Příjemci podpory:

Vysoké učení technické v Brně
Fakulta informačních technologií

Poskytovatel:

Ministerstvo vnitra ČR

***Integrovaná platforma pro zpracování digitálních dat z bezpečnostních incidentů* (TARZAN)**

Identifikační kód VI20172020062

Název předkládaného výsledku: Big Data cluster založený na kontejnerech

Typ výsledku dle UV č. 837/2017	Evidenční číslo (příjemce)	Rok vzniku
O ostatní	FIT-TR-2019-4	2019
ISBN-ISSN	**Webový odkaz na výsledek**	**Kde a kdy publikováno**
	https://www.fit.vut.cz/research/publication/12141/	Technická zpráva FIT VUT v Brně

Anotace k výsledku:

V dnešní době hledáme stále nové způsoby jak zjednodušit nasazení aplikací a služeb jednotným způsobem nezávisle na hardware. Proč tímto způsobem nenasazovat i nástroje pro zpracování velkých dat jako jsou Apache Spark a Apache Ignite. Technologiemi, které toto umožňují jsou Docker Swarm a Kubernetes. Na jejich základě je možné postavit Big Data výpočetní cluster.

Řešitelský tým: *Petr Matoušek (manažer a hlavní řešitel), Kamil Jeřábek (realizační tým)*

Big Data cluster založený na kontejnerech

Big Data výpočetní cluster založený na Docker Swarm a Kubernetes

FIT-TR-2019-04

Kamil Jeřábek

Dokument k projektu TARZAN
Fakulta informačních technologií, Vysoké učení technické v Brně

Naposledy změněno: 2. ledna 2020

Big Data cluster - Uživatelská příručka

Kamil Jeřábek

Vysoké učení technické v Brně, email: ijerabek@fit.vutbr.cz

Abstrakt V dnešní době hledáme stále nové způsoby jak zjednodušit nasazení aplikací a služeb jednotným způsobem nezávisle na hardware. Proč tímto způsobem nenasazovat i nástroje pro zpracování velkých dat jako jsou Apache Spark a Apache Ignite. Technologiemi, které toto umožňují jsou Docker Swarm a Kubernetes. Na jejich základě je možné postavit Big Data výpočetní cluster.

1 Úvod

V dnešní době stále hledáme způsoby jak postavit výpočetní clustery čím jak nejjednodušším způsobem. Pro vývoj a testování je velice výhodné mít možnost rychle a jednoduše zkoušet různé kombinace a verze služeb. Zároveň je výhodné používat jeden management systém, který je použitelný i pro ostatní služby. Což pak snižuje také náročnost a cenu udržitelnosti takových systémů.

Vyvíjený Big Data cluster je založen na technologii Docker v kombinaci s cluster management nástroji Docker Swarm a Kubernetes. Poskytuje konfigurační soubory pro nasazení jednotlivých služeb potřebných pro sestavení Big Data processing pipeline. Tyto konfigurační soubory umožňují nasazení služeb Apache Spark, Apache Ignite, Apache Cassandra, Apache Hadoop, Apache Kafka, Apache Zeppelin, Nginx.

Jako hlavní výpočetní framework je myšlen Apache Spark, nicméně je možné použít taktéž Apache Ignite. Apache Ignite je možno použít taktéž jako úložiště ve formě klíč-hodnota, či chache vrstva. Apache Cassandra je zde myšlena jako primární úložiště pro zpracovaná data. Apache Hadoop zde slouží jako distribuovaný souborový systém pro ukládání zdrojových dat. V neposlední řadě je zde Apache Kafka, která taktéž může sloužit jako vstupní prvek systému. Apache Zeppelin pak slouží jako interaktivní rozhraní pro přístup k jednotlivým službám.

Pro cluster management je možné zvolit buď Docker Swarm nebo Kubernetes, nikoli obě zároveň. Docker Swarm poskytuje jednoduché nasazení, Kubernetes poskytuje možnost nasazení většího množství služeb a zároveň lepší udržitelnosti.

