnym rastom možných procesov.

4.1.4 Použitie Debian GNU/Linux

Ako serverový operačný systém som si zvolila Linuxovú distribúciu Debian GNU/Linux vo verzii pre procesory s architektúrou x86_64 (AMD64). Nakoľko sú však servre použité pre tento projekt hostované u poskytovateľov IaaS, samotnú inštaláciu systému som nemala možnosť priamo ovplyvniť, všetky tri VPS boli dodané už s prihlasovacími root údajmi.

Zabezpečenie

V rámci zabezpečenia operačného systému som pristúpila k niekoľkým krokom:

- Firewall zahadzuje všetky prichádzajúce spojenia okrem vybratých (TCP/22, TCP/80, TCP/443, ICMP)
- SSH umožňuje prihlasovanie sa len kľúčmi
- Pridaný balík automatického updateovania systému (unattended-upgrades) nastavený na samočinnú inštaláciu bezpečnostných záplat raz denne

4.1.5 Architektúra LEMP

Linux, Nginx, MySQL, PHP. Oproti klasickej LAMP architektúre použitie webového servera Nginx. Ten by mal byť schopný vykonať viac požiadaviek s menším vyťažením výpočtových zdrojov vďaka svojej architektúre. Pozostáva z nadradeného procesu, ktorý deleguje prácu jednému alebo viacerým bežiacim pracovným procesom. Každý „pracovný“ proces obsluhuje viacero požiadaviek príležitostným a asynchrónnym spôsobom využívajúc špeciálnu funkcionalitu z linuxového kernelu (epoll/select/poll). To umožňuje Ngnixu obsluhovať veľký počet konkurenčných požiadaviek rýchlo a nízkoréžijne. Aj Apache sa môže skonfigurovať na podobný

spôsob spracovania, jeho využitie pamäte a vyťažovanie procesora je pri rovnakom počte požiadaviek neporovnateľne vyššie ako Nginx.

4.1.6 Použitie IP Sec VPN

Na všetých troch nodách je nainštalovaný balíček OpenSWAN aj OpenSWAN DKMS Modules. Na každej node sú vytvorené dva nezávislé tunely k ďalším nodám. Všetky tunely používajú Preshared Key autentifikáciu (PSK), AES-256 šifrovací protokol, SHA-512 hashovací protokol, diffie hellman group 2 protokol výmeny kľúčov a majú vypnuté perfect forward secrecy (PFS).

Ukážková konfigurácia:

```
conn sk2
      left=213.81.215.111
      leftsubnet=10.0.0.1/32
      right=92.240.248.244
      rightsubnet=10.0.0.2/32
      pfs=no
      keyexchange=ike
      auth=esp
      esp=aes256-sha1
      ike=aes256-sha1-modp1024!
      authby=secret
      keyingtries=0
      keylife=86400s
      ikelifetime=3600s
      auto=start
```

Server SK1 používa adresu 10.0.0.1/32 a má nasledovné tunely:

```
10.0.0.1/32===213.81.215.111<213.81.215.111>[+S=C]...92.240.248.244<92.240
.248.244>[+S=C]===10.0.0.2/32; erouted; eroute owner: #2

10.0.0.1/32===213.81.215.111<213.81.215.111>[+S=C]...193.164.131.179<193.1
64.131.179>[+S=C]===10.0.0.3/32; erouted; eroute owner: #6
```

Server SK2 používa adresu 10.0.0.2/32 a má nasledovné tunely:

```
10.0.0.2/32===92.240.248.244<92.240.248.244>[+S=C]...213.81.215.111<213.81
.215.111>[+S=C]===10.0.0.1/32; erouted; eroute owner: #13
```

```
10.0.0.2/32===92.240.248.244<92.240.248.244>[+S=C]...193.164.131.179<193.1
64.131.179>[+S=C]===10.0.0.3/32; erouted; eroute owner: #4
```

Server DE1 používa adresu 10.0.0.3/32 a má nasledovné tunely:

```
10.0.0.3/32===193.164.131.179<193.164.131.179>[+S=C]...213.81.215.111<213.
81.215.111>[+S=C]===10.0.0.1/32; erouted; eroute owner: #13
```

```
10.0.0.3/32===193.164.131.179<193.164.131.179>[+S=C]...92.240.248.244<92.2
40.248.244>[+S=C]===10.0.0.2/32; erouted; eroute owner: #4
```

4.1.7 Použitie DRBD a NFS

Na zabezpečenie redundancie dátového úložiska funguje medzi servermi BA1 a DE1 systém replikovania blokového zariadenia DRBD. Samotné DRBD využíva na oboch serveroch nezávislé blokové zariadenie do ktorého ukladá svoje metadáta a samotné cez DRBD zapísané dáta, na tento účel je na oboch serveroch vyhradený logical volume (20 GiB) typu LVM2.

