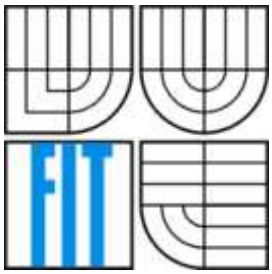
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DISTRIBUOVANÉ ZPRACOVÁNÍ DAT O IP TOCÍCH

DISTRIBUTED PROCESSING OF IP FLOW DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Pavel Krobot

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Martin Žádník, Ph.D.

BRNO 2015

Abstrakt

Tato práce se zabývá distribuovaným zpracování dat o IP tocích. Konkrétně je pak hlavním cílem poskytnutí řešení softwarového kolektoru, který bude umožňovat zpracování a ukládání masivního objemu dat. V rámci této práce je zkoumána volně dostupná implementace rámce pro distribuované ukládání a výpočty nad daty Hadoop, která využívá modelu MapReduce. Nad tímto systémem byly následně provedeny experimenty, jejichž smyslem bylo získat představu o výkonnosti tohoto řešení oproti řešením stávajícím a odhalit slabiny systému. Na základě získaných poznatků byla pak vytvořena specifikace a návrh rozšíření stávajícího softwarového kolektoru. Dle vytvořeného návrhu následně vznikla implementace dotazovací části navrhovaného kolektoru, která se při distribuovaném zpracování dat o IP tocích jeví jako nejvíce kritická. Výsledky experimentů s touto implementací ukázaly výrazné zvýšení výkonu při dotazování a schopnost lineární škálovatelnosti na některých typech dotazů.

Abstract

This thesis deals with the subject of distributed processing of IP flow. Main goal is to provide an implementation of a software collector which allows storing and processing huge amount of a network data in particular. There was studied an open-source implementation of a framework for the distributed processing of large data sets called Hadoop, which is based on MapReduce paradigm. There were made some experiments with this system which provided the comparison with the current systems and shown weaknesses of this framework. Based on this knowledge there was created a specification and scheme for an extension of current software collector within this work. In terms of the created scheme there was created an implementation of query framework for formed collector, which is considered as most critical in the field of distributed processing of IP flow data. Results of experiments with created implementation show significant performance growth and ability of linear scalability with some types of queries.

Klíčová slova

Distribuce, výpočet, úložiště, databáze, MapReduce, Hadoop, Nfdump, IPFIX.

Keywords

Distribution, computation, storage, database, MapReduce, Hadoop, Nfdump, IPFIX.

Citace

Pavel Krobot: Distribuované zpracování dat o IP tocích, diplomová práce, Brno, FIT VUT v Brně, 2015.

Distribuované zpracování dat o IP tocích

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením Ing. Martina Žádníka, Ph.D. a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Pavel Krobot
1. 6. 2015

Poděkování

Rád bych velmi poděkoval zejména Ing. Martinovi Žádníkovi, Ph.D. za odbornou pomoc a vynikající spolupráci při vytváření celé této diplomové práce. Dále bych chtěl poděkovat Ing. Tomáši Čejkovi, Ing. Václavu Bartošovi a Ing. Tomáši Podérmanskému za odbornou a vstřícnou pomoc v implementační fázi této práce, která využívá pomocné nástroje a knihovny, vytvořené právě těmito pány. V neposlední řadě bych chtěl poděkovat také virtuální organizaci MetaCentrum za poskytnutí přístupu k úložištím a výpočetním zdrojům, vlastněným subjekty přispívajícími do Národní gridové infrastruktury MetaCentrum, poskytovaným v rámci programu „Velká infrastruktura CESNET“ (LM2010005), dále pak Ing. Františku Dvořákovi za odbornou asistenci a rychlé řešení problémů při prováděných experimentech. Závěrem tohoto poděkování bych chtěl vyjádřit svůj dík partnerce Janě Večerkové, mým rodičům a celé rodině za podporu, kterou mi při práci na tomto díle poskytly.

Obsah

1 Úvod

Počítačové sítě jsou dnes hojně využívány v různých oblastech našeho každodenního života. Jejich uplatnění nacházíme v různorodých případech využití, zahrnujících obecně přístup k informacím, zábavu, vzdělání, ale i například zaměstnání, bezpečnost či zdravotnictví. Do počítačové sítě, a zejména pak do Internetu, se dnes připojuje stále více a více zařízení. To způsobuje rychlý nárůst dat přenášených počítačovou sítí, která je potřeba pro účely zajištění funkcionality, dostupnosti, kvality a bezpečnosti poskytovaných služeb sbírat, uchovávat a analyzovat. Množství těchto dat, zvláště pak u velkých operátorů a provozovatelů služeb již dnes přesahuje možnosti poskytovaných komerčních řešení. Tato řešení pak musí při zpracování dat přijmout jisté kompromisy a ústupky, které mohou znamenat nižší přesnost a spolehlivost, způsobenou vzorkováním dat na síti, omezení informací o infrastruktuře, distribuci spolu souvisejících dat na více systémů, které ale nejsou vzájemně provázané apod.

Kromě klasických zařízení (stolní počítače, notebooky, atd.), jsou dnes připojovány do Internetu i další různorodá zařízení, která s sebou přinášejí nové možnosti nebo poskytují další komfort a kvalitu poskytovaných služeb. Tato zařízení jsou souhrnně označována jako „Internet of Things“ či „Internet of Everything“. Aktuální trend, kdy těchto zařízení stále přibývá, se také významně podílí na nárůstu provozu, jenž je přenášen počítačovou síti a na jeho různorodosti. Navíc tato nově připojovaná zařízení disponují pouze omezeným výkonem, který je ve většině případů vyhrazen pouze aplikaci, nikoliv její ochraně.

Dalším trendem je centralizace služeb do datových center či cloudů a vznik nových služeb, založených na technologiích, které dříve nebyly dostupné. Centralizace a zavedení těchto nových služeb vede opět na zvýšení množství dat, přenášených síťovou infrastrukturou. Klíčovými faktory pro poskytovatele těchto služeb jsou pak bezpečnost, spolehlivost a kvalita poskytované služby. Její výpadek či napadení útočníkem může totiž znamenat jak obrovské ekonomické ztráty, tak i narušení či ohrožení některé základní služby poskytované státem, které může mít dopad jednak na majetek, jednak i na zdraví lidí.

Dalším významným aspektem této oblasti je kyberkriminalita. Ta se projevuje v podobě velkého množství síťových útoků, využívajících vlastnosti či zranitelnosti aplikací, dostupných prostřednictvím sítě, nedostatky v síťových protokolech apod. Útoky a hrozby se stávají mohutnějšími[1] nebo sofistikovanějšími a právě touto intenzitou či vlastnostmi zůstávají mimo detekční schopnosti současných metod.

Uvedené trendy ukazují, že vzrůstá potřeba škálovatelného řešení, jež bude schopno sbírat, uchovávat a analyzovat obrovské množství dat o síťovém provozu. Toto řešení musí umožnit hloubkovou analýzu nasbíraných dat s cílem odhalení problémů daných služeb a síťové infrastruktury. Důležitým parametrem je také rychlost systému a s ním spojená krátká doba odezvy na zadané dotazy, jenž umožní provozovateli služeb či bezpečnostnímu týmu rychle lokalizovat problém a následně na něj včas a vhodným způsobem reagovat. Důležité je zde také zohlednit nedostatky současných řešení. Významným nedostatkem je vysoká pořizovací cena hardware, na kterém jsou současné systémy pro sběr a analýzu dat provozovány. Tento nedostatek znamená velkou počáteční investici a vysoké náklady na pořízení nových zařízení v závislosti na růstu infrastruktury a jejich potřeb. Kromě pořizovacích nákladů jsou zde také méně viditelné náklady spojené s provozem těchto složitých zařízení, která navíc kladou vysoké požadavky na čas a znalosti uživatele. Ke správnému

[1] Na začátku minulého roku byly reportovány DDoS s rychlostí přenosu dat převyšující 300Gb/s.

a plnohodnotnému použití je potřeba systém řádně nastavit pro konkrétní síť a získané výsledky vhodným a správným způsobem interpretovat. To vyžaduje velmi zkušené správce, kteří jsou ale obvykle již přetíženi dalšími činnostmi.

Cílem této práce je nalézt řešení pro tento problém, které navazuje na již existující principy. Tím je myšlena architektura zahrnující množství sond, jež shromažďují data o provozu na síti. Tato data následně odesílají na kolektor, který je sbírá, spojuje dohromady a následně ukládá pro další zpracování. Právě tato část, kdy je potřeba efektivně manipulovat s masivními objemy dat jak z hlediska rychlého uložení, tak z pohledu rychlého přístupu a práce s daty, je předmětem této práce. Cílem je tedy vyvinout kolektor s masivním výkonem, který bude zmíněné požadavky naplňovat. Navržený kolektor pak bude sloužit jako vhodný základ pro zpracování a analýzu dat o datových tocích sbíraných z vysokorychlostních sítí, a to navíc na dynamické úrovni detailu. Jak již bylo naznačeno, úroveň detailu sbíraných dat je totiž dalším bodem, který mnohdy nevyhovuje povaze síťového provozu, s jakým se dnes setkáváme. Často jsou dnes ukládána a následně analyzována data s pevnou strukturou. Některé metody však pro dosažení lepších a přesnějších výsledků potřebují zdrojová data popsat podrobněji, přičemž jiné přístupy zase tato data nepotřebují a jejich ukládání by pak znamenalo plýtvání prostoru datového úložiště. Příkladem zde může být požadavek na údaje z aplikačního protokolu IP telefonní komunikace, umožňující přesnější a spolehlivější detekci podvodných volání, u protokolu DNS mohou další informace z vyšších vrstev pomoci k přesnější identifikaci botnetů apod.

V rámci naplnění cíle této práce budou zkoumány a vyvíjeny inovativní algoritmy pro sběr a uchování velkého množství dat o IP tocích, které umožní následnou analýzu nad těmito daty. Vyvinuté algoritmy a řešení pak budou umožňovat zpracování extrémního množství sbíraných dat o síťovém provozu na různé úrovni detailu pomocí různých protokolů, pokládání dotazů nad těmito daty s odezvou v reálném čase či dolování dat se zaměřením na problémy a hrozby směřující na aplikace a uživatele. Těchto vlastností a parametrů bude dosaženo díky využití masivní paralelizace a virtualizace vyvinutého programového vybavení. Bude zde také uplatněn koncept zpracování velkých objemů dat[2] s využitím prostředí datových center a cloudů. Tento přístup pak může umožnit potencionálním uživatelům využít výpočetní a kapacitní prostředky v jejich datových centrech nebo využít systém v podobě cloudové služby. Problém s pořízením hardware je pak touto cestou přesunut na provozovatele centra či cloudové služby.

Součástí dosažení cíle této práce je také testování vyvíjeného řešení na reálné přenosové síti velkého rozsahu CESNET2. Na této síti jsou provozovány kritické služby, jako je přenos multimediálních dat (např. online operace pacientů), poskytování datových úložišť, IP telefonie, roaming uživatelů (tj. připojení uživatele do sítě v navštívené organizaci) a další. Testování navrhovaného řešení v prostředí reálných sítí umožní ověřit vlastnosti a chování daného řešení v produkčním prostředí a poskytne tak zpětnou vazbu pro ladění a optimalizaci výsledného řešení.

V následující kapitole bude shrnut současný stav poznání, spolu s teoretickým jádrem této práce. Ve třetí kapitole bude uvedena specifikace cíleného řešení včetně předpokládaných případů užití, následovaná popisem experimentů se systémem pro distribuované zpracování dat Hadoop dalších platforem na tomto systému založených. Poté bude v další kapitole uveden návrh vyvíjeného kolektoru. Za návrhem řešení následuje popis implementace nejkritičtější části vyvíjeného kolektoru, dotazovacího frameworku, který vznikl jako součást této práce. V předposlední kapitole jsou uvedeny výsledky experimentů s implementovaným programem v distribuovaném prostředí. Poslední kapitola uzavře tuto práci shrnutím zjištěných poznatků a zhodnocením dosažených výsledků.

[2] tzv. koncept Big Data

2 Současný stav poznání

Na národní i mezinárodní úrovni využívají organizace různých řešení pro sběr dat ze sítě, jejich uložení a následnou analýzu. Typické řešení se skládá ze dvou částí, kdy jedna část systému řeší sběr, uchování a přístup k datům, druhá pak zajišťuje samotnou analýzu a další práci s daty. Jak již bylo naznačeno, v rámci této práce budu vycházet právě z tohoto paradigmatu, přičemž hlavní důraz bude kladem na část systému, která zajišťuje sběr, uchování a manipulaci s daty (kolektor). Rozšíření některého stávajícího kolektoru, jež je předmětem této práce, bude umožňovat masivně paralelní zpracování velkého objemu dat ze síťového provozu na dynamické úrovni detailu, analýzu těchto dat a dolování informací z nich. Primárním protokolem pro přenos dat o tocích bude v rámci této práce protokol IPFIX, jehož specifikace je uvedena v [1].

Současné kolektory jsou dostupné buď jako open-source produkty (NFDUMP, VERMONT, yaf, IPFIXcol) nebo jako systémy komerční (StealthWatch, FlowCollector, Plixer Scrutinizer, HP Netflow Insight Web). Důležitými vlastnostmi kolektorů jsou jejich výkon při sběru dat, při dotazování nad těmito uloženými daty a rozšiřitelnost samotného kolektoru. Současné systémy nevyužívají masivní paralelizaci pro dosažení větší propustnosti ukládání dat a urychlení přístupu k uloženým datům. Dolování informací z uložených dat tak u těchto kolektorů trvá řádově hodiny i dny. Stávající systémy řeší tento problém spouštěním zadaných dotazů na pozadí a podporují různé způsoby notifikace uživatele o dokončení těchto dotazů. Tento způsob obsluhy dotazů je však pro interaktivní práci s daty nepoužitelný. V rámci této práce bude pro dosažení větší propustnosti dat využito distribuce úloh na více zařízení, kdy pak jednotlivé relativně pomalé propustnosti k diskům poskytnou jako celek výrazně větší propustnost dat. Další nevýhodou současných řešení je podpora jednotné úrovně detailu a typu IP toků. Jak již ale bylo naznačeno v úvodu, moderní přístupy pro analýzu provozu na síti vyžadují data v různých úrovních detailu tak, aby bylo dosaženo sběru relevantních dat pro řešení potencionálních problémů sítě, které se mohou svojí povahou značně odlišovat.

Stávající řešení, jež nedovolují práci se všemi daty ze sítě (vzorkování apod.), omezují následnou analýzu těchto dat, ať už se jedná o statickou analýzu, strojové učení či třeba rozpoznávání vzorů chování. Výsledky dnešních systému pro analýzu těchto dat trpí zejména následujícími nedostatky:

- produkují falešně pozitivní hlášení a
- umožňují analyzovat pouze omezené množství dat do určité hloubky detailu.

To vede k neschopnosti odhalit problémy, jako jsou například výkonnostní problémy aplikací a služeb či pokročilé a přetrvávající hrozby[3]. Mimo tyto hlavní nedostatky jsou dalšími nevýhodami stávajících systémů nízký výkon a omezená škálovatelnost, bránící ve zpracování dat z větších sítí nebo poskytnutí sdíleného řešení pro více zákazníků zároveň (tzv. multitenance). Aby bylo možné tyto nedostatky překonat a odstranit, je nutné zkoumat a vyvíjet nové přístupy, umožňující komplexní analýzu sbíraných dat. Tyto nové přístupy musí umožňovat škálovatelnost a masivní paralelizaci. Díky tomu bude možné odhalovat problémy rychleji, přesněji a v neposlední řadě bude také možné detekovat dosud skryté hrozby. K tomu také přispěje možnost analýzy datových toků na aplikační úrovni a korelace událostí v historii, což dále podporuje výzkum a vývoj metod, dovolující uchovávat informace o zachycených událostech a následné vytváření databáze znalostí o reputaci entit či oblastí. Tato znalost může dále poskytovat zpětnou vazbu sondám, které pak mohou o problémových entitách

[3] tzv. Advanced Persistent Threat

sbírat specifičtější informace ze sítě. Tuto historii událostí by mohly také využívat metody zaměřující se na hlubší analýzu a korelaci s analyzovanými daty.

Zpracování velkého objemu síťových dat však vyžaduje nové přístupy, jelikož ty stávající již dosáhly hranice svých možností. Zejména nelze pro tyto účely použít řešení založená na relačních databázích a SQL, hlavně kvůli tomu, že zpracovávaná data mají být dynamická, tj. nemají pevnou a předem stanovenou strukturu. Pro interaktivní práci s daty je nutné se zaměřit na mechanismy, jež umožní data zpracovávat dostatečně efektivně, v závislosti na jejich aktuální velikosti a dostupných výpočetních a úložných zdrojích. Je tedy potřeba hledat řešení, která dokážou reagovat a přizpůsobovat se dostupným kapacitám a dynamicky navyšovat a snižovat jejich využití podle potřeb uživatele. Podobně je potřeba hledat nové přístupy pro dolování dat o velkých objemech. Slibnou cestou jsou metody, které využívají paralelizace. Efektivní využití těchto metod ovšem vyžaduje velkou změnu v náhledu na zpracování síťových dat, protože se velmi liší od tradičně používaných metod. Jako jeden z použitelných přístupů se jeví model „MapReduce“, který bude podrobněji popsán v následující podkapitole. Tato práce pak vychází z poznatků a snaží se navázat na výzkum popsaný v [2][3][4].

2.1 Model MapReduce

Následující popis modelu MapReduce vychází z [5] a [6]. MapReduce je programovací model představený společností Google v roce 2004. Inspirací pro vznik tohoto paradigmatu byly funkce ve funkcionálních jazycích. Dnes se jedná o nejrozšířenější model v oblasti paralelního zpracování dat v řádech terabytů až petabytů. Kromě společnosti Google jej pro zpracování masivního množství dat využívají i například Yahoo!, Amazon či Facebook. Pojem MapReduce zahrnuje i mimo samotné paradigma s ním spojenou implementaci. Kromě proprietální implementace společnosti Google existují i volně dostupné. Jednou z nejrozšířenějších implementací, využívajících modelu MapReduce je Hadoop, který je mj. také předmětem pozornosti této práce a bude blíže popsán v následující kapitole. Model MapReduce kombinuje klasické sekvenční výpočty spolu s paralelním zpracováním. Jednotlivé *Map* či *Reduce* úlohy mohou být prováděny souběžně a umožňují tak využití celého výpočetního clusteru namísto jediné stanice.

Obecně lze model MapReduce rozdělit na několik fází, z nichž hlavní dvě jsou *Map* a *Reduce*:

- *Vstup a rozdělení*: Na začátku celého výpočtu jsou načtena vstupní data a následně jsou rozdělena mezi jednotlivé uzly, provádějící funkci *Map*. Vstupní data jsou v této fázi transformována do dvojic (klíč; hodnota), v této podobě jsou totiž očekávány funkcí *Map*.
- *Map*: Tato fáze zahrnuje paralelní provedení funkce *Map* jednotlivými výpočetními uzly. Samotná funkce *Map* přijímá jako vstup jeden pár (klíč; hodnota), který zpracuje a následně vrátí seznam párů (klíč; hodnota). Formálně lze zapsat funkci *Map* takto:

$$Map: (k_1, h_1) \rightarrow seznam(k_2, h_2) \quad (1)$$

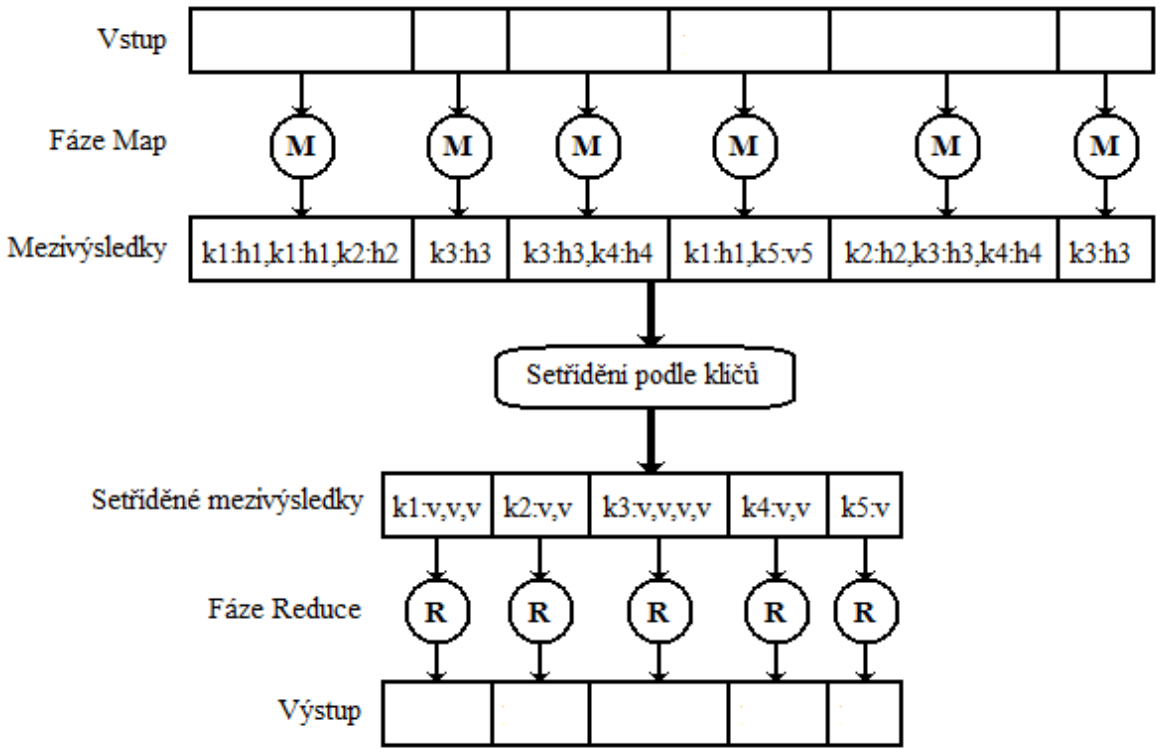
kde k_1, k_2 jsou klíče a h_1, h_2 jsou hodnoty. Hodnota klíče vstupních dat k_1 určuje rozdělení mezi jednotlivé uzly. Vlastní data pro zpracování jsou pak reprezentována hodnotou h_1. Hodnota klíče k_2 se již v průběhu následujících fází nemění a je tedy hodnotou výstupního klíče celého procesu. Data h_2 jsou mezistupněm výsledku zpracování dat a jsou získána aplikací funkce *Map* na jednotlivé vstupy h_1. Důležitým rysem funkce *Map* je to, že pracuje na svém vstupu pouze s jediným párem. Tím je zajištěna nezávislost vstupů a umožněna distribuce výpočtů na různé výpočetní uzly, které následně mohou tyto výpočty provést paralelně.