Dokument je členěn následovně: Sekce 2 popisuje technologie, které jsou použity při tvorbě a nasazení na cluster. Sekce 3 pak popisuje zvolený návrh nasazení clusteru jak pro Docker Swarm tak Kubernetes. Sekce 4 popisuje výkonnostní měření, a výsledky, kterých bylo na daném clusteru ve zvolené konfiguraci dosáhnuto. Sekce 5 obsahuje závěrečné poznámky, a dokument je pak uzavřen seznamem s tématem souvisejících odkazů a bibliografických zdrojů.

2 Technologie

Tato sekce je věnována základnímu představení technologií, které jsou použity pro sestavení big data výpočetního clusteru. Jelikož je nutné porozumět nejen cluster management systému, ale taktéž jenotlivým nasazovaným službám, je zde věnována část i těmto technologiím.

Docker [6] je technologie, která přináší možnost izolovat nasazení samostatných aplikací. Tento koncept snižuje nadměrnou zátěž na systém, než v případě kdy jsou používány virtuální stroje. Docker izoluje pouze samostatnou aplikaci a základ souborového systému, nicméně využívá stejné jádro operačního systému, na kterém je spuštěn. Zároveň umožňuje vytvářet virtuální síťová rozhraní pro propojení kontejnerů.

Docker Swarm [7] je nativní cluster management nástroj pro Docker. Umožňuje propojení několika strojů s nainstalovaným a spuštěným docker daemonem do jednoho funkčního celku, clusteru. Nastavení a ovládání je velmi snadné. Jednotlivé kontejnery jsou samostatně nasazovány na spravované stroje. Tyto kontejnery lze sdružovat do služeb (stejný kontejner je několikrát replikován), případě do aplikací obsahujících více služeb tzv. stacků. Nasazení je konfigurováno pomocí souborů docker compose.

Kubernetes [13] je cluster management nástroj pro různé kontejnery. Zde se nemusíme omezovat pouze na Docker. Tento nástroj umožňuje cluster management pro několik strojů. Jeho nastavení je komplikovanější, ale výsledný běh systému je stabilnější. Zároveň přidává abstrakci nad samostatnými kontejnery ve formě obálky zvané Pod. Tento Pod pak obsahuje jeden či více kontejnerů a poskytuje konektivitu pro kontejnery, které jsou jeho součástí. Jedná se o základní jednotku Kubernetes. Kubernetes taktéž umožňuje sdružení těchto Podů do služeb. Nasazení je konfigurováno pomocí speciálních yaml souborů.

Apache Spark představuje rozšíření Map-Reduce modelu o podporu více efektivních výpočtů přímo v paměti. Každá aplikace napsaná pro Apache Spark obsahuje jeden tzv. driver process a jeden nebo více tzv. executor procesů. Driver proces se stará primárně o analýzu, distribuci a plánování částí výpočtu přes všechny executory. Executor je pak výpočetní jednotkou, která se stará o spouštění částí úloh, které ji driver proces přidělil. Následně pak oznamuje stav zpátky driver procesu. Apache Spark rozlišuje dva hlavní typy uzlů, jedná se o master uzel a worker uzel. Na každém worker uzlu je vždy spuštěn alespoň jeden executor proces.

Apache Spark poskytuje nízkoúrovňové rozhraní a strukturované rozhraní. Nízkoůrovňové rozhraní je reprezentováno primárně datovou strukturou zvanou Resilient Distributed Dataset (RDD) a operacemi nad touto strukturou. Tato struktura je základem pro ukládání všech datových objektů použitých při běhu aplikace. Strukturované rozhraní je pak reprezentováno datovou strukturou zvanou Data Frame (DF), kterou můžeme považovat za tabulku obsahující řádky se specifikovaným schématem. Obě dvě datové struktury jsou neměnné. Spark obsahuje operace nad každou z datových struktur zvané transformace a akce. Transformace jsou abstraktní operace nad těmito datovými strukturami, které spadají do kategorie lazy evaluation. Apache Spark vždy tvoří plán výpočtu, kdy

žádná transformace není provedena nad datovou strukturou dokud není zavolána nějaká akce. Takovouto akcí může být například operace `count`, jejíž návratovou hodnotou je počet záznamů v datové struktuře.

