MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Monitorování zátěže a využití výpočetních zdrojů v heterogenním výpočetním prostředí

DIPLOMOVÁ PRÁCA

Juraj Leždík

Brno, jar 2016

Prehlásenie

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

Vedúci práce: Mgr. Miroslav Ruda

Poďakovanie

Zhrnutie

Kľúčové slová

Contents

Chapter 1

Úvod

Cloudová infraštruktúra MetaCentra poskytuje výpočetné prostriedky pre mnohé vedecké a výskumné organizácie. Spracovávajú sa v nej veľké objemy dát. Je tvorená mnohými uzlami rozmiestnenými naprieč celou Českou republikou. Podstatou fungovania cloudového modelu je zdieľanie veľkého výpočetného výkonu viacerými subjektami, pre ktoré by zabezpečenie vlastnej infraštruktúry predstavovalo neúmernú ekonomickú, personálnu a prevádzkovú záťaž. Princípom zdieľania je, že prostriedky by mali byť ideálne využívané všetkými rovnako. Nemalo by dochádzať k tomu, že jeden klient vyťaží cloud natoľko, že výpočetné úlohy ostatných dostanú nepomerne malý priestor. To je možné docieliť monitorovaním využitia výpočtového výkonu a periférií. Ak máme informácie o tom, kto koľko využíval zdroje, je možné tot využívanie účtovať a zároveň do budúcnosti primerane obmedzovať.

Aby bolo možné správne vyvodzovať závery o zaťažení cloudu, je potrebné zberať údaje o tom periodicky a kontinuálne v čase. Je potrebné sledovať parametre v pravidelných intervaloch. Každému intervalu prináleží hodnota, ktorá vypovedá o využití prostriedkov v danom momente. Takéto dáta sa nazývajú časové rady. Cloud zahŕňa množstvo uzlov, kde na každom môže byť spustených množstvo úloh. O každej je potrebné zberať viacero parametrov - vyťaženie procesora, pamäte, periférií. To predstavuje veľké množstvo metrických údajov, ktoré je potrebné ukladať a nejakým spôsobom vyhodnocovať. Slúžia na to databázy časových rád. Účtovanie sa tiež deje v určitých pravidelných intervaloch. Je preto potrebné mať možnosť spätne dohľadať údaje o využívaní zdrojov. Uchovávať podrobné metrické dáta má význam na určité obdobie. Nie je potrebné vedieť ako bol využívaný cloud úlohou pred piatimi rokmi každých päť sekúnd. Okrem toho by to predstavovalo veľké požiadavky na úložné kapacity, ktoré by rástli lineárne vzhľadom na počet úloh a čas. Je preto žiaduce po nejakej dobe dáta agregovať do väčších celkov a tiež mazať už nazbierané podrobné časové rady.

Aby boli monitorovacie údaje spoľahlivé, je potrebné zabezpečiť pe-

riodický zber metrických údajov. Monitorovanie sa uskutočňuje pravidelným pýtaním sa na vyťaženie daného zdroja. Aplikácia, ktorá zdroj využíva, vytvorí odpoveď a pošle ju späť monitorovacej aplikácií. Problém nastáva, ak odpoveď nie je vytvorená v dostatočne krátkom intervale. Povedzme, že chcem zberať údaje každých päť sekúnd. Aplikácia, ktorá práve rieši úlohy ale môže byť plne zaneprázdená a nebude odpovedať na výzvy na údaje o vyťaženosti. Tento problém je potrebné riešiť. Ak k tomu dôjde, nemali by byť vytvárané ďalšie výzvy na túto zaneprázdnenú aplikáciu v takej miere, ako keď plynulo odpovedala. Monitorovacia aplikácia by si mala zapamätať poslednú nameranú hodnotu a tú odoslať ako aktuálnu. Zároveň by sa mala zaneprázdnenej aplikácií pýtať v menej častých intervaloch. Ak aplikácia začne odpovedať, všetko je v poriadku. Ak nie, je možné reagovať reštartovaním monitorovacej aplikácie. Ak ani táto reakcia problém nevyrieši, komplikácia nastala zrejme v dotazovanej aplikácií.

Cloud predstavuje heterogénnu infraštruktúru. Na riešenie výpočetných úloh sa používajú rôzne technológie a aplikácie. Rôzne aplikácie pristupujú k výpočetným problém odlišnými postupmi. Všetky úlohy sú ale počítané na akoby jednom veľkom a veľmi výkonnom počítači. Tieto aplikácie teda používajú spoločné zdroje. Metrické dáta, ktoré budem zberať, by takisto mali vypovedať o vyťažení tých istých druhov zdrojov. Je preto potrebné identifikovať, ktoré metrické dáta z jednej aplikácie je možné porovnať s údajmi z inej aplikácie. Takto môžeme dostať celkový obraz o tom, ako rôzne technológie využívajú jednu skupinu zdrojov a len takto je možné vzájomne tieto technológie zrovnávať a účtovať ich využitie.

Chapter 2

Cloud

Cloudové technológie umožňujú využívať veľkú množinu výkonných výpočetných zdrojov mnohým subjektom. Každý užívateľ cloudu využíva časť výpočetného výkonu. Spolu s rozdelením výkonu prichádzajú aj ďalšie výhody. Užívateľ sa nemusí starať o podliehajúci hardvér. Typ fyzického procesora alebo rýchlosť pamäte uzla cloudu sú v určitom zmysle dôležité, ale pre výkon je rozhodujúca veľkosť pamäte, kapacita disku resp. počet procesorových jadier. Tieto požiadavky cloud umožňuje jednoducho spravovať a transparentne meniť. Má to význam, ak sa rozrastú požiadavky užívateľa na výkon, alebo naopak z dôvodu obmedzenej prevádzky či nedostatku výpočetných úloh sa nároky môžu zmenšiť. Klient môže dané prostriedky využívať hneď, bez veľkých počiatočných investicií, ktoré by si buď nemohol dovoliť, alebo ak sú výpočetné úlohy krátkodobejšieho rázu, nákup stroja s požadovaným výkonom by bol nevhodnou investíciou. Užívateľ sa nemusí starať o nákup hardvéru, jeho zostavovanie do funkčných serverov a ich následné rozširovanie a správu. Tieto služby zabezpečuje prevádzkovateľ cloudovej infraštruktúry. Pre neho je zase dôležité mať prehľad o tom, ako sú jeho inštalované kapacity využívané. Či už z pohľadu skvalitňovania vlastných poskytovaných služieb alebo vo vzťahu ku klientovi a k tomu, v akej miere spotrebováva poskytovaný výkon. Jednou z charakteristík cloudu je aj heterogénnosť v prístupe k zdrojom. Výpočetné prostriedky sú poskytované viacerými spôsobmi:

Infrastructure as a Service (IaaS) - Tento spôsob poskytuje užívateľovi priamo hardvérové prostriedky infraštruktúry. Ten má možnosť určiť, koľko pamäte, procesorov alebo diskovej kapacity požaduje. Je jeho voľbou aký operačný systém použije, aký softvér nainštaluje a aké výpočetné úlohy bude realizovať alebo aké služby bude prevádzkovať.

Platform as a Service (PaaS) - Užívateľ využíva priamo platformu prevádzkovateľa. Jedná sa vrstvu o úroveň vyššie ako pri využití infraštruktúry. Hardvérová konfigurácia je daná, ale užívateľ využíva

operačný systém a súbor aplikácií na vývoj a prevádzku vlastných programov. Vlastník cloudu volí platformu a inštaláciu ďalšieho softvéru. Užívateľ ho môže používať a prípadne meniť. Tento spôsob poskytovania zdrojov znižuje záťaž na infraštruktúru tak, že viacerí užívatelia využívajú jedno bežiace jadro operačného systému.

Software as a Service (SaaS) - V cloudovej infraštruktúre môže byť nainštalovaný softvér, ktorý špecifickým spôsobom zefektívňuje prácu s distribuovanými výpočetnými zdrojmi. Je preto niekedy vhodné poskytovať samotné prostredie jednej aplikácie ako službu. Užívateľ navrhuje výpočetné úlohy alebo aplikácie s využitím knižníc a architektúry konrétnej aplikácie a táto aplikácia sa zároveň stará o ich vykonávanie.

2.0.1 Kontrola zdrojov cloudu

Bez ohľadu na to, na akej vrstve si klient zvolí prístup k využívaniu zdieľaných výpočetných prostriedkov, jeho výpočty sa v konečnom dôsledku budú musieť realizovať na fyzickom hardvéri poskytovateľa. Aj keď sa daná úloha vypočítava na viacerých uzloch cloudu a rozličnými postupmi, vlastník by mal mať možnosť nejakým jednotným spôsobom určiť, ako je reálne celá infraštruktúra vyťažovaná.

2.1 Cloudové technológie

V prostredí cloudu je výpočtový výkon poskytovaný viacerými technológiami. Dôvodom sú rozdielne požiadavky užívateľov. Niektorí požadujú komplexné služby, napr. na prevádzku informačného systému. Ďalším spôsobom využitia sú výpočty náročné na výkon, kedy sa jedná o úzko špecializované využitie infraštruktúry. V nasledujúcich sekciách uvádzam bežne používané technológie v cloude.

2.1.1 Virtuálne stroje

Virtuálne stroje poskytujú úplnú virtualizáciu fyzickej hardvérovej štruktúry. Na jednom hosťujúcom počítači môže byť spustených viacero virtuálnych strojov. Každý má svoj vlastný virtuálny procesor, pamäť, grafický procesor, pevný disk a periférie. Operačný systém spustený vo virtuálnom stroji je izolovaný od hosťovského opračného systému (ak ho hosťovský počítač má). Takéto riešenie má jednu bezpečnostnú výhodu oproti aplikačným kontajnerom. Nežiadúce fungovanie jedného virtuálneho stroja

neovplyvňuje beh ostatných.
V súčasnosti existuje mnoho úrovní virtuálnych strojov. Emulácia inštrukčnej sady, prekladanie programov za behu a ich optimalizácia, vysokoúrovňové virtuálne stroje (napr. Java) a systémové virtuálne stroje používané ako jednotlivcami tak na serveroch.

Hypervízor

Hypervízor je softvér, ktorý vytvára virtuálne stroje a zabezpečuje ich beh. Rozlišujeme 2 typy:

natívny - na hosťujúcom počítači nie je nainštalovaný žiadny operačný systém. Hypervízor spravuje hardvér hosťujúceho počítača a kontroluje beh operačných systémov, ktoré sa javia ako procesy. Príkladom je VMware ESX/ESXi, Oracle VM Server for x86 alebo Citrix XenServer.

hosťovaný - hypervízor je spustený ako bežný program v operačnom systéme hosťujúceho počítača. Príkladom je QEMU, VMware Workstation alebo VirtualBox.

[?]

2.1.2 Aplikačné kontajnery

Kontajnery predstavujú odlišný prístup k virtualizácií ako virtuálne stroje. Tiež ide o snahu spúšťať softvér v prostredí oddelenom od skutočného hardvéru a operačného systému. Na rozdiel od úplných virtuálnych strojov nie je virtualizovaný celý hardvér, ale len softvérové vybavenie nevyhnutné na spustenie programu. Rozdiel v architektúre ilustruje nasledovný obrázok: Kontajnery zdieľajú jadro operačného systému hosťovského počítača a jeho množinu prostriedkov. Jednotlivé kontajnery dostávajú kontrolovaný prístup k výpočtovému výkonu, pamäti, úložnej kapacite, sieti a prípadne ďalším prostriedkom. Takýto spôsob virtualizácie predstavuje zníženú záťaž na hosťujúci počítač, pretože nie je virtualizovaný celý operačný systém a ani hardvér.
Kontajnery môžu byť spustené aj v rámci virtuálnych strojov. Cieľom tejto práce ale nie je monitorovanie takto vnorených kontajnerov.