- *Setřídění a rozdělení*: Jednotlivé výstupy předcházející fáze jsou setříděny a podle klíčů rozděleny na výpočetní uzly pro fázi *Reduce* tak, že jsou vždy jednomu uzlu přiřazena všechna data týkající se jednoho klíče. Tato fáze je prováděna automaticky systémem implementujícím MapReduce model a nezávisí tedy přímo na programátorovi.
- *Reduce*: Ve fázi *Reduce* je na každý seznam hodnot pro daný klíč aplikována funkce *Reduce*. Ta z tohoto vstupního seznamu vytvoří seznam výstupních párů, přičemž samotný klíč hodnot nemění. Zde se projevuje sekvenční aspekt výpočtu, kdy je nutné s touto fází vyčkat až do provedení celé sekce *Map*. Formální zápis funkce *Reduce* lze zapsat následovně:

$$Reduce: \left(k_2, seznam(h_2)\right) \rightarrow seznam(k_2, h_3) \quad (2)$$

  kde $k_2$je klíč, který byl použit na výstupu funkce *Map* a podle kterého byly výsledky této funkce setříděny, seznam hodnot h_2 je seznam výsledků funkce *Map* pro odpovídající zdrojové hodnoty. Na výstupu *Reduce* je pak seznam párů s výstupními hodnotami h_3, typicky se však jedná o jeden pár, představující jedinou hodnotu pro daný klíč.
- *Výstup*: Poslední část výpočtu už jen sesbírá jednotlivé výsledky a předá je na výstup jakožto výsledek celé operace MapReduce.

Obrázek 2-1 znázorňuje průběh celého výpočtu MapReduce. Z pohledu programátora je z celého výpočtu nutno řešit fázi vstupu dat a následně dodat implementace funkcí *Map* a *Reduce*, o zbytek se stará již samotný subsystém.



Obrázek 2-1 Průběh výpočtu MapReduce modelu. Vytvořeno na základě originálu v [7].

.

3 Specifikace kolektoru a experimenty

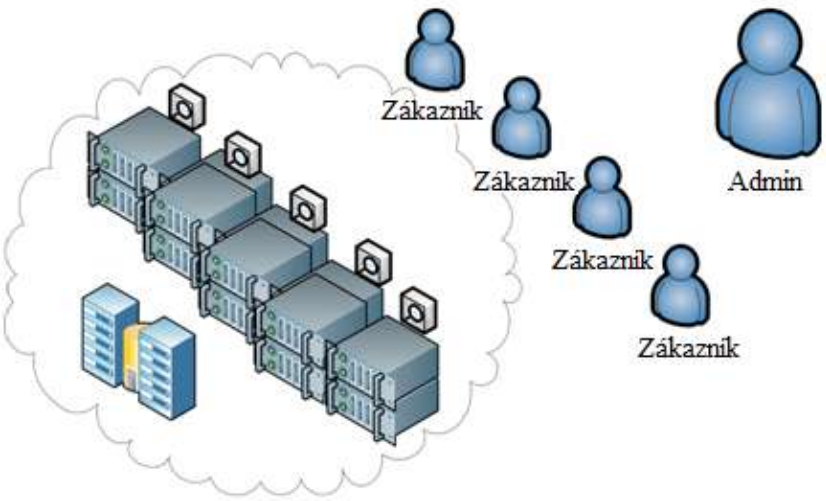
Tato kapitola obsahuje předpokládané případy užití vyvíjeného kolektoru, na základě kterých byla následně vytvořena detailní specifikace jednotlivých částí. Vyvíjený kolektor bude realizovat příjem a zpracování extrémního množství dat ze síťového provozu na různé úrovni detailu. Dále bude realizovat dotazy nad velkým množství dat s odezvou ideálně v reálném čase a bude umožňovat vysoce výpočetně náročnou analýzu těchto dat v oblasti bezpečnosti. Jednotlivé případy užití budou popsány v následující podkapitole, následovány zmíněnou specifikací.

Ve třetí podkapitole jsou uvedeny výsledky experimentů s existujícími platformami pro distribuované zpracování dat. Smyslem těchto experimentů bylo zjistit vhodnost těchto již implementovaných řešení pro potřeby distribuovaného kolektoru a dále také odhalit případné nedostatky a slabá místa, na která se bude nutné zaměřit, pokud by byla některá z těchto platforem zvolena pro realizaci vyvíjeného kolektoru. Součástí těchto experimentů je také srovnání s prováděním dotazů nástrojem Nfdump, který poskytuje velmi dobrou odezvu při zpracovávání menšího množství dat o IP tocích na jednom stroji.

3.1 Případy užití

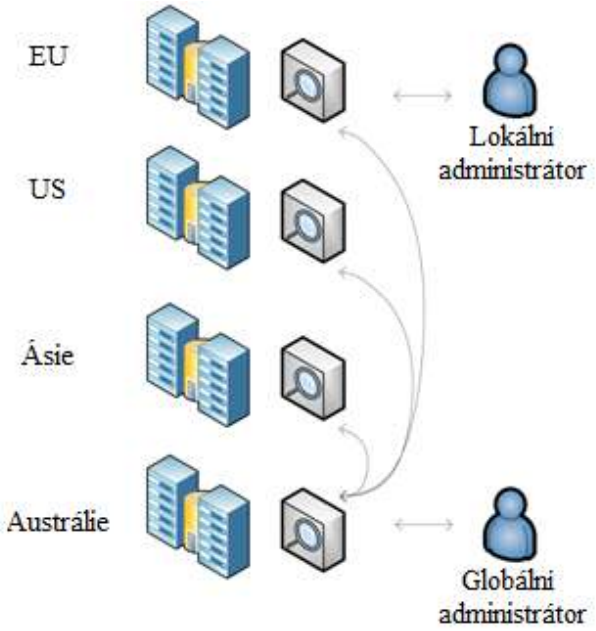
V této sekci jsou popsány dva očekávané případy užití vyvíjeného kolektoru. Z těchto případů užití vychází specifikace a následně i návrh tohoto kolektoru.

Prvním případem užití je situace, kdy poskytovatel služeb, jako jsou např. infrastruktura jako služba (IaaS), platforma jako služba (PaaS) aj., má zájem poskytnout náhled na data ze sítě svým zákazníkům. Každému zákazníkovi však poskytovatel umožní náhled pouze na ta data, která přísluší danému zákazníkovi. Současně správce tohoto poskytovatele vidí data od všech zákazníků. Předpokladem u tohoto případu užití je provozování vlastního datového centra poskytovatelem a také že tento poskytovatel má dostatek svých výpočetních prostředků. Samotný kolektor pak bude provozován právě na infrastruktuře datového centra.



Obrázek 3-1 Případ užití: poskytovatel služby umožňuje zákazníkům přístup k datům při vzájemně striktním oddělení dat.

Druhý případ užití bere v potaz nadnárodní organizaci. Tato organizace může provozovat několik datových center, jež jsou distribuovány v různých zemích. Z každého datového centra jsou data ukládána na lokální kolektor, provozovaný v dané lokalitě, přičemž se nemusí jednat o vyhrazený server (kolektor může být provozován na infrastruktuře datového centra). Existují zde dvě úrovně správců. Na lokální úrovni mají správci přístup pouze k lokálním datům. Globální správci pak mají přístup ke všem datům ze všech lokalit (tj. na všech kolektorech) přes jednotné rozhraní. V tomto případě bude v rámci této práce řešena pouze lokální část kolektoru.



Obrázek 3-2 Případ užití: nadnárodní organizace, využívající lokálního a globálního přístupu k datům.

3.2 Specifikace požadavků

Specifikace požadavků vychází z případů užití a poznatků, uvedených v předcházejícím textu. V úvodu této podsekce jsou shrnuty základní požadavky na vyvíjený kolektor. Dále je pak specifikace požadavků rozdělena do několika podkapitol podle jednotlivých úkonů kolektoru. Na začátku každé této podsekce je vždy nejprve ve formě tabulky uveden souhrn požadovaných vlastností a funkcionalit, vztahujících se k dané části. V těchto tabulkách je uvedena míra splnění požadavků ve třech úrovních – minimální, dostatečná a ideální. Splnění minimálního požadavku je nezbytné pro fungování výsledného řešení. Předpokladem je však dosažení dostatečné úrovně splnění požadavků. Stupeň ideální pak označuje nadprůměrně dobrý výsledek. Jednotlivé parametry, které jsou v tabulkách uvedeny, budou následně v rámci dané sekce komentovány podrobněji.

Následující seznam shrnuje základní obecné požadavky, kladené na vyvíjený kolektor:

- Podpora masivní škálovatelnosti, přidávání zdrojů (CPU, RAM, HDD) vhodným způsobem (přidávání virtuálních strojů, možnost rozšiřovat zdroje stávajících virtuálních strojů).
- Absence úzkého místa, zajištěná rozkládáním zátěže mezi více strojů, a to na všech úrovních.
- Multitenance, tj. obsluha více zákazníků z jedné instance kolektoru při striktním oddělení jednotlivých zákazníků od sebe.
- Poskytnutí možnosti adresovat a bezpečně odstranit data konkrétního zákazníka.
- Flexibilita zpracovávaných dat.

- Podpora pro rozšíření typu plug-in, které budou přistupovat k datům. Tato rozšíření se mohou obecně lišit podle zákazníka.

3.2.1 Příjem dat

První fází zpracování dat ze sítě je jejich přijetí. Hlavním vstupem těchto dat do kolektoru jsou záznamy o IP tocích, jež jsou přijímány ze zařízení schopných zachycená data exportovat. Pro zpracování vstupu kolektoru je zde předpoklad několika vyhrazených uzlů, přičemž jednotlivé zdroje dat budou rozděleny do disjunktních podmnožin a každý z vyhrazených uzlů bude přijímat a zpracovávat data právě z jedné této podmnožiny. Na vstupu je potřeba zpracovávat různé protokoly, dle jejichž specifikací jsou data přenášena. Výsledkem zpracování těchto protokolů je vnitřní reprezentace záznamu o toku, tj. vstupní data jsou do jisté míry normalizována. Nad vstupními záznamy by také mělo být umožněno provádění některých základních operací, jako je například vzorkování dat (např. při zahlcení). Tabulka 3-1 uvádí parametry kolektoru, vztahující se k příjmu vstupních dat.

Požadavek	Úroveň splnění požadavku		
	Minimální	**Dostatečné**	**Ideální**
Transport dat po síti	UDP	UDP, TCP	UDP, TCP, SCTP, SSL
Formát a typ dat	NetFlow v5	NetFlow v5/v9, IPFIX	NetFlow v5/v9, IPFIX, SNMP, sFlow, NSEL, syslog
Autentizace	Ne	Na základě zdrojové IP adresy	Oboustranná autentizace pomocí certifikátů
Počet zdrojů dat	10	200	1000
Množství toků z jednoho zdroje dat	100 tis. toků/s	200 tis. toků/s	500 tis. toků/s
Celkové množství přijímaných toků	500 tis. toků/s	1 mil. toků/s	2 mil. toků/s (až 3 mil. toků/s po dobu 5s)
Zvyšování výkonu za běhu	Ne	Ano	Ano
Zotavení po výpadku zdroje	Ne	Ano	Ano, zaznamenávání událostí
Efektivita příjmu dat vůči nfcapd	>25%	>45%	>70%
Sledování zátěže	Ne	toky/s	toky/s, CPU, paměť
Detekce přetížení	Ne	Ano	Ano
Volitelné vzorkování toků	Ne	Ano	Ano, adaptivní dle zátěže
Přeposílání dat	Ne	Ne	Ano
Časová značka přijetí dat	Ne	Ano	Ano
Administrativní rozhraní	Ne	Základní	Pokročilé

Tabulka 3-1 Parametry kolektoru pro příjem vstupních dat.

- *Transport dat po síti*: Data ze zařízení, která exportují zachycené toky, mohou být po síti přenášena pomocí několika transportních protokolů. Základem je zde nespolehlivý přenos, a tedy protokol UDP. Spolehlivý přenos je zajišťován protokolem TCP, v případě nutnosti zabezpečení

přenosu dat se jedná o protokoly SCTP či SSL. Tento parametr uvádí, jaké transportní protokoly bude kolektor na svém vstupu podporovat.

- *Formát a typ dat*: Zařízení odesílající záznamy o tocích mohou pro export těchto záznamů používat jak některý ze standardizovaných, tak i nějaký z proprietálních protokolů. Některé z těchto protokolů umožňují exportovat pouze omezenou množinu dat (např. NetFlow v5). Typ dat tedy skrze podporovaný protokol vyjadřuje tuto množinu položek a jejich variabilitu. Jedná se zde od základních položek podporovaných protokolem NetFlow verze 5, až po libovolné položky proměnné délky exportované protokolem IPFIX. Data, obsahující IP toky, kdy klíčem toku je typicky pětice skládající se ze zdrojové/cílové IP adresy, zdrojového/cílového portu a protokolu, mohou být rozšířeny např. o HTTP, SIP či DNS položky. V ideálním případě je třeba počítat s flexibilitou vstupních dat, tj. položky nemají statickou velikost a mohou být libovolně přidávány a odebírány za běhu kolektoru. Parametr zde uvádí, které protokoly pro formátování záznamů budou kolektorem podporovány.
- *Autentizace*: Kolektor bude veřejně dostupný na síti a tím pádem na něj může kdokoliv posílat data. Aby nedošlo k podvržení vstupních dat je zde vhodná autentizace zdroje, ze kterého jsou data přijímána. Parametr určuje autentizační mechanizmus.
- *Počet zdrojů dat*: Na kolektor mohou data přicházet současně z více zdrojů. Parametr zde uvádí jejich maximální množství, které je kolektorem pro příjem dat podporováno.
- *Množství toků z jednoho zdroje dat*: Parametr představuje maximální intenzitu, jakou bude moci zdroj, exportující nejvíce toků, zasílat data. Základní předpoklad je zde ten, že kolektor bude přijímat data z velkého množství zdrojů s různorodou intenzitou generování toků (v rozmezí 1 tis. – 200 tis. toků/s) nebo bude vstupní data přijímat od několika zdrojů dat s vysokou intenzitou (200 tis. - 400 tis. toků/s).
- *Celkové množství přijímaných toků*: Tímto parametrem je zde vyjádřena dlouhodobá výkonnost kolektoru v počtu přijatých toků za 1 sekundu. Dále uvádí i krátkodobou výkonnost pro zpracování většího množství toků během krátkého intervalu.
- *Zvyšování výkonu za běhu*: Tento parametr udává, zda je možné při alokaci dalšího hardware, začlenit tento hardware do běžícího kolektoru a zvyšovat tak propustnost vstupu dat za běhu. Toto zvyšování výkonu je možné jen pro nově přidávané zdroje dat, tj. příjem dat z již exportujících zdrojů není možné přesunout na nově přidaný hardware.
- *Zotavení po výpadku zdroje*: Během provozu může dojít k výpadku zdroje dat či spojení s tímto zdrojem. Parametr zde udává, zda je kolektor schopen takovýto výpadek překonat a ustanovit znovu příjem dat. V ideálním případě musí kolektor zajistit opětovné připojení zdroje dat bez nutnosti zásahu uživatele (po obnovení komunikace).
- *Efektivita příjmu dat vůči nfcapd*: Navrhovaný kolektor bude distribuovaný na více výpočetních uzlů, což s sebou přináší režii, nutnou k zajištění běhu kolektoru. Parametr udává propustnost příjmu dat přepočítanou na jeden uzel, která nesmí být horší než procento dosažené při propustnosti kolektoru NfSen při zpracování stejného typu dat, běžícím v jedné instanci a na stejném hardware.
- *Sledování zátěže*: Kolektor by měl být schopen sledovat vytížení svých zdrojů, aby mohl předcházet jejich vyčerpání. Parametr zde definuje, v jakém rozsahu bude vytížení prostředků monitorováno.
- *Detekce přetížení*: Souvisí s předchozím parametrem. Kolektor může být zahlcen velkým množstvím příchozích dat a je vhodné tyto situace detekovat. Parametr udává, zda je tato detekce implementována.

- *Volitelné vzorkování toků*: Vzorkování toků využívá kolektor pro snížení zátěže a k úspoře diskového prostoru. Parametr udává, jaké způsoby vzorkování jsou podporovány. V případě adaptivního vzorkování je vzorkování automaticky spuštěno subsystémem při detekci přetížení.
- *Přeposílání dat*: V některých případech je vhodné přeposílat data na další kolektor. Parametr značí, zdali je přeposílání implementováno.
- *Časová značka přijetí dat*: Čas přijetí dat hraje významnou roli při analýze těchto dat. Tento parametr určuje, zda je kolektor schopen uložit časovou značku doby přijetí do každého toku (předpokládaná přesnost je v řádu milisekund).
- *Administrativní rozhraní*: Vstupní procesy je potřeba za běhu konfigurovat a sledovat jejich stav. Základní administrativní rozhraní umožňuje zjišťovat informace o počtu přijatých a zahozených toků v průběhu činnosti kolektoru. Pokročilé administrativní rozhraní přidává možnost sledování více čítačů a nastavovat parametry přijímajících procesů za běhu.

3.2.2 Předzpracování dat a tvorba profilů

Po fázi příjmu a normalizace dat dochází před jejich uložením do úložiště (případně dalším zpracováním a analýzou) k předzpracování. Cílem tohoto předzpracování dat je obohatit data o informace, které mohou být použiti pro jejich analýzu či pro poznačení příslušnosti dat k jednotlivým zákazníkům. Tato funkcionalita bude součástí vstupní části kolektoru a bude prováděna na stejných uzlech jako příjem dat.

Pojem profil zde značí způsob filtrování vstupních dat. Výchozím profilem je „live/all“, který obsahuje všechna přijímaná data. Profily mohou být dvojího typu. Jednak jsou to běžné profily, které obsahují kopii vyfiltrovaných dat, jednak jsou to profily virtuální (tzv. stínové), které data nekopírují. Jednotlivé profily jsou definovány filtrovacím výrazem a jejich vytváření je umožněno každému zákazníkovi. Typickými příklady profilů mohou být jednotlivé sítě zákazníka, provoz z/do internetu, vybrané klíčové podnikové aplikace atd. Tabulka 3-2 zobrazuje požadavky na parametry této části kolektoru.

Požadavek	Úroveň splnění požadavku		
	Minimální	**Dostatečné**	**Ideální**
Identifikace zákazníka	IP adresa zdroje dat	IP adresa zdroje dat, IP prefix	IP adresa zdroje dat, IP prefix, MPLS
Deduplikace dat	Ne	Ne	Ano
Profily	Virtuální	Reálné i virtuální, <10 profilů na zákazníka	Reálné i virtuální, >100 profilů na zákazníka
Podprofily	Ne	Ano	Ano, více než 1 úroveň zanoření
Přiřazení profilů/podprofilů uživateli	Ne	Ano	Ano
Počet zákazníků	>10	>100	>500
Počet uživatelů na zákazníka	1	>10	>100
Předpočítávání agregovaných statistik	Ne	Ano, >10 na profil	Ano, >100 na profil
Obohacení dat	Ne	Ano, geoIP	Ano, geoIP, uživatel

Vzorkování	Ne	Ano, plošně	Ano, selektivně dle zákazníka
Sledování vytížení	Ne	toky/s	toky/s, CPU, paměť
Administrativní rozhraní	Ne	Základní	Pokročilé

Tabulka 3-2 Parametry kolektoru pro předzpracování dat a tvorbu profilů.

- *Identifikace zákazníka*: Kolektor bude přijímat data od mnoha zákazníků. Parametr udává, jakým způsobem je možné jednotlivé zákazníky od sebe odlišit. Při ideálním splnění tohoto požadavku bude zákazník identifikován pomocí tzv. wildcard pravidel, obsahujících zdroj dat o IP tocích (observation domain IP nebo zdrojová IP adresa), IP prefix nacházející se v datech, MPLS štítek. Kolektor musí být schopen řešit překryv pravidel i příslušnost toků k více pravidlům (např. při komunikaci dvou zákazníků přes jedno exportní zařízení).
- *Deduplikace dat*: V situaci, kdy má jeden zákazník více zdrojů dat o IP tocích, dochází k hlášení stejných toků z více zdrojů. Deduplikace zajišťuje hlášení těchto toků pouze jednou. Parametr zde vyjadřuje, zdali je tato funkcionalita podporována.
- *Profily*: Parametr udává, jakým způsobem je podporována práce s profily a jejich počet pro jednoho zákazníka.
- *Podprofily*: Jedná se o tvorbu profilů, které jsou aplikovány na již stávající profily. Tímto způsobem je možné vytvářet stromovou strukturu profilů. Parametr vyjadřuje podporu této funkcionality a míru zanoření profilů.
- *Přiřazení profilů/podprofilů uživateli*: Zákazník v roli uživatele s právy administrátora může definovat své vlastní profily. K těmto profilům může následně přiřadit uživatele, který může v rámci svého profilu definovat další podprofily a tyto následně také přiřadit dalším uživatelům. Parametr udává, zda je toto přiřazení uživatele k profilům podporováno.
- *Počet zákazníků*: Parametr vyjadřuje počet zákazníků, pro kolik je kolektor schopen předzpracovat data.
- *Počet uživatelů na zákazníka*: Hodnota parametru udává počet uživatelů u jednoho zákazníka, pro který je kolektor schopen vytvářet profily.
- *Počet profilů na uživatele*: Parametr udává počet profilů, jenž je kolektor schopen průměrně vypočítat na jednoho uživatele.
- *Předpočítávání agregovaných statistik*: Tento parametr vyjadřuje, kolik agregovaných statistik je kolektor schopen vypočítat v daném profilu.
- *Obohacení dat*: Obohacení dat se zde vztahuje ke zdrojové a cílové IP adrese. Parametr udává, jakou informací budou data obohacována.
- *Vzorkování*: Hodnota parametru určuje, zdali je implementována podpora vzorkování a možnost nastavení tohoto vzorkování individuálně dle zákazníka.
- *Sledování vytížení*: Předzpracování dat může být výpočetně velmi náročné. Je proto potřeba sledovat vytížení této části systému a v případě potřeby včas alokovat další zdroje. Parametr zde opět udává podporu této funkcionality a typ sledovaných ukazatelů.
- *Administrativní rozhraní*: Procesy ve fázi předzpracování dat je potřeba za běhu sledovat a konfigurovat. Základní administrativní rozhraní umožňuje získávat informace o počtu přijatých a zahozených toků. Pokročilé administrativní rozhraní přidává možnost sledovat za běhu další statistiky, týkající se vytížení hardwarových zdrojů a nastavovat parametry procesů.

