Univerzita Pavla Jozefa Šafárika v Košiciach

Prírodovedecká fakulta

MONITOROVANIE INFORMAČNÝCH SYSTÉMOV

BAKALÁRSKA PRÁCA

Študijný odbor: Informatika
Školiace pracovisko: Ústav informatiky
Vedúci záverečnej práce: Doc. RNDr. Gabriel Semanišin PhD.
Konzultant: RNDr. Erik Bruoth PhD.

Košice 2015

Pavol Kozák

Poďakovanie

Chcem sa poďakovať vedúcemu tejto práce doc. RNDr. Gabrielovi Semanišinovi PhD., konzultantovi RNDr. Erikovi Bruothovi PhD. a Mgr. Slavomírovi Varchulovi za cenné rady, odbornú pomoc a konzultácie, ktoré mi poskytli pri vypracovaní tejto bakalárskej práce.

Osobitné poďakovanie patrí mojej rodine a priateľom za podporu a pochopenie.

Abstrakt

Hlavným cieľom našej bakalárskej práce je navrhnúť spôsob monitorovania informačného systému za účelom zbierania užitočných informácií ohľadom výkonu systému a používateľských preferencií. Zaoberáme sa najmä otázkami, aké informácie zbierať a akými nástrojmi ich zbierať. Existuje viacero nástrojov sčasti riešiacich našu problematiku. Zameriame sa na open-source nástroje. Vpráci si rozoberieme tie najpoužívanejšie, ich výhody, nevýhody a vybrané nástroje integrujeme do akademického informačného systému AiS2. Pomocou nich sa pokúsime zbierať a zobrazovať metriky, ktoré nám pomôžu identifikovať problémy a funkcie, ktoré je potrebné vylepšiť.

Kľúčové slová: metrika, monitorovanie, informačný systém

Abstract

The main aim of our bachelor thesis is to design a way to monitor information system in order to gather useful information about system performance and user preferences. We are dealing mainly with questions what information to gather and what tools to use for collecting. There are several available open-source tools to solve certain parts of our problems. In this thesis we will be describing advantages and disadvantages of the best known of the tools and integrating choosen ones into the academical information system AiS2. We will gather and vizualize useful metrics using the tools to identify problems and functions to be improved.

Key words: metric, monitoring, information system

Obsah

Úvod **5**

1 Základné pojmy **6**

1.1 Informačný systém 6

1.2 Webová aplikácia 6

1.3 Metrika 7

1.3.1 Rozdelenie metrík 7

1.3.2 Aká je dobrá metrika? 8

1.3.3 Typy metrík 8

1.4 Ilustračný príklad 9

2 Monitorovanie informačných systémov **10**

2.1 Monitorovanie systému AiS2 10

2.2 Problémy 11

2.3 Prehľad súčasného stavu 11

2.4 Ciele práce 11

3 Algoritmus monitorovania informačného systému **13**

4 Typy monitorovania **17**

4.1 Monitorovanie systému v reálnom čase 17

4.2 Dlhodobé monitorovanie systému 17

4.3 Monitorovanie systému za účelom hĺbkovej analýzy 18

5 Nástroje na zber metrík **19**

5.1 Nástroj Metrics 19

5.2 Nástroj Statsd 20

5.3 Metrics vs Statsd 20

6 Nástroje na ukladanie a zobrazenie metrík **22**
6.1 Nástroj Graphite 22
6.1.1 Výhody nástroja Graphite 23
6.1.2 Nevýhody nástroja Graphite 23
6.2 Nástroj Zabbix 24
6.2.1 Výhody nástroja Zabbix 25
6.2.2 Nevýhody nástroja Zabbix 25
6.3 Nástroj Ganglia 25
6.3.1 Výhody nástroja Ganglia 28
6.3.2 Nevýhody nástroja Ganglia 28
6.4 Graphite vs Zabbix vs Ganglia 28

7 Monitorovanie systému AiS2 **29**
7.1 Testovanie nástrojov 29
7.2 Monitorovanie systému AiS2 v reálnom čase 29
7.3 Monitorovanie systému AiS2 za účelom hĺbkovej analýzy 31

Záver **33**

Prílohy **35**

Úvod

S informačnými systémami sa stretávame v dnešnej dobe čoraz častejšie. Mnohé systémy sa presúvajú z papierovej formy do elektronickej. Môže to byť spôsobené mnohými výhodami tohto prechodu, napríklad jednoduchším prístupom k informáciám, alebo len modernizáciou systému kvôli pokroku doby.

V tejto informačnej spoločnosti každý túži po informáciách. S informačnými systémami sa stretávame na školách, na univerzitách, na pracoviskách, v zdravotníctve a v rôznych iných oblastiach.

Vývojári informačných systémov sa snažia poskytnúť svojim zákazníkom informácie, ktoré hľadajú, kvôli ktorým navštívili ich informačný systém. Aby vedeli, či ich systém funguje spoľahlivo, potrebujú ho nejakým spôsobom odmerať.

Na meranie systémov slúžia metriky. Dávajú nám komplexnú informáciu o systéme, o jeho nedostatkoch, o tom, čo vyžaduje zlepšenie, o rýchlosti a spoľahlivosti systému. Pri zlepšovaní systému sú metriky veľmi užitočné. Odhalia slabé miesta systému, ktoré by sme pri bežnom testovaní nespozorovali.

Kapitola 1

Základné pojmy

1.1 Informačný systém

Informačný systém je zostavený z ľudí a počítačov, ktorí spracúvajú a interpretujú informácie. Niekedy sa tento termín používa na označenie softvéru, ktorý využíva počítačovú databázu alebo len označenie počítačového systému. [1]

V tejto práci sa budeme zameriavať na informačné systémy, ktoré sú vo forme webovej aplikácie.

1.2 Webová aplikácia

Webová aplikácia je aplikácia typu klient-server vytvorená pre prostredie internetu alebo intranetu. V softvérovom inžinierstve je aplikácia poskytovaná užívateľom z webového servera cez počítačovú sieť Internet, alebo jej vnútropodnikovú obdobu (intranet). Webové aplikácie sú populárne predovšetkým pre všadeprítomnosť webového prehliadača ako klienta. Ten sa nazýva tenkým klientom, pretože sám o sebe logiku aplikácie nepozná. Výhodou webových aplikácií je rovnaké užívateľské rozhranie kdekoľvek bez nutnosti inštalácie špeciálneho softvéru. Veľkou výhodou pre užívateľa, ale aj pre prevádzkovateľa aplikácie, je jednoduchá aktualizácia. Tá sa vykonáva len na jednom mieste - na serveri, na ktorom webová aplikácia beží. Nevýhodou je nutnosť internetového pripojenia. [2]

1.3 Metrika

Slovo metrika je často používaný pojem v oblasti monitorovania systémov. Metriky v oblasti monitorovania sú spôsoby a nástroje merania, vyjadrujú stav určitého systému, napríklad jeho kvality, efektívnosti a nadobúdajú pri tom rôzne hodnoty. Pri riadení sa používajú indikátory aj pre definíciu a dosahovanie cieľov, prípadne ich žiaducich hodnôt. Používajú sa tiež špecializované pojmy Performance indicator alebo Key performance indicator. [3] Metriky nám pomáhajú "odmerať" systém a tým nám dávajú komplexnú informáciu o systéme a pomáhajú odhaliť jeho slabé miesta. Vhodne zvolené metriky dokážu nasmerovať vývojárov správnym smerom a poukázať na možné problémy a tým v konečnom dôsledku pomôžu zlepšiť monitorovaný systém.

1.3.1 Rozdelenie metrík

Medzi základné delenia metrík patrí delenie na kvalitatívne a kvantitatívne metriky. Kvantitatívne metriky udávajú číselnú hodnotu, zatiaľ čo kvalitatívne metriky nečíselnú hodnotu. Kvantitatívne metriky sa ľahšie zobrazujú, interpretujú a sú zrozumiteľnejšie.

Metriky môžeme rozdeliť podľa druhu na:

- výkonnostné metriky
- softvérové metriky
- aplikačné metriky
- systémové metriky
- používateľské metriky

Výkonnostné metriky udávajú informáciu o výkone. Výkonnostná metrika môže byť napríklad maximálny počet záťaže, ktorú monitorovaný systém ešte znesie.

Ak vyvýjame softvér a chceme ho monitorovať, pomôžu nám softvérové metriky. Softvérovou metrikou môže byť napríklad hodnota pokrytia testami.

Informácie ohľadom fungovania a stavu aplikácie nám poskytujú aplikačné metriky. Môžeme zbierať metriky ako počet využívania konkrétnej funkcie.

Systémové metriky sú často používané metriky. Určujú stav, v ktorom sa systém nachádza. Príkladom systémovej metriky je zaťaženie procesora, počet voľného miesta na disku, či napríklad dostupnosť služby v systéme.

Ak potrebujeme zistiť informácie o používateľoch systému, zameriame sa na používateľské metriky. Používateľská metrika je napríklad čas od prihlásenia používateľa do systému po odhlásenie.