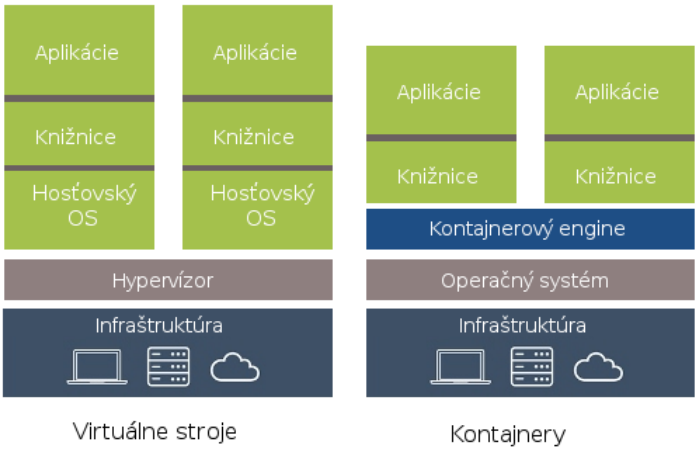

Figure 2.1: Porovnanie architektúry kontajnerov a virtuálnych strojov

2.1.3 MapReduce aplikačné prostredia

2.1.4 Gridové počítanie

2.2 Cloudové technológie MetaCentra

V prostredí MetaCentra sú nasadené do produkčnej prevádzky softvéry, ktoré umožňujú využívať výkon spomínanými technológiami. Jedná sa o heterogénnu infraštruktúru, ktorú tvoria výkonné clustre ale aj jednotlivé uzly rozmiestnené po celej republike. Virtuálne stroje sú poskytované prostredníctvom linuxového modulu jadra *KVM*. Na prácu s nimi je využívaná knižnica *libvirt*. Na virtualizáciu v podobe kontajnerov je nasadený softvér *Docker*. Technológiu distribuovaného počítania v podobe MapReduce aplikácií zabezpečuje *Apache Hadoop*. Koordináciu gridových výpočtov zabezpečuje *Torque*. V nasledujúcich sekciách sa podrobnejšie venujem popisu jednotlivých softvérov.

2.2.1 libvirt/KVM

KVM[1] je plné virtualizačné riešenie pre Linux na x86 hardvér, obsahujúce virtualizačné rozšírenia (Intel VT or AMD-V). Pozostáva z nahrateľného

1. Kernel-based Virtual Machine

modulu jadra, kvm.ko, ktorý poskytuje základ virtualizačnej infraštruktúry a špecifický modul, kvm-intel.ko alebo kvm-amd.ko, ktoré simulujú procesor od daného výrobcu. Je možné virtualizovať obrazy s operačnými systémami Linux, Windows, OS X alebo aj ďalšími. Každý virtuálny stroj má vlastný virtualizovaný hardvér: procesor, pamäť, sieťovú kartu, disk, grafický adaptér atď. KVM je open-source softvér. Virtualizačný modul jadra sa nachádza v Linuxe od verzie 2.6.20. [**?**]
libvirt je sada nástrojov na prácu s virtualizačnými schopnosťami Linuxu (a ostatných OS). Je to voľný softvér dostupný pod licenciou GNU LGPL. Obsahuje API v jazyku C a väzby pre bežné programovacie jazyky, ako napr. Python, C, Java, Ruby alebo Go. [**?**]
Umožňuje vytvoriť a zrušiť virtuálny stroj, spúšťať a vypínať, meniť jeho konfiguráciu a pridelené zdroje a tiež zisťovať údaje o využití zdrojov.

2.2.2 Docker

Docker umožňuje zabaliť aplikáciu so všetkými jej závislosťami do štandardizovanej jednotky (tzv. kontajnery) určenej na softvérový vývoj. Kontajnery Dockeru obaľujú softvér kompletným súborovým systémom, ktorý zahŕňa všetko, čo daný softvér potrebuje na spustenie: kód, nástroje potrebné na beh, systémové nástroje a knižnice. Toto zaručuje, že program bude pracovať rovnako bez ohľadu na prostredie, v ktorom je spustený. [**?**]
Docker rozlišuje kontajnery a obrazy. Obraz predstavuje softvérový balík so všetkými závislosťami, kontajner reprezentuje jeho beh. Preto môže existovať viacero kontajnerov, v ktorých je spustený ten istý softvér.

2.2.3 Hadoop

Projekt Apache Hadoop vyvíja open-source softvér na spoľahlivé, škálovateľné, distribuované výpočty. Apache Hadoop je prostredie, ktoré umožňuje distribuované spracovávanie veľkých množstiev dát naprieč clustermi, používajúcimi jednoduché programovacie modely. Je navrhnutý tak, aby bol škálovateľný od jednotlivých serverov po tisícky strojov, kde každý poskytuje lokálny výpočetný výkon a úložný priestor. Nespolieha sa na vysokú dosupnosť hardvérových prostriedkov, ale je navrhnutý, aby detekoval a zvládal chyby na aplikačnej vrstve, takže poskytuje vysoko dostupnú službu nad clusterom počítačov, z ktorých každý je náchylný na chyby. [**?**]

Projekt pozostáva z týchto modulov:

Hadoop Common: spoločné nástroje, ktoré podporujú ostatné Hadoop

moduly

Hadoop Distributed File System (HDFS™): distribuovaný súborový systém, ktorý poskytuje vysokú priepustnosť

Hadoop YARN: prostredie pre plánovanie úloh a správu zdrojov clustera

Hadoop MapReduce: systém založený na YARN pre paralelné spracovávanie veľkých množstiev dát, prostredie pre plánovanie úloh a správu zdrojov clustera

Chapter 3

Zber a uchovávanie monitorovacích dát

Monitorovanie akejkoľvek (nielen počítačovej) prevádzky je dôležitá oblasť, ktorá má význam pre jej správne fungovanie a jej správu, zdokonaľovanie a servis. Na priblíženie uvediem príklad železničnej spoločnosti. Je vhodné vedieť, koľko ľudí prepraví spoločnosť na určitom spoji. Tak môže identifikovať nerentabilné spoje, prípadne spoje, ktoré sú preťažené a je potrebné nejakým spôsobom zväčšiť ich kapacitu. Iným príkladom môže počet vlakov, ktoré prejdú za deň jednou stanicou. Vo všeobecnosti sa teda jedná o vyťaženie zdrojov vlakovej spoločnosti, či už sú to samotné vlaky alebo stanice, prípadne personál atď. Monitorovanie nám poskytuje dáta, ktoré majú viacero aplikácií:

Prehľad o aktuálnom vyťažení zdrojov Pri zbere dát toto zodpovedá meraniu konkrétnej hodnoty sledovaného parametru. Kvantifikáciou sledovaného zdroja dostaneme predstavu o tom, ako je používaný "teraz", čiže v prítomnosti. Môžeme tak detegovať prípadné preťaženie zdroja, jeho dostupnosť alebo zlyhanie.

Prehľad o vyťažení zdrojov za určitú dobu Vyťaženie zdrojov sa v priebehu času mení. Nielen z krátkodobého hľadiska, kedy napríklad viac ľudí cestuje ráno za prácou a do školy, ale aj zo strednodobého (v období leta ľudia viac cestujú na dovolenky, čiže sa menia vyťažené spoj) a dlhodobého hľadiska (cestujúcich stále pribúda). Uchovávanie monitorovacích dát predstavuje kľúčovú požiadavku, aby sme mohli sledovať, aké trendy vo využívaní mali jednotlivé sledované zdroje v priebehu nejakej doby. Z takto nahromadených dát môžeme vypočítavať rôzne štatistiky a ďalej ich analyzovať.

Prehľad o vyťažení zdrojov podľa parametrov Vyťaženie zdrojov predstavuje určitú skalárnu hodnotu či už pre jednotlivé uzly alebo naprieč celou infraštruktúrou. Niekedy je však potrebné zistiť, ako boli využívané zdroje len v určitej oblasti cloudu, alebo napr. ako boli využívané disky s kapacitou nad 1 TB. Okrem jednoduchého zberu

dát o vyťažení nám dobre navhrnuté monitorovanie poskytuje aj komplexnejšie informácie, na základe ktorých je potom možné selektívne určovať vyťaženie zdrojov. To je možné dosiahnuť zberom doplňujúcich dát pripísaných jednotlivým metrikám.

Prehľad o stave zdrojov z hľadiska správy a plánovania Nezanedbateľný význam má monitorovanie z hľadiska údržby a servisu poskytovaných služieb. Z dostupných dát vieme indentifikovať nefunkčný zdroj a nahradiť ho novým. V prípade vysokého vyťaženie zas môžeme vyvodiť záver, že zdroje už nie sú dostačujúce a je potrebné ich nejakým spôsobom rozšíriť prípadne zlepšiť efektivitu ich využívania. Takisto vieme do určitej miery predpovedať, ako budú zdroje v budúcnosti využívané, čo je podstatné pri samotnom plánovaní odstávok, servisnej činnosti alebo dočasného rozšírenia zdrojov.

Účtovanie vyťaženia zdrojov Každá poskytovaná služba má svojho spotrebiteľa. Či už sa jedná o zákazníkov železničnej spoločnosti alebo klientov výpočetného strediska. Dáta o používaní zdrojov sú jediným možným spôsobom ako rozumne stanoviť cenu za používané zdroje. Taktiež na ich základe môžeme sledovať činnosť jednotlivých užívateľov v systémoch a stanoviť im akceptovateľné podmienky na využívanie služieb.

3.1 Všeobecné problémy monitorovania

Problematika monitorovania v sebe zahŕňa viacero aspektov, ktoré možno oddeliť, zistiť pre ne najlepšie riešenia a spojiť ich tak do funkčného celku.

Identifikácia relevantných metrík
V prvom rade je potrebné identifikovať, čo vlastne potrebujeme pre konkrétny systém monitorovať. Jedná sa o určenie zdrojov kľúčových pre dané prostredie. Pre železničnú spoločnosť to môže byť vyťaženie vlakov alebo množstvo spotrebovanej energie. Niektoré údaje, ktoré potrebujeme vedieť, je možné odvodiť z iných už nameraných hodnôt. Takéto meriky budem nazývať sekundárne. Nakoľko môžu byť dopočítané, nie je vždy nevyhnutné ich v čase sledovať a uchovávať.

Analýza v reálnom čase a varovania

Namerané hodnoty metrík je okrem neskoršej štatistickej analýzy potrebné

sledovať a analyzovať v reálnom čase. Metrika môže mať svoje kritické hodnoty. Sú to hodnoty, ktoré naznačujú, že daný zdroj sa vymyká bežným očakávaniam o využití. Na toto je vhodné reagovať. Či už sa jedná o zapísanie hlásenia do logu, zobrazenie varovania alebo prípadné zaslanie emailu či sms správy s popisom udalosti.

Meracie intervaly
Ďalším čiastkovým problémom je granularita metrík. Zdroje sú kontinuálne využívané v čase. Tieto dáta je potrebné nejakým spôsobom digitalizovať. V určitom momente sa pozrieme na zdroj, kvanitifikujeme, do akej miery je využitý a danú hodnotu zobrazíme prípadne uložíme. Meranie je po uplynutí určitého intervalu zopakovať, aby sme získali opäť aktuálne údaje. Rôzne zdroje sa menia v čase rôznou intenzitou. Napr. kapacita vlaku sa v priebehu jazdy mení zriedkavo, zatiaľčo počet cestujúcich sa mení v každej stanici. Je preto dôležité určiť aj periodicitu zberania jednotlivých metrík. Niektoré metriky sa môžu meniť takým spôsobom, že nie je efektívne sledovať ich zmenu pravidelne v nejakých intervaloch. Namiesto toho je efektívnejšie pri zmene daného parametru túto zmenu ohlásiť spolu s novou hodnotou. U vlakov je to napríklad kapacita vlaku, ktorá sa mení len zriedkavo počas jazdy, napríklad pri prepriahaní vozňov z jedného vlaku do druhého.