3.2.3 Uložení dat a indexace

Primární a předzpracovaná data budou dlouhodobě ukládána v distribuované architektuře (na více uzlech). Cílem využití distribuovaného přístupu je dosažení vyšší celkové propustnosti při ukládání dat do trvalé paměti a při jejich opětovném získávání. Dalším klíčovým požadavkem je vysoká dostupnost a spolehlivost dat. Těchto požadavků bude dosaženo pomocí replikace ukládaných dat na více uzlů. V případě některých dotazů je pak možné urychlit získání dat tvorbou indexu při jejich ukládání. Tabulka 3-3 opět shrnuje požadované parametry, vztahující se k této problematice.

Požadavek	Úroveň splnění požadavku		
	Minimální	**Dostatečné**	**Ideální**
Kapacita trvalé paměti	>1 bil. záznamů	>10 bil. záznamů	>30 bil. záznamů
Přidávání kapacity trvalé paměti za běhu	Ne	Ano, přidáním disku	Ano, přidáním uzlu
Zotavení z výpadku hardware	Ne	Ano, selhání disku	Ano, selhání uzlu
Sledování zaplnění trvalé paměti	Ne	Ano	Ano, prediktivní, sdílené
Varování při nedostatku úložného prostoru	Ne	Ano, celkově	Ano, celkově i na zákazníka
Výkonnost dlouhodobá	300 tis. toků/s	500 tis. toků/s	1 mil. toků/s
Výkonnost krátkodobá	500 tis. toků/s	1 mil. toků/s	2 mil. toků/s
Efektivní uložení dat vůči NfSen	30%	50%	70%
Indexování	Ne	Ne	Ano
Efektivita indexu	-	-	40% kapacity při 200% výkonu
Flexibilita dat	Ne	Dynamická sada položek se statickou velikostí	Dynamická sada položek, které mohou mít proměnnou délku
Administrativní rozhraní	Ne	Základní	Pokročilé

Tabulka 3-3 Parametry kolektoru pro ukládání dat a tvorbu indexů.

- *Kapacita trvalé paměti*: Kolektor musí být schopen uchovávat velké množství záznamů o tocích. V závislosti na reprezentaci (binární či textové) se mění prostor na disku, potřebný pro uložení záznamu. Proto je tedy tento parametr uveden v počtech záznamů namísto běžně používaných bytů. (Pozn. Více než 30 bil. záznamů o tocích odpovídá při konstantní rychlosti příjmu 1 mil. toků/s zhruba 360 dnům historie.)
- *Přidávání kapacity trvalé paměti za běhu*: Udává možnost rozšíření úložného prostoru při běhu kolektoru.
- *Zotavení z výpadku hardware*: Je pravděpodobné, že v prostředí distribuovaného výpočetního systému bude docházet k selhání hardware. Parametr udává odolnost kolektoru vůči těmto výpadkům a tedy i ztrátě dat. V ideálním případě bude kolektor schopen zajistit okamžitou a transparentní obnovu dat při selhání hardware.
- *Sledování zaplnění trvalé paměti*: Kolektor musí průběžně sledovat stav zaplnění paměti, aby mohl včas signalizovat nedostatek místa. V základní variantě je sledováno pouze zaplnění

z pohledu množství vyčerpané paměti. Prediktivní sledování umožňuje poskytnout informaci vztaženu k času (tj. např. místo informace typu „zbývá 50GB prostoru“ poskytne informaci „zbývající prostor bude vyčerpán za 12 hodin“). Toto zaplnění je vhodné sledovat společně pro celý systém i pro jednotlivé zákazníky. Parametr udává, zdali je sledování zaplnění paměti implementováno a případně jakým způsobem.

- *Varování při nedostatku úložného prostoru*: Vychází z předchozího bodu a z předpokladu, že zákazník bude mít k dispozici určitou kapacitu úložného prostoru. Parametr udává, jestli bude možné při vyčerpání zdrojů pro ukládání dat (např. při 90% zaplnění) zákazníka upozornit na tuto skutečnost. Zaplnění úložného prostoru je nutné sledovat jak pro jednotlivé zákazníky, tak i z pohledu celkového systému a v případě nutnosti včas varovat správce kolektoru na docházející fyzické místo.
- *Výkonnost dlouhodobá*: Kolektor musí být schopen ukládat soustavný proud dat. Tento parametr určuje množství v počtu toků za vteřinu, jež je kolektor schopen zpracovat bez ztráty dat.
- *Výkonnost krátkodobá (špičková)*: Během příjmu a ukládání dat může dojít k dočasnému zvýšení objemu proudu příchozích toků (např. během DoS útoku). Parametr udává, kolik toků za vteřinu je kolektor schopen krátkodobě[4] zpracovat.
- *Efektivita uložení dat vůči NfSen*: Data budou v úložišti ukládána ve speciálních formátech a strukturách. Efektivita uložení dat vyjadřuje poměr uložení stejného typu dat, obsahujících záznamy o tocích, kolektorem Nfsen vůči navrhovanému kolektoru. Do efektivity uložení se nezapočítává nutná replikace dat, při splnění podmínky zotavení.
- *Indexování*: Určuje, zdali kolektor podporuje indexování dat pro urychlení vybraných dotazů (např. výběr na základě IP adresy).
- *Efektivita indexu*: Vhodný index musí urychlit zpracování dotazu a zároveň nesmí přesáhnout určité procento kapacity, kterou zabírají primární data. Parametr udává přijatelnou míru zaplnění prostoru v úložišti vůči zrychlení.
- *Flexibilita dat*: Přijaté záznamy mohou být ukládány do pevně dané struktury s předem určenými statickými položkami. Tento přístup však není příliš vhodný, proto je nutná podpora dynamické volby položek s případnou variabilní délkou.
- *Administrativní rozhraní*: V základní variantě by mělo administrativní rozhraní umožnit sledování zaplnění diskové kapacity. Pokročilé rozhraní pak bude navíc umožňovat toto sledování na úrovni zákazníků, predikci zaplnění a změny parametrů úložiště (kvóty, indexování apod.).

3.2.4 Dotazování

Dotazování nad daty uloženými v distribuované architektuře bude využívat vyšší celkové propustnosti a výpočetní výkonnosti. Během dotazování jsou data získávána z trvalé paměti na jednotlivých uzlech a zároveň se na každém tomto uzlu vykonává část dotazu nad lokálně přístupnými daty. Další část dotazu, kterou není možné provést distribuovaně, je dopočítána nad daty, získanými z jednotlivých uzlů. Stejně jako u předchozích sekcí, shrnuje Tabulka 3-4 parametry týkající se této části kolektoru.

[4] Předpokládám zde interval 5 vteřin.

Požadavek	Úroveň splnění požadavku		
	Minimální	**Dostatečné**	**Ideální**
Dokončení dotazu	Ne	Ano	Ano
Odhad délky trvání dotazu	Ne	Ano	Ano
Ukazatel průběhu dotazu	Ne	Ne	Ano
Spuštění dotazu na pozadí	Ne	Ano	Ano, vícenásobné
Prioritizace ukládání	Ne	Ne	Ano
Podpora multitenance	Ne	Ano	Ano
Výkonnost (bez režie)	5 mil. toků/s	10 mil. toků/s	30 mil. toků/s
Režie	<20s	<10s	<5s
Efektivita vůči NfSen	>25%	>40%	>50%
Administrativní rozhraní	Ne	Základní	Pokročilé

Tabulka 3-4 Parametry kolektoru pro dotazování se nad uloženými daty.

- *Dokončení dotazu*: Zpracování každého dotazu by mělo být ošetřeno tak, aby dotaz skončil v konečném čase (ne chybou programu). Parametr udává, zdali je toto ošetření implementováno.
- *Odhad délky trvání dotazu* : Před samotným spuštěním dotazu bude na základě množství toků k dispozici odhad doby trvání dotazu.
- *Ukazatel průběhu dotazu*: Parametr udává, jestli je uživatel v průběhu dotazu informován o stavu dotazu, délce dosavadního výpočtu a jeho očekávaném ukončení.
- *Spouštění dotazů na pozadí*: Některé náročné dotazy mohou běžet relativně dlouhou dobu. Parametr definuje možnost uživatele spouštět dotazy na pozadí. V tomto případě musí systém informovat uživatele v okamžiku, kdy je výsledek k dispozici.
- *Prioritizace ukládání*: Průběžné ukládání dat kolektorem má přednost před výpočty dotazů. Parametr značí, zdali je podpora přednostního ukládání implementována.
- *Podpora multitenance*: Parametr určuje, zdali systém umožňuje obsloužit z jedné instance kolektoru více zákazníku při striktním oddělení dat jednotlivých zákazníků.
- *Výkonnost*: Dotazování bez využití indexace musí dosáhnout výkonnosti při dotazu typu filtrace určitého počtu toků za vteřinu. To prakticky znamená, že musí být kolektor schopen načíst z úložiště uvedené množství toků za sekundu a porovnat tyto toky na podmínku filtrace (tj. dotaz, při kterém není nutné udržovat stavovou informaci). Při použití indexu se tato výkonnost počítá jako by index neexistoval a načítaly se všechny toky. Výkonnost, uvedená pro tento parametr, nezahrnuje nutnou režii distribuovaného zpracování.
- *Režie*: Parametr uvádí maximální latenci, způsobenou režií dotazovacího subsystému. Je zde nutné uvažovat určitou latenci i pro ty nejjednodušší dotazy. Tato latence není započítávána do výkonnosti dotazování.
- *Efektivita dotazování vůči NfSen*: Kolektor musí dosahovat alespoň určité efektivity dotazování na jeden uzel ve srovnání s Nfsen kolektorem. Porovnává se zde výkonnost bez latence. Parametr uvádí výkonnost kolektoru, vztaženou na uzel, vůči kolektoru Nfsen.
- *Administrativní rozhraní*: Základní administrativní rozhraní zde bude poskytovat odhad doby trvání dotazů a informaci o jejich ukončení. Pokročilé rozhraní navíc bude poskytovat informaci o průběhu jednotlivých dotazů.

3.3 Experimenty

V následujícím textu jsou popsány experimenty, které byly provedeny v distribuovaném prostředí za účelem získání prvotní představy o možnostech distribuovaného zpracování dat o IP tocích s využitím existujících řešení. Smyslem těchto experimentů bylo také odhalit nedostatky a slabá místa, dále snaha optimalizace parametrů a nastavení těchto platforem pro potřeby vyvíjeného kolektoru.

Pro realizaci distribuovaného úložiště byla zvolena volně dostupná implementace Apache Hadoop[5], která umožňuje spolehlivé uložení a provádění distribuovaných výpočtů na velkých objemech dat s využitím počítačového clusteru. Pro práci s tímto frameworkem byly vybrány čtyři přístupy. První z nich je ruční implementace dotazů, zvolených pro experimenty, v jazyce JAVA. Tato ruční implementace je pak spouštěna nad daty ve formátu CSV a nad binárními soubory. Dále byly použity dvě nástavby pro Hadoop, poskytující rozhraní pro dotazování nad uloženými daty. Jedná se o nástavby Hive a Pig[6]. Jako poslední varianta přibyl v pozdější fázi experimentů produkt Spark[7], který je schopen pracovat nad distribuovaným úložištěm Hadoopu. Měření doby provedení dotazů pomocí těchto přístupů bylo porovnáváno vůči výsledkům, získaným pomocí nástroje Nfdump[8], který byl spuštěn nad stejnými daty v komprimované či nekomprimované formě na jednom stroji. V následujících sekcích bude stručně popsáno použité technické a programové vybavení. Tuto kapitolu pak uzavře poslední sekce popisem dat a dotazů, s nimiž bylo experimentováno a samozřejmě také výsledky experimentů.

3.3.1 Technické vybavení

Experimenty s uvedenými platformami a nástroji probíhali na několika různých systémech. Experimenty s nástrojem Nfdump, které poskytly referenční hodnoty pro porovnání a pro odhad míry zrychlení doby provádění dotazů, běžely na jediném stroji, protože nástroj Nfdump neobsahuje podporu distribuovaného zpracování. Tyto testy byly spuštěny na fyzickém stroji (bez vizualizace) s čtyřjádrovým procesorem, s 10GB paměti a jedním diskem bez RAID. Prostředí pro testování ostatních platforem poskytla virtuální organizace (dále VO) MetaCentrum[9]. Počáteční testy, které měli ukázat rychlost testovaných systémů vůči nástroji Nfdump probíhaly na clusteru o 6 strojích, z nichž hlavní uzel poskytoval procesor o 8 jádrech, 32GB paměti s diskem bez RAID. Výpočetní uzly pak byly v konfiguraci s procesory o 4 jádrech, 8GB paměti a se 2 disky v RIAD0, sdílenými s dalšími virtuálními stroji.

V pozdější fázi experimentů poskytla VO MetaCentrum specializovaný Hadoop cluster[10]. Tento cluster sestává z 27 strojů, vybavených šestnáctijádrovými procesory, 128GB paměti na každém z nich a úložiště o celkové velikosti 1PB. Experimenty prováděné na tomto větším clusteru byly již zaměřeny na optimalizaci těchto systémů pro rychlé dotazování nad daty o IP tocích a na vyhodnocení vhodnosti těchto systémů pro vyvíjený kolektor v souladu s vytvořenou specifikací. Smyslem těchto testů bylo tedy odhalit slabá místa testovaných systémů, zjistit možnosti vyladění prostřednictvím změny konfiguračních parametrů či změřit schopnost zpracovávat více dotazů současně a případnou škálovatelnost při současném zpracování více dotazů.

[5] http://hadoop.apache.org/
[6] https://hive.apache.org/ a http://pig.apache.org/
[7] https://spark.apache.org/
[8] http://nfdump.sourceforge.net/
[9] https://metavo.metacentrum.cz/
[10] http://www.metacentrum.cz/cs/hadoop/

Prováděné experimenty se zaměřují zejména na dotazovací část vyvíjeného kolektoru, protože právě ta je hlavní náplní implementační části této práce a jeví se jako nejvíce kritická. V této fázi je totiž potřeba zpracovávat největší množství dat s co nejkratší dobou odezvy.

3.3.2 NfDump

Nfdump je nástroj ovládaný z příkazové řádky, který je součástí stejnojmenné sady nástrojů pro zobrazení a analýzu netflow dat. Nfdump načítá soubory uložené pomocí nástroje Nfcapd a zpracovává v nich uložené toky podle pravidel, určených parametry programu. Syntaxe filtrů, které lze v parametrech použít, je srovnatelná s filtry používanými programem Tcpdump a je rozšířena o možnosti práce s netflow daty. Nfdump umožňuje poskytnout výpis uložených záznamů, které mohou být na základě různých položek filtrovány (např. pro získání záznamů, týkajících se jedné služby, může být použito filtrování na základě čísel portů). Dále je možné pomocí tohoto nástroje počítat top-n statistiky či prohlížet agregované záznamy, typicky např. pro IP adresy, porty (služby), atd. Nfdump je relativně výkonný a v dnešní době běžně používaný nástroj pro analýzu dat o IP tocích, a proto byl použit při experimentování jako referenční bod měření doby trvání dotazů.

3.3.3 Hadoop

Hadoop, vyvíjený společností Apache software foundation, je rámec pro spolehlivé, škálovatelné, distribuované výpočty nad rozsáhlými daty. Může pracovat v rámci jedné stanice, ale jeho hlavní účel a síla spočívá ve využití velkého množství samostatných, spolupracujících zařízení (tzv. clusteru), která nemusí být nijak zásadně specializovaná a může se jednat o běžně dostupný hardware. Celý rámec Hadoop sestává z několika částí. V první řadě se jedná o distribuovaný souborový systém HDFS (Hadoop Distributed File System), dále je to implementace samotného MapReduce paradigmatu, v nové verzi[11] framework Hadoop YARN zajišťující správu zdrojů a plánování distribuovaných úloh a sada pomocných knihoven Hadoop Common.

Oproti klasickému přístupu, kdy jsou uložená data přenášena do místa výpočtu požadované funkce, využívá Hadoop opačného modelu, kde jsou výpočetní funkce posílány k datům v distribuovaném úložišti prostřednictvím úloh (tzv. jobs). Pomocí tohoto přístupu je umožněno zpracování obrovského množství dat, jež by běžnými prostředky nebylo z technického či ekonomického hlediska možné.

Další důležitou vlastností Hadoopu je replikace dat. Ta vychází z předpokladu výskytu chyb, ke kterým zajisté bude v rámci velkého výpočetního clusteru docházet. Replikace dat zajišťuje jednak spolehlivost uložených dat a odolnost vůči jejich ztrátě při selhání úložného zařízení, jednak také umožňuje redundanci výpočtu, díky níž je možné dosáhnout lepší spolehlivosti provedení distribuovaného dotazu.

Ukládání dat a výpočty nad těmito daty v rámci systému Hadoop probíhají nad distribuovaným souborovým systémem HDFS, který byl navržen pro běh na běžně dostupných zařízeních. Jedná se o virtuální souborový systém, vybudovaný nad běžnými souborovými systémy jednotlivých uzlů a zabývá se tedy pouze nalezením úložiště a přístupu k datům (tj. nezajišťuje přímo fyzické uložení dat na uzlu). HDFS byl inspirován souborovým systémem GFS, navrženým společností Google pro účely vlastní implementace MapReduce modelu. HDFS zajišťuje optimální uložení dat, přístup k těmto datům a odolnost vůči jejich výpadkům skrze replikaci. Souborový systém HDFS byl navržen pro zpracování velmi velkých souborů, které jsou v rámci HDFS uloženy v blocích fixní velikosti

[11] hadoop-0.23

(typicky 64MB či 128MB). Distribuce dat a jejich replikace pak probíhá na úrovni těchto bloků. Manipulaci s daty v HDFS zajišťují dva typy uzlů, které jsou označovány jako *namenode* a *datanode*.

Namenode je v systému jen jeden. Tento uzel spravuje celý souborový systém a obsahuje metadata souborů. Namenode řídí rozdělení dat na bloky a přiřazení těchto bloků na jednotlivé datanode, přičemž uživatelská data tímto uzlem nikdy přímo neprochází, namenode pouze řídí proces ukládání dat. Dále tento uzel obsluhuje operace jako je otevírání či uzavírání souborů, přejmenování souborů nebo adresářů apod. Samotné operace čtení či zápisu provádí jednotlivé datanode uzly. V průběhu svojí činnosti si namenode stále udržuje informaci o umístění bloků všech souborů. Pokud je potřeba k některému ze souborů přistoupit, je pak tento přístup zprostředkován právě tímto uzlem. Dále namenode kontroluje stav a zdraví jednotlivých datanode, které pravidelně signalizují svůj stav pomocí tzv. *heartbeat* zpráv a informují o datech na nich uložených podle speciálních správ, nazývaných *blockreport*.

Mimo HDFS uzlů existují v rámci Hadoop další dva typy, jedná se o tzv. *job-tracker* a *task-tracker* uzly. Job-tracker přijímá a řídí požadavky na MapReduce operace. Obdobně jako namenode je job-tracker v rámci celého výpočetního clusteru pouze jeden a musí být proto spolehlivý. Job-tracker tedy určuje rozdělení Map a Reduce úloh na jednotlivé task-tracker uzly a to tak, aby každý task-tracker prováděl tyto operace nad jemu blízkými daty. Spolehlivost uzlů typu task-tracker není nikterak kritická, v případě jejich výpadku zopakuje job-tracker výpočet na jiném dostupném uzlu.

V novější verzi systému Hadoop je tento přístup k vykonávání MapReduce operací změněn. Nový způsob provádění úloh zprostředkovává tzv. YARN, někdy také označovaným jako NextGen MapReduce nebo MapReduce 2.0 (MRv2). Hlavní změna v tomto novém přístupu zamýšlí rozdělit správu prostředků a plánování či sledování jednotlivých úloh, prováděnou na job-tracker uzlu. Myšlenkou je mít v rámci Hadoop clusteru jeden globální *ResourceManager* uzel, který se bude skládat z *plánovače* (angl. *scheduler*) a *AplicationsManageru*. Smyslem plánovače je čistě alokace zdrojů spouštěných úloh a nestará se o sledování stavu jednotlivých aplikací. ApplicationManager se pak stará o příjem uživatelských úloh, jejich přípravu a poskytuje mechanismus pro případné restartování nezdařených úkolů. Další komponentou v rámci nového přístupu je *NodeManager*, jenž se nachází na každém výpočetním uzlu. Tento uzel má za úkol správu lokálních zdrojů a sledování lokálních prostředků (vytížení procesoru, spotřebovaná paměť či diskový prostor, vytížení sítě) a hlášení těchto informací ResourceManageru. Tyto dvě hlavní komponenty pak spojuje *ApplicationMaster*, který je jeden pro každou aplikaci a jeho úkolem je vyjednání potřebných zdrojů mezi plánovačem a NodeManagerem, sledování stavu dané úlohy a zjišťování postupu výpočtu.