V tejto práci nás budú zaujímať aplikačné, systémové a používateľské metriky.

1.3.2 Aká je dobrá metrika?

Aby metrika plnila svoj cieľ, teda podala správnu informáciu o systéme, mala by byť:

- jednoduchá
- relevantná
- aktuálna
- okamžite ukončiteľná

Metrika by mala byť jednoduchá, aby bola ľahko interpretovateľná a pochopiteľná. Relevantná metrika je taká, ktorá pomôže splniť cieľ, dosiahnuť to, na čo bola určená. Neaktuálne metriky nám poskytnú možno už nepravdivé dáta, a preto by metriky mali byť čo najaktuálnejšie. Len aktuálne dáta nám popíšu systém spoľahlivo. Od metriky požadujeme, aby bola okamžite ukončiteľná, aby hneď po jej získaní bola pripravená na spracovanie a tak viedla k novým poznatkom. [4]

1.3.3 Typy metrík

Medzi základné typy metrík patria údaje o počte (counting), údaje o čase (timing), iné číselné hodnoty (celočíselné alebo desatinné), alebo niekedy aj nečíselné hodnoty (gauges).

Údajmi o počte môžeme získavať metriky ako napríklad počet požiadaviek, ktoré prišli na server za nejaký časový interval, počet volaní metódy v kóde, počet kliknutí na tlačidlo v informačnom systéme. Metriky tohto typu zbierame inkrementovaním počítadla v nástroji ako Metrics alebo Statsd v klientskej časti.

Často potrebujeme zmerať časový údaj ako napríklad čas pripojenia k databáze, čas obsluhy požiadavky alebo čas strávený vykonávaním určitého kusu kódu. Zmeraný čas zaznamenáme nástrojom na zbieranie metrík.

Niekedy je užitočné zbierať hodnotu, ktorá nezávisí od predchádzajúcich hodnôt napríklad počet objektov v rade, veľkosť dát odoslanej odpovede, alebo zaťaženie procesora. Tieto hodnoty môžu byť číselné alebo nečíselné. Tieto hodnoty pravidelne zbierame a odosielame na spracovanie a zobrazenie.

1.4 Ilustračný príklad

Dobrým príkladom informačného systému je akademický informačný systém. Monitorovanie takéhoto systému môže výrazne pomôcť k jeho vylepšeniu a rozvíjaniu.

Ak by mal vývojár informáciu o prehliadačoch, ktorými sa používatelia pripojili do systému, vedel by sa lepšie rozhodnúť, ktoré prehliadače má podporovať, aby pokryl čo najväčšiu skupinu používateľov.

Zaujímavou metrikou je aj rozlíšenie obrazoviek klientov, lebo s takouto informáciou sa dá lepšie rozvrhnúť používateľské rozhranie a zabezpečiť, aby klienti našli informácie, ktoré potrebujú rýchlejšie. Ak by napríklad väčšina používateľov systému k nemu pristupovala s malou obrazovkou cez mobilný telefón, vývojár by investoval viac času a prostriedkov do prispôsobenia rozhrania pre mobilné telefóny.

Ďalšou zaujímavou metrikou je čas odpovede na požiadavku od klienta. Ak zaznamenáme výrazne zvýšenie času odpovede vzhľadom na dlhodobý historický trend, vieme, že v systéme dochádza k problému a treba ho čo najskôr začať riešiť.

Kapitola 2

Monitorovanie informačných systémov

2.1 Monitorovanie systému AiS2

V našej bakalárskej práci chceme navrhnúť systém monitorovania akademického informačného systému AiS2. Problémom je to, že vývojári systému AiS2 nie sú správcami systému. Nemôžu preto získavať metriky ako napríklad zaťaženie serverov, voľná operačná pamäť, či iné systémové metriky, pretože tie sú skreslené virtualizáciou. Tento problém sa týka aj cloudových riešení, ktorých v dnešnej dobe rýchlo pribúda. Nemôžeme teda monitorovať priamo infraštruktúru aplikácie. Aby vývojári zabezpečili hladký chod systému, musia sa spoľahnúť na aplikačné metriky. Tie nám dajú informácie o aktuálnom stave aplikácie a o jej správaní.

Keďže systém AiS2 beží vo viacerých inštaláciách, je potrebné monitorovať viacero inštalácií tohto systému samostatne, aj ako celok. Chceme ich vedieť porovnať medzi sebou, aby sme vedeli odlíšiť problémy v konkrétnej inštalácii a globálne problémy týkajúce sa systému samotného. Na porovnanie inštalácií potrebujeme získané metriky mať na jednom mieste, na jednom serveri, čo môže výrazne skomplikovať zber metrík. Potrebujeme zobrazovať agregované metriky zo všetkých inštalácií. Bude jednoduchšie inštalácie porovnať a analyzovať nazbierané metriky.

Ďalším cieľom monitorovania systému AiS2 je optimalizácia interakcie systému s používateľom. Chceme detekovať zle navrhnuté dialógy, ich neprehľadné umiestnenie v systéme a analyzovať interakciu s používateľmi, napríklad aj dynamickým upravovaním používateľského rozhrania.

2.2 Problémy

Hlavným cieľom monitorovania informačného systému je identifikovať problémy skôr ako používateľ. Keď zbadá problém používateľ, už je neskoro. Medzitým, ako nahlási problém vývojárom informačného systému, tento problém už mohlo zaznamenať množstvo ďalších používateľov. Vývojári potom musia rýchlo problém identifikovať a opraviť, čo môže byť netriviálna záležitosť. Ak by sme problém zaznamenali hneď ako nastane, mali by sme viac času na jeho riešenie.

Problém môže nastať, ak máme metrík veľa. Monitorovanie systému nesmie systém výrazne zaťažiť, lebo by monitorovanie nebolo efektívne.

2.3 Prehľad súčasného stavu

V tomto čase je množstvo informačných systémov, ktoré sú neustále vylepšované a zdokonaľované, a tak je potrebné ich monitorovať, aby sa dali ešte viac zlepšiť a prispôsobiť zákazníkom. Každý systém je špecifický, a tak aj jeho monitorovanie je jedinečné. Vývojári sa snažia upraviť monitorovanie ich systému a prispôsobiť ho tak, aby im poskytol čo najviac vedomostí o ich systéme.

Existuje niekoľko nástrojov, ktoré čiastočne ponúkajú riešenia našich problémov či už ide o zbieranie metrík alebo ich vizualizáciu. Tieto nástroje sú však väčšinou zamerané na zbieranie systémových metrík a my potrebujeme zbierať hlavne aplikačné metriky. Patria tu ako platené (Datadog, Stackdriver, Librato), tak aj open-source nástroje (Statsd, Metrics, Cacti, Graphite, Zabbix, Ganglia, Naggios). My sa zaoberáme open-source nástrojmi.

2.4 Ciele práce

Hlavným cieľom našej bakalárskej práce je navrhnúť spôsob monitorovania informačného systému v reálnom čase za účelom zhromažďovania informácií o výkone systému a používateľských preferenciách. Chceme navrhnúť systém monitorovania akademického informačného systému AiS2.

Ďalším cieľom je navrhnúť dátovú štruktúru pre uložené metriky z jednotlivých inštalácií a vytvoriť jednoduché štatistické prehľady nad zozbieranými údajmi v závislosti od charakteru údajov. Potrebujeme určiť čo najvhodnejší spôsob ukladania met-

rík, ich agregácie z viacerých inštalácií na jedno miesto a vhodným spôsobom ich zobraziť.

Chceme tiež analyzovať možnosti využitia monitorovania pre účely optimalizácie interakcie s používateľom, teda zistiť, ako sa monitorovaním dá dosiahnuť skvalitnenie informačného systému pre jeho používateľov.

Keďže už existujú nástroje, ktoré čiastočne riešia naše problémy, naším cieľom bude aj porovnať tieto nástroje, popísať si ich výhody a nevýhody a určiť nástroje, ktoré by boli najvhodnejšie pre monitorovanie informačného systému AiS2.

Kapitola 3

Algoritmus monitorovania informačného systému

Ak chceme získať užitočné dáta o systéme, aby sme ich mohli analyzovať, musíme ich najprv zozbierať. Pred zbieraním metrík si treba najprv dobre premyslieť, čo chceme zbierať, aké metriky sú pre nás vhodné a aký cieľ chceme zberom metrík dosiahnuť. Treba si dobre a jasne definovať metriky, aby sme z nich mali čo najväčší osoh.

Monitorovanie informačného systému od prípravy na monitorovanie až po analýzu výsledkov monitorovania sa dá popísať nasledujúcim algoritmom:

1. Určenie cieľov monitorovania informačného systému.
2. Výber vhodných metrík.
3. Výber nástrojov na monitorovanie a zobrazovanie metrík.
4. Integrácia nástrojov a metrík do informačného systému.
5. Zber metrík.
6. Analýza získaných metrík.
7. Vyvodenie dôsledkov monitorovania.