Počet zdrojov
Množstvo zdrojov predstavuje ďalšiu oblasť, ktorú je potrebné zvážiť. Môžeme disponovať rôznym počtom zdrojov viacerých druhov, rozmiestnenými na viacerých miestach pod správou mnohých ľudí - prípadne oddelení. Metrické dáta vo svojej podstate len kvantifikujú využitie zdroja vo všeobecnosti. Zaobchádzajú s ním v tom zmysle, ako by bol len jeden. Nehovoria nič bližšie o tom, o aký zdroj sa jedná, kde je ho možné nájsť. V rámci širšieho systému je preto dôležitá jednoznačná identifikácia daného zdroja. O danom zdroji preto chceme zistiť napr. názov uzla, kde sa nachádza, prípadne výrobný model. Takéto údaje sa nazývajú metadáta - čiže dáta o (v tomto prípade o metrických) dátach.

Ukladanie dát
Uchovávanie metrických dát je kľúčovou sučasťou monitorovacieho systému. Doba, po ktorú vlastníci zdrojov uchovávajú metrické dáta, sa líši systém od systému. V niektorých oblastiach to je dokonca upravené zákonmi - napr. telekomunikácie, v iných je to na zvážení majiteľa ifraštruktúry. Množstvo dát, ktoré je treba uložiť, závisí od viacerých parametrov.

Sú nimi počet druhov zdrojov, počet jednotlivých inštancií zdrojov, počet metrík, ktoré sa o zdrojoch uchovávajú, periodicita zbieraných metrík a do určitej miery aj spôsob identifikácie daného zdroja. Prikladám tabuľku, ktorá ilustruje ako sa mení počet záznamov v závislosti na uvedených parametroch.
Ukladanie metrických dát predstavuje zároveň úzke hrdlo monitorovacieho riešenia. Zistenie hodnôt metrík pre kontrétny zdroj predstavuje v zásade serializovanú činnosť. V určitom čase sú všetky zdroje opýtané na to, ako sú vyťažené. Po jednom zistia svoje hodnoty a odošlú ich na uloženie. Počet metrík pre jeden zdroj je v bežnej praxi len zlomkom počtu zdrojov. Na časť systému, ktorá je zodpovedná za ukladanie metrík, sú preto v pravidelných intervaloch kladené pomerne veľké nároky. Je ich už ale možné spracovávať paralelne. Centralizácia úložiska by znamenala prílišnú záťaž a monitorovacie uzly, preto je vhodné, aby takéto úložisko bolo distribuované a prispôsobené na paralelné spracovávanie veľkých objemov dát.

Vizualizácia a analýza dát
Zozbierané metrické dáta je ďalej potrebné nejakým spôsobom zobraziť. Na to sú vhodné čiarové grafy s dvoma osami, kde horizontálna os predstavuje čas a vertikálna os predstavuje hodnotu nameranej metriky. Analýza zozbieraných dát slúži na odhalenie trendov vo využívaní zdrojov a môže viesť k zdokonaľovaniu systému a k lepšiemu plánovaniu využívania zdrojov.

3.1.1 Monitorovanie v heterogénnom výpočetnom prostredí

V oblasti cloudového a gridového výpočetného prostredia je pre monitorovanie kľúčové sledovať vyťaženie výpočetných zdrojov. Konkrétne procesor, výpočetná pamäť, trvalý ukladací priestor a sieťová infraštruktúra. Toto je spoločné pre všetky technológie a okrem týchto zdrojov je vhodné sledovať aj parametre špecifické pre jednotlivé oblasti. Tým sa budem konkrétne venovať v kapitole Metriky.
Automatizované monitorovanie výpočetného prostredia so sebou prináša aj špecifické problémy. Odvíjajú sa od toho, že monitorovanie je vykonávané strojovo (tj. pomocou počítača) a monitorované zdroje sú takisto stroje, ktorých stav z pohľadu využitia sa mení veľmi dynamicky.
Rozdielny je aj prístup jednotlivých technológií v tom, ako poskytujú výpočetné zdroje. Aplikačné kontajnery a virtuálne stroje sa snažia rozdeliť celú infraštruktúru na menšie viac-menej uzavreté celky, ktoré sú potom

sprostredkované užívateľom. Každý tento celok má pridelené zdroje, ktoré potom využíva. Množstvo týchto zdrojov je možné meniť, ale súvisí to s reštartovaním virtuálneho stroja alebo kontajnera. Z pohľadu gridu sú zanedbávané zdroje jedného uzla a na vypočítavanie úloh sa berú do úvahy všetky zdroje celého clustera.

Problematika intervalov

U vysoko výkonných výpočetných systémov sa jednotlivé operácie vykonávajú vo veľmi krátkych časových intervaloch. Rádovo sú to nanosekundy. Jednotlivé procesy a výpočetné úlohy dostávajú na krátky čas k dispozícií všetky zdroje, čím sa z vyššieho pohľadu zabezpečuje paralelizácia. To spôsobuje vyťaženie množstva inštancií zdrojov mnohými učastníkmi. Celkový stav infraštruktúry z pohľadu vykonaných úloh a prenesených dát sa preto mení veľmi dynamicky a je vhodné zbierať dáta o zdrojoch v rozmedzí jednej až troch sekúnd.
Nezanedbateľnú rolu hrá aj čas potrebný na získanie jedného takéhoto "snímku" využitia zdrojov. Kým napr. v prípade vlakov je dostatok času na spočítanie cestujúcich medzi jednotlivými zastávkami, v prípade heterogénneho prostredia môže byť tento čas kritický. Ak sa každé dve sekundy pýtame na metrické údaje, očakávame, že odpoveď príde v kratšom intervale. V najlepšom prípade by táto doba odpovede mala byť len malý zlomok periódy danej metriky. Vopred nevieme, koľko bude trvať, kým príde odpoveď. To je v priamej závislosti od toho, koľko subjektov infraštruktúru využíva a koľko úloh spúšťa. Ak je však doba odpovede dlhšia ako samotný interval metriky, je to potrebné nejako riešiť. Nie je vhodné automaticky paralelne vygenerovať ďalšiu požiadavku na "snímku". Žiaducejším spôsobom riadenia je také, ktoré počká, kým sa daná požiadavka vykoná, a potom v nasledovnom intervale je meranie vykonané znova. Týmto sa predíde nadmernej záťaži celého systému.
To, ako sa systém vysporiadava z takýmto neočakávaným správaním, som zohľadňoval pri jednotlivých dostupných softvérových riešeniach.

Detekcia nových zdrojov a užívateľov

Heterogénna štruktúra v čase mení aj množstvo a kapacitu svojich poskytovaných zdrojov. Nie je možné staticky definovať zoznam procesorov, diskov, alebo virtuálnych strojov ktoré treba monitorovať. Podobne je to aj s užívateľmi. Proces vytvárania nových užívateľov je zautomatizovaný, takisto užívatelia môžu automatizovať vykonávanie svojich výpočetných

úloh. Monitorovacie riešenie sa preto musí vedieť vysporiadať s týmito dynamickými zmenami, musí ich vedieť automaticky detegovať a zberať o nich metrické dáta.

3.2 Časové rady

Monitorovacie dáta vo svojej podstate predstavujú časové rady. Časová rada je sekvencia dát, kde danému časovému okamihu zodpovedá jedna hodnota. Príkladom je zaznamenávanie teplôt v priebehu roka, výšky oceánskeho prílivu alebo množstvo áut, ktoré za určitú dobu prejde jedným bodom diaľnice. Efektívnou metódou vizuálizácie dát časových rád sú čiarové grafy. Horizontálna os reprezentuje plynutie času a na vertikálnej osi sú znázornené hodnoty v danom čase.

3.2.1 Analýza časových rád

Analýza časových rád sa primárne zaoberá získavaniu štatistík o zozbieraných dátach, napr. priemerná teplota počas celého roka. Medzi ďalšie úlohy patrí:

Exploračná analýza dát

Aproximácia na funkciu

Predpovedanie

Klasifikácia

Mám sa viac o nich rozpísať?

3.2.2 Zaobchádzanie s historickými dátami

Niečo o tom, ako sa v rámci šetrenia priestoru dáta zhlukjú a vypočítavajú sa agregované štatistiky.

Chapter 4

Aktuálne monitorovacie riešenia

Problematike monitorovania softvéru a infraštruktúry sa venuje viacero komerčných alebo open-source aplikácií, prípadne aplikácií zadarmo. Rôznymi technológiami riešia zber dát, ich uchovávanie, vizualizáciu a operácie nad nimi. Najprv sa venujem popisu aplikácií, ktoré súvisia so samotným monitorovaním, neskôr rozoberám dostupné riešenia v oblasti uchovávania časových rád.

4.1 Monitorovacie riešenia

Monitorovací softvér vo všeobecnosti poskytuje ucelené monitorovacie riešenie od zberu dát, cez uchovávanie a vizualizáciu. Pri konkrétnych riešeniach sa venujem aj tomu, akým spôsobom je ich možné rozšíriť o zber ďalších dát a ako sa správajú v prípade, že metriky nie sú dostupné v požadovaných intervaloch.

4.1.1 Nagios

Nagios je aplikácia, ktorá poskytuje komplexné riešenie na monitorovanie systémov. Poskytuje informácie o kľúčových komponentoch infraštruktúry vrátane aplikácií, služieb, operačného systému, sieťových protokolov, systémových metrík a sieťovej infraštruktúry. [?] Aplikácia pozostáva z jadra, ktoré riadi zber údajov, a z množstva pluginov tretích strán. Tie sa zaoberajú monitorovaním jednotlivých oblastí systému. Ďalej aplikácia poskytuje grafické užívateľské rozhranie v podobe webového rozhrania. K dispozícií sú rôzne vizualizácie nameraných dát, grafy a histogramy. Nagios disponuje aj systémom užívateľských účtov. Tie sa delia na dva typy: administrátorov a bežných užívatešov. Bežní užívatelia majú prístup k nameraným hodnotám a zobrazovaniu grafov. Administrátori môžu konfigurovať aplikáciu, pridávať služby, ktoré je potrebné monitorovať, upravovať parametre monitorovania a spravovať užívateľské účty. K dispozícií

je aj uchovávanie konfigurácie aplikácie a spravovanie týchto konfigurácií. Súčasťou je aj systém na upozorňovanie na kritické hodnoty, či už formou emailu alebo SMS správou. Aplikácia je vyvíjaná pre platformu CentOS a Red Hat Enterprise Linux. Namerané metriky sa uchovávajú v logovacích súboroch. Pomocou pluginov je možné ich odosielať do MySQL databázy prípadne PostgreSQL. Nagios predstavuje centralizované riešenie pre monitorovanie, kde výkonné jadro riadi zberanie metrík naprieč celým systémom. Jadro a pluginy sú dostupné zdarma, komplexné riešenie je potrebné zakúpiť. Cena sa odvíja od množstva zariadení, ktoré je potrebné monitorovať. Zariadením sa rozumie niečo, čo má IP adresu, prípadne doménové meno - či už sa jedná o firewall, switch, router, pracovnú stanicu alebo server. V súčasnosti Nagios disponuje pluginmi Docker, Hadoop aj libvirt/KVM, no ani jeden z pluginov nemonitoruje požadované metriky.