Pro ovládání systému Hadoop jsou mimo možnosti vývoje vlastních aplikací v programovacím jazyce Java, který je nativní pro Hadoop, k dispozici různé nadstavby. Jejich účelem je usnadnit práci se systémem a zejména pak dotazování nad daty. Dvě z nástaveb, jež byly při experimentech se systémem Hadoop použity, jsou stručně popsány v následujících sekcích. Poslední podkapitola zabývající se programovým vybavením, použitým pro experimenty pak stručně popisuje systém Spark, který je také schopný pracovat nad distribuovaným úložištěm HDFS a jenž používá odlišný přístup zpracování dat.51[9]

3.3.4 Hive a Pig

Účel nástavbových softwarových produktů Apache Hive a Apache Pig je poskytnout jednodušší rozhraní pro dotazování a správu dat ukládaných prostřednictvím Hadoopu. Produkt Hive vznikl pro potřebu společnosti Facebook zpracovávat rozsáhlé a neustále rostoucí objemy dat. Jeho

cílem bylo poskytnout rozhraní podobné jazyku SQL pro uživatele, kteří byli specializováni na práci s SQL databázemi tak, aby i tito uživatelé mohli rychle a snadno analyzovat data, uložená v HDFS. Hive poskytuje prostředky pro zavedení dodatečné struktury těchto dat a pro dotazování nad daty používá jazyk HiveQL, jenž se svojí syntaxí podobá právě jazyku SQL, nicméně neposkytuje plnou funkcionalitu jazyka SQL. Současně Hive umožňuje prostřednictvím jazyka HiveQL zapojení vlastních funkcí Map a Reduce do výpočtu.

Další testovanou nástavbou pro systém Hadoop je Apache Pig. Tato platforma se také zaměřuje na analýzu rozsáhlé sady dat. Oproti holému Hadoopu poskytuje vyšší míru abstrakce, kdy se snaží odstínit MapReduce model od uživatele, pro kterého je tak snazší zadávat jednotlivé dotazy na uložená data. Poskytuje také některé operace, které nejsou v Hadoopu implicitně podporovány (např. operace *join*). Pig sestává z jazyka pro reprezentaci toku dat, nazývaný Pig Latin a z prostředí pro provádění úloh. Tímto prostředím může být buď lokální instance JVM[12] nebo distribuované prostředí poskytované systémem Hadoop. [9][10]

3.3.5 Spark

Apache Spark je jinou alternativou k Hadoop MapReduce přístupu. Oproti Hadoopu se více zaměřuje na rychlost odezvy jednotlivých dotazů. Pro dosažení tohoto cíle se snaží mj. pracovat s daty co nejvíce v paměti a ukládat dílčí výsledky na disk pouze v případě nutnosti. Základním konceptem práce s daty v systému Spark jsou tzv. Resilient Distributed Datasets (RDD), které mohou obsahovat jakákoliv data. Prostřednictvím RDD lze kontrolovat výpočet jednotlivých úloh a v případě chyby dopočítat ztracená data. Dotazy nad uloženými daty jsou pak realizovány dvěma druhy operací nad RDD – *transformacemi* a *akcemi*. Transformace berou na svém vstupu nějaký RDD, aplikují na něj požadovanou operaci a vrátí nový RDD. Transformací může být např. i nějaká funkce map. Akce dále bere na vstupu také RDD, výsledkem akce je však hodnota. Jako příklad akce lze uvést operaci reduce, count, first a další. Pro psaní dotazů podporuje Spark jazyky Java, Scala či Python.[11][12]

3.3.6 Popis použitých dotazů

Pro účely experimentů s distribuovaným zpracováním dat byly navrženy 4 dotazy, které se snaží svou strukturou zachytit typické dotazy, používané při analýze dat ze síťového provozu. Tyto dotazy byly následně spouštěny nad anonymizovanými daty z jednoho dne[13] reálného provozu, která byla rozdělena na soubory s přírůstkem dat odpovídajícím jedné hodině provozu (tj. první soubor neobsahuje žádná data[14], druhý soubor obsahuje data z jedné hodiny, třetí soubor data ze dvou hodin atd.). Celkově tato data obsahovala zhruba 880 mil. záznamů o zhruba 30 mld. paketech a 27 terabytech. Pro každý blok dat byl každý ze 4 dotazů spuštěn třikrát. Výsledná doba trvání dotazu se pak spočítala jako průměr těchto hodnot.

První z vytvořených dotazů počítá ze všech záznamů o tocích sumu počtu paketů a sumu počtu bytů v nich obsažených. Výsledkem jsou tedy tři hodnoty – počet toků, paketů a bytů. Druhý dotaz získává celkový počet všech záznamů s cílovým portem 53, tj. zaměřuje se na dotazy protokolu DNS. Výsledkem tohoto dotazu je pak jediná hodnota. Třetí dotaz vybírá jen zvolená pole (časová značka příchodu záznamu, protokol, zdrojová a cílová IP adresa, zdrojový a cílový port, počet paketů a počet bytů) pro záznamy o IP tocích přenášené spolehlivým protokolem TCP na portu 53. Výstupem dotazu

[12] Java Virtual Machine
[13] Data byla ze dne 1. 4. 2014.
[14] Ve skutečnosti obsahoval první blok 10 prvních toků.

jsou jednotlivé řádky, obsahující požadované záznamy. Poslední ze sady dotazů pro každou zdrojovou adresu počítá sumu paketů, bytů a celkový počet záznamů přenesených z této adresy pomocí protokolu TCP. Výsledkem jsou jednotlivé agregované záznamy, předávané na výstup po řádcích.

3.3.7 Výsledky experimentů

V následujícím textu jsou experimenty rozděleny na tři celky. V první části proběhlo základní testování, kdy byly všechny 4 dotazy spuštěny nad nástrojem Nfdump, nad Hadoopem a jeho prvními dvěma nástavbami. Smyslem těchto experimentů bylo získat srovnání s aktuálně dostupným a používaným řešením, běžícím v rámci jedné stanice, vůči kterému lze posuzovat přínos distribuovaného zpracování. Distribuované úlohy v rámci tohoto běhu byly spuštěny nad menším clusterem VO MetaCentra.

Ve druhé části experimentů již byly porovnávány pouze přístupy založené na Hadoopu. Součástí těchto experimentů bylo hledání možností optimalizace konfiguračních parametrů jednotlivých platforem, s cílem minimalizovat dobu odezvy se současným zachováním důležitých vlastností (spolehlivost výpočtů, redundance dat, apod.). Dotazy v rámci těchto pokusů byly spouštěny na Hadoop clusteru MetaCentra o 27 stanicích.

Poslední běh experimentů byl zaměřen na platformu Spark, jež byla do testování zahrnuta v pozdější fázi, kdy se testované systémy začínaly jevit jako nevyhovující. Experimenty s touto platformou však byly menšího rozsahu, jelikož se systém Spark ukázal jako zcela nevhodný pro některé druhy dotazů, jak bude uvedeno ve výsledcích testování tohoto systému. Testování platformy Spark proběhlo na stejném clusteru jako experimenty zaměřené na optimalizaci.

Základní experimenty – zjištění potenciálu distribuovaného zpracování

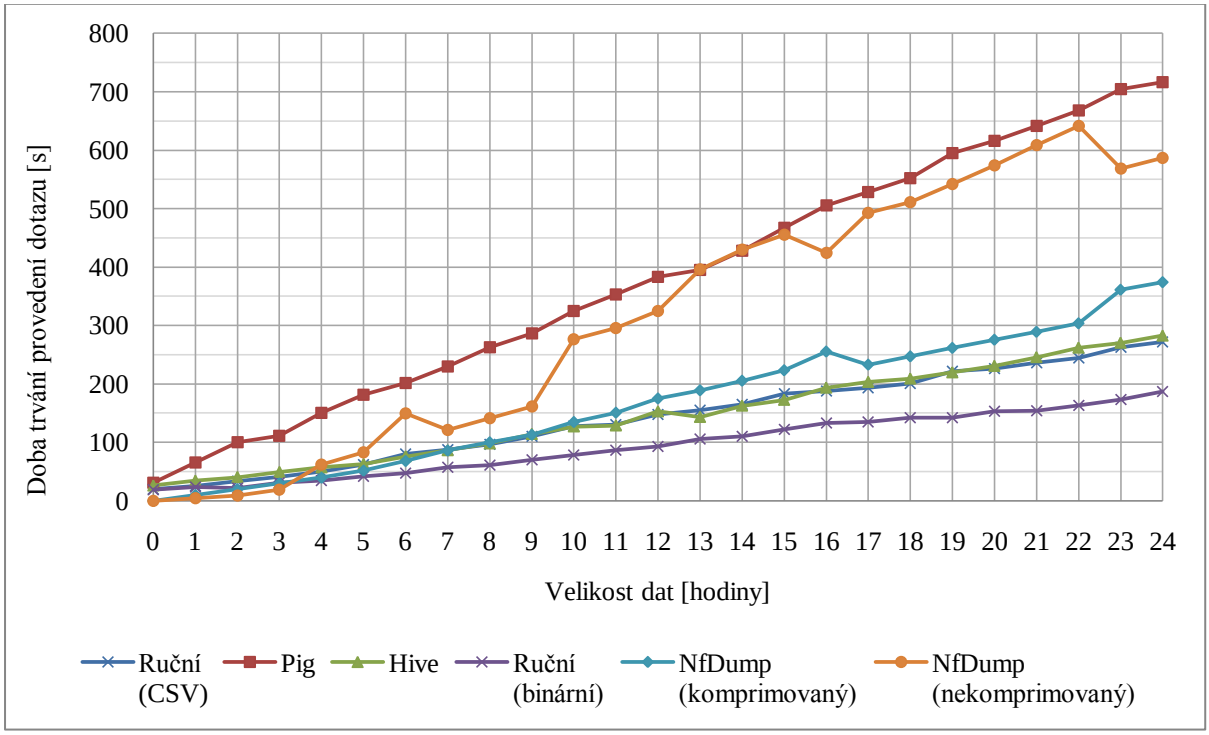
V následujícím textu jsou pro každý z popsaných dotazů uvedeny grafy, zachycující dobu odezvy těchto dotazů pro testované platformy.

Obrázek 3-3 zobrazuje výsledky provedení prvního navrženého dotazu. V grafu na tomto obrázku (a všech následujících) je na ose x velikost dat, vyjádřena v počtu hodin zpracovávaných dat. Na ose y je pak průměrná doba vykonání dotazu v sekundách. Graf pak zahrnuje srovnání všech testovaných platforem, kromě platformy Spark, která byla do experimentů zařazena dodatečně. Následující grafy pak zobrazují stejným způsobem výsledky provádění dotazů č. 2 – 4.

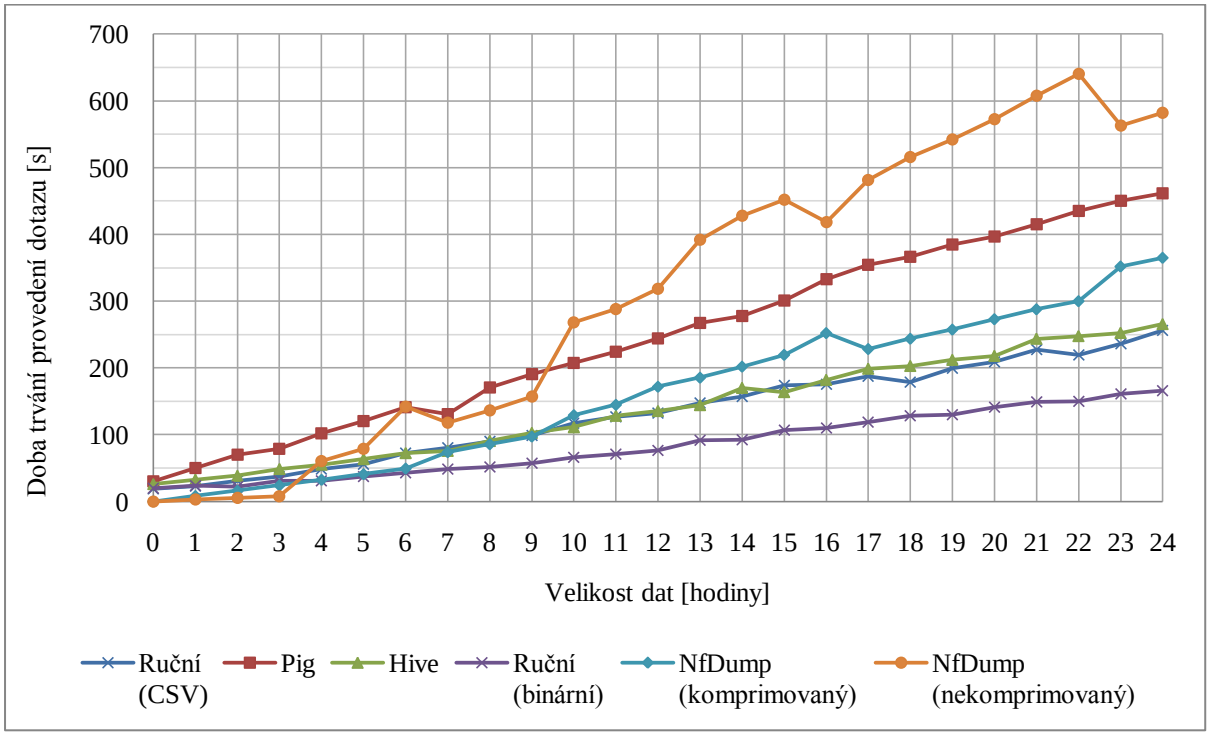
Z uvedených výsledků experimentů dosáhla nejlepších výsledků ruční implementace dotazů, realizovaná nad binárními daty. Dobrých výsledků dosahovala tato ruční implementace i nad daty ve formátu CSV (tj. textový soubor). Stejně tak za zmínku stojí i nástavba Hive, která oproti implementaci konkrétního dotazu poskytuje rozhraní pro zadávání obecně libovolných dotazů. Nejhůře ze všech testovaných možností dopadla nástavba Pig.

Další informací, kterou je možné z výsledků vyčíst je stálost doby trvání dotazů prováděných nástrojem Nfdump. Ta je způsobena tím, že Nfdump při vykonávání těchto dotazů musí vždy sekvenčně projít všechna data, což je z celého dotazu časově nejnáročnější část. Samotný výpočet pak již způsoboval odchylky pouze v rámci vteřin. Důležitým jevem, na který se bude nutné v následujícím výzkumu zaměřit, je počáteční doba latence dotazů spouštěných nad systémem Hadoop. Ta se i pro dotazy nad velmi malým množstvím dat pohybuje okolo 20 vteřin, což není pro navrhovaný kolektor vhodné. Jak je na grafech možno vidět, tato latence se projevovala u všech typů dotazů a je pravděpodobně zapříčiněna zejména způsobem komunikace jednotlivých uzlů prostřednictvím heartbeat zpráv. Celkově lze však z těchto výsledků vypozorovat zrychlení doby

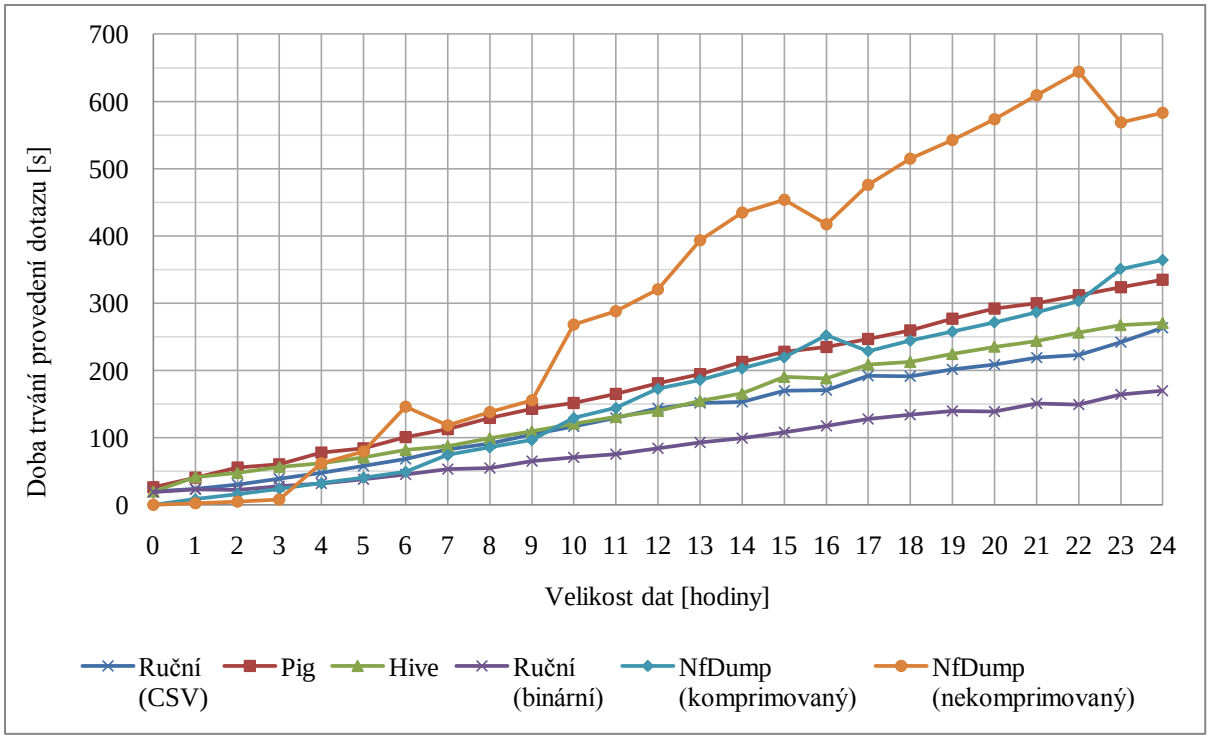
provedení jednotlivých dotazů, zvláště pak s rostoucím objemem dat. Nejedná se o pětinásobné zrychlení, které by se dalo intuitivně očekávat při použití 5 výpočetních uzlů. Nicméně je potřeba zohlednit jednak fakt, že použité distribuované systémy nejsou na rozdíl od nástroje Nfdump specializovány a optimalizovány na data o IP tocích. Dále pak skutečnost, že distribuované řešení s sebou přináší potřebu zajištění spolehlivosti a dostupnosti dat skrze redundanci, čímž narůstá režie při práci s těmito daty.



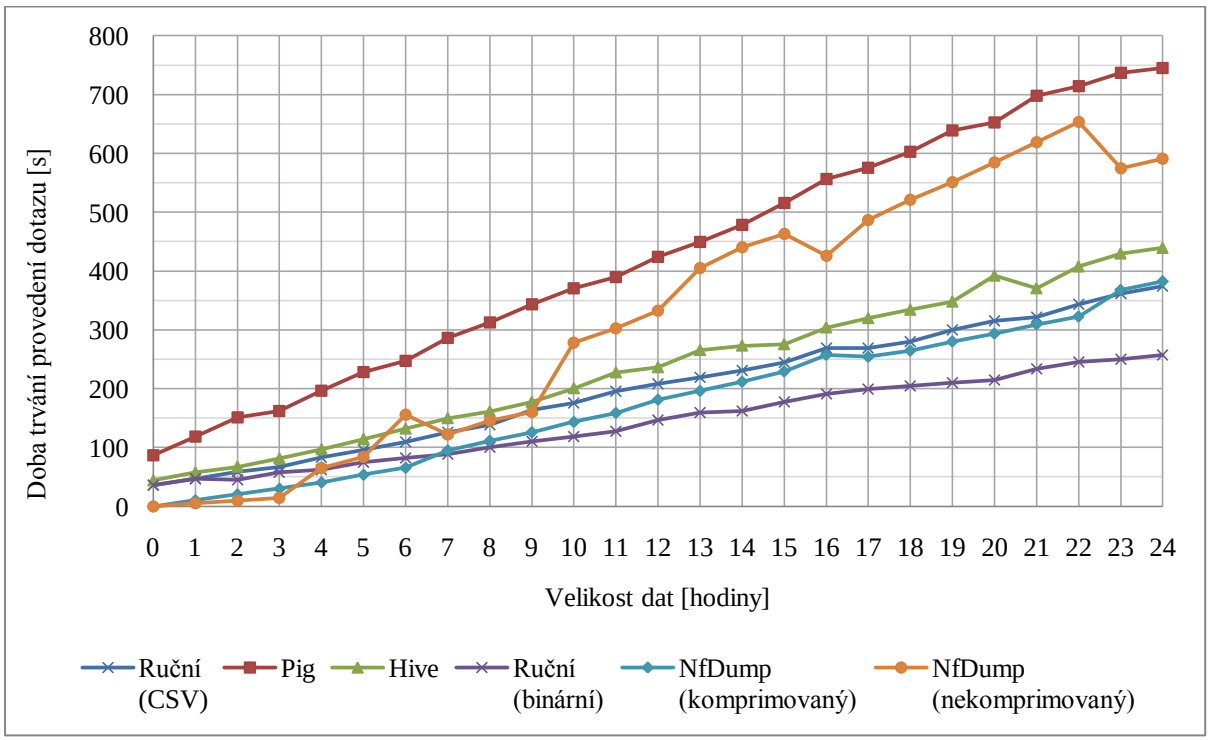
Obrázek 3-3 Dotaz č. 1: suma počtu paketů a bytů.



Obrázek 3-4 Dotaz č. 2: počet všech záznamů s cílovým portem 53.



Obrázek 3-5 Dotaz č. 3: výpis zvolených položek pro záznamy přenášené prostřednictvím protokolu TCP na portu 53.