Keďze som si zvolila bežný, neclusterovateľný súborový systém EXT4 je DRBD nastavené v režime PRIMARY/SECONDARY a používa replikačný protkol B (semi-synchronous). Takto sú lokálne operácnie zápisu na primárnej node považované za dokončené v momente keď sa dáta zapíšu na lokálny disk primárnej nody a packet nesúci informáciu o zápise k sekundárnej node opustil sieťové rozhranie primárnej nody. V tomto režime DRBD neumožňuje zápis dát na oboch nodách a tak je súborový systém nad DRBD blokovým zariadením pripojený vždy len na jednej node do */mnt/drbd-store*.

Na oboch nodách je ďalej nainštalovaný a na primárnej node vždy spustený NFS(v4) server ktorý zdieľa */mnt/drbd-store* do siete *10.0.0.0/8* cez IP Sec VPN tunel. Každá noda (vrátane tej kde beží NFS server) má pripojený tento zdielaný súborový systém do */mnt/shared*.

Takto je možné na každej node vykonávať operácie čítania aj zápisu dát do/zo spoločného úložiska.

4.1.8 Použitie Heartbeat

Aby aj v prípade výpadku celého jedného servre fungoval moja monitorovacia služba ďalej, nasadila som balíček HeartBeat (v2). Komunikácia prebieha cez nasadené IP Sec tunnely iba medzi nodami SK1 a DE1 nakoľko noda SK2 slúži len na bez web frontendu a teda všetky jej služby bežia simultánne a nezávislo na ďalších dvoch nodách.

Služby obhospodarované HeartBeatom sú: MySQL server, NFS server, Ping Dispatcher, Ping Tester, HTTP Dispatcher, HTTP Tester, SMTP Dispatcher, SMTP Tester a Response Evaluator

Ukážkový *haresources*:

```
sk1  drbddisk::shared  Filesystem::/dev/drbd0::/mnt/shared::ext4  nfs-server
php-http-dispatcher php-http-tester php-response-evaluator

de1 mysql php-icmp-dispatcher php-icmp-tester php-smtp-dispacher php-smtp-
tester
```

4.1.9 Použitie Round Robin DNS

Pre doménu 123monitoring.com a jej subdoménu www.123monitoring.com som nastavila tri samostatné A záznamy smerujúce na všetky tri servre. Ako doménový server som na tento účel použila software KNOT DNS z českých CZ.NIC Labs.

Konfiguracia zóny:

```
$ORIGIN 123monitoring.com.
123monitoring.com. 600 IN SOA ns1.szabados.sk. (
krasna.szabados.sk.
2012082522
86400
7200
604800
86400
)

@ NS ns1.szabados.sk.
@ NS ns2.szabados.sk.

@ A 213.81.215.111
@ A 193.164.131.179
@ A 92.240.248.244
```

```
mail A 193.164.131.179
* CNAME @
@ MX 0 mail.szabados.sk.
```

Odpoveď servera na dotaz:

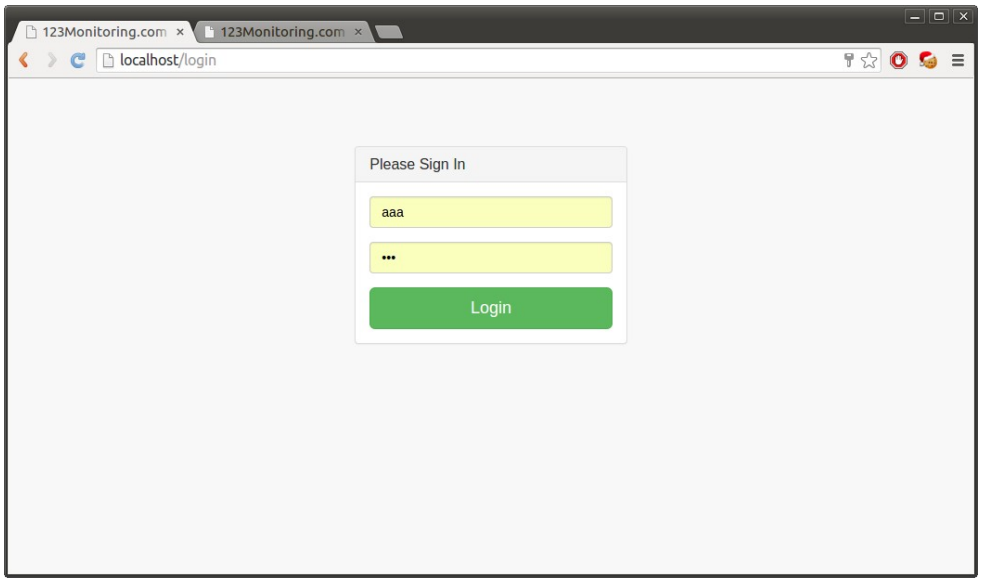
```
kk@CatVer:~$ dig 123monitoring.com @szabados.sk

; <<>> DiG 9.9.5-3ubuntu0.1-Ubuntu <<>> 123monitoring.com @szabados.sk
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16887
;; flags: qr aa rd; QUERY: 1, ANSWER: 3, AUTHORITY: 2, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;123monitoring.com.          IN    A

;; ANSWER SECTION:
123monitoring.com.      3600  IN    A     92.240.248.244
123monitoring.com.      3600  IN    A     193.164.131.179
123monitoring.com.      3600  IN    A     213.81.215.111

;; AUTHORITY SECTION:
123monitoring.com.      3600  IN    NS    ns1.szabados.sk.
123monitoring.com.      3600  IN    NS    ns2.szabados.sk.

;; Query time: 22 msec
;; SERVER: 193.164.131.179#53(193.164.131.179)
;; WHEN: Mon Dec 15 01:36:02 CET 2014
;; MSG SIZE  rcvd: 141

kk@CatVer:~$
```