Určenie cieľov monitorovania informačného systému

Určenie cieľov monitorovania informačného systému je dôležité. Musíme si dobre premyslieť, na čo chceme monitorovanie zamerať, čo chceme zlepšiť, čo potrebujeme odmerať. Taktiež je dôležité premyslieť si, čo urobíme s výsledkom monitorovania a nakoľko bude výsledok merania relevantný a pravdivý. To závisí aj od metrík, ktoré si

zvolíme, od času a obdobia, v ktorom systém budeme monitorovať. Treba si ujasniť problémy, ktoré chceme vyriešiť pomocou monitorovania.

Výber vhodných metrík

Na to, aby sme monitorovaním dosiahli stanovené ciele, musíme si zvoliť vhodné metriky, ktoré nám poskytnú odpovede na naše otázky. Každý systém je odlišný, a tak neexistuje univerzálny návod, aké metriky si zvoliť.

Treba sa zamyslieť nad tým, aké údaje by nám pomohli, keby sme ich mali k dispozícii. Ak potrebujeme získať informácie o používateľoch a o ich správaní v systéme, pomohli by nám používateľské metriky. Ak potrebujeme získavať údaje o stave aplikácie, potom je pre nás vhodné zbierať aplikačné metriky. Keď chceme zistiť, aký je výkon systému a ako sa správa systém pri zaťažení a pod akou záťažou je v konkrétnych obdobiach, potom by sme sa mali zamerať na systémové metriky.

Výber nástrojov na monitorovanie a zobrazovanie metrík

Ak už máme definované metriky, ktoré chceme zbierať, v ďalšej časti prípravy na monitorovanie si zvolíme vhodné nástroje na zber metrík a na ukladanie a zobrazovanie metrík. Môžeme si zbieranie, ukladanie a zobrazovanie metrík navrhnúť a implementovať aj po svojom, ale keďže je monitorovanie informačných systémov rozšírené, existujú už špecializované nástroje, ktoré nám pomôžu s monitorovaním a zabezpečia nám kompatibilitu s ďalšími nástrojmi.

Nástrojov, ktoré máme k dispozícii, je niekoľko. V tejto práci si rozoberieme open-source nástroje na zbieranie metrík Statsd a Metrics a nástroje na zobrazovanie metrík Graphite, Zabbix a Ganglia. Všetky nástroje sú odlišné a majú svoje výhody a nevýhody, ktoré si podrobnejšie popíšeme. Nástroje si vyberieme podľa funkcionality, náročnosti integrácie a podľa veľkosti informačného systému.

Integrácia nástrojov a metrík do informačného systému

Po výbere nástrojov na monitorovanie nasleduje integrácia vybraných nástrojov a metrík do systému. Zložitosť integrácie a konfigurácie závisí od nástrojov, ktoré sme si zvolili a od komplexnosti monitorovaného systému. Metriky zbierame v podobe counterov, timerov a gauges, prípadne im podobným alebo inak pomenovaným alternatívnym spôsobom podľa vybraných nástrojov.

Zber metrík

Ak máme nakonfigurované nástroje na zber metrík a funkčné prepojenie nástroja na zber metrík s nástrojom na ukladanie a zobrazovanie metrík, nasleduje fáza samotného zbierania metrík a monitorovania informačného systému. V tejto fáze sledujeme metriky, ktoré nám nástroj na zobrazovanie metrík vykresľuje v prehľadných grafoch. Niektoré zobrazovacie nástroje umožňujú aj agregáciu grafov pre jednotlivé metriky, to znamená, že súvisiace metriky si zobrazíme v jednom spoločnom grafe.

Niektoré nástroje, alebo ich jednoduché prepojenie s ďalším nástrojom ponúkajú možnosť notifikácie administrátora o udalostiach, ktoré nastali v systéme. Môžeme tak byť notifikovaný napríklad o priveľkej záťaži na systém, o nedostatku voľnej operačnej pamäte alebo o pridlhom čase čakania klienta na odpoveď zo servera. To nám pomôže identifikovať aktuálne nastávajúci problém, ktorý je potrebné čo najskôr vyriešiť. Takisto nám to pomôže odhaliť problém skôr, ako ho odhalí používateľ nášho systému.

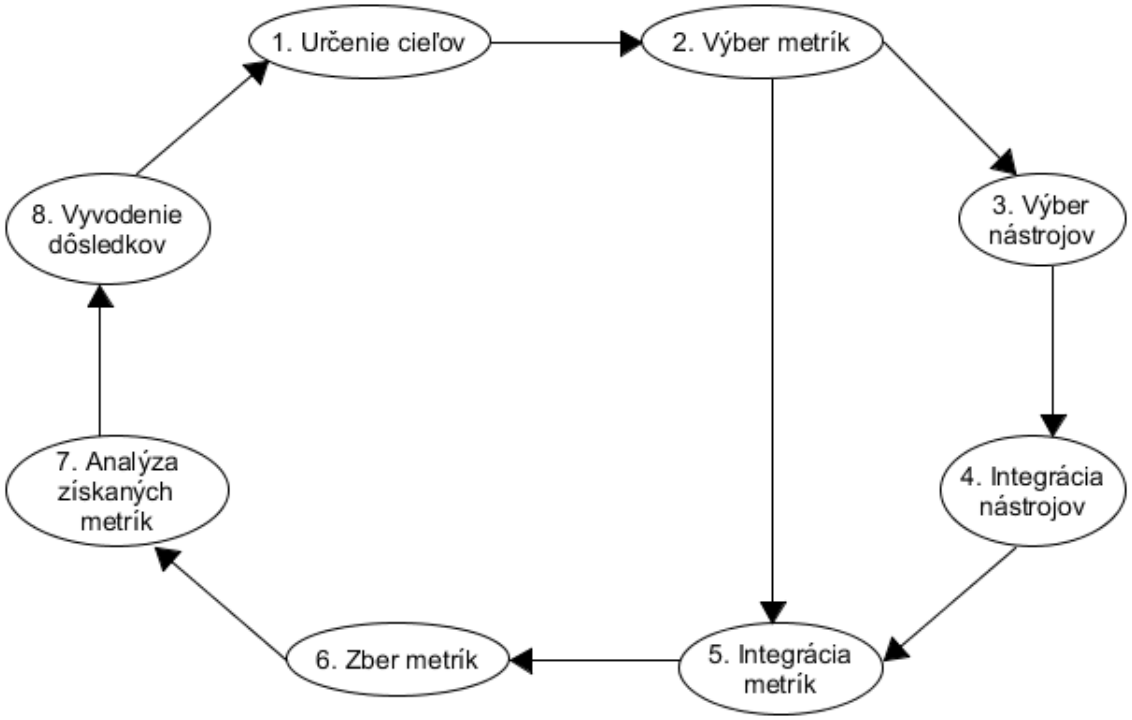
Analýza získaných metrík

Ak už zbierame metriky dlhšiu dobu, máme dostatok údajov na ich analýzu a ďalšie spracovanie. Môžeme analyzovať, ako sa systém správa v určitých obdobiach, ako sa správajú používatelia v systéme a ako sa správa systém, keď je pod záťažou. To nám pomôže pripraviť sa na ďalšiu záťaž, alebo zistíme, že potrebujeme zbierať ďalšie metriky, na ktoré sme na začiatku nemysleli, alebo sa nám zdali irelevantné.

Vyvodenie dôsledkov monitorovania

V ďalšej fáze vyvodíme dôsledky monitorovania. Zamyslíme sa, či nám monitorovanie pomohlo splniť ciele, ktoré sme si na začiatky stanovili a ako nám to pomohlo zlepšiť systém. Porozmýšľame, čo nám ešte treba zlepšiť, aké závery nám plynú z monitorovania a určíme ďalšie kroky k ešte lepšiemu monitorovaniu.

Monitorovanie informačného systému je dlhodobý proces a počas monitorovania sa môže meniť a upravovať s meniacim sa systémom a už získanými poznatkami z monitorovania. Môžeme pridávať nové metriky aj s novou funkcionalitou alebo tie nepodstatné metriky zmazať. V tejto fáze upravíme systém podľa výsledkov, ktoré sme získali monitorovaním. Napríklad, ak sme zistili, že náš systém potrebuje optimalizáciu dopytov do databázy, optimalizujeme ich a sledujeme, aký to ma dopad na systém



Obr. 3.1: Životný cyklus monitorovania informačného systému

a na rýchlosť odpovede na požiadavky od klientov.

Na obrázku 3.1 môžeme vidieť životný cyklus monitorovania informačného systému.

Kapitola 4

Typy monitorovania

4.1 Monitorovanie systému v reálnom čase

Monitorovanie systému v reálnom čase nám pomôže detekovať problém skôr, ako ho zbadá používateľ. Prehľadné grafy zobrazujúce nazbierané metriky nám ukážu dôležité informácie o stave systému. Niektoré nástroje nás dokonca upozornia, ak v systéme nastane problém. Po vyriešení problému zas z grafov vyčítame, či sme detekovaný problém vyriešili a ako dobre sme ho vyriešili.