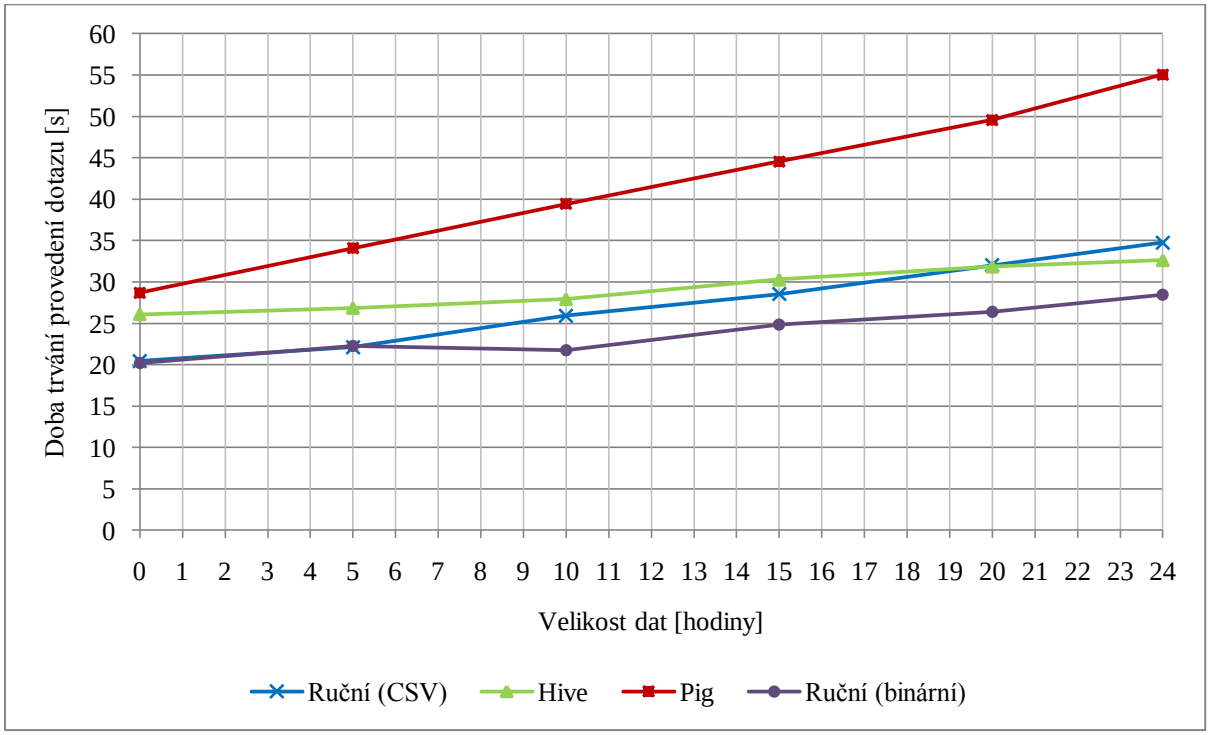


Obrázek 3-6 Dotaz č. 4: výpočet sumy paketů, bytů a celkového počtu záznamů pro každou zdrojovou adresu.

Pokročilé experimenty – optimalizace platforem založených na systému Hadoop

V následujícím textu budou v jednotlivých částech uvedeny různé parametry, s jejichž hodnotami bylo experimentováno. Výsledky těchto experimentů jsou z důvodu omezení rozsahu demonstrovány výsledky z provádění dotazu č. 2. Kompletní výsledky ze všech dotazů jsou uvedeny v příloze D tohoto dokumentu.

První graf zobrazuje výsledky měření odezvy při výchozím nastavení. V kontextu dalších měření to znamená heartbeat interval pro výměnu zpráv mezi jednotlivými uzly nastavený na 3 sekundy a replikační faktor nastaven na hodnotu 4. V rámci tohoto pokusu byl spouštěn vždy jediný dotaz samostatně v daný čas (tj. nedocházelo k paralelnímu spuštění úloh). Ve výsledcích měření je také možné vidět ve srovnání se základními experimenty výrazné zrychlení při zpracování větších objemů dat za použití většího clusteru.

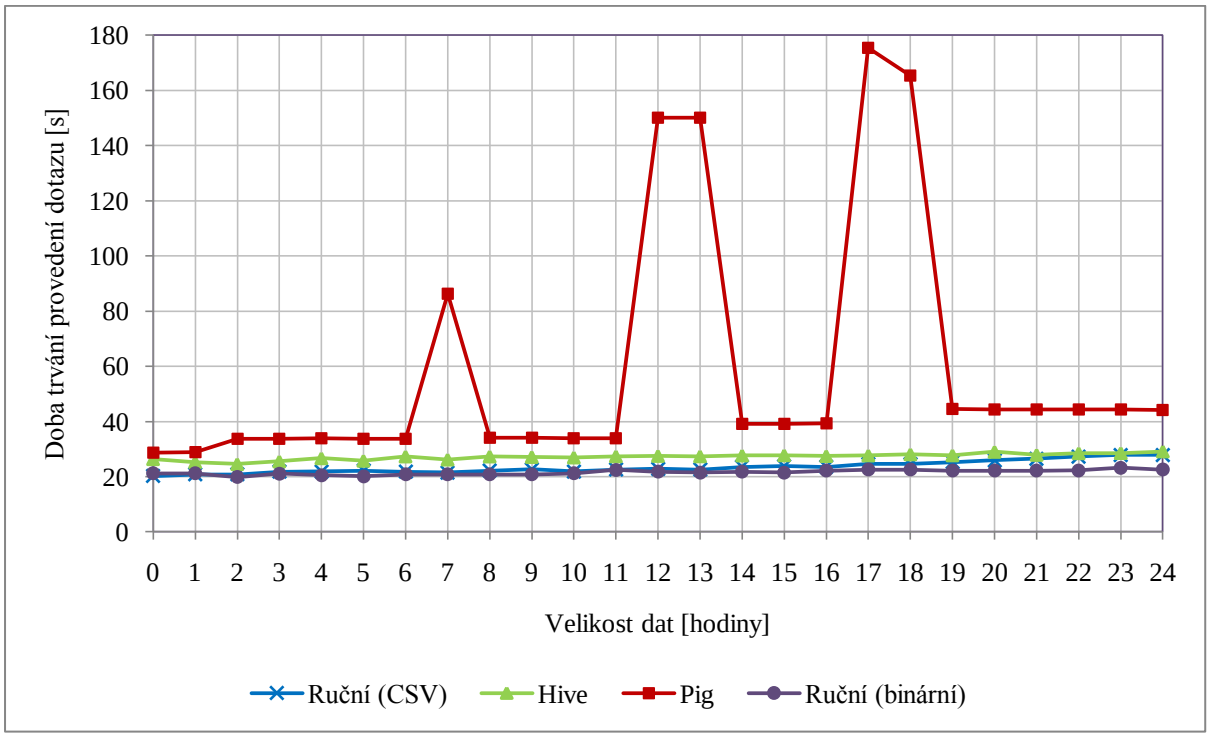


Obrázek 3-7 Dotaz z č. 2 s výchozím nastavení parametrů Hadoopu

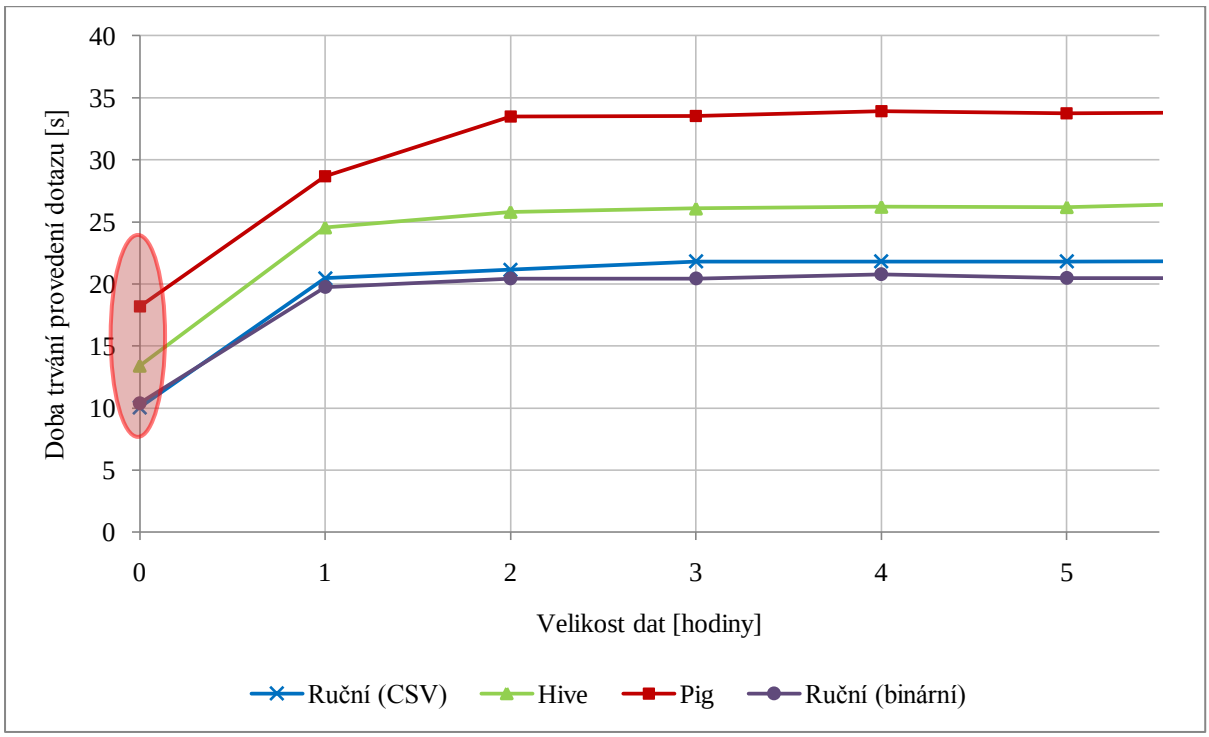
První z parametrů, na který jsem se při optimalizaci konfigurace zaměřil, byl interval výměny zpráv jednotlivých uzlů, tzv. heartbeat interval. Cílem ladění tohoto parametru bylo snížení počáteční latence při zpracování dotazů menšího až středního rozsahu. Obrázek 3-8 zobrazuje graf naměřených hodnot při nastavení heartbeat intervalu na 1 sekundu. Z výsledků je vidět, že změna tohoto parametru nepřinesla žádné zlepšení v délce odezvy provádění dotazu. Naopak přinesla spíše zhoršení spolehlivosti, které lze pozorovat u platformy Pig.

Pro dotazy pracující s menšími objemy dat byl v dalším experimentu testován tzv. *über* mód, který je poskytován Hadoopem v novější verzi právě pro úlohy menšího rozsahu, jenž je možné provést lokálně na jednom z pracovních uzlů. Ve výchozím stavu je tento mód vypnutý. Na třetím grafu jsou zobrazeny výsledky testování se spuštěným über módem. Z grafu je zobrazena pouze část s několika prvními testy, na které bylo toto testování zaměřeno nejvíce. Je zde vidět zlepšení při dotazu na nejmenší množství dat, nicméně tento dotaz pracuje pouze s malým souborem o deseti tocích, který je v experimentech zařazen spíše pro odhad režie při minimální výpočetní zátěži. Z principu činnosti über módu lze odhadnout, že přínos pro zpracování požadavku lze očekávat pouze při úlohách pracujících s daty ne většími, než je velikost bloku, protože právě taková data se mohou

vyskytovat na jednom výpočetním uzlu. Přestože über mód přináší zrychlení pro velmi malé dotazy, je počáteční prodleva relativně dlouhá a pro vyvíjený kolektor nevhodná. Dalším negativem při použití über módu, byl zvýšený výskyt chyb při zpracování dotazů prostřednictvím nástavby Hive.

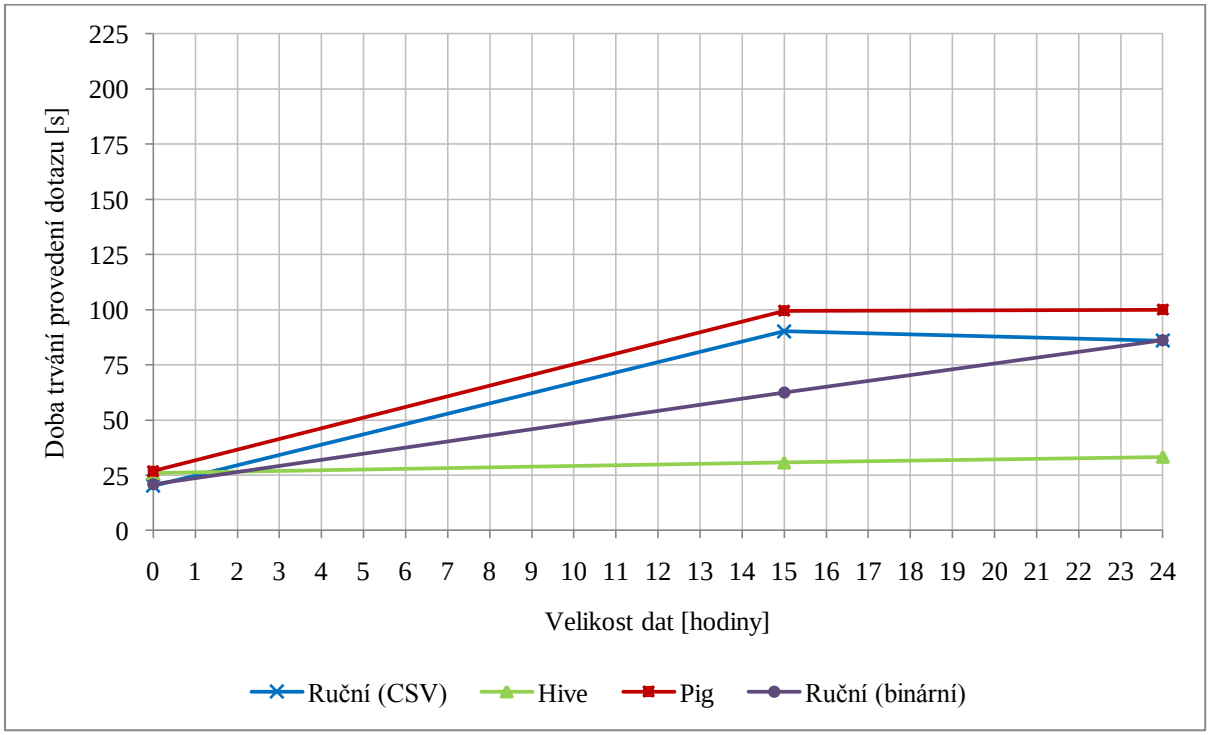


Obrázek 3-8 Dotaz č. 2 s nastavením heartbeat intervalu na 1 sekundu

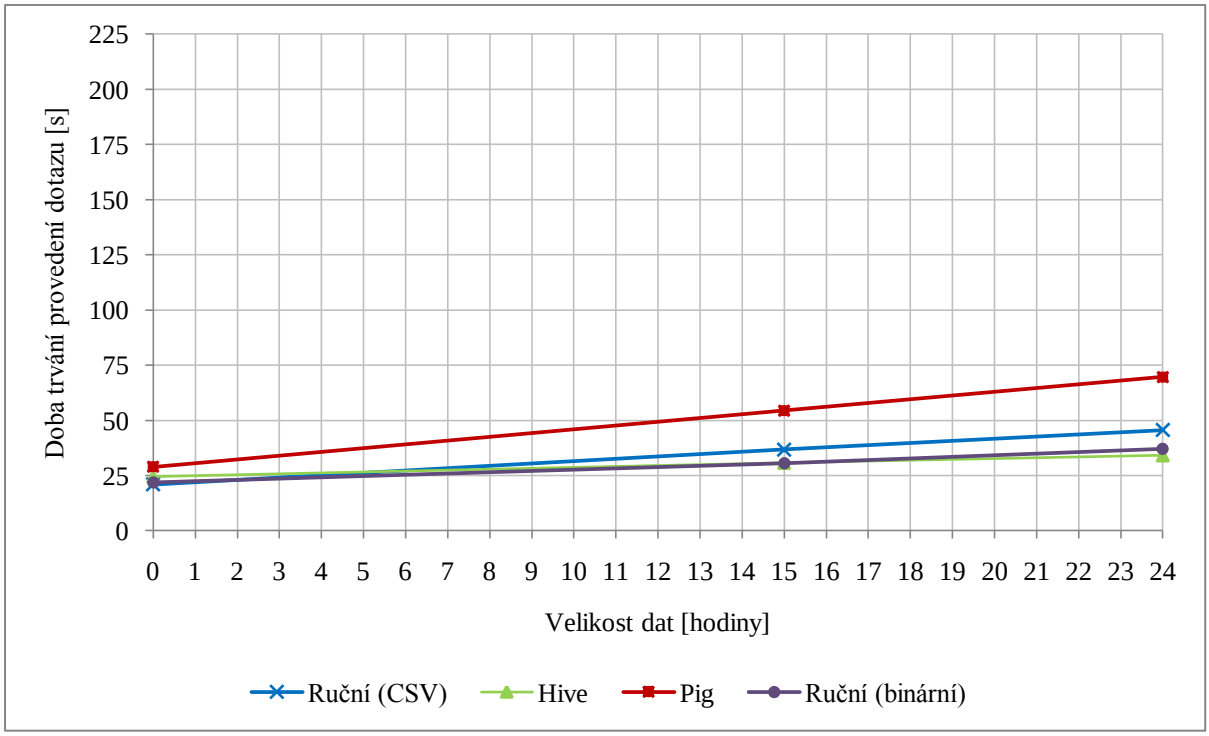


Obrázek 3-9 Dotaz č. 2 se spuštěným über módem (prvních 5 hodin dat)

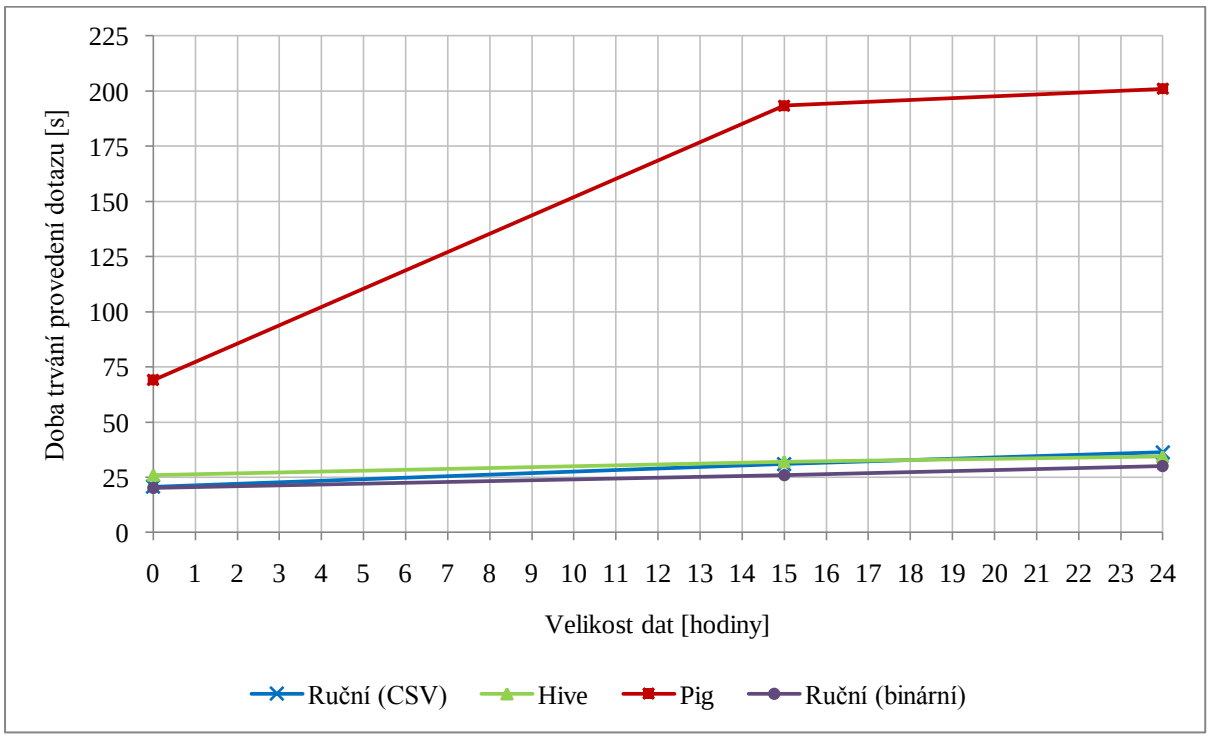
Dalším sledovaným parametrem byl replikační faktor. Jeho změna může mít vliv jednak na rychlost výpočtu, kdy přítomnost více kopií dat v systému umožňuje jemnější rozdělení dílčích úloh a lepší rozložení zátěže mezi uzly. Dále replikační faktor ovlivňuje také spolehlivost výpočtu a zálohování dat, kdy je při výpadku jednoho uzlu zajištěna přítomnost jiné kopie dat v systému. Kromě výchozí hodnoty replikačního faktoru, jenž byl roven čtyřem kopiím každého bloku dat, jsou na následujících grafech zobrazeny výsledky z pokusů pro hodnoty replikačního faktoru 1, tj. bez replikace, 2 a 6. Z výsledků je možné pozorovat výrazné zpomalení výpočtu při zrušení replikace (replikační faktor 1) a o něco mírnější zhoršení při replikačním faktoru 2. Je tomu tak kvůli neoptimálnímu rozložení zátěže, kdy může docházet k nerovnoměrnému vytížení uzlů, majících uloženu větší část aktuálně požadovaných dat a ke zvýšeným přenosům dat mezi uzly, které nemají k dispozici momentálně potřebná data. Dalším negativem těchto dvou nastavení je zhoršení zálohování dat, jenž však nebylo hlavním předmětem těchto experimentů. Ve srovnání s experimenty s výchozím nastavením Hadoopu bylo dosaženo s replikačním faktorem 6 srovnatelných výsledků z hlediska délky provádění dotazů. Vzhledem k tomu, že toto nastavení nepřináší zkrácení odezvy, lze považovat další zvyšování replikačního faktoru za nevhodné, jelikož s sebou přináší prodloužení doby uložení dat a nárůst prostoru potřebného pro uložení dat. Zlepšení zálohovacích vlastností oproti replikačnímu faktoru 4 již není zásadní. Při těchto experimentech nebyly z důvodu úspory času při dlouho trvajících testech spouštěny dotazy nad daty z každé hodiny, ale pouze nad vybranými datovými celky.