4.1.10 Používateľské Rozhranie

Používateľské rozhranie som koncipovala ako modernú webovú aplikáciu postavenú na štandardoch HTML 5 a CSS 3 s využitím rôznych JavaScriptových a designových knižníc (Jquery, Bootstrap 3). Ako základ designu som si zvolila slobodnú a voľne šíriteľnú šablónu StartBootstrap 2. Webové rozhranie som rozdelila na logické celky úvodný dashboard, ICMP ping testy, SMTP testy a HTTP get testy.

Prihlásenie

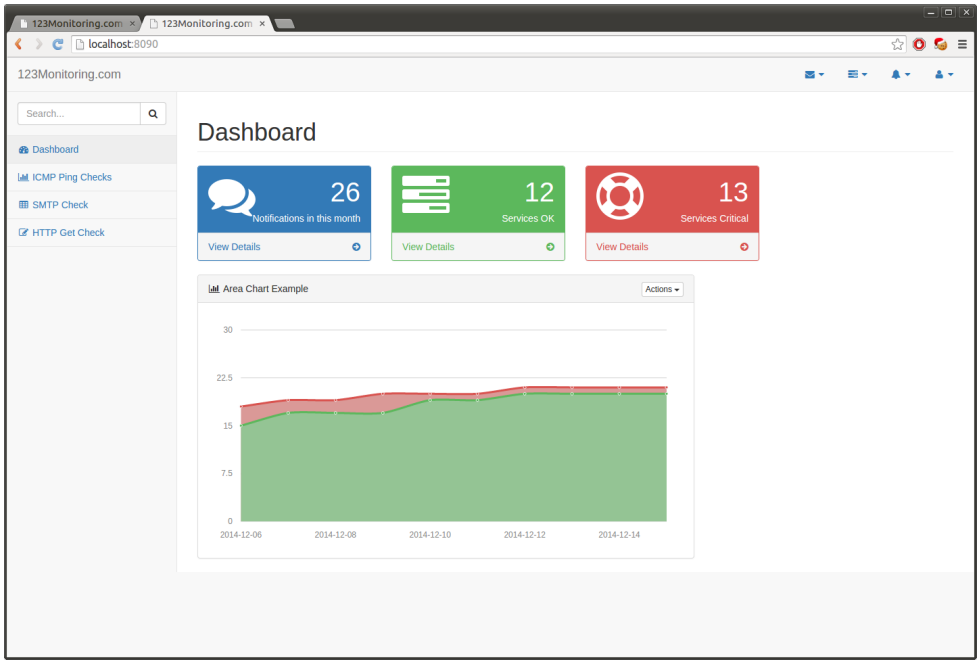
Pred začatím práce s monitorovacím systémom je nutné sa prihlásiť pomocou emailovej adresy a hesla.



