Masarykova univerzita
Fakulta informatiky

Distributed Complex Event Processing

Diplomová práce

Bc. Tomáš Skopal

Brno, jaro 2016

Místo tohoto listu vložte kopie oficiálního podepsaného zadání práce a prohlášení autora školního díla.

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Bc. Tomáš Skopal

Vedoucí práce: RNDr. Filip Nguyen

Shrnutí

Goal of this thesis is to develop a Peer to Peer algorithm for distributed Event Pattern matching.. The application should be able to run any number of processing nodes. For the needs of this thesis, example of 4 nodes will be sufficient.

Klíčová slova

keyword1, keyword2, …

Obsah

1 Úvod

Cílem této práce je vytvoření ...

2 Zpracování událostí

Se zvyšujícím se počtem zařízení, která jsou schopna produkovat data, se zvyšuje potřeba tato data analyzovat. Běžně rozšířeným způsobem je zpracování dat dávkově. Tedy, data se uloží a ve vhodnou dobu, typicky v noci, se analyzují.

Pokud však uvažujeme reálný provoz na síti, který se dnes v centrálních uzlech pohybuje okolo $1Tb/s$, je dávkové zpracování nereálné. Potřebujeme data analyzovat za běhu (angl. real time).

Jednotkou zpracování dat je událost (angl. event). Událost je základním pojmem používaným v oblasti zpracování událostí. Je definována jako objekt, který reprezentuje záznam o aktivitě v daném systému. Událost může mít vlastnosti. Typickým příkladem takové vlastnosti je čas vzniku události, příčina jejího vzniku nebo její typ. [1] Jednoduchým příkladem události může být paket. Je to datová schránka, která obsahuje informace, které můžeme analyzovat. Samostatný paket nemá téměř žádnou vypovídající hodnotu, kdežto proud paketů je základem Internetu.

Takový proud událostí skrývá množství dat, která je možné získat až při komplexní analýze, která zohledňuje více událostí v řadě. To nazýváme *komplexní zpracování dat (angl. complex event processing neboli CEP)*

2.1 CEP

Je těžké shrnout celý vědní obor pod jednu všeobjímající definici. David Luckham ve své knize THE POWER OF EVENTS: AN INTRODUCTION TO COMPLEX EVENT PROCESSING IN DISTRIBUTED ENTERPRISE SYSTEMS [1] říká, že CEP je soubor technik a nástrojů, které pomáhají k pochopení a kontrole událostmi řízených systémů.

Jak už bylo řečeno, množství událostí v systémech je enormní. Při jejich zpracování se setkáváme s pojmem *komplexní událost*. Taková událost se může vyskytnout pouze jako reakce na sled jiných, dílčích, událostí. Dílčí události mohou spolu souviset mnoha různými způsoby, nejčastěji je však spojujeme na základě vlastností (čas vzniku, příčina vzniku, typ, atd).

Příkladem komplexní události může být akce nakoupení produktu v internetovém obchodě. Je to v dnešní době elektronického marketingu velké téma. Běžnému uživateli Internetu je v mnoha kanálech (Facebook ads, Google ads, mailing) zobrazována reklama. Některý uživatel nakoupí produkt při prvním zobrazení určité reklamy. Jiný uživatel potřebuje reklamu vidět alespoň pětkrát, než nakoupí. Způsob jak tuto "cestu"měřit se jmenuje atribuční model. Atribuční model je tak defacto soubor událostí, které vyvolaly konečnou, komplexní, událost. Tedy nákup produktu v obchodě. S roustoucím počtem zařízení, roste počet reklam a také se komplikují atribuční modely. Vhodnými technikami zpracování dílčích událostí (zobrazení reklamy, kliknutí na reklamu, nainstalování aplikace) lze například predikovat chování uživatele.

CEP nabízí techniky pro definici a využití vztahů mezi událostmi. Může být využíván pro analýzu libovolného typu událostí v aplikaci, počítačové síti nebo v informačním systému. Jednou z těchto technik je i definování vlastních událostí, jakožto pravidla. Jinak řečeno, můžeme vytvořit vlastní reakci na soubor určitých událostí v našem systému. Touto cestou můžeme pochopit co se v našem systému odehrává.

To zvyšuje míru flexibility. Uživatel může za pomocí CEP specifikovat taková pravidla, která jej aktuálně zajímají a jsou pro něj přínosem. Může analyzovat jak nízko-úrovňové, tak vysoko-úrovňové procesy. Různé druhy událostí mohou být v CEP monitorovány současně. Velkou výhodou je, že pravidla mohou být měněna, odebírána a přidávána za běhu, tedy bez výpadku systému.

Zpracování proudu událostí a vyhodnocení, jestli se pravidlo vyskytlo, stojí samozřejmě určitý výpočetní výkon. V závislosti na typu a množství dat. Pokud je dat hodně (například analýza síťového provozu), musíme výpočet distribuovat.