Obrázek 3-10 Dotaz č. 2 s nastavením replikačního faktoru na hodnotu 1



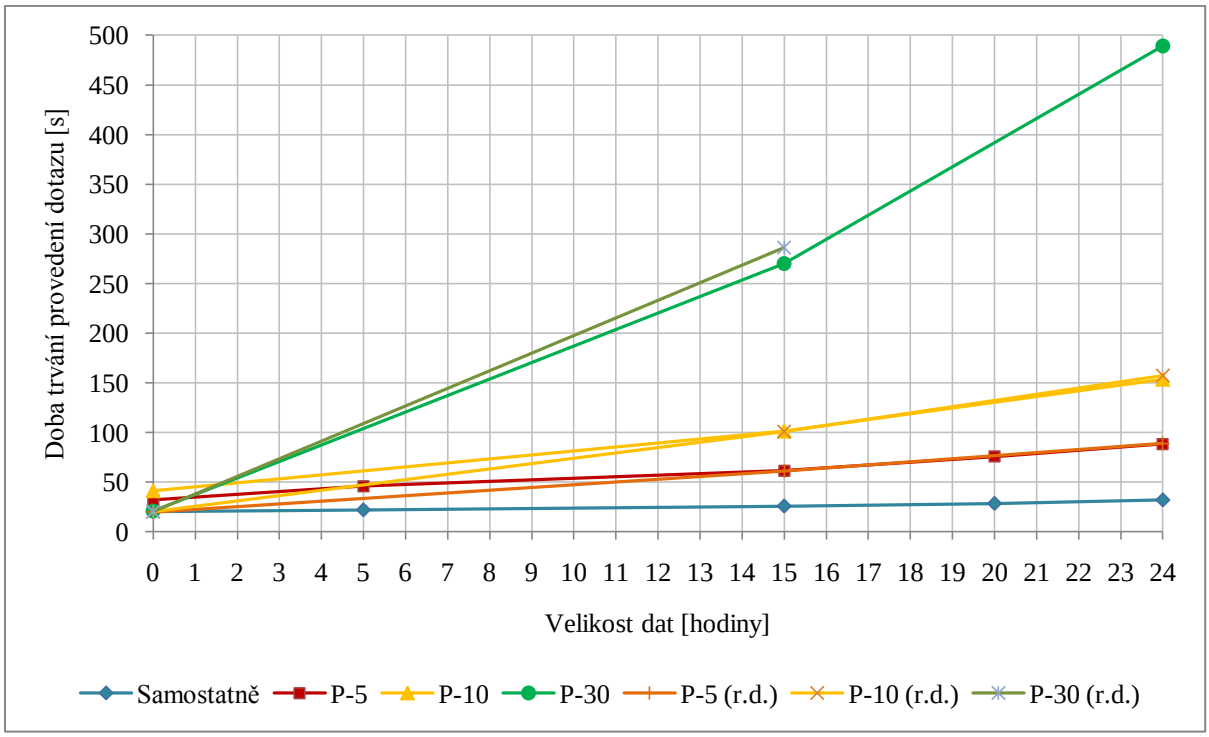
Obrázek 3-11 Dotaz č. 2 s nastavením replikačního faktoru na hodnotu 2



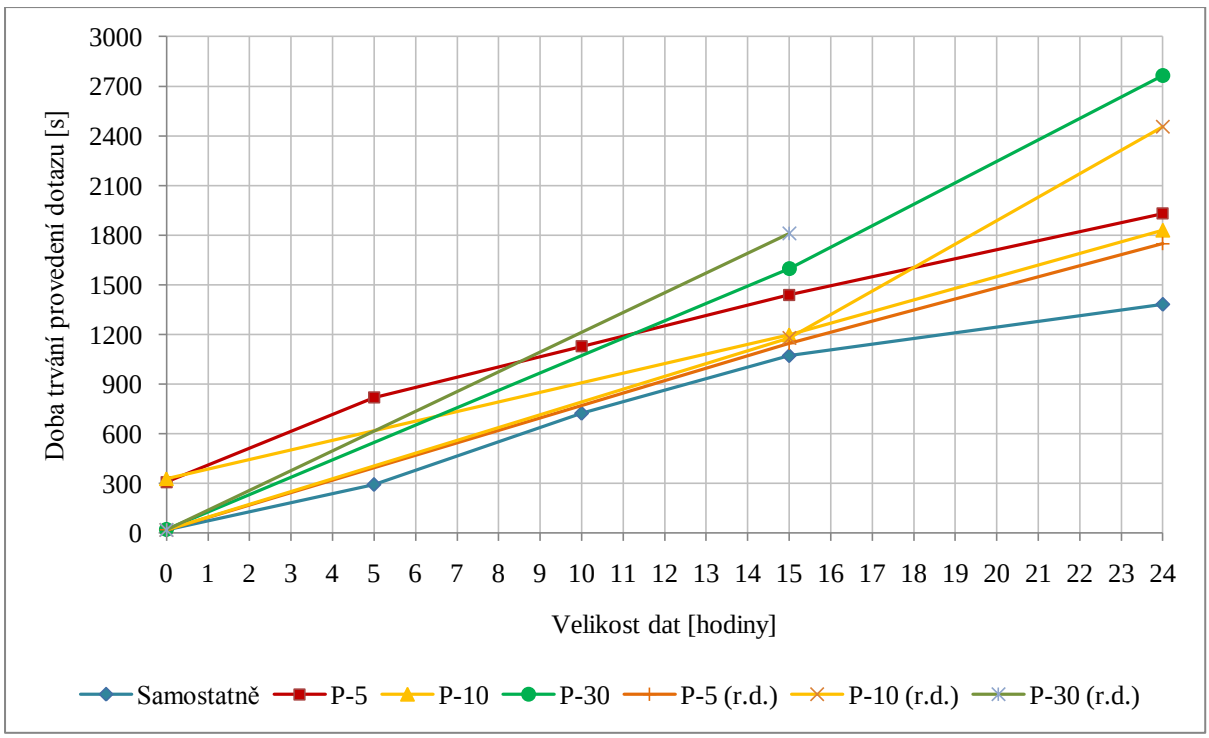
Obrázek 3-12 Dotaz č. 2 s nastavením replikačního faktoru na hodnotu 6

Dalším sledovaným aspektem testovaných systémů byla schopnost zpracování více dotazů současně. Obrázek 3-13 ukazuje vliv spuštění více paralelních úloh. Obrázek 3-14 pak zobrazuje výsledky stejného typu naměřené pro dotaz č. 4. Ty jsou zde uvedeny, protože oproti ostatním experimentům poskytují vůči dotazu č. 2 další dodatečnou informaci v podobě projevu většího přenosu dat na dobu trvání dotazů při zpracování více úloh současně. Na těchto grafech jsou zobrazeny výsledky pro spuštění jediné úlohy (referenční hodnota), pro vícenásobné spuštění dané

úlohy 5x, 10x a 30x nad stejným datovým souborem (v grafech značeno *P-5*, *P-10* a *P-30*) a stejné počty úloh pro rozdílné datové soubory, kdy byla pro každou spuštěnou úlohu vytvořena zvláštní kopie zdrojového datového souboru (hodnoty označeny navíc *(r.d.)*).

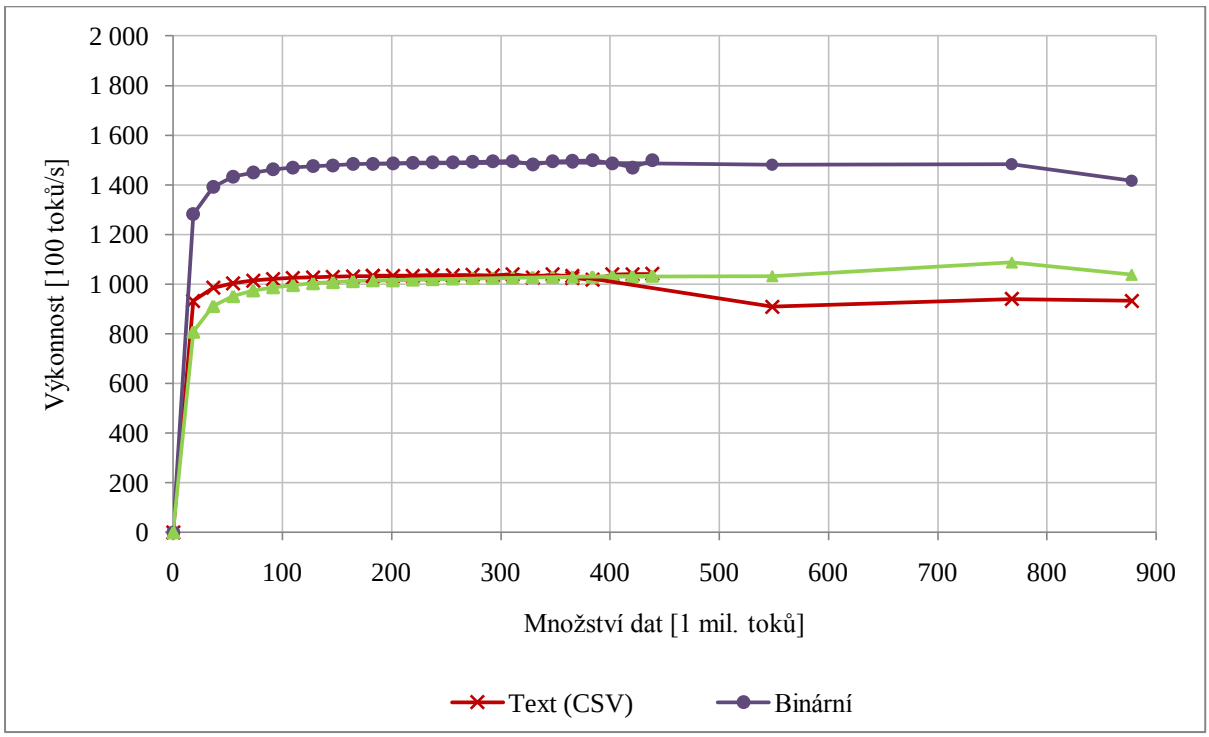


Obrázek 3-13 Dotaz č. 2 při paralelním spuštění úloh



Obrázek 3-14 Dotaz č. 4 při paralelním spuštění úloh

Jako předposlední naměřený výsledek zde uvádím výkonnost systému z pohledu ukládání dat. Obrázek 3-15 zobrazuje průběh doby trvání ukládání jednotlivých datových souborů, použitých pro výše uvedené experimenty. Z grafu lze pozorovat, že se tato doba relativně rychle ustálí na hodnotě přibližně 1,5 mil. záznamů za sekundu pro data uložená v binárním formátu a zhruba 1 mil. záznamů pro textová data či Hive formát. Poslední zkoumaný ukazatel systému Hadoop je jeho výkonnost z pohledu počtu zpracovaných toků za sekundu. Ta se v průměru ze všech typů dotazů pohybuje okolo 500 tis. toků za vteřinu na jeden uzel.



Obrázek 3-15 Doba trvání ukládání dat do HDFS

Testování platformy Spark

Po ne příliš uspokojivých výsledcích experimentů cílených na optimalizaci systému Hadoop pro zpracování dat o IP tocích jsem zkoumal další možné alternativy. Jako nejslibnější platforma se jevil Apache Spark, který je schopen pracovat mj. nad úložištěm HDFS a využívat tak výhody, které toto úložiště poskytuje (distribuované úložiště, zálohování skrze replikaci a další). Pro experimenty s tímto systémem byl zvolen dotaz č. 2, jenž byl pro platformu Spark přepsán do jazyka Scala. Při prvním spuštění vybraného dotazu však trvalo zpracování výsledku déle než dvě minuty, což bylo ve srovnání s přibližně třiceti vteřinovou odezvou Hadoopu nepřijatelné. Při dalších snahách optimalizovat tento dotaz a způsob jeho provádění jsem nebyl schopen dosáhnout doby zpracování dotazu kratší než 60 vteřin. Po dalším přezkoumání jsem došel k závěru, že největší síla platformy Spark spočívá v ukládání mezivýsledků v paměti. Tím je při specifických typech dotazů, které jsou zaměřeny na opakovanou práci se stejnými daty, dosaženo výrazného zkrácení doby zpracování daného dotazu. Experimenty s touto platformou byly následně ukončeny a v implementaci dalších dotazů pro systém Spark jsem nepokračoval.

Shrnutí poznatků

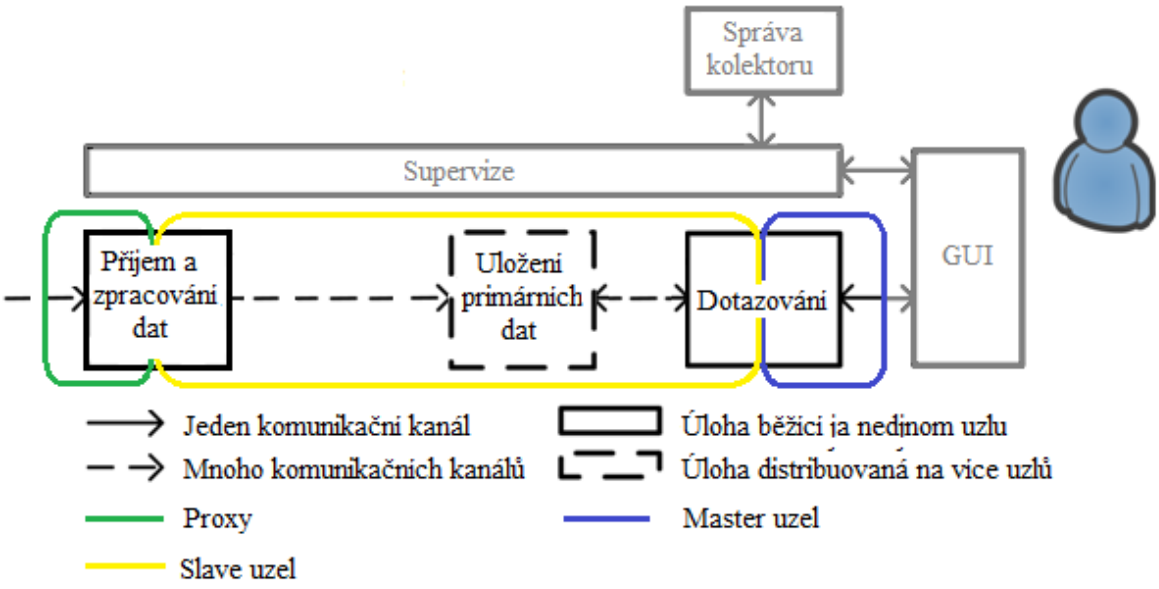
V rámci této práce byly provedeny rozsáhle experimenty se stávajícími platformami pro distribuované zpracování dat. Tyto experimenty částečně ukázaly potenciál distribuovaného

zpracování větších objemů dat. Současně také přinesly důležité poznatky o existujících platformách, určených ke zpracování masivních objemů dat s využitím počítačového clusteru. Na základě těchto poznatků se však vybraná řešení jeví spíše jako nevhodná. Důvodem je zejména režie výpočtu, která je v kontrastu dotazů malého až středního rozsahu pro použití pro dotazování ve vyvíjeném kolektoru neúnosná. Další významnou nevýhodou je relativně malá výkonnost, která zřídka přesahuje hodnotu 500 000 toků za vteřinu na jeden uzel. Pozitivem těchto systémů je řešení zálohy dat a spolehlivosti výpočtů prostřednictvím replikace datových bloků, nicméně toto jediné významné pozitivum nepřesahuje svým přínosem nedostatečnou výkonnost testovaných řešení. Další nevýhodou těchto systémů je jazyk JAVA, ve kterém jsou tyto systémy napsány. Chybějící rozhraní pro jazyk C/C++, jenž je nativní pro zdrojové kódy rozšiřovaného kolektoru IPFIXcol představují významnou překážku při propojení těchto dvou platforem.

S ohledem na dosažené výsledky jsem se nakonec rozhodl implementovat vlastní řešení, jež bude založeno na nástroji Nfdump, který je poměrně výkonný při práci s daty o IP tocích a je v oblasti analýzy síťového provozu běžně používán.

4 Návrh kolektoru

V této sekci je popsán návrh architektury vyvíjeného kolektoru, který vychází z výše uvedené specifikace. Obrázek 4-1 zobrazuje zjednodušené blokové schéma navrženého kolektoru. Zvýrazněná část schématu je hlavním předmětem této práce. Popis jednotlivých částí a rozhraní bude uvedeno v následujících kapitolách.



Obrázek 4-1 Schéma vyvíjeného kolektoru.

Hlavní vstup do kolektoru tvoří záznamy o IP tocích (primární data). Ty jsou přijímány ze zařízení, která jsou schopna tyto záznamy exportovat. Při přijetí dat je zpracován samotný protokol, použití pro přenos záznamů o IP tocích a následně jsou tato data předzpracována. Fáze předzpracování provádí především obohacení dat, asociaci dat ke zdrojům, zákazníkům a profilům a případně i k výpočtu některých hodnot časových řad. Předzpracovaná data jsou dále uložena do distribuovaného persistentního úložiště. Dále je zde poskytnuta možnost odeslání duplikátu primárních dat (např. k proudovému zpracování). Nad úložištěm primárních dat je prováděno dotazování, které může být i součástí grafického uživatelského rozhraní (GUI). Veškeré procesy v úložišti jsou monitorovány skrze supervizi, která sděluje prostřednictvím GUI aktuální stav kolektoru (celkově či pro daného zákazníka). Blok supervize dále poskytuje data a konfigurační rozhraní pro správu celého kolektoru. V rámci této práce bude řešena pouze vyznačená část, tedy příjem dat, jejich předzpracování a uložení a vrstva pro následné dotazování nad uloženými daty. Implementován pak bude pouze dotazovací framework, který se jeví z pohledu výkonnosti jako nejkritičtější. Část zabývající se předzpracováním a uložením dat bude implementována v souladu se zde uvedeným návrhem v rámci bakalářské práce Michala Kozubíka [14]. V dalším textu následuje podrobnější popis návrhu těchto částí.

4.1 Architektura distribuovaného kolektoru

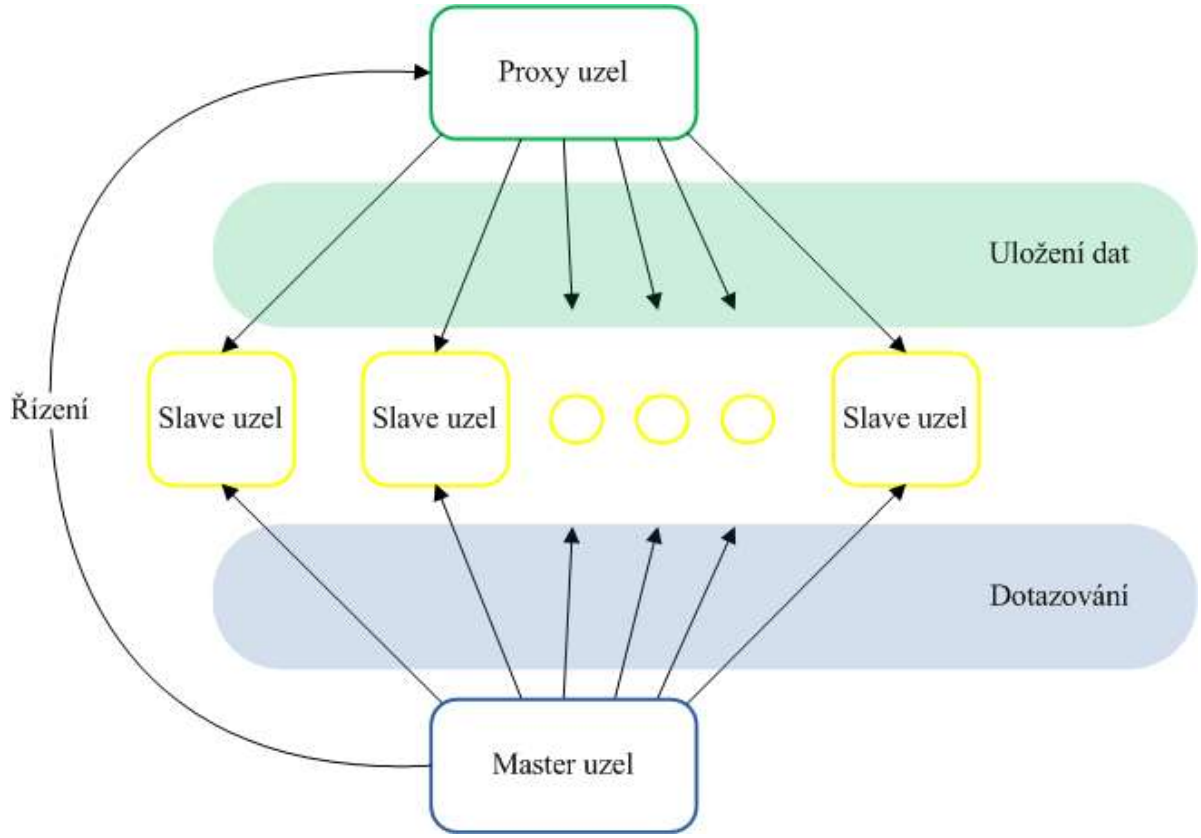
Distribuovaný kolektor se bude skládat ze tří částí:

- *Proxy* uzel jako první z těchto částí má za úkol distribuci příchozích dat. Data ze sběrných bodů jsou posílána právě na tento jediný uzel, který data dále odesílá na *slave* uzly, jež mají za úkol

sbíraná data ukládat a uchovávat. Data jsou rozesílána po jednotlivých IPFIX paketech způsobem round-robin, tj. první paket na první slave uzel, druhý paket na druhý slave uzel atd. Proxy uzlem může být samotný slave uzel, nebo hlavní *master* uzel. Implementace této části není součástí této práce, vzniká však v rámci výše uvedené bakalářské práce.