4.2 Dlhodobé monitorovanie systému

Krátkodobé merania systému nám podajú užitočnú informáciu o aktuálnom stave systému a o jeho používateľoch. Nás však zaujímajú aj dlhodobé merania. Tie nám povedia, kedy v priebehu roka je systém najviac zaťažený, kedy najmenej, a aké funkcionality používatelia využívajú v konkrétnych obdobiach. To sa nám pomôže pripraviť na najväčšiu záťaž.

Po vylepšení niektorej funkcionality v systéme vieme porovnať nové hodnoty metrík s historickými hodnotami metrík pre danú funkcionalitu. Dozvieme sa tak, či sme funkcionalitu, naopak, nezhoršili a či bude schopná vydržať záťaž.

4.3 Monitorovanie systému za účelom hĺbkovej analýzy

Dlhodobými meraniami vieme zaznamenávať vzory správania používateľov v našom informačnom systéme. Takýmto vzorom môže byť v akademickom informačnom systéme napríklad:

- Používatelia si najčastejšie po prihlásení do aplikácie prečítajú nové správy.
- Po prečítaní správ sa odhlásia alebo:
  - Ak je začiatok semestra, pozrú si rozvrh hodín.
  - Ak je stred semestra, pozrú sa na priebežné hodnotenie.
  - Ak je koniec semestra, prihlásia sa na skúšku.

Ak získame takéto vzory, môžeme napríklad dynamicky meniť používateľské rozhranie počas priebehu semestra a sprístupniť tak najčastejšie hľadané informácie podľa vzorov na viditeľnejšie miesta v našom systéme.

Dlhodobými meraniami systému vieme získať aj ďalšie zaujímavé informácie. Nad množstvom vyzbieraných metrík sa dá robiť hĺbková analýza v podobe dolovania dát. No na hĺbkovú analýzu budeme potrebovať väčšinou odlišné dáta od tých, ktoré pre nás majú zmysel pri monitorovaní v reálnom čase. Budú nás zaujímať iné metriky a aj ciele takejto analýzy budeme mať odlišné.

Kapitola 5

Nástroje na zber metrík

Ak už máme jasne definované metriky, môžeme ich začať zbierať. Zber metrík je často riešený problém, v zbere metrík nám môžu pomôcť rôzne nástroje. Medzi najpopulárnejšie open-source nástroje na zber metrík patria Metrics a Statsd. Sú to nástroje určené na odosielanie nameraných metrík do nástrojov na uloženie a zobrazenie metrík v podobe grafov.

Integrácia oboch nástrojov si však vyžaduje určité zmeny v kóde systému. Každý systém je jedinečný, a teda aj jeho monitorovanie je odlišné. To si však žiada odlišné metriky a tie sa zbierajú na odlišných miestach v kóde systému.

Oba nástroje sú funkcionalitou podobné, no odlišné z hľadiska integrácie a architektúry. Riešia ten istý problém - definujú metriky v kóde systému a fungujú ako cache pre metriky a po nastavenom časovom intervale ich posielajú ďalším nástrojom. Naším čiastočným cieľom je porovnať ich. V nasledujúcich odsekoch si popíšeme ich výhody a nevýhody.

5.1 Nástroj Metrics

Metrics je open-source java knižnica na zber metrík. Získané metriky posiela do nástrojov na zobrazovanie metrík ako napríklad Graphite alebo Ganglia, alebo ukladá do databázy prostredníctvom logovacích nástrojov.

Keďže Metrics je java knižnica, integrácia nástroja do java webovej aplikácie je záležitosť pridania niekoľkých závislostí. Následne definujeme registre, v ktorých budú metriky uložené. Ďalej je potrebné nastaviť reportovanie do ďalších nástrojov. Potom na vhodných miestach v aplikácii získame metriky pomocou counterov, timerov a gauges, ktoré chceme zbierať. Nástroj Metrics nám pomôže dostať namerané metriky

do ďalších nástrojov.

Asi najväčšou konkurenciou k Metrics je nástroj Statsd.

5.2 Nástroj Statsd

Open-source nástroj Statsd je určený na zber metrík a následné preposlanie do nástrojov na spracovanie a zobrazenie. Má klientov pre množstvo programovacích jazykov ako napríklad node.js, java, python, ruby, perl, php, c, cpp, .net, či dokonca plugin pre wordpress.

Statsd klient na rozdiel od Metrics posiela získane metriky cez sieťovú komunikáciu na otvorený UDP port na oddelený Statsd server, ktorý ich prijíma, akumuluje a odosiela do ďalších nástrojov. Slúži ako rýchla cache pre metriky. Umožňuje posielať dáta do nástrojov ako napríklad Graphite, Ganglia, Librato, Zabbix, či Mongo, Mysql, Datadog a mnoho ďalších.

Architektúra tohto nástroja je trochu zložitejšia.

5.3 Metrics vs Statsd

Obe nástroje slúžia na zbieranie metrík, majú však rozdielnu architektúru. Ponúkajú rôzne možnosti integrácie s ďalšími nástrojmi. Kvôli prehľadnosti sú základné rozdiely popísané v tabuľke 5.1. Údaje v tabuľke 5.1 sú aktuálne vzhľadom k času písania bakalárskej práce.

Tab. 5.1: Porovnanie nástrojov Metrics a Statsd

Atribút porovnania	Metrics	Statsd
Architektúra	Server totožný s klientom	Server a klient oddelený
Integrácia s ďalšími nástrojmi	Ganglia, Graphite, Librato, Spring, Sematext, Wicket, Guice, Scala, Clojure, Cassandra, Elasticsearch, Statsd, Datadog, Influxdb, Cdi, Aspectj, Apache Camel	Amqp, Ganglia, Librato, Socket.io, Statsd, Mongo, Mysql, Datadog, Monitis, Instrumental, Hosted Graphite, Zabbix, Opentsdb, Influxdb, Stackdiver, Couchdb
Implementácia Servera	Iba Java	Node.js, Python, Ruby, C, Go, Clojure, Perl
Implementácia Klienta	Klient totožný so serverom	Node.js, Java, Python, Ruby, Perl, Php, Clojure, C, Cpp, .Net, Go, Apache, Io, Varnish, PowerShell, JavaScript, Cocoa, ActionScript, Wordpress
Licencia	Apache 2	MIT
Zložitosť integrácie do java webovej aplikácie	Import knižnice, definovanie metrík, nastavenie reportovania	Inštalácia Node.js, Statsd servera, import Statsd klientskej knižnice, definovanie metrík, inštalácia reportovacích backendov, konfigurácia Statsd servera
Počet prispievateľov	127	154
Počet verzií	51	11
Prvý príspevok do repozitára	27.4.2011	30.12.2010
Počet príspevkov do repozitára	1958	823

Kapitola 6

Nástroje na ukladanie a zobrazenie metrík

Nástroje na zbieranie metrík majú často za úlohu agregovať metriky a preposielať ich do ďalších nástrojov. Existujú rôzne nástroje na ukladanie a zobrazenie metrík. Niektoré z nich sú komerčné, iné open-source. Medzi často používané open-source nástroje patria nástroje Graphite, Zabbix a Ganglia. Tieto nástroje vedia prijať dáta z iných nástrojov, uložiť ich v databáze a vhodným spôsobom ich zobraziť, aby sa dali ľahšie interpretovať.

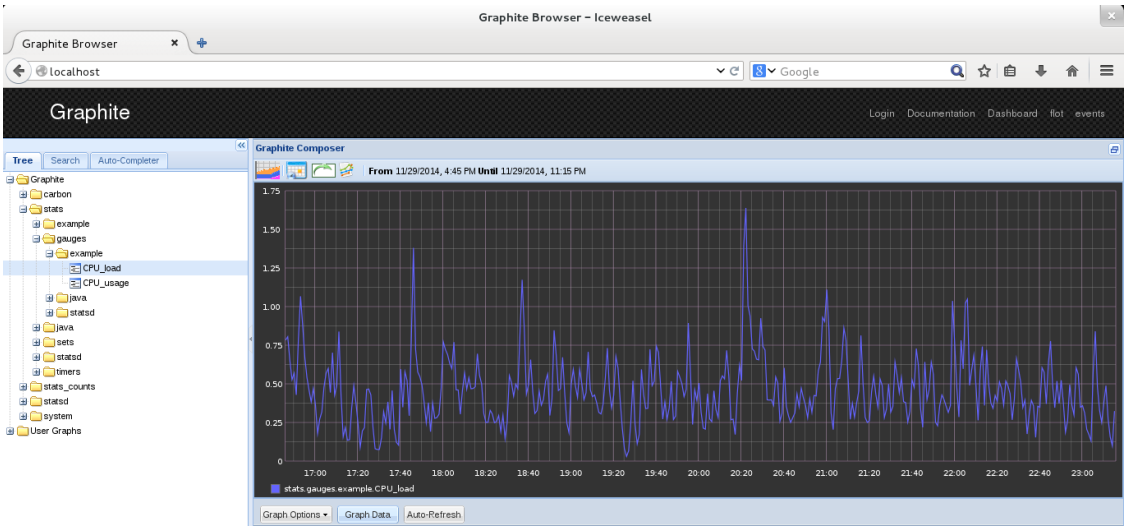
Rôzne nástroje na zbieranie metrík majú rôznu architektúru a ponúkajú rozličnú funkcionalitu. Záleží od veľkosti a komplexnosti systému a aj od požadovanej funkcionality, ktorý nástroj si vyberieme.