Rozšírenia

Nagios pluginy existujú vo forme skriptov alebo spustiteľných programov. Aby mohli fungovať ako pluginy ich činnosť musí spĺňať dve kritériá. Prvým je výpis aspoň jedného riadku na štandardný výstup. Ten sa týka nameraných metrík. Od verzie 3 je podporovaných aj viacerov riadkov na výstupe. Druhým kritériom je návratová hodnota. API rozoznáva štyri návratové hodnoty. Tie zodpovedajú stavom monitorovanej služby - či je služba v poriadku, či vygenerovala nejaké varovanie, či je jej stav kritický, alebo neznámy. Prednastavená maximálna dĺžka výstupu je 4 kB, je ju však možé pomocou konfigurácie zmeniť.

Riadenie intervalov zberu metrík

Pre jednotlivé monitorovacie sondy je možné nadefinovať interval v počte sekúnd, v ktorom je potrebné aby sonda ukončila svoju činnosť. Ak sa tak nestane, proces sondy je ukončený a jej návratová hodnota zaznamenaná ako kritická. Taktiež je toto zaznamenané do logov Nagiosu. Tento mechanizmus predstavuje akúsi poslednú záchranu pred zahltením systému pluginmi, ktoré sa nesprávajú podľa očakávaní.

4.1.2 Zabbix

Zabbix je open-source aplikácia na monitorovanie systémov od malých systémov s malým počtom uzlov až po veľké firemné prostredia s tisíckami strojov. Architektúra aplikácie pozostáva zo serveru a agentov. Úlo-

hou agentov je zber monitorovacích dát a ich odosielanie serveru. Komunikácia týchto dvoch častí môže prebiehať dvoma spôsobmi. V prvom prípade si agent vyžiada zoznam metrík, ktoré má sledovať. Následne serveru odosiela všetky tieto metriky po jednom. Druhou alternatívou je postup, kedy sa server agenta pýta na jednotlivé hodnoty metrík a ten mu ich odosiela. Server spravuje konfiguráciu jednotlivých agentov, čo uľahčuje ovládanie ich správania naprieč celou infraštruktúrou.
Aplikácia zberá údaje o dostupných zdrojoch uzlov, o počte procesorov, dostupnej pamäti a úložnej kapacite. Taktiež zberá údaje o aktuálnom vyťažení týchto zdrojov. Okrem toho sleduje dostupnosť a parametre služieb ako FTP, DNS, HTTP, SMTP, SSH a rôznych ďalších. Taktiež poskytuje údaje o proces bežiacich v systéme a o užívateľoch. Zaujímavým prvkom je monitorovanie logov, ich analýza a vytváranie varovaní v prípade, že je to nutné.
Zabbix vie zberané dáta vizualizovať pomocou grafov. Súčasťou je aj sledovanie zberaných hodnôt, ich kontrola na požadovaný rozsah a generovanie notifikácií. Poskytuje aj manažment užívateľov samotnej aplikácie a ich práv na zaobchádzanie s ňou. Zabbix podporuje autodetekciou nových prvkov v infraštruktúre. Kontroluje zadaný sieťový rozsah na nové uzly, služby na nich bežiace alebo automaticky registruje nových spustených agentov.
Na zvládanie záťaže vo veľkých systémoch je zavedený systém proxy serverov. Tie zhlukujú dáta z niekoľkých agentov a až potom sú odosielané centrálnemu serveru.
Zabbix má vlastnú databázu na uchovávanie zozbieraných dát. Rozdeluje ich na históriu a trendy. História sú dáta tak, ako boli namerané, trendy predstavujú agregované dáta z pohľadu dlhších období, napr. v rádoch rokov.
V súčasnosti Zabbix nepodporuje zber metrík z Hadoopu. Je možné nainštalovať agenta do kontajneru alebo virtuálneho stroja. To ale nepredstavuje ideálne riešenie, nakoľko tieto služby v prvom rade slúžia pre potreby užívateľov a je chodné nechať na nich, čo v danom kontajneri alebo virtuálnom stroji chcú mať spustené.

Rozšírenia

Aplikáciu je možné rozšíriť o vlastné moduly zberajúce dáta tromi spôsobmi. [**?**]

užívateľské parametre - v tomto prípade užívateľ definuje v agentovi názov metriky a príkaz, ktorý sa má vykonať na zber jej hodnoty

externé kontroly - jedná sa o kontrolu na strane servera, kedy je tiež definovaný skript zberajúci dáta a server ho spúšťa

system.run - jedná sa o kontrolu na strane agenta, ktorá podporuje väčší výstup spusteného príkazu ako len hodnota metriky

Okrem toho poskytuje aj programové API a vývoj vlastných modulov ako zdieľaných knižníc, ktoré musia implementovať požadované funkcie.

Riadenie intervalov zberu metrík

Konrola zberu jednotlivých metrík z hľadiska doby trvania tohoto zberu sa v Zabbixe odohráva na úrovni agenta a serveru. Na úrovni agenta ide o čas, ktorý je možné stráviť spracovávaním a získavaním jednotlivých metrík. Pre server je potom definovaná doba, ktorú je ochotný čakať na odpoveď agenta. Tieto doby je možné konfigurovať v rozsahu od 1 do 30 sekúnd. [**?**] [**?**] Zabbix nespracuje jednoduchú kontrolu na hodnotu metriky, ktorá trvá dlhšie ako túto dobu. [**?**] Dokumentácia však neuvádza, ako sa aplikácia riadi beh externých procesov, ktoré zberajú dáta.

4.1.3 Icinga

Icinga predstavuje open-source monitorovací nástroj, ktorý je nástupcom Nagiosu. Je založená na princípe paralelnej činnosti viacerých vlákien, čo poskytuje možnosť vykonávať množstvo meraní za krátky časový interval. Zabezpečuje monitorovanie záťaže zdrojov ako disk, procesor, pamäť a sieť a to v operačných systémoch na základe Linuxu a aj v prostredí Windows. Ďalej podporuje zberanie dát o mnohých ďalších službách, ako je monitoring databáz, odozvy serverov, webových služieb, výkonu JVM, sledovanie logov. Podporuje monitorovanie virtualizačnej platformy VMWare. Poskytuje aj bežné mechanizmy v oblasti monitorovania ako generovanie notifikácií o neštandardnom správaní, na ktoré je možné reagovať spustením nejakého systémové príkazu. Na ukladanie dát používa svoju databázu, no podporuje aj odosielanie dát do databázy časových rád InfluxDB.
V súčasnosti Icniga nie je schopná monitorovať ani jednu zo služieb v prostredí MetaCentra.

Rozšíriteľnosť aplikácie

Icinga poskytuje systém, akým je možné do nej pridať kontroly nových metrík, prostredníctvom tzv. check príkazov. Jedná sa o kofiguračné ob-

jekty, kde užívateľ definuje príkaz, ktorý sa má spustiť, spolu s hodnotami jeho argumentov. Icinga určí stav tejto kontroly podľa návratovej hodnoty príkazu. [**?**] Hodnoty a názvy metrík sú vracané vo forme textového výstupu.

4.1.4 Ganglia

Ganglia je škáľovateľný distribuovaný systém na monitorovanie výkonných systémov ako sú clustre a gridy. Je založená na hierarchickom návrhu som zameraním na federácie clusterov. Využíva bežné štandardy, ako napr. XML na reprezentáciu dát, XDR na prenos dát a RRDTool na ukladanie a vizualizáciu dát. [**?**] Cieľom aplikácie je zabezpečiť čo najmenšiu nabytočnú záťaž na uzloch a vysokú mieru súbežnosti. Ganglia vznikla na University of California v rámci Berkeley Millennium Project. Je to open-source projekt pod licenciou BSD. V súčasnosti je nasadená na mnohých operačných systémoch a podporuje viacero procesorových architektúr.
Ganglia má architektúru monitorovacieho démona, ktorý je spustený na každom monitorovanom uzle, a démona, ktorý uchováva zozbierané údaje do round-robin archívu. K dispozícií sú aj nástroje na získanie aktuálnych údajov konkrétnej sondy a tiež webové rozhranie na zobrazovanie zaznamenaných údajov.
Hadoop je možné nakonfigurovať tak, aby odosielal metriky zbernému démonovi Ganglie. Ostatné integrácie v súčasnosti neobsahuje.

Rozšíriteľnosť aplikácie

K dispozícií je mnoho pzásuvných modulov v podobe skriptov operačného systému alebo programov napísaných v skriptovacích jazykoch ako Ruby, Perl či Python. Pluginy poskytujú monitorovanie prihlásených užívateľov v systéme, rôznych aplikácií a služieb, ako napr. DNS, HTTP, ďalej podporujú zber senzorických údajov (teplota procesora, otáčky ventilátora), monitorovanie rýchlosti siete či napr. ZFS súborového systému. [**?**]

4.1.5 AppDynamics

AppDynamics predstavuje zaujímavé riešnie v oblasti monitorovania. Snaží sa do istej miery zjednotiť a prepojiť monitorovanie aplikácií a infaštruktúry a následne tak ľahšie identifikovať problémy s výkonom. Obsahuje bohaté vizualizácie nie len v podobe grafov o vyťažení, ale aj mapy a plošné grafy pozostávajúce s uzlov prepojených čiarami, ktoré reprezentujú

vyťaženie spojenia jednotlivých služieb - napr. webový server, databáza a servery aplikačnej logiky. Monitoruje vyťaženie zdrojov ako sú procesor, pamäť, vstupno-výstupné operácie a sieť u jednotlivých uzlov infraštruktúry a dáva ich do súvisu s tým, aké role tieto uzly vykonávajú v danej business aplikácií.
Okrem toho disponuje aj integráciami s IaaS a PaaS cloudovými službami od Amazonu, RedHatu či Microsoftu a integráciami s mnohými aplikáciami vrátane zberu metrík o zdrojoch v Hadoope a Dockeri. [**?**]
AppDynamics predstavuje komerčné riešenie. Verzia s obmedzeným monitorovacím záberom a s obmedzenou dĺžkou ukladania údajov je k dispozícií aj zdarma, plná verzia je platená.

Rozšíriteľnosť aplikácie

Nakoľko je AppDynamics komerčný softvér, rozšírenia sú k dispozícií od výrobcu. Neexistuje programové rozhranie ani iný spôsobo, napr. spúšťanie rozšírení ako samostatncých programov a analýza ich výstupu.

4.2 Ďalšie nástroje

V súvislosti s monitorovaním existujú aj ďalšie programy a nástroje, ktoré sa dajú využiť na zber dát. Niektoré poskytujú riešenia v oblasti riadenia zberu a zápisu dát (napr. collectd), iné sa zaoberajú riadením prístupu k zdrojom a môžu slúžiť ako zdroj dát.

4.2.1 Linux cgroups

Linux cgroups je technológia linuxového jadra, ktorá umožňuje limitovať, sledovať a izolovať spotrebu prostriedkov systému jednotlivými procesmi. Zavádza stromovo organizované kontrolné skupiny. Kontrolná skupina obsahuje obmedzenia pre jeden systémový prostriedok, tzv. susbsystém. Príklad subsystémov, ktoré poskytuje Red Hat Enterprise Linux:

blkio - tento subsystém nastavuje limity na zápis a čítanie z blokových zariadení, ako napríklad HDD, SSD alebo USB disky.

cpu - tento subsystém využíva systémový plánovač na poskytovanie prístupu k procesoru pre jednotlivé úlohy.

cpuacct - tento subsystém generuje automatické správy o tom, koľko procesorových zdrojov využila úloha v danej skupine.

cpuset - tento subsystém priraďuje jednotlivé procesory (na viacprocesorovom systéme) a pamäťové uzly úlohám v skupine.

devices - tento subsystém povoľuje alebo zakazuje prístup k zariadeniam podľa úloh v skupine.

freezer - tento subsystém zastavuje alebo obnovuje vykonávanie úloh v skupine

memory - tento subsystém nastavuje limity na využívanie pamäte úlohami v skupine a generuje automatické správy na pamäťové zdroje použité úlohami.

net_cls - tento susbsystém označuje sieťové pakety identifikátorom triedy, čo umožňuje linuxovému ovládaču prevádzky identifikovať pakety, ktoré pochádzajú od konkrétnej úlohy.

net_prio - tento subsystém poskytuje spôsob, ako dynamicky nastavovať prioritu sieťovej prevádzky pre sieťové rozhrania.

ns - toto je menný subsystém.