- *Master* uzel je v rámci dané instance kolektoru jediný. Tento bod spravuje jednotlivé slave uzly a má také na starost obsluhu uživatelských dotazů. Ty deleguje dále na své podřízené slave uzly, které dotaz provedou lokálně nad svými daty a výsledky pak pošlou zpět na master uzel. Ten přijaté výsledky patřičným způsobem zpracuje a předá na výstup uživateli. Master uzel je tedy jediný prvek, se kterým musí (a může) uživatel interagovat. Způsob jakým jsou dílčí výsledky získávány a spojovány dohromady bude popsán v následující kapitole, popisující výslednou implementaci.
- *Slave* uzlů je typicky více. Jak již bylo naznačeno, jejich účelem je příjem dat od proxy uzlu, jejich uchovávání a následně provádění lokálních dotazů nad těmito uzly. Jednotlivé slave uzly neví o existenci ostatních slave uzlů ani o datech, která jsou na jiných uzlech uložena.

Implementace skupin uživatelů je zde umožněna právě pomocí proxy uzlů, které jsou s podřízenými slave uzly konfigurovány do jedné skupiny. Každý slave či proxy uzel může být součástí pouze jediné takovéto skupiny. Tato konfigurace je uchovávána a spravována master uzlem, který tak má k dispozici informace o tom, kde se záznamy o IP tocích jednotlivých zákazníků nacházejí. Obrázek 4-2 zobrazuje jednoduché schéma této architekturu.



Obrázek 4-2 Schéma architektury distribuovaného kolektoru

Master uzel se může zdát jako slabé místo systému (tzv. single point of failure), pro případ jeho selhání však postačí záloha konfigurace distribuovaného kolektoru, jenž může být použita na záložním stroji pro pokračování v činnosti. Implementace záložního mechanismu však není součástí této práce.

Výhodou výše uvedeného přístupu je rovnoměrná zátěž jednotlivých slave uzlů jak při ukládání dat, tak při jejich dotazování. Další výhodou je jednoduchost přidání dalších slave uzlů do systému, kdy stačí tento uzel pouze přidat do konfigurace dané skupiny. Další výhodou je skutečnost, že dané konfigurované skupiny uzlů uchovávají data z určité oblasti, čehož lze dále využít např. při generování profilů či detekci anomálií.

Jako nevýhodu navrženého řešení je možné považovat nerovnoměrné zatížení slave uzlů v rámci odlišných skupin. Další nevýhodou je pak přítomnost jediného master uzlu, na kterém může vzniknout úzké hrdlo při zpracovávání uživatelských dotazů.

Jak již bylo naznačeno v předchozím textu, součástí této práce je implementace komponenty, která bude sloužit jako dotazovací vrstva, v dalším textu nazývaná DistDump. Tato aplikace bude nutně tvořena dvěma částmi pro role master a slave uzlů. Komunikace mezi těmito částmi bude zprostředkována pomocí knihovny libtrap, s využitím formátu zpráv UniRec. Čtení lokálních dat na slave uzlech a jejich zpracování bude umožněno knihovnou libnf. Popis těchto komponent je uveden v následujících podkapitolách.

4.2 Příjem a předzpracování dat

Na vstupu kolektoru je především potřeba zpracovat různé protokoly, pomocí kterých jsou záznamy o tocích přenášeny. Jedná se zde o protokoly rodiny NetFlow, IPFIX a NSEL. Přenos těchto protokolů bude zajištěn v otevřené podobě přes UDP či TCP. Spolehlivé TCP spojení navíc umožní přenos dat šifrovat s využitím TLS. Jednotlivým protokolům bude kolektor naslouchat na předem definovaných portech. Zpracování bude probíhat distribuovaně na několika uzlech, které budou vyhrazeny pro tento účel. Každý zdroj dat bude zasílat své záznamy o IP tocích vždy jen na jeden přijímající proxy uzel. Distribuce zátěže mezi jednotlivé vstupní uzly bude realizována prostřednictvím proxy uzlu. Každý slave uzel pak přijímá pouze poměrnou část dat z dané podmnožiny.

Příchozí záznamy o tocích budou moci být rozšířeny o některé další položky. Prvními z nich budou data, rozšiřující informace o IP adresách. Za tímto účelem bude udržována a pravidelně aktualizována databáze informací o IP adresách. Tato databáze bude využívána pro obohacení záznamu o geolokaci IP adresy a číslo autonomního systému. Dalšími rozšiřujícími položkami budou data, popisující identitu uživatelů. Pro tuto funkcionalitu bude využito informací získaných z DHCP a podobných mechanismů v prostředí IPv6. Protokolem pro získávání těchto dat bude syslog.

Hlavním výstupem fáze příjmu a předzpracování dat jsou záznamy o tocích ve vnitřní reprezentaci kolektoru. Tato data budou následně ukládána do úložiště. Množina nejčastěji používaných položek vychází z konfigurace reálně používaného kolektoru IPFIXcol[15], dostupná v [8]. Tato množina položek záznamu není konečná a úložiště musí počítat s budoucím možným rozšířením množiny těchto položek. Výstupní rozhraní přijímající části kolektoru bude data přímo zapisovat do distribuovaného úložiště primárních dat. Konkrétní typ rozhraní bude reflektovat nativní vstupní rozhraní úložiště (např. zápis do souboru ve formátu JSON). Mimo to zde bude také rozhraní administrativní, které umožní předávání některých parametrů, načtení a modifikaci konfigurace a reportování vytíženosti vstupní části či chybových stavů.

Příjem a zpracování dat bude implementováno pomocí stávajícího softwarého kolektoru IPFIXcol. Tento software poskytuje základní funkcionalitu pro příjem a zpracování protokolů pro

[15] https://www.liberouter.org/ipfixcol/

export záznamů o IP tocích. IPFIXcol bude dále rozšiřován o funkcionalitu, potřebnou pro splnění požadavků na vyvíjený kolektor. Výsledný software pak bude tvořit ucelený proces, který poběží v 1 až N instancích v rámci distribuovaného kolektoru na vstupních uzlech.

IPFIXcol pracuje následujícím způsobem. Na vstupu dochází ke zpracování transportního protokolu a protokolu pro export záznamů o tocích pomocí vstupních zásuvných modulů, tzv. pluginů. Přijímané pakety se záznamy jsou převáděny do interní struktury, odpovídající IPFIX paketu. Tento vytvořený paket dále prochází mediačními pluginy, které upravují či rozšiřují samotný paket či metadata, asociovaná k paketu. Následně je paket zpracováván výstupními pluginy, jenž mají na starost přenos paketů na výstupní rozhraní ve formátu daného rozhraní. Pro potřeby splnění výše uvedené specifikace bude nutné rozšířit stávající IPFIXcol software o další funkcionalitu, jejíž popis následuje.

Ke každému internímu IPFIX paketu budou přiřazeny informace, rozšiřující stávající informace o příchozím paketu a o záznamech toků. Informace vztahující se k celému záznamu budou rozšířeny o časovou značku příchodu paketu na kolektor. Dále budou jednotlivé záznamy obohaceny o tyto údaje:

- odkaz na záznam v IPFIX paketu,
- identifikátor zdrojového autonomního systému (dále AS),
- identifikátor cílového AS,
- identifikátor zdrojové země,
- identifikátor cílové země,
- zdrojová identita,
- cílová identita,
- odkaz na informace k zákazníkům a profilům.

Pro obohacení záznamů o *identifikátor země* (angl. country code) a *číslo AS* bude použita knihovna, umožňující tyto hodnoty získat bez nutnosti dotazovat se externích databází. Zdrojová IP adresa v záznamu bude vyhledána v knihovně a zdrojový identifikátor země, resp. číslo zdrojového AS jsou uloženy do identifikátoru zdrojové země, resp. čísla zdrojového AS. Analogicky se vyhledání a uložení provede pro cílovou IP adresu.

Zdrojová a cílová identita uživatele je reprezentována řetězcem o 32 znacích. Identita uživatele bude spravována v databázi běžící mimo část pro příjem a zpracování primárních dat. Části pro příjem a dotazování se budou při zpracování každého záznamu dotazovat přes databázové rozhraní této externí databáze. Obohacení záznamu o identitu bude řízeno příslušností záznamu k danému zákazníkovi a číslu pravidla zdroje. Každá databáze bude spravovat identitu uživatelů právě pro jednoho zákazníka. Na základě příslušnosti záznamu k zákazníkovi bude prohledána daná databáze. Dle příznaku náležícího k identifikátoru pravidla zdroje bude prohledána identita všech IP adres nebo pouze IP adres veřejných. Pokud záznam patří ke dvěma zákazníkům, tzn. zákazník A komunikuje se zákazníkem B, pak musí být zachováno soukromí druhého zákazníka, tj. záznam ukládán do úložiště pro zákazníka A bude obsahovat pouze identifikovaného uživatele ze sítě zákazníka A, stejné omezení platí i pro ukládání záznamu v případě zákazníka B.

Poslední rozšiřující položka, která odkazuje na *informace k zákazníkům a profilům*, adresuje následující dodatečné položky:

- identifikátor zákazníka (číslo),
- identifikátor pravidla zdroje (číslo),
- odkaz na zdrojovou identitu,
- odkaz na cílovou identitu,

- odkaz do seznamu profilů.

Seznam profilů obsahuje identifikátory profilů, kterým daný záznam o IP tocích odpovídá. Pravidla, určující příslušnost záznamu k zákazníkům, jsou umístěna do seznamu po načtení konfigurace. Na pořadí pravidel v seznamu nezáleží, při vyhledávání záznamů je seznam procházen sekvenčně, dokud není dosaženo jeho konce. V případě, kdy pro daného zákazníka existuje více pravidel a bylo již nalezeno pravidlo přiřazující záznam danému zákazníkovi, je možné následující pravidla tohoto zákazníka přeskočit. Každé pravidlo se skládá z následujících částí:

- Predikát definující konjunkci IP adresy IPFIX paketu, cílového čísla portu IPFIX paketu, observer domain ID (ODID), IP prefixu a MPLS značek (MPLS1-4), VLAN tagů (VLAN1-2), zdrojové a cílové MAC adresy. IP adresa IPFIX paketu je povinný prvek, ostatní jsou volitelné. Dle zdrojové IP adresy IPFIX paketu lze následně optimalizovat vyhledávání.
- Identifikátor zákazníka,
- identifikátor pravidla zdroje,
- příznak pro identitu.

Při prohledávání seznamu pravidel vzniká seznam trojic, jež obsahují identifikátor zákazníka, pravidla zdroje a příznaky určující identitu. Zároveň jsou během vyhledávání vyřazovány duplicitní výsledky, tj. výsledky, pro které v seznamu již existuje trojice se shodným identifikátorem zákazníka. Na základě výsledků vyhledání příslušnosti záznamu k zákazníkovi jsou prohledány seznamy profilů. Každý zákazník má přiřazen seznam pravidel profilů. Tato pravidla se skládají z:

- identifikátoru profilu a
- filtračního predikátu, který odpovídá vyjadřovací silou filtrační podmínce programu Nfdump.

Pokud záznam odpovídá filtračnímu predikátu, pak je identifikátor profilu uložen do seznamu identifikátorů profilů, který je odkazován z informací o zákaznících a profilech. Pokud tedy shrnu výše uvedené, každý záznam může náležet několika zákazníkům a v rámci každého zákazníka libovolnému počtu profilů.

Na výstupním rozhraní směrem do úložiště dat bude pole identifikátorů zákazníků a profilů využito pro oddělení dat pro jednotlivé zákazníky (pokud data patří více zákazníkům, tak budou při zápisu duplikována). Dále bude informace o profilech využita pro další duplikaci dat při zápisu, pokud bude záznam odpovídat více profilům. Časová značka paketu bude sloužit pro označení množiny záznamů, které v daném časovém okamžiku budou přijaty na kolektoru. Volitelně může tato časová značka rozšířit samotný záznam o toku.

4.3 Uložení primárních dat

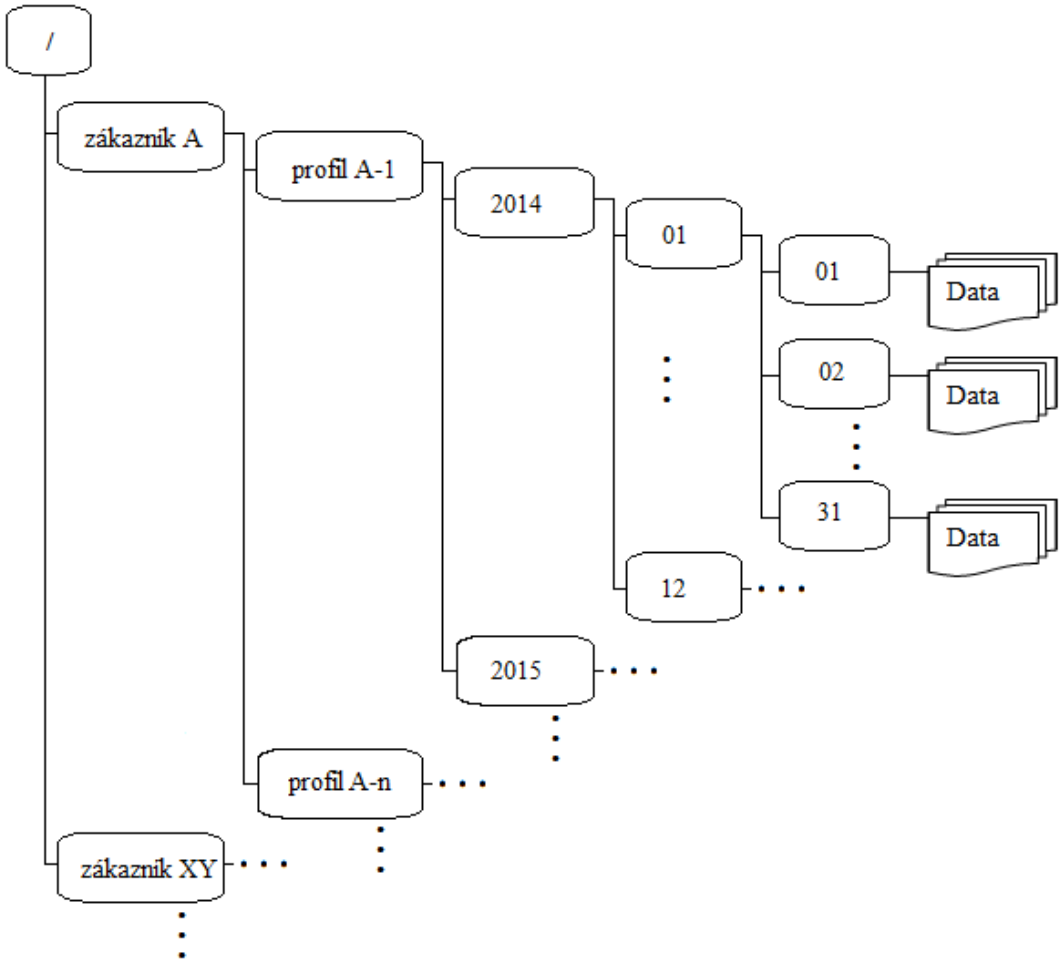
Vstup pro část uložení primárních dat poskytuje zápisové rozhraní pro přímé ukládání záznamů do úložiště bez dalších mezivrstev. Zápis záznamů skrze toto zápisové rozhraní je řešen již na výstupu fáze příjmu a předzpracování dat. Zápisové rozhraní bude umožňovat oddělit ukládání dat od různých zákazníků a v rámci profilů.

Výstupní rozhraní bude poskytovat možnost tvorby, spuštění a ukončení dotazovacích požadavků z GUI a pro předání výsledků dotazů do GUI. V následujícím textu je popsána navržená adresářová struktura a formát souborů, v nichž budou ukládány přijaté záznamy o tocích.

Adresářová struktura začíná na úrovni zákazníka. Každý adresář daného zákazníka dále obsahuje složky, odpovídající jednotlivým profilům tohoto zákazníka. V další struktuře jsou data rozdělena podle roku, měsíce a dne. V adresářích odpovídajícím jednotlivým dnům jsou pak samotná

data o tocích uložena podle nastavitelné granularity (např. po 5 minutách, po 1 hodině apod.). Obrázek 4-3 obsahuje schéma shrnující tuto strukturu.

Návrh formátu ukládaných souborů vychází z binárního formátu, používaného nástrojem Nfdump. Tento formát byl rozšířen o možnost práce s dynamickými položkami. Každý takovýto soubor začíná *hlavičkou souboru*. Za tou následuje *záznam se statistikami* o souboru a dále již samotné *datové bloky*, které obsahují jednotlivé *záznamy* o tocích. Obrázek 4-4 zobrazuje strukturu souboru, z pohledu této nejvyšší úrovně. Jednotlivé bloky budou popsány v následujícím textu podrobněji.



Obrázek 4-3 Adresářová struktura úložiště dat.

Hlavička souboru	Záznam se statistikami	Datový blok1	Datový blok 2	...	Datový blok n

Obrázek 4-4 Schéma datového souboru (Nfdump).

Hlavička souboru

Hlavička souboru určuje obecné nastavení souboru jako je identifikace souboru, počet datových bloků, informace o použití komprese apod. Tyto informace jsou obsaženy v následujících položkách hlavičky:

- Speciální hodnota, která jednak určuje, že se jedná o požadovaný typ souboru, jednak udává endianitu daného systému – hodnota musí být přečtena jako 0xA50C.
- Verze binárního souboru. Tato položka umožňuje do budoucna manipulovat s formátem souboru a poznačení této skutečnosti odlišnou verzí.
- Pole příznaků, jehož hodnoty určují následující vlastnosti souboru:
    - soubor je komprimovaný (hodnota hexadecimálně zapsaná jako 0x1),
    - soubor obsahuje anonymizovaná data (hodnota 0x2),
    - příznak udává, že za hlavičkou souboru následuje záznam se statistikami (hodnota 0x4).
- Počet datových bloků. Určuje počet datových bloků, které následují za záznamem se statistikami. Tyto bloky pak obsahují samotné záznamy o tocích.
- Identifikátor souboru, což je řetězec znaků pevně dané délky.

Za touto hlavičkou již následuje záznam se statistikami o souboru.

Záznam se statistikami

Tento záznam obsahuje souhrnné statistiky o záznamech o tocích, uložených v tomto souboru. Záznam je nejprve uveden hlavičkou, obsahující tyto dva údaje:

- typ záznamu, který udává, že následuje záznam se statistikami,
- velikost záznamu se statistikami bez této hlavičky.

Položky samotného záznamu jsou pak následující:

- celkový počet záznamů o tocích,
- celkový počet bytů v uložených záznamech,
- celkový počet paketů v uložených záznamech,
- počet záznamů o tocích komunikujících po TCP,
- počet záznamů o tocích komunikujících po UDP,
- počet záznamů o tocích komunikujících po ICMP,
- počet záznamů o tocích komunikujících pomocí jiného transportního protokolu,
- počet bytů přenesených pomocí protokolu TCP,
- počet bytů přenesených pomocí protokolu UDP,
- počet bytů přenesených pomocí protokolu ICMP,
- počet bytů přenesených pomocí jiného transportního protokolu,
- počet paketů přenesených pomocí protokolu TCP,
- počet paketů přenesených pomocí protokolu UDP,
- počet paketů přenesených pomocí protokolu ICMP,
- počet paketů přenesených pomocí jiného transportního protokolu,
- časová značka prvního uloženého toku ve vteřinách,
- časová značka posledního uloženého toku ve vteřinách,
- část milisekund časové značky prvního uloženého záznamu,
- část milisekund časové značky posledního uloženého záznamu.

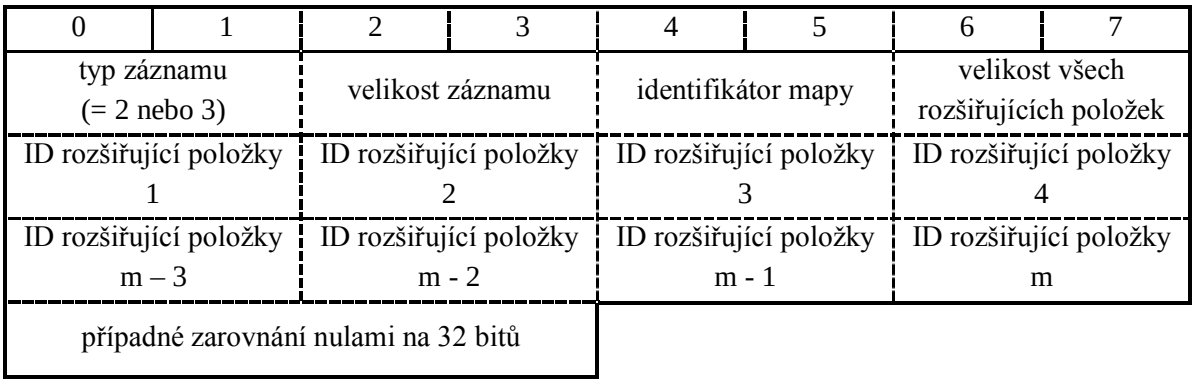
Za tímto záznamem se statistikami již následují samotné datové bloky.

Blok dat

Bloky dat obsahují jednotlivé záznamy o tocích, respektive tyto datové záznamy a dodatečné záznamy popisující jejich strukturu. Tyto strukturní záznamy se nazývají *mapy rozšíření* (angl. *extension maps*). Jedná se o seznam položek – *rozšíření* – které se v záznamu nachází. Mapu rozšíření lze tedy chápat jako šablonu daného záznamu. Oba tyto typy záznamů se mohou v bloku dat objevovat různě, musí ale platit, že každá mapa rozšíření předchází záznamu, který se na tuto mapu odkazuje. Typicky pak platí, že na začátku prvního bloku dat jsou uloženy záznamy všech doposud zaznamenaných map rozšíření. Pokud je zaznamenána nová mapa, uloží se ihned její záznam a v dalším souboru bude již tato mapa zaznamenána spolu s ostatními na začátku prvního bloku. Mapy rozšíření mohou být dvojího druhu, jenž se strukturálně neliší. Odlišují se však položky, na které se odkazují. První typ položek je stejný, jako v případě reprezentace Nfdump. Druhý typ položek je určen pro dynamická pole v záznamech o tocích. Tyto položky nenesou pevnou strukturu, namísto které musí udávat svoji velikost. Z implementačního hlediska to pak znamená, že rozšiřující položky dynamické velikosti budou ve struktuře, popisující tyto položky, obsahovat ukazatel do paměti a velikost dané položky.