Metriky sa najčastejšie zobrazujú v podobe histogramov alebo čiarových diagramov cez používateľské rozhranie vo forme webovej aplikácie.

6.1 Nástroj Graphite

Nástroj Graphite je veľmi obľúbeným nástrojom v oblasti zobrazovania metrík. Nástroje ako napríklad Statsd alebo Metrics pošlú získané metriky do Graphitu a ten ich uloží pomocou integrovaného nástroja Carbon do Whisper databázy. Whisper databáza je veľmi podobná RRD (round robin database) databáze, ale je prispôsobená a napísaná v pythone. Kód graphitu je napísaný v programovacom jazyku python. Metriky stačí poslať do nástroja Graphite a on ich sám zaradí do stromu metrík podľa prefixu. Po inštalácii sa nevyžaduje žiadna dodatočná konfigurácia.

Na obrázku 6.2 môžeme vidieť používateľské rozhranie nástroja Graphite a čiarový



Obr. 6.2: Používateľské rozhranie nástroja Graphite

diagram pre metriku zaťaženie procesora.

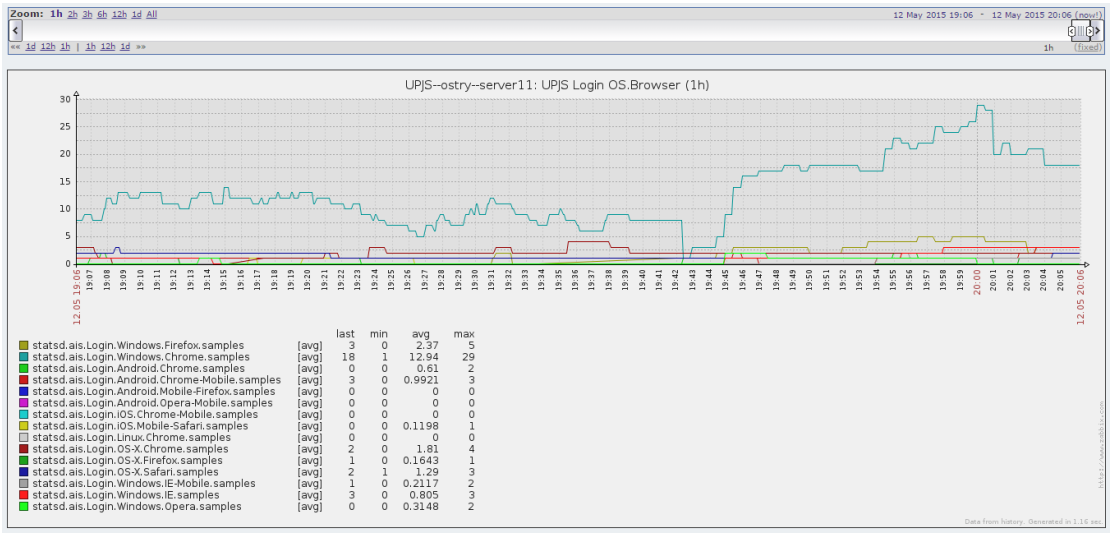
Nástroj Graphite je jednoduchý open-source nástroj určený na ukladanie a zobrazovanie metrík v podobe grafov. Na vykresľovanie grafov využíva python knižnicu PyCairo. V praxi sa používa pri jednoduchých projektoch, ak nám postačuje získané metriky zobrazovať. Pri monitorovaní komplexnejších systémov sa používajú zložitejšie nástroje ako napríklad Zabbix alebo Ganglia.

6.1.1 Výhody nástroja Graphite

Graphite je jednoduchý nástroj určený pre monitorovanie malých jednoduchších systémov. Hlavnou výhodou je, že po inštalácii nástroja sa nevyžaduje žiadna konfigurácia metrík. Ďalšou výhodou je jednoduché zobrazenie viacerých metrík v jednom grafe. Graphite je zameraný na monitorovanie systémov v reálnom čase.

6.1.2 Nevýhody nástroja Graphite

Nástroj Graphite je zameraný len na vykresľovanie metrík a neponúka žiadnu dodatočnú funkcionalitu. Jeho nevýhodou je, že má komplikovanú inštaláciu. Počas jeho inštalácie treba nainštalovať a nakonfigurovať všetky potrebné moduly, čiže Carbon, Whisper, Graphite web, Pycairo a Apache.



Obr. 6.3: Graf prehliadačov používateľov systému AiS2 v Zabbixe

6.2 Nástroj Zabbix

Ďalším používaným nástrojom je Zabbix. Jeho kód je napísaný v jazyku C a používateľské rozhranie v podobe webovej aplikácie je napísané v jazyku PHP. Na ukladanie metrík používa Mysql databázu. Na obrázku 6.3 môžeme vidieť graf operačných systémov a prehliadačov používateľov systému AiS2 v Zabbixe.

Zabbix je komplexný nástroj určený aj pre väčšie systémy. Okrem vizualizácie dokáže zbierať vlastné metriky, pomáha detekovať problém a vie o ňom notifikovať administrátorov. Je to omnoho mohutnejší nástroj ako Graphite.

K informačnému systému patrí aj hardvér, na ktorom systém beží. Informácie o hardvéri a rôzne iné systémové metriky môžeme zbierať pomocou modulu Zabbix agent. Zabbix agent pravidelne posiela získané udaje zo stroja, kde je nasadený na Zabbix server. Môžeme tak jednoducho monitorovať napríklad výkon systému, sieťové zariadenia, virtuálne stroje, databázy, Java virtual machine alebo Java aplikačné servery.

Zabbix tiež umožňuje na zber dát použiť SNMP a IPMI agentov alebo Zabbix sender nástroj, ktorým do Zabbixu vieme poslať ľubovoľné metriky z nástrojov na zbieranie metrík.

Zabbix môže slúžiť aj na detekciu problémov so systémom. Detekcia problémov v Zabbixe funguje pomocou definovania triggerov, ktoré odhalia, že niečo nefunguje. Môže to byť napríklad výpadok stroja, málo voľného miesta na disku monitorovaného zariadenia alebo veľa pripojených používateľov. V triggery sa nastavý prah, ktorým

určíme maximálnu akceptovateľnú hodnotu metriky.

Ak sa táto hodnota prekročí, Zabbix nás o tom môže notifikovať alebo vykoná nastavenú akciu. Môže nás notifikovať prostredníctvom e-mailu, SMS správy alebo správy cez Jabber. Môžeme si tiež nastaviť, aké dáta sa nám v notifikácii zobrazia. Okrem notifikačnej správy Zabbix vie po detekcii problému vykonať napríklad shell príkaz cez ssh.

6.2.1 Výhody nástroja Zabbix

Veľkou výhodou nástroja Zabbix je to, že na ukladanie metrík používa Mysql databázu. Dáta tak máme uložené v relačnej databáze a vieme sa na ne jednoducho dopytovať. To má potenciál na zber dlhodobých metrík za účelom hĺbkovej analýzy a dolovanie dát z týchto údajov.