Obrázok 31: Prihlasovacia obrazovka

Dashboard

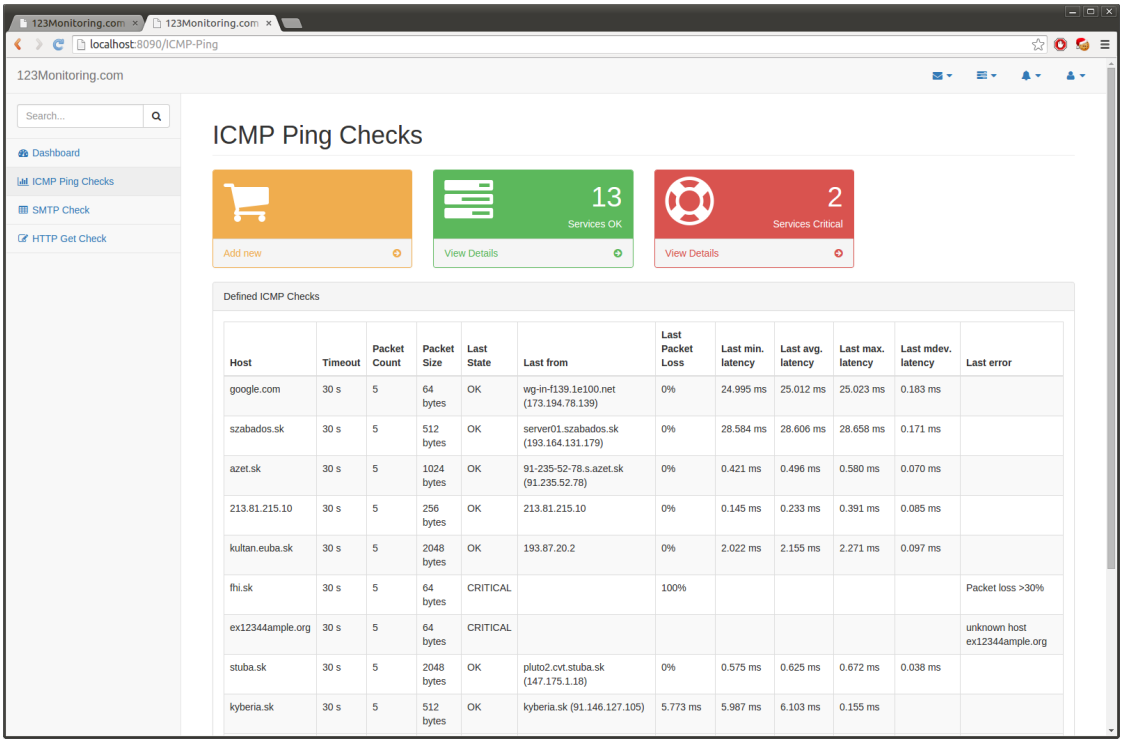
Dashboard slúži na získanie rýchleho prvotného prehľadu o stave monitorovaných služieb. Zobrazuje počet aktuálne meraných služieb v stave critical a ok, zobrazuje počet notifikácií za posledných 30 dní a zobrazuje prehľad o počte critical a ok služieb za posledných 15 dní na prehľadnom grafe.



Obrázok 32: Obrazovka dashboard

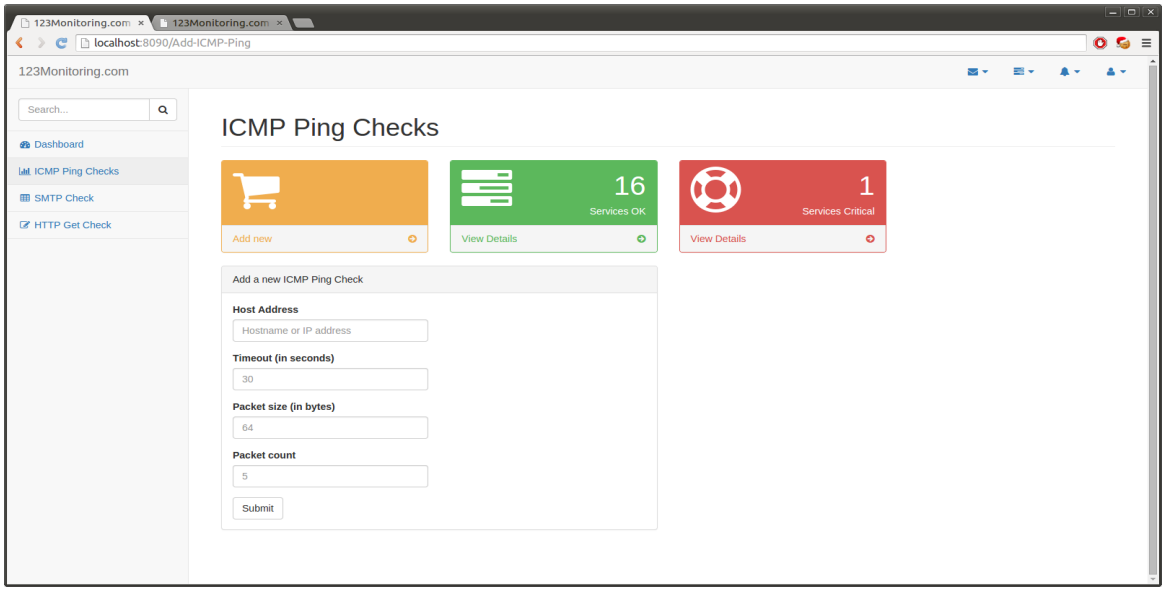
ICMP Ping testy

V hornej časti sekcie ICMP testy získavame okamžitý prehľad o počte služieb v stave ok a critical a máme možnosť pridať ďalšiu službu do monitorovania. Nižšie sa zobrazuje tabuľka s prehľadom všetkých definovaných monitorovaných služieb, nastavení monitorovania a výsledkov posledného testu pre danú službu.