[?]

Pre každý subsystém existuje jeden strom kontrolných skupín. Ďalej skupina obsahuje zoznam procesov, ktoré podliehajú definovaným obmedzeniam. V strome kontrolných skupín sa proces vyskytuje len raz. V cloudovom prostredí však jednotlivé technológie využívajú a spúšťajú mnoho procesov, ktoré by sa obtiažne priraďovalo jednotlivým užívateľom v súvislosti so spustenými aplikáciami, preto tento spôsob monitorovania nie je úplne vhodný. Takisto zberané metriky o subsystémoch sú agregované pre kontrolnú skupinu, tj. predstavujú súčet pre skupinu procesov, takže nie je možné jednoznačne určiť koľko zdrojov spotreboval ten ktorý proces.

4.2.2 collectd

Collectd je open-source monitorovacia aplikácia vyvinutá v jazyku C. Má architektúru jadra a pluginov, ktoré sa starajú o mnoho činností súvisiacich s monitorovaním od zbierania dát, cez zapisovanie metrík a sledovanie prahových hodnôt a upozorňovanie. Jadro predovšetkým ovláda a kontroluje vykonávanie týchto aktivít. V pravidelných intervaloch spúšťa kontroly metrík, analyzuje namerané hodnoty, generuje notifikácie v prípade, že hodnoty sú mimo požadovaný rozsah, alebo ak nedorazili v

požadovaných intervaloch. V súčasnosti je k dispozícií viac než 90 pluginov. Poskytujú monitorovanie základných výpočetných zdrojov ako sú procesor, pamäť, disky a sieťové rozhrania, ďalej monitorovanie mnohých aplikácií (napr. databázy, e-mail, webové servery) a protokolov. Zvolený programovací jazyk dáva dobré predpoklady na výkon a prenositeľnosť, čo umožňuje beh na systémoch bez skriptovacích jazykoc alebo cron démona, ako napríklad zabudované systémy. Zároveň ale disponuje optimalizáciami a prvkami, aby zvládal stovky tisíc metrík. [**?**]
Collectd nepodporuje vizualizáciu dát ani historickú analýzu. Nepredstavuje preto úplné monitorovacie riešenie. Samotný zápis dát je uskutočňovaný pomocou pluginov a teda je možné zvoliť z viacerých spôsobov uchovávania dát. V súčasnosti podporuje zápis do viacerých databáz, napr. MongoDB, OpenTSDB, alebo metódou HTTP POST či do súborov RRD.
V súčasnosti obsahuje plugin pre monitorovanie libvirt/KVM.

Rozšíriteľnosť aplikácie

Collectd poskytuje viacero spôsobov, akým je možné ho rozšíriť. Prvým aspektom je zber metrík. Umožňuje zber dodatočných metrík pomocou spúšťania systémových príkazov, pomocou rozšírení v rôznych programovacích jazykoch (Perl, Python, Java). Ďalej je možné využiť natívne API v jazyku C. Cez toto API je možné aplikáciu rozšíriť v ohľade notifikácií, zápisu metrických dát a spôsobu logovania udalostí.

Riadenie intervalov zberu metrík

V prípade, že hodnota metriky nedorazila v požadovanom intervale, je generované hlásenie. Neprichádza k tomu ale hneď po prvej chybe. Collectd zavádza mechanizmus časových limitov. To znamená, že meranie niekoľkokrát zopakuje v požadovanom intervale a až potom vygeneruje hlásenie s varovaním. Počet opakovaní je možné definovať.

4.3 Databázy časových rád

Na uchovávanie zozbierných dát sú najvhodnejšie databázy časových rád. Ich provradým účelom je uchovávanie veľkého objemu dát a spracovávanie tisícok zápisov za sekundu. Vizualizácia nie je primárnou funkciou, databázy majú buď vlastné zabudované riešenia, alebo sú poskytované ako samostatný softvér.

4.3.1 OpenTSDB

OpenTSDB je open-source databáza na uchovávanie a sprístupňovanie veľkých objemov časových dát. Pozostáva z Time Series Daemon (TSD) a z utilít pre príkazový riadok. Interakcia s OpenTSDB je primárne realizovaná cez jedného alebo viacerých TSD. Každý TSD je nezávislý. Neexistuje žiadny riadiaci proces, žiadny zdieľaný stav, takže je možné spustiť toľko TSD, koľko je potrebné na zvládnutie požadovanej záťaže. Každý TSD používa open-source databázu HBase na ukladanie a vyberanie dát časových rád. HBase schéma je vysoko optimalizovaná na rýchlu agregáciu podobných časových rád, aby minimalizovala požiadavky na úložný priestor. Používatelia TSD nemusia pristupovať do HBase priamo. S TSD je možné komunikovať cez jednoduchý protokol podobný Telnetu, cez HTTP API alebo cez jednoduché GUI. Všetka komunikácia sa deje na tom istom porte (TSD odhadne protokol klienta pohľadom na prvých niekoľko bajtov, ktoré obdrží). [**?**]
OpenTSDB uchováva čas zozbierania metriky, jej hodnotu a súbor popisných dát, ktoré sa nazývajú tagy. Na ich základe je potom možné v nameraných dáta vyhľadávať, zhlukovať ich a analyzovať. Jednou z odlišností od iných databáz je, že má schopnosť uchovávať "surové" nezhlukované dáta po veľmi dlhú dobu (čiže od začiatku merania až do prítomnosti).
Na vizualizáciu dát existuje nástroj Metrilyx. Je to open-source webový engine, ktorý vytvára grafy zo zhromaždených dát. Je možné meniť časové rozpätie, za ktoré sa majú grafy metrík zobraziť. Na obrázku je znázornených viacero metrík, ktorých názov sa nachádza v spodnej časti. Ich hodnoty sa v čase menia, čo je zachtené čiarovým grafom, kde horizontálna os predstavuje čas a vertikálna os hodnotu metriky v tom čase.

4.3.2 InfluxDB

InfluxDB je platforma v jazyku Go na zbieranie, uchovávanie, vizualizáciu a správu časových dát. InfluxDB je určená na použitie ako úložisko dát v takých prípadoch, ktoré zahŕňajú veľké množstvo dát označených časovou známkou, vrátane DevOps monitorovania, aplikačných metrík, senzorových dát internetu vecí, a na analýzu dát v reálnom čase. [**?**] Užívateľ môže vytvoriť viacero nezávislých databáz. Dáta sa zapisujú a čítajú pomocou rozhrania príkazového riadka, rôznych klientskych knižníc, alebo pomocou HTTP API. Na vizualizáciu dát používa modul *chronograf*.

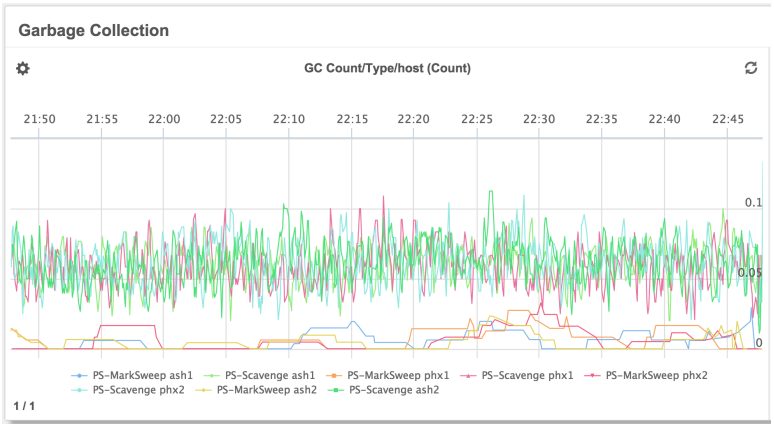

Figure 4.1: Vizualizácia časových rád OpenTSDB pomocou Metrilyx [?]

Politiky udržiavania

Každá databáza obsahuje pravidlá, ktoré definujú, po akú dobu majú byť ukladané dáta časových rád a koľko kópií dát má byť vytvorených. Jedna databáza môže mať niekoľko takýchto politík. Pri zápise do nej je môžné špecifikovať, ktorá politika sa má pre zápis použiť. Pri vytvorení databázy je automaticky vytvorená jedna politika.

Kontinuálne dotazovanie a agregácia

Databáza je schopná periodicky vykonávať požiadavky na dáta. Cieľom je zmenšovanie objemu dát. Ide o zhlukovanie dát s vysokou frekvenciou zberu, čím vzniknú dáta s menšou hustotou zberu. Táto hodnota je potom zvyčajne uložená do inej databázy.

4.3.3 RRDTool

RRDTool je nástroj na uchovávanie, spravovanie a vizuálizáciu časových dát. Využíva round-robin databázu. Je to databáza s dopredu daným maximálnou časovým rozpätím, za ktoré uchováva metrické dáta. V prípade, že príde požiadavka na zápis hodnoty a databáza je už plná, dôjde k prepisu najstaršej hodnoty. Tento nástroj takisto obsahuje funkcie na konsolidáciu dát. Konsolidovaná hodnota je typicky priemer, minimum alebo maximum z viacerých hodnôt zozbieraných za dlhší časový úsek. Tieto hodnoty sú

taktiež ukladané do round-robin archívu.

Chapter 5

Metriky

Každá z technológií cloudového výpočetného strediska je zdrojom mnohých dát o vyťažení zdrojov. Je potrebné určiť, ktoré zdroje monitorovať a ktoré metriky zberať. V prípade MetaCentra podstatné zdroje predstavujú *procesor, pamät, pevné disky a sieťové rozhrania*. Je vhodné mať také metriky pre všetky využívané cloudové technológie, ktoré je možné nejakým spôsobom porovnať medzi sebou. V prípade procesora sa jedná o jeho aktuálne vyťaženie, ale zároveň je zaujímavým údajom aj prepočítaný procesorový čas. Táto metrika môže byť efektívnym nástrojom pre následné účtovanie jednotlivým užívateľom. Keďže majú ale tieto technológie principiálny rozdiel vo svojom určení, nie je možné vždy o každom zdroji zberať rovnaké dáta. O distribuovaných výpočtoch napríklad nie je možné efektívne zistiť aktuálne vyťaženie procesora. Takéto úlohy sa počítajú na viacerých uzloch clustera. O poradí a rozmiestnení požiadaviek na zdroje rozhodouje riadiaca aplikácia, takže sa nejedná o jeden procesor. Táto aplikácia má ale prehľad o tom, koľko času strávil celý cluster počítaním danej úlohy. Takže v určitom ohľade je zrovnateľná s virtuálnym strojom a jeho spotrebou procesorového času.
Podobná situácia nastáva aj u monitorovania sieťových rozhraní. Distribuované výpočty využívajú sieť svojským spôsobom a len na účel vypočítania komplexnejšieho problému. Jedná sa o prepojenie uzlov v rámci clustera. Je to jednoúčelová vysokorýchlostná sieť. Z hľadiska poskytovania výpočetných kapacít dáva väčší zmysel orientácia na využívanie konektivity smerom do internetu.
Názvy metrík uvádzam tak, ako sú pomenované vo výstupoch získavaných od jednotlivých cloudových technológií. Do databázy ukladám metriky o rovnakých zdrojoch a ich parametroch pod rovnakými názvami aj pre rôzne technológie. V prílohe sa nachádza tabuľka, ktorá popisuje prevod týchto názvov.