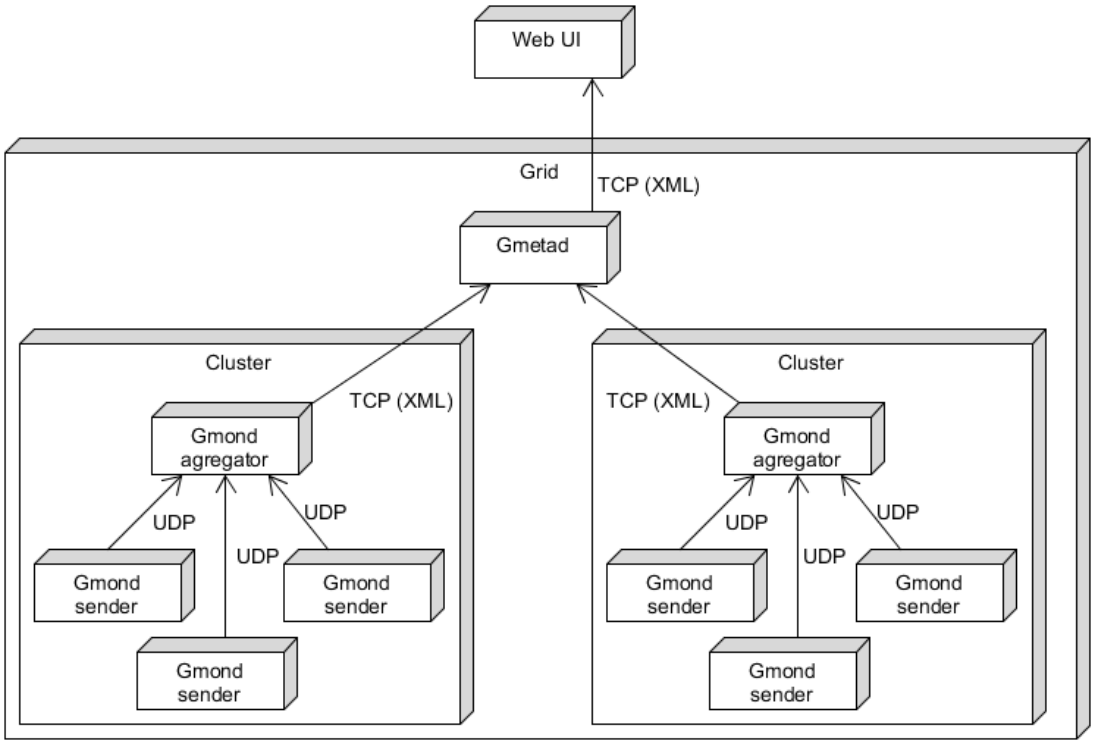
6.2.2 Nevýhody nástroja Zabbix

Hlavná nevýhoda Zabbixu je to, že každá metrika, ktorú chceme monitorovať musí byť dopredu nakonfigurovaná. Ak máme metrík veľa, úvodná konfigurácia je pracná. Síce sa v Zabbixe dajú použiť šablóny, ktoré trochu zjednodušia konfiguráciu, ale stále musíme všetky metriky nakonfigurovať ručne. V systéme AiS2 sme tieto šablóny použili na definovanie rovnakých metrík pre rôzne servery. Pri každej novej funkcionalite informačného systému s predpokladom, že ju chceme monitorovať, je potrebné nakonfigurovať nové metriky aj v Zabbixe.

6.3 Nástroj Ganglia

Ďalším obľúbeným nástrojom je Ganglia. Ganglia je open-source nástroj zameraný na distribuované monitorovanie celých clustrov či gridov. Je naozaj rozšírený, používajú ho napríklad známe spoločnosti ako Twitter, Flickr, National Aeronautics and Space Administration (NASA), Wikipedia, CERN, Cisco, HP, Microsoft, Dell, nVidia, U.S. Air Force. Viac v [5].

Metriky do Ganglie môžeme poslať pomocou Gmond modulov, Gmetric modulu alebo inými nástrojmi, ako napríklad Statsd. Ak posielame metriky do Ganglie, nemusíme ich mať dopredu nakonfiguroavné. Gmetad vytvorí nový RRD súbor pre každú novú metriku, čo značne uľahčí konfiguráciu, ak metrík máme veľa. Webové používateľské rozhranie potom zobrazí základný graf pre každú metriku.



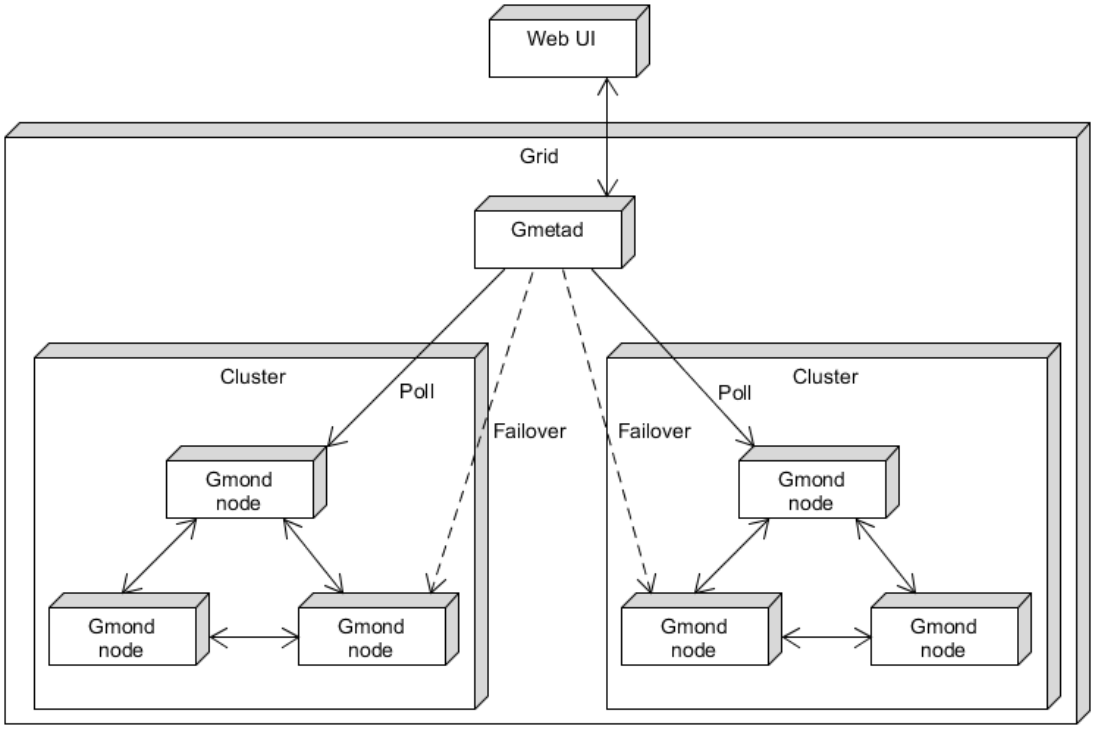
Obr. 6.4: Architektúra Ganglie s použitím unicastu

Ganglia na posielanie metrík na centrálny server používa dvoch démonov a to Ganglia Monitoring Daemon (gmond) a Ganglia Meta Daemon (gmetad). Gmond môže posielať alebo prijímať metriky, ktoré si drží v pamäti. Gmetad v rámci gridu ťahá dáta z jedného gmond démona z každého clustra. Gmond moduly so sebou komunikujú cez UDP protokol a do gmetad posielajú dáta v štruktúre XML cez TCP protokol. Gmetad si medzi sebou posielajú metriky cez TCP vo forme XML. Gmond medzi sebou komunikujú buď prostredníctvom unicastu alebo multicastu.

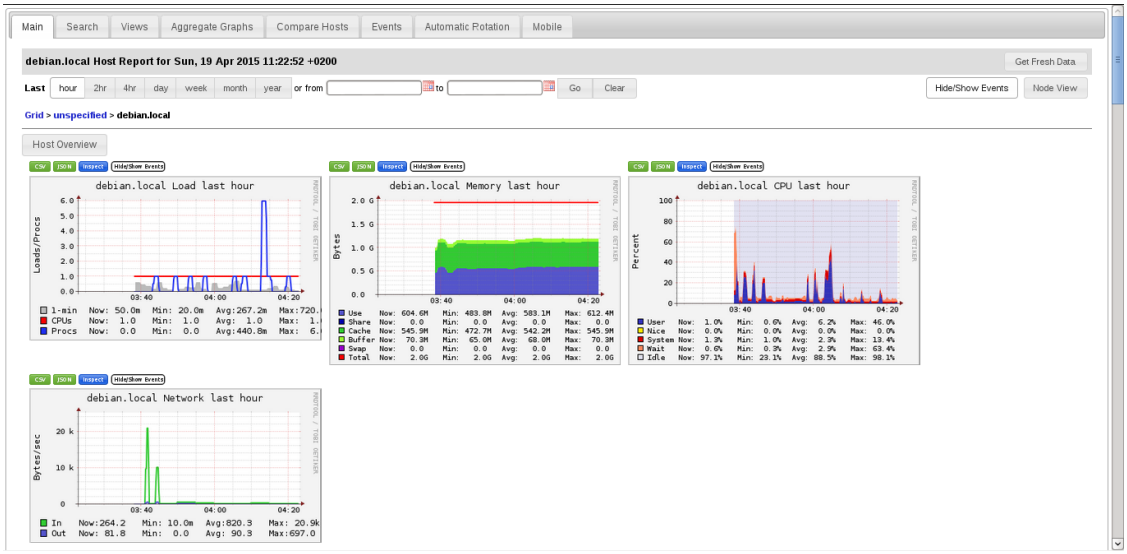
Ak použijeme unicast, komunikácia zo všetkých gmondov smeruje do jedného centrálneho gmond modulu v rámci clustera. Gmetad potom zbiera dáta zo všetkých centrálnych gmond modulov. Táto konfigurácia je výhodná, ak máme veľmi veľa uzlov (1000). Architektúra s použitím unicastu je popísaná na obrázku 6.4.

Ak máme menší počet uzlov v gride, viac sa nám oplatí použiť multicast. Po pridaní nových uzlov do clustra nie je potrebná dodatočná konfigurácia. Architektúra s použitím multicastu je popísaná na obrázku 6.5.

Ganglia priamo nepodporuje detekciu problémov v systéme. Umožňuje však prepojenie s nástrojom Naggios, ktorý je na to určený. Na obrázku 6.6 môžeme vidieť používateľské rozhranie nástroja Ganglia.



Obr. 6.5: Architektúra Ganglie s použitím multicastu



Obr. 6.6: Používateľské rozhranie nástroja Ganglia

6.3.1 Výhody nástroja Ganglia

Asi najväčšou výhodou nástroja Ganglia je to, že dokáže monitorovať celé clustre a gridy. Je určená na agregáciu metrík z viacerých serverov a z komplexných informačných systémov. To nám výrazne zjednoduší porovnávanie metrík získaných z viacerých inštalácií.