Apache Spark rozděluje data na menší části, které jsou rozptýleny po clusteru v průběhu výpočtu a jsou hlavním předmětem parallelismu ve Sparku. Tyto části se nazývají partition. Operace nad těmito částmi jsou vykonávány paralelně. Spark samotný je napsán v programovacím jazyce Scala, nicméně pro programování aplikací je taktéž možné použít další jazyky jako jsou Java, Python nebo R. Taktéž poskytuje rozhraní pro psaní akcelerovaných SQL dotazů [8].

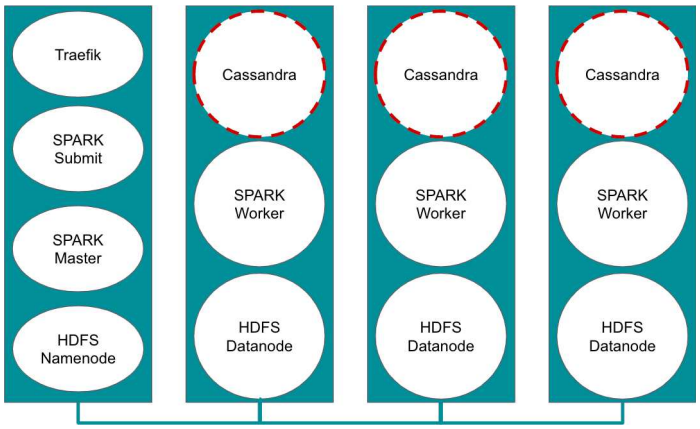
Apache Cassandra je distribuoavaná databáze zaměřená na ukládání velkých dat. Jedná se o NoSQL decentralizovanou databázi. Tato databáze poskytuje vysokou dostupnost, odolnost proti chybám s nastavitelnou konsistencí a škáluje horizontálně. [1] Cassandra je velmi často používána v řešeních pracujících s velkými daty jako uložiště, ve většině případů pak v kombinaci s Apache Spark.

Apache Ignite [3] je distribuovaný framework poskytující velké množství různých funkcionalit. Obecně se jedná o škálovatelné řešení odolné proti chybám, které může růst horizontálně přidáváním dalších uzlů. Ignite cache záznamy lze uložit persistentně v RDBMS a taktéž podporuje NoSQL podobně jako Apache Cassandra. Ignite framework je schopný poskytovat cache mezivrstvu pro různá úložiště jako jsou databáze či distribuovaný souborový systém HDFS. Taktéž je možné jej použít pro sdílení stavů v paměti mezi jednotlivými Spark úlohami, což samotný Spark neposkytuje. Navíc lze Ignite použít jako samostatný systém pro distribuované výpočty a zpracování nekonečných proudů dat. Taktéž poskytuje knihovnu pro strojové učení s velkým množstvím konfigurovatelných algoritmů [14].

Apache Hadoop [4] je první populární framework pro zpracování velkých dat, který pracuje na principu Map-Reduce. Tento framework byl již překonán výše zmíněným Sparkem. Nicméně Hadoop poskytuje distribuovaný souborový systém HDFS, který je stále používaný pro ukládání dat. Konektory pro Apache Spark pro zápis a získávání dat do a z HDFS jsou velmi dobře implementovány. Zároveň slouží jako základ pro další služby jako je například HBase[2].

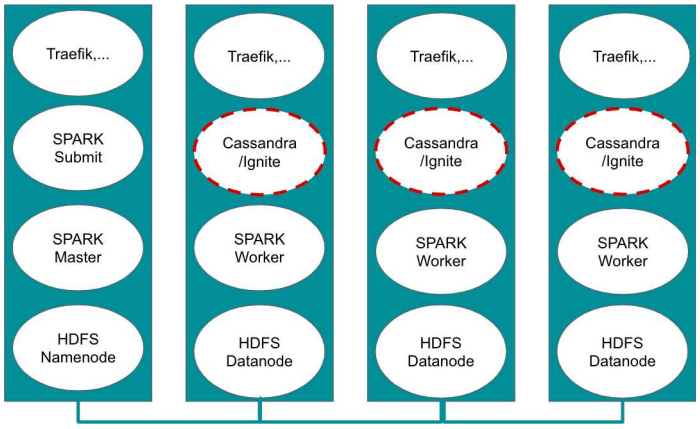
3 Návrh

Návrh rozvržení služeb a kontejnerů předpokládá použití clusteru o čtyřech uzlech. Toto není omezením, v případě použití více uzlů lze pouze zvýšit replikaci kontejnerů dané služby o hodnotu 1 při přidání každého dalšího uzlu do clusteru.