Obrázok 33: Obrazovka ICMP Ping Testov

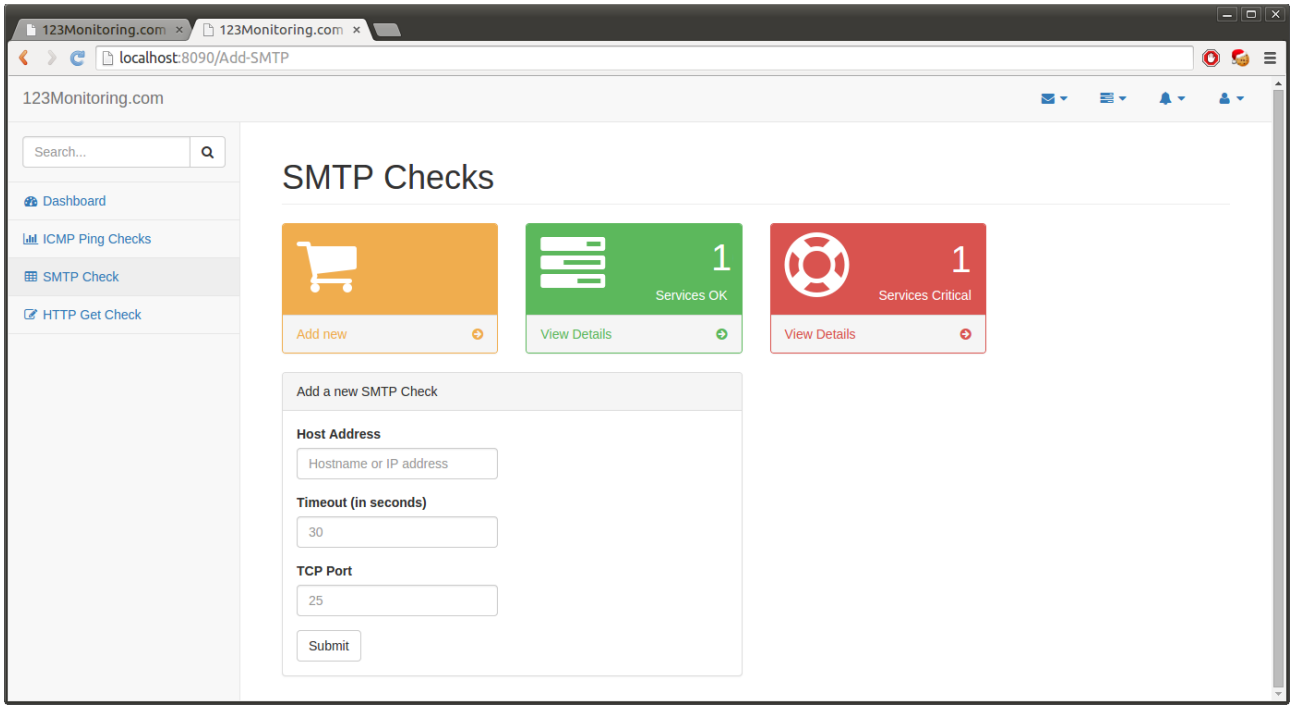
Pre pridanie ďalšej služby do monitorovania je nutné vyplniť prehľadnú formu s políčkami host, čakacia doba, veľkosť a počet odosielaných paketov. Do políčka host je nutné vyplniť plne kvalifikované doménové meno alebo IP adresu servera, ktorý si želáme pingovať. Do políčka počet a veľkosť packetu treba vyplniť požadovaný počet paketov (1 – 10) a ich veľkosť (64 – 4096 bytov). A na záver do políčka čakacia doba je nutné vyplniť maximálnu možnú dobu trvania celého testu (1 – 30 sekúnd).