5.0.4 Význam popisných údajov metrík

Metrické dáta vypovedajú o využití zdrojov. Vieme určiť ako dlhý čas a v akej miere bol využívaný výpočtový výkon. Z nich samotných nevieme presne určiť, kto zdroje využíval. Pre to aby mali tieto dáta zmysel pre neskoršie účtovanie, je potrebné mať možnosť ich priradiť k užívateľom, či už sa jedná o vlastnika virtuálneho stroja alebo uživateľa, ktorý spustil kontrétnu úlohu. Na základe týchto ďalších popisných údajov je potom možné zisťovať, kto zdroje vyťažoval za časové obdobie najviac a podľa toho stanoviť prípadnú cenu za používanie zdrojov. Okrem identity je vhodné metriky popisovať aj ďalšími údajmi, ako je miesto, kde sa zdroj nachádza, názov stroja, ktorý poskytuje svoje kapacity, a ďalšie špecifické údaje, ktoré sa týkajú jednotlivých technológií, ktoré budem popisovať konkrétnejšie v ďalších častiach.

5.0.5 Periodicita zberania metrík

Zberané metriky sa líšia tým, ako veľmi sa v čase menia. Kým záťaž procesora, vstupno-výstupné operácie alebo množstvo prenesených dát sa mení v čase pomerne rýchlo, veľkosť clustera v počte poskytnutých procesorov alebo virtuálnej pamäte sa nemení tak často. Má preto zmysel uvažovať o kontrole intervalu zbierania jednotlivých metrík. Nie len na úrovni zhluku metrík pre konkrétnu technológiu ale aj pre jednotlivé metriky samostatne.

5.0.6 Formát hodnoty metriky

Metriky v podobe záznamov obsahujú čas, kedy bola hodnota nameraná, samotnú hodnotu nejakého sledovaného javu a popisné dáta. Na metrické hodnoty sa môžeme pozerať ako na prírastky alebo ako na absolútne hodnoty. Prírastky hovoria o rozdiele aktuálne nameranej hodnoty a poslednej hodnoty. Absolútne hodnoty predstavujú aktuálnu nameranú hodnotu využitia zdroja.

5.1 Docker

O kontajneroch je možné zberať metriky o všetkých požadovaných zdrojoch, tj. procesor, pamäť, sieť a vstupno-výstupné diskové operácie.

5.1.1 Procesor

Procesor predstavuje jeden z najdôležitejších údajov o vyťažení zdrojov. Docker poskytuje viaceré metriky o procesore, ktorých hodnoty predstavujú výpočetný čas strávený na procesore v jednotkách nanosekúnd:

percpu_usage - obsahuje pole hodnôt, kde každá reprezentuje čas strávený na danom jadre procesora

total_usage - predstavuje sumárnu hodnotu času, ktorý strávil kontajner na všetkých jadrách spolu

usage_in_usermode - predstavuje procesorový čas (pre všetka jadrá), v ktorom došlo k vykonávaniu aplikačného kódu

usage_in_kernelmode - predstavuje procesorový čas (pre všetka jadrá), v ktorom došlo k volaniu kódu jadra operačného systému

5.1.2 Pamäť

Cez API Dockeru je možné získať mnoho podrobných údajov o pamäti. Nie všetky však má význam zberať, preto uvádzam len tie metriky, ktoré zberám. Ich jen

usage - spotreba pamäte v bajtoch

failcnt - počet chýb v pamäti

rss - množstvo pamäte v bajtoch, ktoré kontajner využíva priamo v RAM, tj. okrem stránok uložených na disku alebo okrem častí kódu kontajneru, ktorý ani nebol nahratý do pamäte

pgmajfault - počet prípadov, kedy bolo potrebné načítať chýbajúcu stránku virtuálnej pamäte z disku.

pgfault - počet prípadov, kedy chýbajúca stránka virtuálnej pamäte je načítaná vo fyzickej pamäti, ale nie je o tom záznam v riadacej jednotke pamäte v procesore.

limit - obmedzenie na množstvo použiteľnej pamäte v bajtoch. Buď je táto hodnota nastavená pri spustení kontajnera, alebo predstavuje množstvo pamäte inštalovanej na hosťujúcom počítači.

5.1.3 Sieť

Aby mohli medzi sebou jednotlivé kontajnery komunikovať, Docker im poskytuje sieťové rozhrania. Každé rozhranie má nakonfigurovanú sieť, do ktorej patrí. Na to, aby kontajnery spolu mohli komunikovať, musia byť členmi rovnakej siete. Komunikácia naprieč sieťami nie je možná. Užívatelia si môžu definovať vlastné siete. Docker na vytvorenie týchto sietí poskytuje dva ovládače.

sieť typu most - jednoduchý typ určený pre malé siete.

prekladaná sieť - Docker umožňuje vytvoriť aj sieť, v ktorej sa nachádza viacero hosťujúcich počítačov zároveň. To umožňuje komunikovať medzi sebou aj kontajnerom, ktoré sú spustené v rozličných sieťach, prípadne na inom hosťujúcom počítači.

Metriky siete

Pre jednotlivé sieťové rozhrania je možné zbierať tieto metriky:

rx_bytes - počet prijatých bajtov

rx_dropped - počet prichádzajúcich zahodených bajtov

rx_error - počet chybných bajtov

rx_packets - počet prijatých paketov

tx_bytes - počet odoslaných bajtov

tx_dropped - počet zahodených bajtov pri pokuse o odoslanie

tx_errors - počet odoslaných chybných bajtov

tx_packets - počet odoslaných paketov

5.1.4 Zápis a čítanie z disku

Docker poskytuje metriky o vstupno-výstupných operáciách aj o počte bajtov, ktoré tieto operácie preniesli.

io_service_recursive-Read - predstavuje počet operácií čítania

io_serviced_recursive-Write - predstavuje počet operácií zápisu

io_service_bytes_recursive-Read - predstavuje počet prečítaných bajtov

io_service_bytes_recursive-Write - predstavuje počet zapísaných bajtov

5.1.5 Popisné údaje

Okrem metrických údajov zberám aj nasledovné popisné údaje:

container_id - je to identifikačný reťazec kontajnera, ktorý ho jednoznačne identifikuje na hosťujúcom stroji

host - názov hosťujúceho počítača

image_name - názov Docker obrazu, ktorému zodpovedá spustený kontajner

5.2 libvirt/KVM

Knižnica libvirt taktiež poskytuje údaje o všetkých sledovaných zdrojoch.

5.2.1 Procesor

Libivirt poskytuje údaje o tom, koľko času prepočítal daný virtuálny stroj na svojich virtuálnych procesoroch. Ďalej je možné zistiť, koľko času virtuálny stroj strávil na procesore hosťujúceho počítača. Z hľadiska využitia zdrojov je podstatný druhý údaj. Virtuálne procesory môžu, ale nemusia byť priamo namapované na fyzické procesory. Takisto záťaž na fyzický procesor nepredstavuje len čas prepočítaný na virtuálnom procesore, ale aj režijné náklady spojené s virtualizáciou.

cpu_time - suma času, ktorý strávil virtuálny stroj na procesore hosťujúceho počítača **[?]**

user_time - procesorový čas v nanosekundách, v ktorom došlo k vykonávaniu aplikačného kódu

system_time - procesorový čas v nanosekundách, v ktorom došlo k volaniu kódu jadra operačného systému

5.2.2 Pamäť

O pamäti poskytuje *libivrt* nasledovné štatistiky:

VIR_DOMAIN_MEMORY_STAT_SWAP_IN - množstvo pamäte v kB, ktoré bolo načítané zo swapovacieho priestoru.

VIR_DOMAIN_MEMORY_STAT_SWAP_OUT - množstvo pamäte v kB, ktoré bolo odložené do swapovacieho priestoru.available.

VIR_DOMAIN_MEMORY_STAT_MAJOR_FAULT - počet prípadov, kedy bolo potrebné načítať chýbajúcu stránku virtuálnej pamäte z disku.

VIR_DOMAIN_MEMORY_STAT_MINOR_FAULT - počet prípadov, kedy chýbajúca stránka virtuálnej pamäte je načítaná vo fyzickej pamäti, ale nie je o tom záznam v riadacej jednotke pamäte v procesore.

VIR_DOMAIN_MEMORY_STAT_UNUSED - množstvo pamäte v kB, ktoré je úplne nevyužité systémom.

VIR_DOMAIN_MEMORY_STAT_AVAILABLE - množstvo pamäte v kB, o ktorej virtuálny stroj vie, že ju môže využiť. Títo hodnota môže byť menšia, ako veľkosť pamäte pripísaná virtuálnemu stroju, ak sa používa technika pamäťového balónu.

VIR_DOMAIN_MEMORY_STAT_ACTUAL_BALLOON - aktuálna veľkosť pamäťového balónu v kB.

VIR_DOMAIN_MEMORY_STAT_RSS - množstvo pamäte v kB obsadené procesom, v ktorom beží virtuálny stroj.

VIR_DOMAIN_MEMORY_STAT_USABLE - maximálna veľkosť pamäťového balónu, kedy ešte nedochádza k swapovaniu pamäte virtuálneho stroja na disk.

VIR_DOMAIN_MEMORY_STAT_LAST_UPDATE - Unixová časová známka poslednej aktualizácie štatistík o pamäti.

VIR_DOMAIN_MEMORY_STAT_NR - počet štatistík, ktoré podporuje daný hypervízor.

[**?**] Množstvo poskytovaných štatistík závisí od použitého hypervízora. V prípade MetaCentra sa mi podarilo zberať len niektoré z týchto štatistík. Zároveň ich jednotky neboli kilobajty, ale bajty.

Pamäťový balón

Technika pamäťového balónu umožňuje virtuálnym strojom na jednom hostiteľskom počítači zdieľať pamäť podľa aktuálnych požiadaviek. Ak existujú virtuálne stroje, ktoré pridelenú pamäť využívajú málo, hosťujúci stroj môže túto pamäť prideliť virtuálnym strojom, ktoré jej požadujú viac.

5.2.3 Zápis a čítanie z disku

rd_req - množstvo požiadaviek na čítanie

rd_bytes - počet prečítaných bajtov

wr_req - množstvo požiadaviek na zápis

wr_bytes - počet zapísaných bajtov

5.2.4 Sieť

Pre jednotlivé sieťové rozhrania je možné zbierať tieto metriky:

rx_bytes - počet prijatých bajtov

rx_dropped - počet prichádzajúcich zahodených bajtov

rx_error - počet chybných bajtov

rx_packets - počet prijatých paketov

tx_bytes - počet odoslaných bajtov

tx_dropped - počet zahodených bajtov pri pokuse o odoslanie

tx_errors - počet odoslaných chybných bajtov

tx_packets - počet odoslaných paketov

5.2.5 Popisné údaje

Okrem metrických údajov zberám aj nasledovné popisné údaje:

host - názov hosťujúceho počítača

guest_name - názov virtuálneho stroja

5.3 Hadoop

5.3.1 Metriky clustera

Hadoop poskytuje o clusteri cez svoje Cluster Metrics API YARN-u rôzne údaje. Niektoré reprezentujú využitie zdrojov ako procesor a pamäť, iné poskytujú prehľad o aplikáciách a o uzloch v rámci celého clustera. Využívam všetky informácie poskytnuté týmto API.

appsSubmitted - počet nahratých aplikácií.

appsCompleted - počet dokončených aplikácií.

appsPending - počet aplikácií, ktoré čakajú na svoje spustenie.

appsRunning - počet aplikácií práve počítaných v clusteri.

appsFailed - počet aplikácií, ktorých výpočet zlyhal.

appsKilled - počet aplikácií, ktorých beh bol ukončený pred dokončením výpočtu.

reservedMB - množstvo pamäte v MB rezervovanej pre beh aplikácií.

availableMB - množstvo pamäte v MB, ktorá je dostupná pre beh aplikácií.

allocatedMB - množstvo pamäte v MB, ktorá je práve alokovaná bežiacimi aplikáciami.

totalMB - celkové množstvo pamäte clustera.

reservedVirtualCores - počet virtuálnych jadier rezervovaných pre beh aplikácií.

availableVirtualCores - počet dostupných virtuálnych jadier.

allocatedVirtualCores - počet alokovaných virtuálnych jadier.

totalVirtualCores - celkový počet virtuálnych jadier v clusteri.

containersAllocated - poček alokovaných kontajnerov.

containersReserved - The number of containers reserved

containersPending - The number of containers pending

totalNodes - počet fyzických uzlov clustera.

activeNodes - počet aktívnych uzlov clustera.

lostNodes - počet nedostupných uzlov.

unhealthyNodes - počet "nezdravých" uzlov - tj. uzly, ktoré majú napríklad málo voľného diskového priestoru, alebo sú inak obmedzené.

decommissionedNodes - počet uzlov vyradených z prevádzky.

rebootedNodes - počet reštartovaných uzlov.