Na ukladanie metrík Ganglia používa RRD databázu, takže metriky nemusíme v Ganglii konfigurovať. Výhodné je aj jednoduché prepojenie s nástrojom Naggios.

Ďalším plusom je možnosť použitia multicastu. Ak náhodou niektorý monitorovaný uzol vypadne, gmetad si vyžiada údaje od ďalšieho gmond modulu. Keďže každý gmond má informáciu o metrikách všetkých ostatných gmond modulov v tom istom clusteri, metriky sa nestratia, ale odošlú sa prostredníctvom iného gmond modulu. Dokonca pridanie nových uzlov do clustera je jednoduché a bez ďalšej konfigurácie.

6.3.2 Nevýhody nástroja Ganglia

V istom zmysle je nevýhodou Ganglie to, že nevie alertovať problémy v systéme. To sa dá docieliť prepojením Ganglie s nástrojom Naggios. Musíme teda integrovať do systému až dva nástroje.

Pôvodne sme si vybrali nástroj na ukladanie a zobrazovanie metrík pre systém AiS2 Gangliu. Narazili sme však na problém, že v čase písania tejto práce ešte nebola Ganglia dostupná pre najnovší Red Hat.

Ďalšia nevýhoda Ganglie je že nemá cache pre metriky [6], ale takáto cache je obsiahnutá v oboch nástrojoch na zbieranie metrík. Ich integrácia je ale cez Gmetric, čo ale obchádza infraštruktúru s použitím Gmond modulov, čo je jedným zo základných benefitov návrhu nástroja Ganglia.

6.4 Graphite vs Zabbix vs Ganglia

Ako sme už písali, nástroj Graphite je vhodný na monitorovanie menších systémov. Pre väčšie informačné systémy je lepšie použiť komplexnejší nástroj, ako napríklad Zabbix alebo Gangliu. Tie nám ponúkajú viac užitočných funkcií, ktoré môžeme využiť. Zabbix vie detekovať problémy a tie jednoduchšie aj vyriešiť (napríklad reštartom zlyhanej služby), Ganglia zas dokáže monitorovať celé clustre a gridy a ponúka nám jednoduchú agregáciu grafov. Záleží od komplexnosti systému, ktorý nástroj si vyberieme.

Kapitola 7

Monitorovanie systému AiS2

7.1 Testovanie nástrojov

Aby sme vedeli vyššie spomenuté nástroje porovnať a vybrať si vhodné nástroje na zbieranie a zobrazovanie metrík kvôli monitorovaniu systému AiS2, nástroje sme si najskôr vyskúšali.

Pre testovacie účely sme si vytvorili jednoduchú webovú aplikáciu. Keďže je systém AiS2 napísaný prevažne v programovacom jazyku java, za formu webovej aplikácie sme si zvolili JSP (Java Server Pages) stránky. Vytvorené JSP stránky môžeme vidieť v prílohách A - C.

Vo virtuálnom stroji sme si nainštalovali nástroj Statsd server, Graphite, Zabbix a Gangliu. Po kliknutí na tlačidlo „Special Feature“ v prílohe A sa zavolá príslušná JSP stránka s názvom „sender“, ktorá má inštanciu Statsd klienta (príloha B), alebo integrovanú knižnicu Metrics (príloha C), ktoré odošlú metriku o stlačení tlačidla do príslušného nástroja na uloženie a zobrazenie metriky.

Týmto sme si spomínané nástroje vyskúšali a nabrali sme skúsenosti s inštaláciou, komfortom používania a overili si možnosti integrácie vybraných nástrojov do systému AiS2.

7.2 Monitorovanie systému AiS2 v reálnom čase

Pre monitorovanie systému AiS2 v reálnom čase sme sa rozhodli použiť nástroj Metrics v kombinácii s nástrojom Ganglia.

Metrics sme si vybrali aj preto, lebo komunikácia medzi Statsd klientom a Statsd serverom po sieti môže spôsobiť zbytočné zaťaženie siete. Metrics nepotrebuje odde-

leného klienta od servera. Ďalším dôvodom pre výber Metrics bol ten, že je to java knižnica a vzhľadom k tomu, že systém AiS2 je v jave, integrácia nástroja do systému nebola zložitá. Keby AiS2 nebol v jave, najpravdepodobnejšie by sme sa rozhodli pre nástroj Statsd.

Gangliu sme si vybrali hlavne preto, že ponúka riešenie pre agregáciu metrík z viacerých inštalácií systému AiS2 na jedno miesto. Metriky zo všetkých inštalácií potom môžeme zobraziť prehľadne v jednom grafe a porovnať jednotlivé inštalácie medzi sebou.

Gangliu sa nám však nepodarilo nasadiť na najnovšiu verziu Red Hat distribúcie, pretože Ganglia v čase písania tejto práce ešte nebola pre túto verziu dostupná. Vybrali sme si teda nástroj Zabbix, ktorý má vďaka používaniu MySQL databázy potenciál okrem zbierania metrík v reálnom čase zbierať aj metriky dlhodobo za účelom hĺbkovej analýzy.

Zabbix sme teda nasadili na testovacie, vývojové a ostré prostredie do akademického informačného systému AiS2. Určili sme si, že budeme zbierať metriky ohľadom funkcionality systému - počet spustení dialógov a metriky ohľadom operačného systému a prehliadača používateľov systému, za časový interval 3 sekundy. Po nasadení zberu metrík na ostrý server sme v špičke zaznamenali výrazné zaťaženie siete kvôli trojsekundovému intervalu odosielania metrík. Rozhodli sme sa preto tento interval zmeniť na tridsať sekúnd.

Ešte skôr, ako sa nám podarilo zbierať prvé metriky zo systému Ais2, museli sme upraviť názvy metrík, ktoré produkoval nástroj Metrics a to tak, aby ich bol Zabbix schopný akceptovať. Každú metriku však v Zabbixe bolo potrebné najprv nakonfigurovať.

V konfigurácii metriky bolo treba nastaviť jej meno, jednoznačný kľúč vzhľadom na všetky metriky, spôsob prijatia metriky (v našom prípade to bol Zabbix trapper, čo je modul na prijatie metriky z iného nástroja) a jej typ (celé číslo, desatinné číslo, ...).

Konfigurácia Zabbixu bola pri 142 metrikách veľmi komplikovaná a náročná na čas. Ak by sme potrebovali v budúcnosti pridať ďalšie metriky pre novú funkcionalitu, museli by sme dodatočne Zabbix konfigurovať. To je dosť nevýhodné. Ostatné nástroje, ktoré používajú RRD databázu sú tak oveľa flexibilnejšie a nepotrebujú konfiguráciu metrík. Ak zaznamenajú novú metriku, jednoducho v strome metrík si vytvoria nový adresár podľa názvu a prefixu metriky. Konfigurácia nových metrík je teda nulová.

Zabbix navyše nepodporuje jednoduché zlievanie metrík z viacerých inštalácií

na jedno miesto s možnosťou porovnávania inštalácií. Potvrdila sa nám teda naša hypotéza, že Zabbix nie je vhodný nástroj pre monitorovanie systému AiS2 v reálnom čase.

7.3 Monitorovanie systému AiS2 za účelom hĺbkovej analýzy

Monitorovanie systému AiS2 v reálnom čase nám dá dôležité informácie o tom, čo sa deje so systémom práve teraz. Ak zbierame takéto dáta dostatočne dlho, vieme zistiť trendy, ako sa metriky menili v minulosti. Zistíme tak, kedy sa systém najviac využíva, kedy najmenej a tomu môžeme prispôsobiť napríklad plánované odstávky systému.

Nás však zaujímajú aj komplexnejšie informácie o systéme a jeho používateľoch, ktoré sa jednoduchou metrikou pokryť nedajú. Radi by sme zistili vzory používania jednotlivých dialógov v systéme AiS2, hlavne čo sa týka vhodnosti ich návrhu. Potrebovali by sme identifikovať zle navrhnuté dialógy v systéme, aby sa dali sprehľadniť a zjednodušiť, aby neboli pre používateľov informačného systému mätúce.

Táto informácia by sa dala získať monitorovaním používateľa v systéme a to tak, že by sme zaznamenávali spustené dialógy, čas na nich strávený, jeho spôsobu práce s dialógmi a následne zatváranie týchto dialógov v rámci jednej session. Ak by sa napríklad stalo, že používateľ by otvoril dialóg, o chvíľu by ho zavrel, pootváral by ďalšie, možno podobné dialógy, a nakoniec by sa vrátil k prvému, vyplnil by ho a odoslal, je pravdepodobné že správny dialóg hľadal, ale nenašiel ho hneď. Ak sa tento scenár opakuje často u rôznych používateľov systému, pravdepodobne nie je dialóg, či ľubovoľná iná informácia v systéme vhodne navrhnutá alebo je v systéme zle umiestnená.