2.2 Distribuované CEP

Úvodem kapitoly je potřeba jasně vymezit co v tomto kontextu chápeme pod pojmem "distribuovaný". Distribuované systémy definujeme jako systémy, ve kterých spolu hardwarové nebo softwarové komponenty komunikují pouze zasíláním zpráv. Distribuovaný výpo-

čet, je pak takový výpočet, který probíhá v distribuovaném systému. [2]

Požadavky na distribuované zpracování událostí (DCEP) jsou v mnoha ohledech jiné než na zpracování centralizované. Dvě hlavní vlastnosti, které vyžadujeme jsou vysoká míra dostupnosti (angl. availability) a nízké zpoždění (angl. latency). Cílem je pak maximalizovat dostupnost a minimalizovat zpoždění. Bohužel u většiny návrhů platí, že tyto vlastnosti jsou závislé a zlepšení jedné, zhoršuje druhou. Zlepšení dostupnosti, zvýší zpoždění, protože potřebujeme více času na synchronizaci všech řídících informací. [3]

Zmíněné dvě vlastnosti nejsou jediné. Mezi vlastnosti distribuovaného zpracování události můžeme dále zařadit:

- rovnoměrné rozdělení dat mezi výpočetní uzly (angl. data partitioning)
- automatické škálování výpočtu
- tolerance chyb
- správa datového úložiště

3 Nástroje pro distribuované zpracování událostí

3.1 Apache Samsa

3.2 Apache Storm

3.3 CAVE

http://dl.acm.org/citation.cfm?doid=2675743.2771834

4 Analýza a návrh

se lehce dostknu toho co zminenym technologiim chybi a co by navrhovane reseni (ktere budu popisovat dale) umoznilo delat jinak. Bude to priprava ctenare na to co prijde. Mely by zde byt take nastineny vstupy a vystupy a obecny navrh reseni. Bez znalosti toho jak to konkretne bude udelane. Defacto interface. Zde bude zakomponovan i prepis filipova zadani

5 Vytvoření clusteru v rámci sítě

5.1 Motivace

5.2 Technologie

Tato kapitola popisuje jednotlivé technologie, které jsou použity při implementaci algoritmu. Na konci každé subsekce bude popis toho jak konkrétně je technologie použita v mém řešení.

5.2.1 Apache Maven

5.2.2 Apache Kafka

5.2.3 Apache ZooKeeper

Popis technologie podle dokumentace

Konkrétní použití: Jak se sestavuje strom a k čemu slouží (je důležité zmínit, že jde pouze o virtuální stav a je nutné myslet na to, že aplikace běží na daném uzlu pouze jednou). Jaké má zookeeper tree výhody (dá se zjistit jestli a jaké má uzel potomky, dá se kterýkoli uzel modifikovat, což způsobí příslušnou změnu v aplikaci).

CuratorFramework

5.2.4 Esper

5.3 Události v systému

Kapitola popisující hlavní myšlenku povahy dat, která bude algoritmus vyhodnocovat. Také zde budou ukázky používaných dat.

5.3.1 Hrubozrnné

5.3.2 Jemnozrnné

5.4 Konfigurace

5.5 Implementace

Pro implementaci byla zvolena Java (konkrétně ve verzi 1.8), protože všechny použité technologie mají dobrá API pro Javu a většina příkladů je právě v Javě. Dalším důvodem je také to, že Java se dobře hodí pro běh aplikací tohoto druhu, protože má dobrou práci s vlákny. Posledním, méně důležitým, důvodem je popularita Javy a snadná čitelnost kódu.

5.5.1 Architektura aplikace

Popis jednotlivých maven modulů. K čemu který slouží

5.5.2 Iniciální spuštění

5.5.3 Přechody mezi stavy

Jedna z nejdůležitějších kapitol, která ukazuje co způsobí, že aplikace začne vyhodnocovat jemnozrnné události. Počínaje tím, že Esper vyhodnotí proud událostí a emituje ep-událost. Dále přes zpracování ep-události, nastavení příslušných dat jednotlivým zk-uzlům v zk-stromě, či modifikaci zk-stromu. Až po ukončení zpracovávání jemnozrnný událostí a návrat k iniciálnímu stavu.

Bude zde také zmíněno dynamické nasazování nových ep pravidel. To by se dalo shrnout pod nadpis "ovládání clusteru z venčí"- řešeno přes nastavování dat jednotlivým uzlům v zookeeperu.

5.5.4 Esper pravidla

Mini kapitola, kterou bych věnoval použitým esper pravidlům.

5.6 Demo

Nějaké print screeny. Jednoduše ukázka běhu programu.

5.7 Známá omezení

Diskuse nedostatků nebo možných vylepšení výše navrženého řešení.

- Momentální nemožnost spustit více consumerů na jednom stroji.
- Velmi náročný monitoring a spouštění jednotlivých uzlů.
- Zatím nevím jak je to s dynamickým přidáváním nových uzlů do hierarchie.
- Nejsou vůbec otestovány výpadky některých uzlů. Zookeeper to zvládne, kafka také, ale co se stane s virtuálním zk-tree v aplikaci?

6 Závěr

Závěr bude v tomto případě obsahovat obšírnější zhodnocení toho jak se povedlo splnit zadání. Že výsledkem práce je navržené řešení za použití kafky, zk, esperu, Javy.

Bibliografie

[1] LUCKHAM, David. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Pearson Education, Inc., 2002, ISBN 9780201727890

[2] COULOURIS, George F. *Distributed systems: concepts and design* [online]. 5th ed. Boston: Addison-Wesley, c2012. ISBN 01-321-4301-1. Dostupné z: `https://azmuri.files.wordpress.com/2013/09/george-coulouris-distributed-systems-concepts-and-design-5th-edition.pdf`

[3] KAMBURUGAMUVE, Supun; FOX, Geoffrey; LEAKE, David and QIU, Judy. *Survey of Distributed Stream Processing for Large Stream Sources*. Technical report, 2013. [online]. Dostupné z: `http://grids.ucs.indiana.edu/ptliupages/publications/survey_stream_processing.pdf`