5.3.2 Metriky aplikácií

Cluster Application API YARN-u poskytuje informácie o jednotlivých aplikáciách, ktoré sú spustené na clusteri. Niektoré údaje využívam ako zdroj metrických dát, iné ako metadáta. Niektoré nemá význam zberať. Zbieram nasledovné metrické dáta o spustených aplikáciách:

allocatedMB - suma pamäte v megabajtoch, ktorá je alokovaná kontajnermi, v ktorých beží aplikácia.

allocatedVCores - súčet virtuálnych jadier, ktoré sú alokované kontajmermi, v ktorých beží aplikácia.

elapsedTime - čas v milisekundách, ktorý uplynul od spustenia aplikácie.

runningContainers - počet kontajnerov, v ktorých je spustená aplikácia.

memorySeconds - množstvo pamäte v megabajtoch, ktoré aplikácia alokovala, vynásobené časom behu aplikácie v milisekundách.

vcoreSeconds - počet procesorových jadier, ktoré aplikácia alokovala, vynásobený časom behu aplikácie v milisekundách.

Ako popisné dáta k metrikám pripájam tieto údaje:

id - identifikátor aplikácie v rámci clustera.

user - užívateľ, ktorý spustil aplikáciu.

name - názov aplikácie.

clusterId - identifikátor clustera, v ktorom je spustená aplikácia.

5.3.3 Metriky uzlov

Nodes API systému YARN poskytuje metriky o využití jednotlivých uzlov clustera. Podobne ako aplikačné API, poskytuje aj nadbytočné informácie, ktoré nemajú pre monitorovanie využitia zdrojov význam. Zbieram nasledovné metrické dáta o využití zdrojov uzlov:

usedMemoryMB - celkové množstvo pamäte v megabajtoch aktuálne využívané uzlom na beh aplikácií.

availMemoryMB - celkové množstvo pamäte v megabajtoch aktuálne dostupné uzlu na beh aplikácií.

usedVirtualCores - počet virtuálnych jadier, ktoré uzol využíva na beh aplikácií.

availableVirtualCores - počet virtuálnych jadier, ktoré uzol využíva na beh aplikácií.The total number of vCores available on the node

numContainers - celkový počet kontajnerov aktuálne spustených na uzle.

Ako popisné dáta k metrikám pripájam tieto údaje:

rack - umiestnenie uzla v racku.

id - identifikátor uzla.

nodeHostName - názov uzla.

Chapter 6

Analýza a návrh

6.1 Heterogénna infraštruktúra MetaCentrum

Projekt MetaCentrum vznikol v roku 1996 a od roku 1999 je jeho činnosť zastrešovaná organizáciou CESNET. Zaoberá sa budovaním národnej gridovej infraštruktúry a prepojením s podobnými projektami za hranicami Českej republiky. Projekt je oficiálnou súčasťou Európskej gridovej iniciatívy (EGI). Úlohou MetaCentra je predovšetkým koordinácia a rozširovanie infraštruktúry či už o vlastné zdroje alebo prostredníctvom partnerov, ktorý poskytujú výpočetný výkon svojich clusterov. Jedná sa hlavne o akademickú spoluprácu. MetaCentrum spravuje výpočetné prostriedky a dátové úložiská AV, JČU, MU, MZLU, UK, VUT, ZČU. V súčasnosti disponuje (stav k 30.7. 2010) 1500 procesorovými jadrami, 100 TB využiteľnej diskovej kapacity v podobe poľa a 400 TB kapacity v podobe pások. Služby využíva 385 registrovaných aktívnych užívateľov, ktorí spolu na 750 tisíc úlohách využili 7 miliónov hodín procesorového času.

MetaCentrum primárne poskytuje svoj výpočetný výkon a úložnú kapacitu. Taktiež sprístupňuje svoje programové vybavenie a vývojové prostredie a hlavne množstvo aplikácií využívaných na výskumné účely, ako napr. Ansys, Gaussian, Matlab, Mathematica. Taktiež sa venuje vývoju v oblasti gridového a cloudového počítania, napr. v oblasti plánovania, gridového middleware, optimalizácie a paralelizácie výpočtov a virtualizácie infraštruktúry. Dôležitou funkciou je účasť na medzinárodných projektoch, využívanie medzinárodnej výpočetnej infraštruktúry a využívanie skúseností na rozvoj v domácom prostredí.

Prostredie MetaCentra predstavuje heterogénnu infraštruktúru. Zahŕňa dedikované Hadoop klastre na spúšťanie špecifických aplikácií, ktoré riešia náročné úlohy. Pre potreby zložitejšieho využitia výkonu MetaCentrum poskytuje možnosť vytvárania virtuálnych strojov alebo spúšťanie aplikačných kontajnerov.

6.2 Požiadavky na aplikáciu

6.2.1 Vysoký monitorovací výkon

Cloudová infraštruktúra MetaCentra pozostáva z mnohých výpočetných uzlov. Sú prepojené vysokorýchlostnými sieťami. Je potrebné zbierať dáta o využití množstva zapojených clusterov a uzlov, na ktorých je tiež spustených mnoho výpočetných úloh či virtuálnych strojov. Metriky sú zbierané periodicky v určitých intervaloch z jednotlivých uzlov. Databáza, do ktorej sú pravidelne odosielané metrické dáta, musí byť schopná spracovávať desiatky tisíc záznamov za sekundu.

6.2.2 Nízka nadbytočná záťaž

Primárnou úlohou cloudu je poskytovanie svojho výpočetného výkonu a prostriedkov. Je žiadúce, aby monitorovacia aplikácia predstavovala čo najmenšiu záťaž pre systémy, na ktorých je spustená. Ak by monitorovanie samotné spotrebovávalo príliš veľa zdrojov, takto získané metriky by nemali požadovanú presnosť, nakoľko by samotná kapacita infraštruktúry bola obmedzená už len behom monitorovacej aplikácie. Je to možné docieliť výberom efektívneho programovacieho jazyka, napr. C, C++, prípadne rýchle skriptovacie jazyky ako Python, Ruby či Go. Tie ale predsalen predstavujú istú nadmernú záťaž súvisiacu so spúšťaním interpretera. Nie je príliš vhodné používať jazyky, ktoré na svoj beh potrebujú ďalšiu vrstvu v podobe virtuálneho stroja, ako napr. Java.
Pre porovnanie uvádzam test niekoľkých aplikácií, ktorý sleduje spotrebu zdrojov jednotlivých monitorovacích aplikácií. Tieto

6.2.3 Škálovateľnosť

Monitorovacia aplikácia by mala poskytovať zrovnateľné výsledky v oblasti rýchlosti odozvy v prípade, že bude spustená na jednom uzle, ale aj v prípade, že jej úlohou bude monitorovať stovky výpočetných uzlov s množstvom spustených inštancií, ktoré treba sledovať. To je možné docieliť paralelizovaním dotazov na metriky.

6.3 Miesto nasadenia zbernej aplikácie

Zber metrík je možné realizovať v prípade virtuálnych strojov alebo aplikačných kontajnerov aj z ich vnútra. Bolo by možné v každej takejto virtualizovanej inštancií spustiť jednoduchú monitorovaciu aplikáciu, ktorá

by interagovala priamo so systémom a potom by odosielala dáta. To ale nie je vhodné riešenie, nakoľko sa nedá predpokladať virtualizovaný operačný systém a zároveň to predstavuje zásah do užívateľského priestoru. Je lepšie realizovať zber metrík na základe integrácií s aplikáciami, kedy je monitorovacia aplikácia nasadená na fyzickom operačnom systéme a zberá metriky o virtualizačných aplikáciách, ktoré sú využívané.

6.4 Návrh monitorovacieho riešenia

Monitorovacie riešenie bude využívať viacero súčastí, ktoré zabezpečujú jednotlivé činnosti súvisiace s monitorovaním tak, ako som ich rozoberal v časi 2.1 Všeobecné problémy monitorovania. Aplikácia bude zberať metriky, ktoré sú popísané v predošlej kapitole.
Samotný proces zberu metrických dát bude zabezpečený aplikáciou collectd. Oproti ostatným dostupným aplikáciám disponuje veľkou mierou rozšíriteľnosti, kde je možné rozširovať nie len škálu dát, ktoré sa majú zberať, ale aj spôsobom, ako a kam sa budú dané dáta zapisovať. Ďalej umožňuje rozšíriť funkcionalitu aj v systéme upozornení. Collectd bude zabezpečovať prípadné odosielanie notifikácií prostredníctvom emailu. Ďalšou z výhod zvolenej aplikácie je, že na uskutočňovanie zberu a zápisu používa viacero paralelne bežiacich vlákien, ktoré si medzi sebou rovnomerne delia záťaž. Ich počet je možné definovať pri konfigurácií.
Jednou z ďalších výhod collectd oproti iným softvérom je spôsob, akým pristupuje k spúšťaniu kontrol. Podľa konfigurácie pri štarte zistí, ktoré moduly sa budú používať. Takisto dôjde ku konfigurácií jednotlivých modulov, napr. nastavenie potrebných ciest k požadovaným súborom. Následne dôjde k inicializácií jednotlivých modulov. V tejto fáze moduly inicializujú prostriedky, ktoré potrebujú v priebehu zberu metrík. Napr. pripojenie na správcu kontajnerov alebo hypervízora. Nie je efektívne, aby boli tieto prostriedky incializované pri každej požiadavke na metriku, pretože by to spomaľovalo proces samotného zberu dát. Takýto spôsob bol jedinou možnosťou pri niektorých iných aplikáciách.
Potom nasleduje fáza behu. Collectd beží v režime démona na pozadí a periodicky spúšťa jednotlivé moduly, ktoré zisťujú metrické dáta. V rámci modulov sa periodicky zisťuje, ktoré virtuálne stroje alebo kontajnery sú spustené a ktorých sa tým pádom budú zberať metriky.
Po nameraní hodnoty collectd inicializuje zápis prostredníctvom modulu Write TSDB. Modul má optimalizovanú činnosť v podobe cachovania hodnôt a ich odosielania nie po jednom ale vo väčších dávkach.

Démon takisto riadi ukončovanie činnosti modulov. Až v tejto fáze moduly uvoľnia všetky prostriedky, ktoré mali naalokované.
Samotný collectd bez pluginov predstavuje riadiaciu aplikáciu v podobe démona, ktorá na svoju inštaláciu vyžaduje minimum závislostí. To je ďalšou z výhod, keďže démon na zber metrík bude nainštalovaný na každom monitorovanom uzle.
Na zápis monitorovacích dát bude využitá databáza časových rád. Budem sa jej venovať v nasledujúcej časti.
Takýto návrh zabezpečuje decentralizované a škálovateľné riešenie, kedy je distribuovaný nielen zber metrických údajov, ale aj ich zápis a uchovávanie.
Collectd rozšírim o jednotlivé pluginy, ktoré budú zberať metrické údaje o využití zdrojov jednotlivými technológiami. Pluginy budú konfigurovateľné, vyvinuté s dôrazom na paralelizáciu. Budú mať schopnosť automatickej detekcie nových inštancií kontajnerov a virtuálnych strojov.