Obrázok 34: Obrazovka pridania ICMP Ping testu

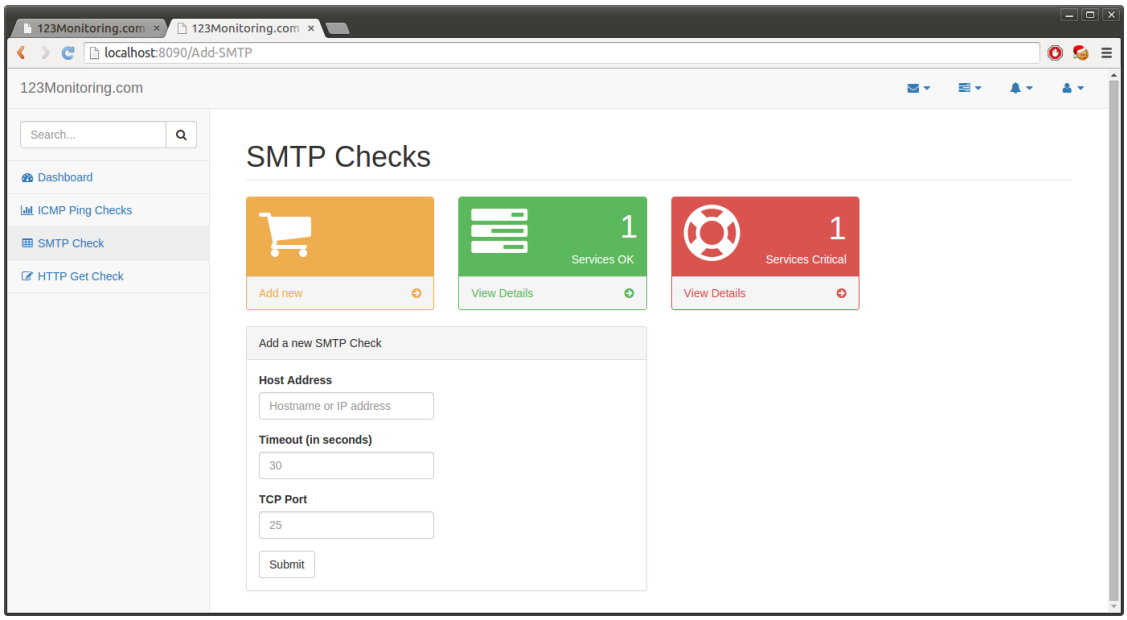
SMTP testy

V hornej časti sekcie SMTP testy získavame okamžitý prehľad o počte služieb v stave ok a critical a máme možnosť pridať ďalšiu službu do monitorovania. Nižšie sa zobrazuje tabuľka s prehľadom všetkých definovaných monitorovaných služieb, nastavení monitorovania a výsledkov posledného testu pre danú službu.



Obrázok 35: Obrazovka SMTP Testov

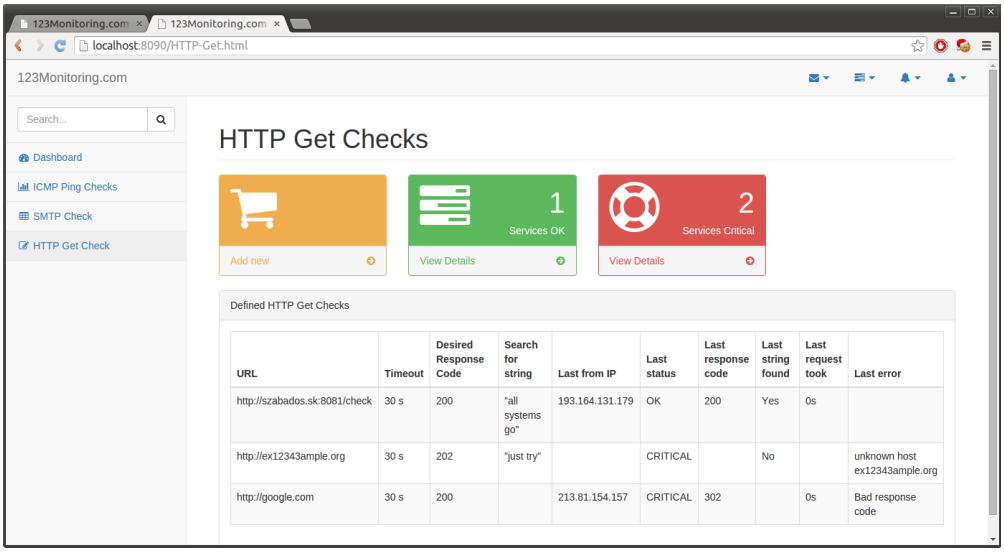
Pre pridanie ďalšej služby do monitorovania je nutné vyplniť prehľadnú formu s políčkami host, čakacia doba a port.