Obrázek 1: Logická struktura clusteru založeném na Docker Swarm.

V rámci návrhu byl zvolen jeden hlavní řídící uzel. Tento uzel je nastaven jako hlavní uzel pro většinu služeb. Taktéž slouží jako hlavní přístupový bod.



Obrázek 2: Logická struktura clusteru založeném na Kubernetes.

Návrh rozmístění kontejnerů jednotlivých služeb se mírně liší za použití Docker Swarm oproti Kubernetes. Toto rozložení je zobrazeno na obrázku 1.

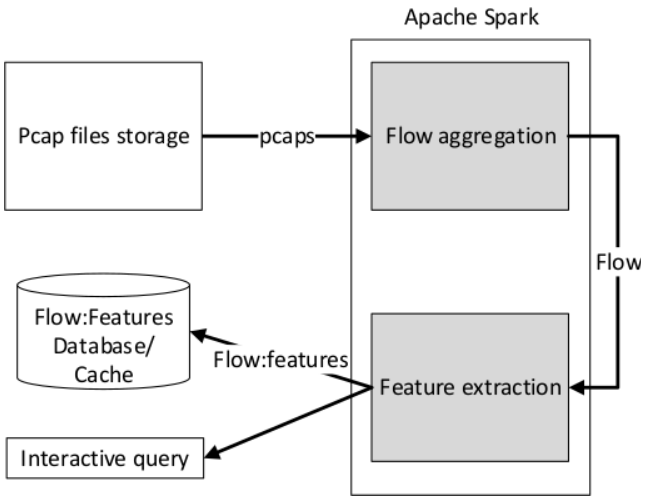
Návrh rozmístění kontejnerů pro jednotlivé služby pro Kubernetes je pak zobrazen na obrázku 2. V případě Kubernetes obsahuje každý uzel pomocné kontejnery rozmístěné na každém uzlu včetně reverzní proxy, v tomto případě je využívána proxy Traefik[5], zajišťující přístup ke službám mimo cluster. V případě Docker Swarm je spuštěna reverzní proxy Traefik pouze na jednom uzlu. Kubernetes oproti Docker Swarm může navíc obsahovat zprovozněnou službu s Apache Ignite.

Zbylé části služeb a kontejnerů jsou rozděleny identicky pro oba případy.

4 Měření výkonnosti

Pro celý nastavený systém bylo provedeno měření výkonnosti již spuštěných služeb na úloze zpracování 10GB pcap souborů, agregace paketů do flow, extrakce statistických hodnot z flow a následného uložení do databáze. Tato úloha by měla poskytnout dostatečné ověření funkčnosti a výkonnosti systému, jelikož zapojuje všechny části systému, a to jak úložiště různých typů (databáze, distribuovaný souborový systém), tak primární výpočetní část.

Platforma, která byla při nastavení clusteru výkonnostně laděna byl Supermicro SuperTwin26026TT-TF server osazený osmi Intel (R) Xeon E5520 @ 2.26GHz. Celý cluster čítá 4 uzly. Tři uzly jsou osazeny 48 GB RAM paměti a 16 CPU jádry. Čtvrtý uzel je pak osazen pouze 23 GB RAM a 16 CPU jádry. Všechny uzly mají nainstalován 1 TB SSD disk, který slouží jako uložiště pro data pro HDFS, Cassandru nebo Ignite, či další služby.