6.4.1 Paralelizácia v rámci pluginov

Collectd poskytuje mechanizmus pracovných vlákien, v ktorých vykonávajú pluginy zber dát. To umožňuje paralelne spúšťať jednotlivé pluginy. Môže ale nastať situácia, kedy plugin zberá dáta o stovkách kontajnerov či virtuálnych strojov. Pluginy som preto navrhol tak, že využívajú paralelizáciu pri zisťovaní monitorovacích dát o jednotlivých inštanciách.

6.4.2 Databáza časových rád

Ako databázu na uchovávanie časových dát som si zvolil OpenTSDB. Dôvodom je používanie databázy HBase. Je to distribuovaná databáza určená pre veľké objemy dát v rádoch stoviek miliónov a miliárd záznamov. Je typom NoSQL databázy. Oproti SQL databázam je linárne škálovateľná. Ak dôdje k zdvojnásobeniu výpočetných zdrojov, dôjde aj k zdvojnásobeniu výkonu databázy. To je dôležité pri zbere časových dát z mnohých uzlov, ktoré sa v infraštruktúre MetaCentra nachádzajú. Ďalším dôvodom je, že v MetaCentre je aktuálne databáza HBase využívaná, čo predstavuje zjednodušenie nasadenia.
Monitorovacie dáta môžu byť uchovávané po dobu rokov až desiatky rokov. Súčasťou môjhom riešenia bude aj sada nástrojov, ktoré budú umožňovať zmenu granularity už zapísaných metrických dát. Tým je možné prispieť k úspore úložnej kapacity a zároveň kontinuálne reagovať na zastarávanie údajov.

Pre potreby monitorovania záťaže a reakcie na danú hodnotu je vo svojej podstate významná len aktuálna záťaž prostriedkov, prípadne záťaž v okne niekoľkých hodín dozadu. Potreba uchovávania dát je tu z dôvodu účtovania a pre možnosť sledovať vývoj využívania zdrojov v čase. Podľa toho, ako dlho zberáme metriku, sa mení aj požiadavka na interval v akom je potrebné hodnoty uchovávať. Čím staršie hodnoty sú, tým menšia granularita nám postačuje.
OpenTSDB podporuje podobnú funkciu pri vyhodnocovaní požiadaviek - tj. ak by vo vyhodnotení požiadavky mali byť státisíce dátových bodov, dôdje k súčtu hodnôt za určité obdobie, ktoré je nahradené len jedným bodom s hodnotou tohoto súčtu. Nie je ale podporovaná kontinuálna zmena na úrovni záznamov v databáze.

Zmena granularity dát

Zmena granularity už zapísaných dát je daná dvojicami parametrov, ktoré pre účely tejto práce nazývam *historický interval* a *požadovaná granularita*. *Historický interval* definuje, s akými dátami pracovať a *požadovaná granularita* určuje interval, v ktorom sa za danú dobu majú uchovávať dáta. Napr. dáta staršie ako rok je potrebné uchovávať v intervale jedného dňa a dáta staršie ako mesiac v intervale jednej hodiny.
Jednou súčasťou monitorovacieho riešenia bude nástroj, ktorý upraví dáta v časovej databáze do požadovanej podoby. Taktiež bude možné vybrať, na ktorých metrikách prebehne táto transformácia. Nástroj bude spúšťaný jednorázovo v dobe, keď bude infraštruktúra pomerne slabšie vyťažená. Nebude sa teda jednať o kontinuálny dohľad nad starnutím hodnôt a ich úpravy. Takýto prístup by vyťažoval zdroje neustálym prepočtom hodnôt, ktoré sa plynutím času presúvajú medzi intervalmi s rôznymi granularitami. Vzhľadom na to, aké ciele plní táto funkcionalita, je vhodnejšie spúšťať transformáciu databázy ručne - aj keď prakticky v určitých časových intervaloch, napr. každý týždeň alebo mesiac.
Vo svojej podstate kontinuálny dohľad tiež predstavuje len periodické prepočítavanie, kde je interval prepočtov daný najmenším intervalom zo spomínaných dvojíc. Intervaly *požadovanej granularity* sú spravidla jeden násobkom druhého (napr. hodina a deň), takže by sa periodicky opakovala náročnosť prepočtov a teda aj dĺžka ich trvania. Periodicita však môže byť udaná aj ručným spúšťaním aplikácie, kde množstvo prepočtov závisí od toho, koľko intervalov *požadovanej granularity* bolo prekročených od poslednej agregácie. Avšak prepočty je možné naplánovať na vhodnú dobu, kedy je infraštruktúra menej vyťažená.

Nástroj bude ako svoj vstup využívať historické politiky, ktoré definujú spomínané dvojice intervalov a zároveň aj zoznam metrík a ich príslušnosť k jednotlivým politikám.

Chapter 7

Implementácia

Využijem existujúcu aplikáciu na zbieranie metrických dát *collectd*. Tá poskytuje mechanizmy na periodické spúšťanie merania metrík, analýzu zozbieraných hodnôt a generovanie hlásení. Na zbieranie jednotlivých metrík som vytvoril moduly pre tento program. Tieto údaje následne bude odosielať do databázy OpenTSDB pomocou modulu WriteTSDB.

7.1 Techniky zbierania metrík

Rôzne technológie MetaCentra poskytujú svoje metrické údaje cez rôzne programové rozhrania alebo prostredníctvom textového výstupu zvyčajne dostupného prostredníctvom HTTP servera.

7.1.1 Docker

Na komunikáciu s Dockerom je možné využiť:

príkazy aplikácie v príkazovom riadku

Remote API

Používanie príkazov aplikácie môže byť o čosi rýchlejšie, ale následne by bolo potrebné analyzovať textový výstup programu. Rozhodol som sa použiť Remote API. Toto API funguje pre účely monitorovania na princípe REST a odpovede vracia vo formáte JSON, čo predstavuje zjednodušenie spracovania výstupu. Démon Dockeru "počúva" na lokálnom sockete, čo by nemalo spôsobovať výrazné oneskorenie odpovede. Každému kontajneru zodpovedá adresa, ktorá zahŕňa identifikačný reťazec kontajnera.

7.1.2 libvirt/KVM

Na získavanie metrických údajov o virtuálnych strojoch som použil knižnicu libvirt. Je to knižnica napísaná v jazyku C. Poskytuje množstvo rozhraní,

ktoré umožňujú vytvárať virtuálne stroje, zapínať ich a vypínať, meniť ich konfiguráciu. Poskytuje tiež údaje o tom, ako virtuálny stroj využíva virtualizované zdroje.
Najprv je potrebné vytvoriť spojenie s hosťujúcim počítačom. Toto je udržované počas celého zberu metrík. Cez vytvorené spojenie je možné zisťovať, ktoré virtuálne stroje sú spustené. V periodických intervaloch aktualizujem zoznam bežiacich strojov a odosielam metriky len o nich.
Collectd už obsahoval plugin pre libvirt, no bolo potrebné doplniť niektoré popisné údaje metrík, ako názov hosťujúceho počítača, názov virtuálneho stroja a iné. Taktiež som rozšíril tento plugin o paralelný zber metrík z jednotlivých virtuálnych strojov.

7.1.3 Hadoop

Na monitorovanie Hadoopu som si zvolil REST API YARN-u. YARN zabezpečuje správu zdrojov a plánovanie/monitorovanie vykonávania úloh do dvoch oddelených démonov. [**?**] ResourceManager je autoritou, ktorá rozhoduje o využívaní zdrojov aplikáciami v celom systéme. NodeManager je agent, nainštalovaný na jednotlivých uzloch, ktorý sa stará o beh kontajner a monitorovanie zdrojov, ktoré spotrebovávajú. Toto nahlasuje ResourceManagerovi. Preto som zvolil toto API.
REST API poskytuje odpovede vo formáte JSON a XML. Vo svojej implementácií využívam odpovede vo formáte JSON.

7.2 libcurl

libcurl je voľne šíriteľná klientska knižnica v jazyku C určená na manipuláciu so zdrojmi identifikovanými pomocou URL. Podporuje množstvo protokolov ako FTP, FILE, HTTPS, IMAPS, LDAP, POP3, SMTP, SCP atď. Podpourje taktiež SSL certifikáty, HTTP POST a PUT metódy a rôzne formy autentifikácie. V mojej implementácií túto knižnicu využívam na získavanie metrík zo systému Hadoop. Využívané REST API sú chránené autentifikačným mechanizmom Kerberos. libcurl vie využiť autorizačné tokeny, ktoré sú vygenerované pri autentifikácií pomocou Kerberosu. Posiela ich spolu s HTTP GET žiadosťou a týmto spôsobom sa autentifikuje voči Hadoopu.
libcurl je ľahko prenositeľná medzi rôznymi platformami, ako sú Solaris, BSD platformy, Windows, Linux, OS X a iné. Je vláknovo bezpečná, rýchla, dobre zdokumentovaná a kompatibilná s protokolom IPv6. [**?**]

7.3 libpthread

libpthread je knižnica, ktorá umožňuje programovanie aplikácií s paralelne vykonávanými procedúrami. Tie sa nazývajú vlákna. Paralelizáciu v systéme na vyššej úrovni prináša koncept procesov. Pri vytváraní procesu ale dochádza k režijnej záťaži v operačnom systéme. Procesy si v systéme udržujú viacero parametrov, ako identifikačné číslo, identifikáciu užívateľa, premenné prostredia, zásobník, haldu, registre, zdieľané knižnice a reagujú na signály. Vlákna využívajú tieto atribúty a zavádzajú menšiu množinu parametrov a kontrolných signálov, ktoré umožňujú paralelné vykonávanie aj v rámci jedného procesu.
Túto knižnicu využívam v plugine pre Docker. V pravidelných intervaloch dochádza k zisťovaniu, ktoré kontajnery sú spustené a na REST API zodpovedajúcich kontajnerov sú potom paralelne vytvorené a zaslané požiadavky na metrické údaje.

7.4 cJSON

cJSON je miniatúrna knižnica v jazyku C, ktorej úlohou je prevod reťazca vo formáte JSON do hierarchickej štruktúry objektov. Zjednodušuje to analýzu obdržaných informácií vo formáte JSON. Keďže jazyk C nepozná objekty, namiesto toho sú používané štruktúry jazyka C. Knižnica neobsahuje žiadne závislosti, na jej činnosť sú potrebné dva súbory - jeden so zdrojovými kódmi a jeden hlavičkový súbor.
Knižnicu bolo potrebné mierne upraviť, pretože nesprávne narábala s dátovým typom long, ktorý sa vyskytuje ako hodnota niektorých parametrov vo vracaných JSON reťazcoch. Aj keď knižnica obsahovala tento nedsotatok, zvolil som si ju pre jej minimalistickú formu.

7.5 Zapisovací plugin Write TSDB

Plugin pre collectd s názvom Write TSDB zapisuje metriky do OpenTSDB. [?] Je napísaný v jazyku C. Bolo potrebné ho upraviť, aby požadovaným spôsobom generoval názvy metrík z údajov, ktoré mu predávajú pluginy zberajúce metriky. Plugin využíva mechanizmy dočasného zhlukovania správ a ich odosielania vo väčších dávkach. Je možné nakonfigurovať adresu a port, na ktorom je spustený démon OpenTSDB. Taktiež je možné nakonfigurovať tagy špecifické pre uzol. Toto vo svojej implementácií nevyužívam, všetky tagy metrík sú generované pluginmi.

Chapter 8

Záver

Appendix A

Kapitola priloha