Obrázok 36: Obrazovka pridania SMTP testu

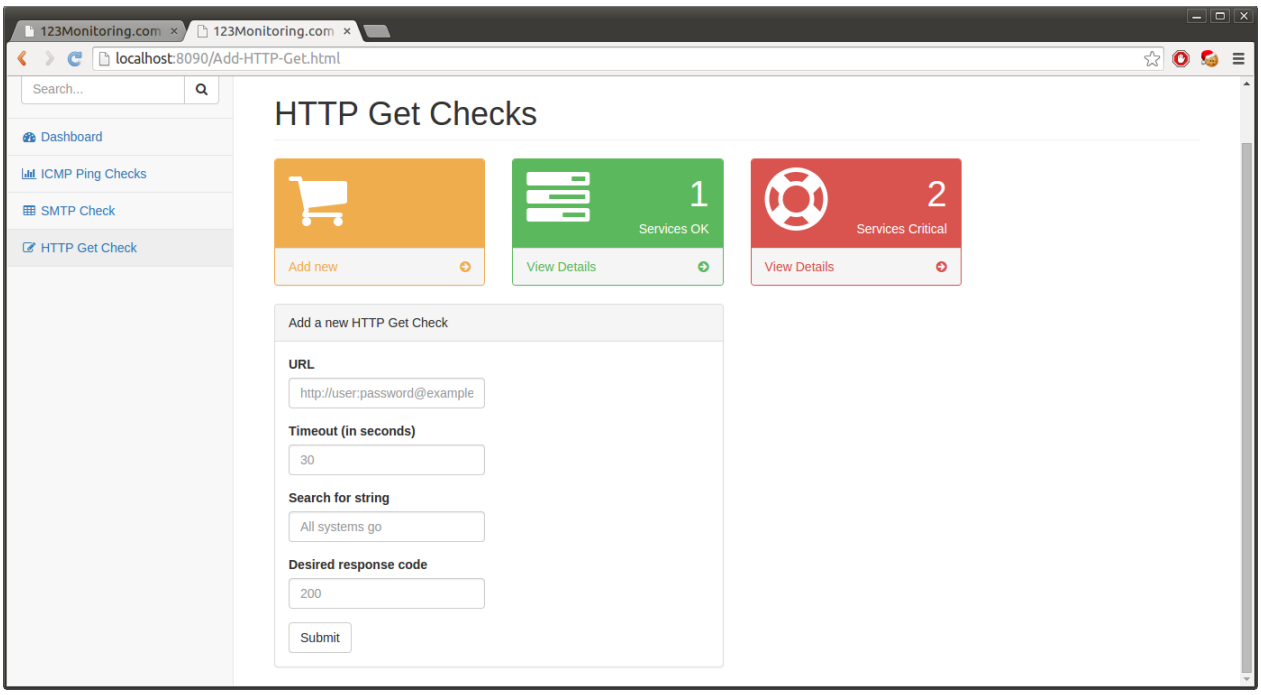
HTTP Get testy

V hornej časti sekcie HTTP Get testy získavame okamžitý prehľad o počte služieb v stave ok a critical a máme možnosť pridať ďalšiu službu do monitorovania. Nižšie sa zobrazuje tabuľka s prehľadom všetkých definovaných monitorovaných služieb, nastavení monitorovania a výsledkov posledného testu pre danú službu.



Obrázok 37: Obrazovka HTTP Get testov

Pre pridanie ďalšej služby do monitorovania je nutné vyplniť prehľadnú formu s políčkami URL, čakacia doba, hľadaný reťazec a požadovaný návratový kód.



Obrázok 38: Obrazovka pridania HTTP Get testu

5 Diskusia

Vytvorená aplikácia je v tomto stave síce dostačujúca pre potreby tejto práce, na uvedenie na trh je však nutné rozšíriť ju o spoplatnenie monitorovania, notifikačné centrum s rôznymi formami poskytovaných notifikácií (email, sms, API).

Taktiež je tu možnosť rozšírenia monitoringu o ďalšie protokoly (SSH, SNMP, POP3, IMAP, LDAP, FTP, SFTP, NTP...), podporu zabezpečených verzií protokolov (HTTPs, SMTPs, POP3s IMAPs, FTPs ...) a o prvky aktívneho monitoringu, aby bolo možné výpadky nielen detekovať ale im aj proaktívne predchádzať.

V rámci geografickej redundancie by bolo optimálne, ak by aplikácia nielen rozdeľovala záťaž medzi jednotlivé nody, ale aj v prípade sieťovej nedostupnosti služby skúsila rovnaký test vykonať z inej nody. Ak by zlyhal aj druhý test, až vtedy by sa výsledok označil za chybový.

Táto práca by mala byť dôkazom, že je možné takúto rozsiahlu infraštruktúru na beh vytvorenej aplikácie zostrojiť aj s nižšími nákladmi, ako v prípade dedikovaného fyzického hardvéru.

Cloudové služby tak naberajú nový zmysel – poskytovatelia softvéru ako služby môžu na beh svojich aplikácií využiť napríklad infraštruktúru ako službu (ako to bolo v tomto prípade). Tým sa vlastne z podniku stáva nielen poskytovateľ cloudovej služby monitoringu, ale zároveň aj odberateľ cloudovej služby serverovej a sieťovej infraštruktúry. Aj vďaka flexibilite cloudových služieb je možné dovoliť si viac virtuálnych výpočtových kapacít, než by si podnik mohol dovoliť fyzických. A práve tento aspekt napomáha vzniku a udržateľnosti mnohých zaujímavých, no často nízkorozpočtových projektov ako je aj táto práca.