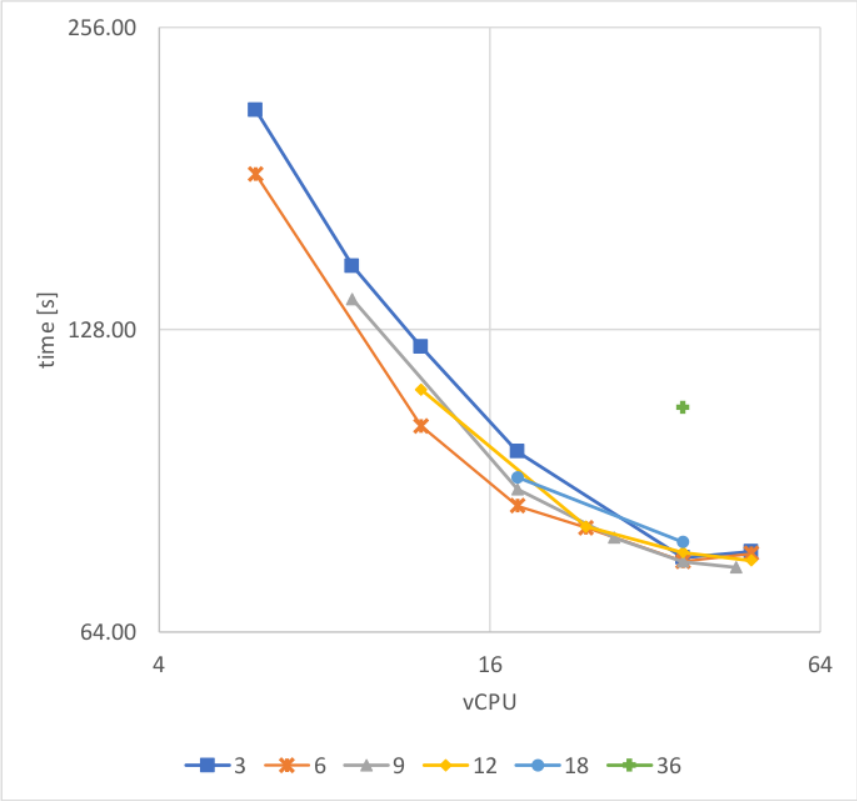


Obrázek 3: Logická struktura úlohy.[12]

Úloha spočívala ve zpracování 10 GB pcap souborů čítajících 27369774 paketů zachycených z honey-potu, uložených na HDFS. Tento zachycený provoz

byl rozdělen do do 231 souborů, každý obsahující maximálně 125000 rámců. Všechny tyto soubory byly menší než 128 MB, což je defaultní velikost bloku použita pro HDFS.

Tyto soubory byly následně zpracovány v prostředí Spark a zpracované informace z agregovaných toků byly uloženy do Cassandra či Ignite databáze. Logická struktura této úlohy je zachycena na obrázku 3. Měření výkonnosti bylo provedeno v několika kategoriích.



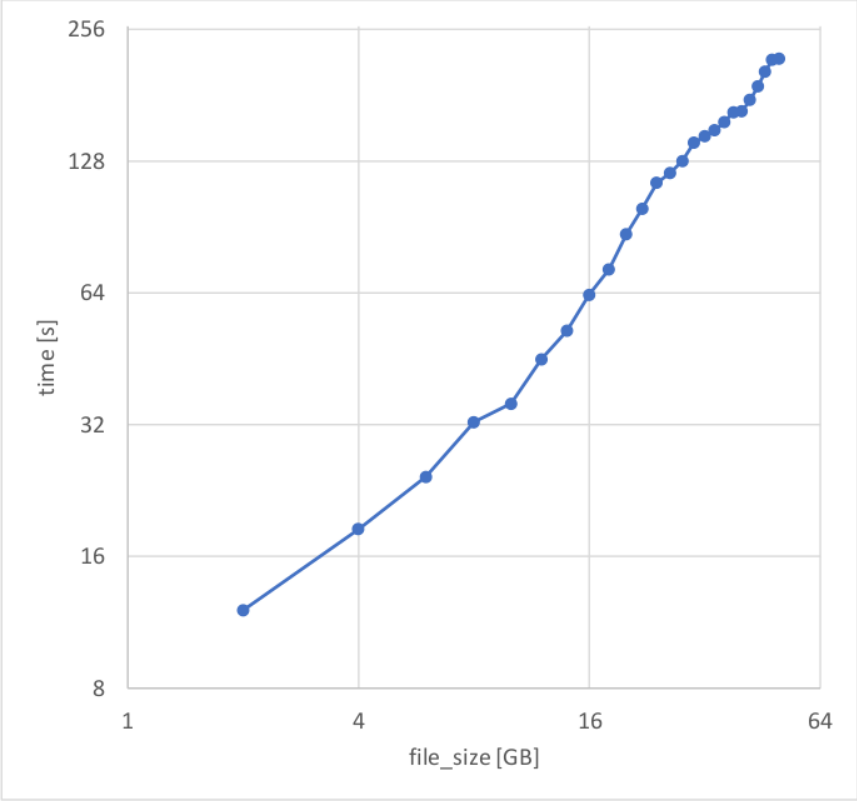
Obrázek 4: Měření škálovatelnosti úlohy v závislosti na počtu použitých jader.[12]