Detekcia takýchto dialógov by výrazne prispela k zlepšeniu a skvalitneniu informačného systému. Používatelia by sa tak v ňom vedeli lepšie orientovať a našli by hľadanú informáciu rýchlejšie. Preto má pre nás detekcia takýchto dialógov zmysel.

Informácia z takéhoto monitorovania je už dosť komplexná. Musí obsahovať minimálne informáciu o identifikátore aktuálnej session, čísla dialógu a čase medzi otvorením a zatvorením dialógu. Tieto údaje už nevieme zakódovať do jednoduchej metriky. Údaje spolu úzko súvisia, z toho dôvodu ich nemôžeme rozdeliť do viacerých metrík, pretože by sme ich nevedeli interpretovať, analyzovať a porovnávať medzi sebou. Na uloženie tejto informácie nevieme použiť RRD databázu. Vhodnejšia je relačná

databáza.

Mysleli sme si, že Zabbix bude vhodný nástroj na ukladanie takýchto štruktúrovaných dát, keďže metriky ukladá v relačnej databáze MySQL. Neskôr sme si ale uvedomili, že síce Zabbix na ukladanie metrík používa relačnú databázu, ale takúto komplexnú metriku mu nevieme poslať zo žiadneho nástroja na zbieranie metrík, ktoré sme vyššie opisovali, keďže podporujú iba zbieranie jednoduchých metrík.

Potrebujeme teda zaznamenať komplexné metriky iným spôsobom. Výhodné môže byť zaznamenávanie týchto dát do logov. Dáta z logov potom môžeme vyhľadať, spracovať a uložiť do databázy nástrojom na spracovanie logov, napríklad open-source nástrojom Logstash.

Dáta z logu by sme v pravidelných časových intervaloch (napríklad raz za mesiac) spracovali pomocou nástroja Logstash a uložili do databázy. Potom by sme už spracované dáta z logu vymazali, aby zbytočne nezaberali miesto na disku, ale uvoľnili ho pre nové metriky.

Ak by sme potom chceli robiť hĺbkovú analýzu metrík, mali by sme ich uložené v štruktúrovanej databáze a vedeli by sme sa na nich jednoducho dopytovať.

Hĺbkovú analýzu získaných dát by sme robili pomocou niektorej z techník dolovania dát. Mali by sme tak užitočné informácie, ktoré by nám pomohli zlepšiť interakciu systému s používateľom informačného systému.

Záver

V tejto bakalárskej práci sme sa zaoberali monitorovaním informačných systémov, prečo je monitorovanie výhodné pre vývoj systému a ako nám môže pomôcť pri jeho rozvíjaní. Analyzovali sme možnosti monitorovania za účelom optimalizácie interakcie s používateľom.

Popísali sme si tri prístupy k monitorovaniu a to monitorovanie informačného systému v reálnom čase, dlhodobé monitorovanie systému a monitorovanie systému za účelom hĺbkovej analýzy.

Navrhli sme algoritmus na monitorovanie informačných systémov od určenia cieľov monitorovania až po analýzu a vyvodenie dôsledkov monitorovania.

Vytvorili sme jednoduchú webovú aplikáciu na demonštráciu nástrojov na zbieranie a zobrazovanie metrík, na ktorej sme si vyskúšali vyššie spomenuté nástroje.

V práci sme navrhli spôsob monitorovania akademického informačného systému AiS2. Porovnali sme open-source nástroje na zbieranie, ukladanie a vizualizáciu metrík, konkrétne sú to nástroje Statsd, Metrics, Graphite, Zabbix a Ganglia. Povedali sme si o ich výhodách a nevýhodách vzhľadom na monitorovanie akademického informačného systému AiS2. Vybrali sme si nástroje Metrics a Zabbix a integrovali sme ich do systému AiS2.

Zistili sme, že by naše problémy ohľadom agregácie metrík z viacerých inštalácií a prehľadné porovnanie týchto inštalácií vyriešil omnoho lepšie nástroj Ganglia. Takisto by nám to uľahčilo konfiguráciu oproti nástroju Zabbix.

Tiež sme zistili, že nástroj Zabbix nebude vhodný ani na dlhodobé monitorovanie systému AiS2 za účelom hĺbkovej analýzy. Namiesto Zabbixu použijeme zápis komplexných metrík do logov a priebežne ich budeme ukladať do štruktúrovanej databázy napríklad nástrojom Logstash. Hĺbkovú analýzu potom môžeme robiť s použitím techník dolovania dát.

Zoznam použitej literatúry

[1] Information system. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-17]. Dostupné z: http://en.wikipedia.org/wiki/Information_system

[2] Webová aplikácia. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-17]. Dostupné z: http://sk.wikipedia.org/wiki/Webov%C3%A1_aplik%C3%A1cia

[3] SMOLEN, Eduard. Metriky [online]. [cit. 2015-05-17]. Dostupné z: http://epodnikanie.euin.org/node/126

[4] KAUSHIK, Avinash. Webová analytika 2.0: kompletní průvodce analýzami návštěvnosti. Vyd. 1. Brno: Computer Press, 2011. ISBN 978-80-251-2964-7.

[5] MATT MASSIE, Bernard Li. Monitoring with Ganglia. 1. ed. Sebastopol, CA: O'Reilly, 2013. ISBN 9781449329709.

[6] ALLSPAW, John a Jesse ROBBINS. Web operations. Sebastopol, CA: O'Reilly, 2010, s. 43-44. ISBN 1449377440.

Prílohy

Príloha A: JSP stránka simulujúca webovú aplikáciu

Príloha B: JSP stránka s integrovaným Statsd klientom

Príloha C: JSP stránka s integrovanou knižnicou Metrics

Príloha A

example.jsp

```
<%@ page import="net.sf.uadetector.service.UADetectorServiceFactory" %>
<%@ page import="net.sf.uadetector.UserAgent" %>
<%@ page import="net.sf.uadetector.UserAgentStringParser" %>
<%@ page import="net.sf.uadetector.ReadableUserAgent" %>
<%@ page import="net.sf.uadetector.*" %>

<HTML>
  <head>
    <script src="jquery-2.1.3.min.js"></script>
  </head>
<BODY>
  <center>
    Hello! <br> <br>

<%
    java.util.Date date = new java.util.Date();
    out.println( String.valueOf( date ) + "<br><br>");
    out.println("Your IP address is "
      + request.getRemoteAddr() + "<br>");

    UserAgentStringParser parser
      = UADetectorServiceFactory.getResourceModuleParser();
    ReadableUserAgent agent
      = parser.parse(request.getHeader("User-Agent"));

    out.println("You're a <em>");
    out.println(agent.getName());
    out.println("</em> on <em>");
    out.println(agent.getOperatingSystem().getName());
    out.println("</em>!<br>");
%>

<script>
    var resolutionWidth = window.screen.width;
    var resolutionHeight = window.screen.height;
    document.write("Your Resolution: "
      + resolutionWidth + " / " + resolutionHeight);
```

```
    $(document).ready(function(){
      $("button").click(function(){
          $.get( "statsd_sender.jsp",
            { width: resolutionWidth, height: resolutionHeight } );
          $.get( "metrics_sender.jsp",
            { width: resolutionWidth, height: resolutionHeight } );
      });
    });
</script>

    <br><br>
    <button>Special Feature</button>
  </center>
</BODY>
</HTML>
```

Príloha B

statsd_sender.jsp

```
<%@ page import="java.io.*,java.util.*" %>
<%@ page import="com.timgroup.statsd.NonBlockingStatsDClient" %>
<%@ page import="com.timgroup.statsd.StatsDClient" %>

<html>
<head></head>
<body>
<%
    String width = request.getParameter("width");
    String height = request.getParameter("height");
    StatsDClient statsd = new NonBlockingStatsDClient
        ("jsp.statsd", "192.168.0.10", 8125);
    statsd.incrementCounter("Special_Feature");
    statsd.incrementCounter(width + "/" + height);
    statsd.recordGaugeValue("width", Long.parseLong(width));
    statsd.recordGaugeValue("height",  Long.parseLong(height));
%>
</body>
</html>
```

Príloha C

metrics_sender.jsp

```
<%@ page import="java.io.*,java.util.*" %>
<%@ page import="java.net.InetSocketAddress" %>
<%@ page import="com.codahale.metrics.*" %>
<%@ page import="com.codahale.metrics.graphite.*" %>
<%@ page import="java.util.concurrent.TimeUnit" %>

<%!
Counter specialFeature;

public void jspInit() {
    final class MyMetrics{};
    MetricRegistry metricsRegistry = new MetricRegistry();
    Graphite graphite
        = new Graphite(new InetSocketAddress("192.168.0.10", 2003));
    final GraphiteReporter reporter = GraphiteReporter
                .forRegistry(metricsRegistry)
                .prefixedWith("jsp.metrics")
                .convertRatesTo(TimeUnit.SECONDS)
                .convertDurationsTo(TimeUnit.MILLISECONDS)
                .filter(MetricFilter.ALL)
                .build(graphite);
    reporter.start(10, TimeUnit.SECONDS);
    specialFeature = metricsRegistry.counter("Special_Feature");
    specialFeature.inc();
}
%>
```