Záver

Monitorovací systém webových služieb ma širokú škálu použiteľnosti, či už sa jedná o webové služby ako emaily, groupware, účtovnícky či fakturačný softvér, webhosting, hostované CRM alebo cloudový servicedesk. Umožňuje overenie dodržiavania SLA zo strany poskytovateľa a dáta z tohto monitorovacieho systému môžu byť použité aj ako súčasť dokazovania skutkového stavu v rámci reklamácie.

V časti analýzy nástrojov na tvorbu som sa zaoberala niekoľkými bežne používanými databázovými systémami, programovacími prostriedkami, operačnými systémami a webovými servermi. Taktiež bolo pre vytvorenie tejto práce nutné oboznámiť sa s jednotlivými sieťovými protokolomi a princípmi ich fungovania.

V návrhu som sa zamerala na vytvorenie cloudovej služby aj so všetkými vopred definovanými súčasťami a na vytvorenie jednoduchej monitorovacej aplikácie, ktorá dokáže overovať dostupnosť webových služieb prostredníctvom troch vybraných sieťových protokolov.

Implementácia na základe predošlej analýzy a návrhu prebehla úspešne, jej výsledky je možné vzhliadnuť na priloženom médiu alebo na webovej stránke www.123monitoring.com, pod týmto doménovým menom je možné funkcionalitu aj prakticky otestovať a overiť si geografickú redundanciu.

Pre zavedenie vytvoreného riešenia na trh a poskytnutie vysokodostupného a spoľahlivého monitorovacieho systému dostupnosti webových služieb je nutné aplikáciu rozšíriť o ďalšie časti a funkcionalitu. Pre potreby podnikateľského zámeru je nevyhnutné službu spoplatniť na základe kalkulácie nákladov s prevádzkou spojených, vlastného prínosu a snahy preraziť na trh. Tiež je nutné stanoviť marketingovú stratégiu orientovanú na cieľovú skupinu, ktorou sú v tomto prípade najmä firemní používatelia cloudových služieb. Nakoľko sa však jedná o veľmi komplexné činnosti, ich analýza a vyhodnotenie presahuje rozsah tejto práce a preto sa s nimi budem zaoberať až neskôr.

ZOZNAM POUŽITEJ LITERATÚRY A ZDROJOV

[1] Bc. HABRMAN, Zdeněk: Monitorování aktivních prvků počítačové sítě, UTB ve Zlíně, 2013

[2] http://www.samuraj-cz.com/clanek/zaciname-s-monitoringem-site/ z dňa 16.11.2014

[3] https://www.pingdom.com/pricing/ aktuálny cenník a poskytované služby z dňa 3.12.2014

[4] http://www.monitoring-serverov.sk/ cenník a portfólio služieb z dňa 3.12.2014

[5] http://en.wikipedia.org/wiki/Cloud_computing z dňa 7.11.2014

[6]http://www.linuxjournal.com/article/10108 ...nginx z 10.12.2014

[7] http://cs.wikipedia.org/wiki/Seznam_%C4%8D%C3%ADsel_port%C5%AF_TCP_a_UDP z dňa 7.11.2014

[8] SOSINSKY, Barrie:*Cloud Computing Bible.* John Wiley & Sons, Inc., 2010. 532 s.

ISBN 97-81118-0239-90.

[9] Bc. KRÁSNA, Kristína: Informačný systém na zadávanie a kontrolu plnenia úloh, EUBA FHI, 2011

[10] BARTH, Wolfgang: *Nagios: System And Network-Monitoring.* Open Source Press, Mníchov: Open Source Press, 2006. ISBN 978-15-932-7179-4

[11]ZÁVODNÝ, Peter; TURŇA, Ľubomír; RUBLÍK, Martin: *Počítačové siete v hospodárskej praxi. Bratislava:* Ekonóm, 2009. 358 s. ISBN 978-80-225-2731-6.

[12] SCHLOSSNAGLE, George: *Pokročilé programování v PHP 5*. 1. Brno : Zoner Press, 2004. 640 s. ISBN 80-86815-14-5.

[13] ŠKULTÉTY, Rastislav. *JavaScript : programujeme internetové aplikace*. 1. Praha : Computer Press, 2001. 208 s. ISBN 80-7226-457-5.

[14] ŠIMŮNEK, Milan. *SQL : Kompletní kapesní průvodce*. 1. Praha : Grada Publishing, 1999. 248 s. ISBN 80-7169-692-7.

[15] THOMSON, Laura; GILMORE, William J; WELLING, Luke. *PHP a MySQL : rozvoj webových aplikací*. Praha : SoftPress, 2002. 718 s. ISBN 80-86497-20-8.

[16] CASTRO, Elizabeth. *HTML 4 pro World Wide Web*. 1. Praha : SoftPress, 2001. 368 s. ISBN 80-86497-08-9.

[17] DRUSKA, Peter. *CSS a XHTML : tvorba dokonalých webových stránek krok za krokem*. 1. Praha : Grada Publishing, 2006. 200 s. ISBN 80-247-1382-9.