Nejprve byl laděn a testován systém běžící na Docker Swarm. V této části byla vyladěna úloha a zapisováno bylo pouze do databáze Cassandra. Zároveň byla měřena škálovatelnost celého systému.

Škálovatelnost systému v závislosti na různém počtu jader zapojených do zpracování úlohy v prostředí Apache Spark je zachyceno na obrázku 4. Zde je možné pozorovat, že při zvyšování počtu jader do určité hodnoty a nastavování poměru počet jader na executor dochází ke zrychlování výpočtu. Při zvýšení počtu jader nad určitou hranici nicméně začalo docházet ke zpomalování. Toto

bylo způsobeno především vytížením jader dalšími službami, které na clusteru byly spuštěny. Optimální hodnota pro počet jader byla 36 jader (6 jader připadá na každého ze 6 executionerů). Toto nastavení bylo využíváno i pro další měření.

Následně bylo taktéž testováno a měřeno jak systém škáluje při zpracování různé velikosti souborů. Kdy počáteční hodnota byla nastavena na 2 GB. A konečná hodnota na 50 GB, přičemž krok byl 2 GB. Na obrázku 5 můžeme vypozorovat lineární škálovatelnost. Tyto výsledky a podrobnější popis úlohy včetně naměřených hodnot je možné nalézt v článku [12].



Obrázek 5: Měření škálovatelnosti zpracování v závislosti na velikosti dat.[12]

Protože je celý cluster postaven na technologiích Docker Swarm nebo Kubernetes. Bylo vhodné otestovat zda-li, případně jaký je rozdíl v rychlosti zpracování stejné úlohy při stejném nastavení při správě clusteru Docker Swarm a Kubernetes. A tedy jak moc velké zatížení přidává jeden management vůči druhému.

Toto porovnání je zachyceno v tabulce 1. V tabulce je možné nalézt dvě Stage a následně jejich sumarizaci. Stage 0 odpovídá části úlohy, kdy jsou data načtena z HDFS do prostředí Spark a zároveň zpracována. Stage 1 odpovídá části úlohy,

Stage	Kubernetes Time [s]	Swarm Time [s]
0	29.32	26.54
1	49.12	48.73
sum	78.44	75.27

Tabulka 1: Výsledky měření stejné Spark úlohy běžící na Kubernetes a na Docker Swarm.

kdy jsou data ukládána do Cassandra databáze. Toto rozdělení je z pohledu prostředí Spark, kde je možné tyto jednotlivé části změřit. Čas zpracování celé úlohy je pak možné najít pod sum.

Z tabulky 1 je možné vypozorovat že zpracování úlohy při použití cluster management nástroje Kubernetes je o 3 sekundy pomalejší než při použití Docker Swarm. Nicméně na druhou stranu, jak se ukázalo, běh služeb při použití Kubernetes je mnohem více stabilnější. Tento časový rozdíl můžeme tedy považovat za stále rozumný a zanedbatelný. Kubernetes využívá více pomocných struktur a kontejnerů pro správný běh managementu což má za následek větší zátěž na celý systém.

Jelikož byla jako potenciální alternativa pro ukládání dat identifikováno uložiště klíč-hodnota v Ignite, byla otestována výkonnost ukládání do této databáze. Zde je možnost nastavovat velké množství parametrů, avšak největší vliv má povolení či zakázání ukládání dat na disk místo čistě do paměti. Pro měření byla použita stejná úloha, pouze se změnou výsledného úložiště. Toto bylo testováno pouze za použití Kubernetes, jelikož při použití Docker Swarm nebylo možné sestavit celý Ignite cluster (propojit všechny uzly). V obou případech byla nastavena pro částečnou zálohu cache na hodnotu 0, tedy žádná záloha.

Stage	Persistence disabled Time [s]	Persistence enabled Time [s]
0	28.54	30.82
1	39.66	76.30
sum	68.20	107.12

Tabulka 2: Výsledky měření zpracování a zápisu času do Ignite klíč-hodnota uložiště se zapnutou a vypnutou persistencí zápisu.

V tabulce 2 je možné pozorovat časový rozdíl v době výpočtu celé úlohy. Rozdíl je především v stage 1, spojené se zápisem dat do databáze. Zde je možné pozorovat značný rozdíl mezi povolenou a nepovolenou persistencí uložiště. Při porovnání mezi dobou běhu úlohy, která zapisovala data do Apache Cassandra (78.44 sekund) a Apache Ignite s povolenou persistencí a zálohou cache 0 (107.12 sekund). Byla zvolena Apache Cassandra jako optimální řešení.

5 Závěr

Technická zpráva podrobně dokumentuje použité technologie, návrh a výkonnostní měření Big Data Výpočetního Clusteru. Tato konfigurace byla vyvinuta v rámci projektu TARZAN [9].

Tato technická zpráva navazuje na naše další publikace:

- Instalační příručku [10];
- Uživatelský a programátorský manuál [11];

Odkazy

[1] Apache Software Foundation, ed. *Apache Cassandra*. 2016. URL: https://cassandra.apache.org/ (cit. 28. 04. 2017).

[2] Apache Software Foundation, ed. *Apache HBase*. 2019. URL: https://hbase.apache.org/ (cit. 20. 12. 2019).

[3] Apache Software Foundation, ed. *Apache Ignite*. 2019. URL: https://ignite.apache.org/ (cit. 20. 12. 2019).

[4] Apache Software Foundation, ed. *Welcome to Apache Hadoop!* 2014. URL: https://hadoop.apache.org/ (cit. 28. 04. 2017).

[5] Containous, ed. *Traefik*. 2018. URL: https://traefik.io/ (cit. 05. 01. 2019).

[6] Docker, ed. *Docker*. 2019. URL: https://www.docker.com/ (cit. 20. 12. 2019).

[7] Docker, ed. *Docker Swarm Overview*. 2019. URL: https://docs.docker.com/engine/swarm/ (cit. 20. 12. 2019).

[8] Bill Chambers a Matei Zaharia. *Spark: the definitive guide: big data processing made simple*. "O'Reilly Media, Inc.", 2018.

[9] *Integrovaná platforma pro zpracování digitálních dat z bezpečnostních incidentů*. https://www.fit.vut.cz/research/project/1063/. navstiveno: 2019-09-10.

[10] Kamil Jeřábek. *Big Data Výpočetní Cluster: Instalační přírucka*. Tech. zpr. 2019.

[11] Kamil Jeřábek. *Big Data Výpočetní Cluster: Uživatelský a programátorský manuál*. Tech. zpr. 2019.

[12] Kamil Jeřábek a Ondřej Ryšavý. „Big Data Network Flow Processing Using Apache Spark“. In: *Proceedings of the 6th Conference on the Engineering of Computer Based Systems*. ACM. 2019, s. 9.

[13] The Linux Foundation, ed. *Kubernetes*. 2019. URL: https://kubernetes.io/ (cit. 20. 12. 2019).

[14] Michael Zheludkov, Timur Isachenko et al. *High Performance in-memory computing with Apache Ignite*. Lulu. com, 2017.