Typ záznamu, obsahující mapu rozšíření (pro položky statické i variabilní velikosti), je složen z těchto položek:

- typ záznamu, který obsahuje hodnotu 2 pro položky statické velikosti a hodnotu 3 pro položky s proměnnou délkou,
- velikost této mapy,
- identifikátor mapy, pomocí kterého se záznamy o tocích na tuto mapu odkazují,
- velikost všech rozšiřujících položek, uvedených v této mapě,
- dále následuje seznam identifikátorů jednotlivých rozšiřujících položek, které definují strukturu záznamu.



Obrázek 4-5 Složení záznamu mapy rozšíření, popisující strukturu záznamu toku.

Typ záznamu obsahující samotná data o tocích pak obsahuje tyto položky, které je z každého IP toku možné vždy určit (buď dle hodnoty, nebo z principu považovat za prázdné):

- typ záznamu, jenž zde obsahuje hodnotu 1,
- velikost záznamu,
- příznaky,
- tag pro volitelné označení záznamu,
- identifikátor odkazující se na příslušnou mapu rozšíření pro položky statické velikosti,
- část milisekund časové značky prvního toku,
- část milisekund časové značky posledního toku,

- časovou značku prvního toku ve vteřinách,
- časovou značku posledního toku ve vteřinách,
- forwarding status (položka Netflow v9, id 89),
- TCP příznaky,
- protokol,
- typ služby na zdrojovém rozhraní (dle Netflow v9 položka src ToS, id 5),
- zdrojový port
- cílový port nebo typ paketu ICMP (v závislosti na transportním protokolu).
- identifikátor odkazující se na příslušnou mapu rozšíření pro položky s proměnnou délkou (0 pokud záznam neobsahuje dynamické položky),
- Dále již následují samotná data, jejichž strukturu popisují výše uvedené mapy příznaků.

Kromě této povinné struktury musí každý záznam navíc obsahovat rozšiřující položky zdrojové a cílové IP adresy, počtu paketů a počtu bytů.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="2">typ záznamu (= 1)</td><td colspan="2">velikost záznamu</td><td>příznaky</td><td>tag</td><td colspan="2">mapa rozšíření (statické položky)</td></tr>
<tr><td colspan="2">milisekundy (příchod prvního toku)</td><td colspan="2">milisekundy (příchod posledního toku)</td><td colspan="4">časová značka příchodu prvního toku (s)</td></tr>
<tr><td colspan="4">časová značka příchodu posledního toku (s)</td><td>fwd status</td><td>TCP příznaky</td><td>protokol</td><td>zdrojový TOS</td></tr>
<tr><td colspan="2">zdrojový port</td><td colspan="2">cílový port / ICMP</td><td colspan="2">mapa rozšíření (dynamické položky)</td><td colspan="2">data ...</td></tr>
</table>

Obrázek 4-6 Složení záznamu IP toku.

4.4 Dotazování

Poslední částí vyvíjeného kolektoru je část pro dotazování, která má za úkol poskytnout uživateli rozhraní, jenž bude svojí funkcionalitou a vlastnostmi odpovídat výše uvedené specifikaci. Tato fáze se jeví z pohledu zpracování dat jako nejkritičtější, jelikož je v této fázi zapotřebí zpracovávat velké objemy dat s co nejmenší dobou odezvy. Implementace této části je také předmětem této práce a její popis bude uveden v následující kapitole.

Z pohledu návrhu bude dotazování rozděleno do dvou částí, jednak se bude jednat o dotazování nad primárními daty, jednak bude dotazování probíhat nad daty sekundárními. Z pohledu uživatele se však bude jednat o jednotné rozhraní. To, na kterou databázi bude dotaz směřován, bude určeno prostřednictvím GUI. Dotazovací vrstva tedy obdrží dotaz na jednu z databází.V rámci této fáze je řešena část dotazování nad primárními daty. Dotazy určené pro primární data budou zapisovány v Nfdump formátu. Dotazy směrované na sekundární data budou v GUI zadávána prostřednictvím předdefinovaných možností, jež jsou dále reprezentována v SQL jazyce jako příslušné části SQL dotazu.

Zadané dotazy je nejprve potřeba rozebrat a zkontrolovat jejich platnost z pohledu syntaxe dotazu. Výsledkem tohoto rozboru bude dotaz, připravený ke spuštění, nebo chybové hlášení s určením syntaktické chyby. Pro primární data budou dotazy zadávány ve formátu shodným s programem Nfdump jednak proto, že se tato podoba jeví jako vhodná a dostačující pro analýzu dat

o IP tocích, jednak také proto, že je tento přístup běžně používán a bude uživatelům blízký. Data, nad kterými má být dotaz proveden jsou určena daným zákazníkem, příslušným profilem, nad kterým je dotaz spouštěn a samozřejmě parametry zadaného dotazu. Zde je nutné zachovat hierarchii profilů a dodržet postupné aplikování profilů na zdrojová data. V případě reálných profilů se jedná o přístup do příslušného adresáře. Pokud se ale jedná o profily virtuální (stínové), musí být pravidla tohoto profilu načtena a aplikována na data nadřazená reálného profilu, resp. výsledků aplikace nadřazeného stínového profilu. Při dotazování na sekundární úložiště určí dotazovací vrstva data, vztahující se k danému uživateli a nad nimi pak sestaví samotný SQL dotaz.

Jak již bylo uvedeno v úvodu této kapitoly, dotazovací část se bude skládat ze dvou částí. Na master uzlu bude spouštěna master část vyvíjeného programu. Tomuto programu bude prostřednictvím parametrů předáno nastavení daného dotazu. Spuštěný program ověří zadané parametry a v případě chyby ohlásí tuto skutečnost uživateli a je následně ukončen. V případě úspěchu vytvoří master část inicializační zprávu s nastavením dotazu a po připojení všech slave uzlů odešle tuto zprávu připojeným uzlům pro zpracování. Slave část programu je spuštěna na jednotlivých výpočetních uzlech stále a běží na pozadí jako démon systému. Po přijetí inicializační zprávy s dotazem provedou slave uzly kontrolu přítomnosti dat. V případě nenalezených dat, zadaných uživatelem při spouštění dotazu, oznámí slave uzel tuto skutečnost master uzlu, který v případě nepřítomnosti všech požadovaných dat skončí s chybou. V případě úspěchu odešlou všechny slave uzly master části zprávu informující o úspěšném přijetí dotazu a s velikostí zpracovávaných dat, kterou master uzel vypíše pro poskytnutí odhadu doby trvání dotazu. Následně začnou jednotlivé slave uzly zpracovávat data s cílem získání lokálních mezivýsledků, zatímco master uzel očekává na svých vstupních rozhraních záznamy s výsledky.

V případě zpracování dotazu, požadujícího pouze výpis záznamů jsou tyto záznamy vypisovány v požadovaném rozsahu round-robin způsobem, jenž je očekáván i ve fázi ukládání dat. Pokud je požadována agregace záznamů nebo top-n statistiky, musí master uzel nejprve přijmout data ze všech uzlů, spojit obdržené mezivýsledky a následně vypsat výsledky z těchto spojených záznamů. Při tomto typu dotazů je předpokládáno omezení výstupních záznamů. Dle tohoto omezení je na slave uzlech spočítáno nutné množství záznamů, které je potřeba odeslat, aby byl výsledek korektní. Toto omezení je ve výchozím stavu nastavené na předem určenou hodnotu a v případě potřeby je možné jej vypnout nastavením na hodnotu 0.

Následující implementace nebude obsahovat všechny pokročilé funkce, jako je dynamické sledování stavu výpočtu apod. Implementované řešení bude směřovat svojí funkcionalitou k síle nástroje nfdump, nicméně její hlavní účel je poskytnout nástroj, se kterým je možné provádět základní operace pro analýzu síťových dat a zjistit výkonnostní potenciál vyvinutého řešení. Rozšíření této implementace pak bude předmětem další práce.

5 Implementace

Po vyhodnocení zjištěných poznatků a provedených experimentů jsem se v rámci této práce rozhodnul implementovat vlastní řešení. Důvodem proč nebyla vybrána ani jedna z testovaných platforem, je zejména dlouhá doba odezvy dotazů menšího a středně velkého rozsahu (tj. data pohybující se svým rozsah v rámci 24 hodin). Za touto dlouho odezvou stojí především režie systému Hadoop, jenž je alespoň z části základem všech testovaných nástrojů. Minimální doba této režie se pohybuje okolo 20 sekund a je dána způsobem komunikace mezi jednotlivými uzly systému Hadoop. V současných aplikacích Hadoopu nebývá zpravidla tato režie problém, jelikož se její poměr vůči dotazovaným datům rychle zmenšuje s nárůstem objemu zpracovávaných dat. Pro potřeby kolektoru, vyvíjeného v rámci této práce, je však potřeba uvažovat jak dotazy menšího rozsahu, které se budou využívat při každodenní práci, tak i méně časté velmi rozsáhlé dotazy, jejichž cílem je spíše analýza dlouhodobého chování toku dat na počítačové síti.

Vzhledem k těmto překážkám stávajících řešení pro zpracovávání masivních objemů dat jsem se rozhodl přejít k vlastní implementaci distribuovaného dotazovacího nástroje, který bude vycházet ze stávajícího nástroje Nfdump, určeného pro lokální dotazování nad daty o IP tocích. Implementovaný nástroj si neklade za cíl poskytnout plnou sílu nástroje Nfdump, ale spíše ukázat výkonnostní potenciál distribuovaného zpracování oproti lokálnímu zpracovávání dat. Vyvíjený nástroj pro distribuované zpracovávání dat bude dále nazýván DistDump.

Implementovaná komponenta DistDump, která bude sloužit jako dotazovací vrstva vyvíjeného kolektoru, bude nutně tvořena dvěma částmi pro role master a slave uzlů. Jako implementační jazyk byl zvolen jazyk C. Pro komunikaci mezi jednotlivými uzly byla vybrána knihovna libtrap[16], v kombinaci s formátem zasílání zpráv UniRec[17], jež budou blíže popsány v následujících sekcích. Spojení těchto dvou komponent poskytuje rozhraní pro spolehlivou komunikaci po síti spolu s univerzálním formátem přenášených dat. Platforma libtrap dále poskytuje důležité funkcionality, potřebné pro vznikající nástroj, mezi které patří zejména automatické opětovné navázání komunikace při dočasném výpadku spojení či optimalizace přenosu větších objemů dat s využitím odesílacího zásobníku (bufferu). Dalším prvkem pro vytvoření nástroje DistDump je volně dostupná knihovna libnf, jenž poskytuje rozhraní v jazyce C pro práci se soubory, uloženými ve formátu Nfdump. Stručný popis této knihovny je také uveden v následujícím textu. Kromě popisu využitých existujících komponent bude v následujícím textu dále popsána poskytovaná funkcionalita a způsob komunikace mezi jednotlivými uzly.

5.1 Knihovny třetích stran

V této podkapitole jsou stručně popsány knihovny, jež byly při implementaci nástroje DistDump použity. Jedná se zejména o komunikační platformu Trap, zpřístupněnou prostřednictvím knihovny libtrap. Dále s ní úzce spojený formát zasílaných zpráv UniRec a nakonec knihovna libnf pro práci se soubory uloženými ve formátu nfdump.

[16] Knihovna Traffic Analysis Platform (https://www.liberouter.org/technologies/nemea/)
[17] Unified Record (https://www.liberouter.org/technologies/nemea/)

5.1.1 LibTrap

TRAP (traffic analysis platform) zpřístupněná knihovnou libtrap je platforma, poskytující rozhraní pro komunikaci a výměnu dat. Pro přenos informací nabízí několik různých rozhraní. Pro účely implementace nástroje DistDump bylo použito rozhraní TCP soketu, které realizuje spolehlivý přenos informací počítačovou sítí. Při odesílání zpráv nabízí knihovna libtrap blokující i neblokující přístup. Data jsou odesílána, jakmile jsou k dispozici a nedochází tak tedy ke zbytečným prodlevám.

Platforma libtrap je vhodná také pro přenos větších objemů dat. Pro podporu dosažení vysoké propustnosti obsahuje buffer, jenž snižuje poměr režie vůči přenášeným datům spojením více požadavků na přenos dat, které poté zpracuje naráz. Tento buffer je vhodné použít při přesunu velkého množství dat, při posílání ojedinělých zpráv (např. příkaz) je však vhodné buffer vypnout a poslat danou zprávu ihned, což knihovna umožňuje. Implementace bufferu s sebou přináší také ovládací prvky, kterými jsou funkcionalita automatického odeslání neplného bufferu (tzv. autoflush), jež zamezuje stárnutí dat a možnost nastavení časového intervalu, po který se má na odeslání či přijetí dat čekat (tzv. timeout).

O vzájemné spojení uzlů se knihovna stará automaticky. Další důležitou poskytovanou funkcí je opětovné navázání spojení uzlů při dočasném výpadku. Jelikož je v rámci vyvíjeného nástroje DistDump nutnost přijetí všech dat, je komunikace prostřednictvím knihovny libtrap realizována vždy v blokujícím režimu. Datového bufferu je pak využito ve směru od slave uzlů k master uzlu, kdy je předpokládán přenos záznamů o IP tocích, jež jsou výsledkem daného dotazu. V opačném směru, kterým jsou přesouvány příkazy, je buffer vypnut. Menší nevýhodou této knihovny je jednosměrnost komunikačních kanálů, kdy je pro obousměrný přenos dat nutno využít dvou rozhraní v opačných směrech. [13]

5.1.2 UniRec

UniRec (unified record) je datový formát zpráv, navržený pro přenos pomocí trap rozhraní. Tento formát podporuje různé druhy zpráv, jejichž formát může být volitelně definován. Umožňuje začlenit do správy jak položky pevné velikosti, tak i položky jejichž velikost se může dynamicky měnit. Další výhodou UniRec formátu je jeho efektivnost z pohledu paměti, kdy se zpráva v tomto formátu nezabírá téměř žádnou další paměť ve srovnání s obyčejnou strukturou jazyka C. Implementace UniRec formátu také poskytuje mechanismus pro rychlý přístup k jednotlivým položkám zprávy, čímž také odpadá nutnost ručního rozkládání a zpracování položek dotazu.

Každá zpráva v UniRec formátu sestává z volitelných položek. Každá tato položka má své unikátní jméno a datový typ. Sada těchto položek pak tvoří šablonu dané správy. Pro dané Trap rozhraní je pak použita vždy právě jedna šablona, která musí být shodná na obou stranách komunikačního kanálu. Pro potřeby nástroje DistDump byly vytvořeny dvě nové definice šablon, popsané v dalším textu. [13]

5.1.3 Libnf

Libnf je volně dostupná knihovna pro práci se soubory ve formátu Nfdump. Poskytuje funkce a datové typy pro čtení uložených záznamů o tocích, jejich agregaci a filtraci. Dále umožňuje omezení pracovní množiny položek daného záznamu, čímž činí práci s daty efektivnější. Tato knihovna je využita zejména na straně slave uzlů, nicméně pro agregaci záznamů jsou její struktury a

funkce z části použity i na straně master uzlu. Na rozdíl od předchozích dvou komponent byly zdrojové kódy knihovny libnf upraveny pro práci v distribuovaném prostředí.

5.2 Funkcionalita DistDump nástroje

Při implementaci nástroje DistDump jsem vycházel z funkcionality a rozhraní programu Nfdump. Důvodem pro tento přístup je to, že je nástroj Nfdump již ověřený řadou uživatelů a je vhodný pro analýzu a dotazování se nad daty o IP tocích. Současně volba shodného rozhraní vychází z rozšířenosti programu Nfdump a jejím cílem je usnadnit případný přechod uživatelům, kteří jsou již na rozhraní Nfdumpu zvyklí.

V rámci této práce byla implementována základní funkcionalita pro analýzu uložených dat s rozhraním shodným s nástrojem Nfdump. Podporované druhy dotazů budou popsány v následujícím textu. Konkrétní použití programu DistDump lze pak nalézt v příloze C nebo v souboru README na disku se zdrojovými kódy programu.

Základním druhem dotazu je výpis záznamů, jenž budu dále nazývat jako dotaz typu *list-flows*. Pro tento dotaz může být volitelně vybrána agregace podle některé z položek záznamu. Druhým typem je dotaz na statistiky o provozu, kdy program vypíše prvních N záznamů podle požadované agregace a řazení. Tento druh dotazu budu v dalším textu nazývat jako *top-n* dotaz. V aktuální verzi podporuje DistDump následující položky pro agregované dotazy list-flows či top-n statistiky:

- proto – IP protokol
- srcip/dstip – zdrojová/cílová IP adresa
- srcport/dstport – zdrojový/cílový port
- srcas/dstas – zdrojové/cílové číslo autonomního systému
- inif/outif – číslo vstupního/výstupního rozhraní
- srcmask/dstmask – zdrojová/cílová maska sítě
- srcvlan/dstvlan – číslo zdrojové/cílové VLAN
- insrcmac/indstmac – vstupní zdrojová/cílová MAC adresa
- outsrcmac/outdstmac – výstupní zdrojová/cílová MAC adresa
- srctos/dsttos – zdrojové/cílové číslo identifikátoru služby (Type of Service)

Dále může být pro výsledný výpis zvoleno řazení podle následujících položek:

- flows – počet toků (výchozí)
- packets – počet paketů
- bytes – počet bytů
- bps – spočítaná hodnota bytů za sekundu
- pps – spočítaná hodnota paketů za sekundu
- bpp – spočítaná hodnota bytů na paket

Ve všech typech dotazů lze stanovit limit počtu řádků výpisu, v následujícím textu bude tento limit označován *N*. Podle tohoto limitu je také řízeno zpracování dotazu, s cílem zpracování a zejména přenosu co nejmenšího nutného počtu záznamů. V případě omezení základní varianty list-flows dotazu bez agregace přečte každý ze slave uzlů *N* záznamů a všechny tyto záznamy pošle master uzlu. Každý uzel musí zpracovat všech *N* záznamů, protože v nejhorším případě mohou být požadovaná data pouze na jediném z uzlů, což je zjištěno až na master prvku ve fázi spojování přijatých mezivýsledků. Master uzel pak ze sesbíraných záznamů vypíše právě *N* položek.

Pokud je požadována agregace nebo je zadán dotaz typu top-n, musí každý slave uzel zpracovat všechny záznamy z požadovaného rozsahu (ze všech zadaných souborů), ze kterých vytvoří lokální mezivýsledek agregovaných záznamů či statistik. Při obdržení požadavku na tento typ dotazu získá každý ze slave uzlů také celkový počet uzlů, které se aktuálně podílejí na zpracování dotazu. Dle tohoto počtu je určena hodnota *Z*. Při výpočtu prahu *Z* je přečtena hodnota *N*-té položky (hodnota dle které je požadováno řazení záznamů) a následně je vydělena počtem zpracovávajících uzlů (hodnotou obdrženou v prvotní zprávě, zadávající požadavek dotazu), tuto hodnotu označme *Y*. Následně jsou projity všechny položky záznamů za *N*-tým záznamem a je spočten počet všech řádků lokálního mezivýsledku, které mají hodnotu dané položky větší než *Y*. Počet těchto záznamů + *N* udává celkový práh *Z*. Prvních *Z* záznamů je následně odesláno na master uzel, kde je ze všech top-*Z* mezivýsledků složen celkový výsledek a na výstup vráceno prvních *N* záznamů. Hodnota *Z* se bude typicky na každém slave uzlu lišit podle hodnoty lokální *N*-té položky. Za cenu přenosu určitého množství „zbytečných“ dat je však dosaženo korektního výsledku (celkový top-*N*) s minimální dodatečnou režií výpočtu.

K jakémukoliv typu dotazu může být také volitelně přidán filtrační výraz, kterým je možné vybrat z množiny vyhodnocovaných záznamů pouze konkrétní záznamy (např. konkrétní adresu, port apod.). Syntaxe filtračního výrazu odpovídá také syntaxi používané u nástroje Nfdump.

5.3 Protokol komunikace

V následující sekci bude uveden popis UniRec šablon, které definují podobu správ, vyměňovaných mezi master a slave uzly. Dále zde bude uveden jednoduchý komunikační protokol, prostřednictvím něhož si obě komunikující strany vyměňují informace.

Na výstupní rozhraní master uzlu se připojují všechny zapojené slave uzly, čímž vytvoří servisní komunikační kanál, pomocí kterého master uzel zadává a může řídit průběh vyhodnocení daného dotazu. Zprávy posílané tímto kanálem odpovídají UniRec šabloně DD_COMMAND, jenž se skládá s následujících položek:

- DD_CODE – kód dané zprávy, určující její význam a může nabývat následujících hodnot:
  - DD_M_CMD_INIT – značí počáteční zprávu obsahující nastavení dotazu
  - DD_M_CMD_FAILED – zpráva informující o neúspěchu prováděného dotazu
  - DD_M_CMD_DONE – zpráva značící úspěšné dokončení vyhodnocení dotazu
- DD_REQUEST_TYPE – určuje typ zadaného dotazu, hodnotou může být jeden z:
  - DD_REQ_LIST_FLOWS – požadavek na dotaz typu list-flows
  - DD_REQ_TOP_N – požadavek na dotaz typu top-n
- DD_SLAVE_CNT – počet slave uzlů aktuálně určených k vyhodnocení daného top-n dotazu
- DD_SEND_CNT – limit záznamů pro zpracování či odeslání (dle typu dotazu)
- DD_AGGREG – zadaný agregační výraz
- DD_DATA_PATH – výraz definující data ke zpracování
- DD_FILTER – výraz filtračního pravidla
- DD_ORDERBY – výraz určující řazení záznamů

Obrázek 5-1 zobrazuje schéma této šablony spolu s použitými datovými typy. Pro každý datový kanál používá master uzel jedno vstupní rozhraní, jenž je spojeno s výstupním rozhraním daného slave uzlu. Tímto kanálem pak proudí zejména samotné datové záznamy spolu s případnými řídicími

a stavovými informacemi ze strany slave uzlu. UniRec šablona DD_DATA na tomto rozhraní obsahuje tyto položky:

- DD_CODE – kód dané zprávy, určující její význam, jenž může nabývat následujících hodnot:
  - DD_ S_CMD_ACCEPT – značí přijetí zadaného dotazu
  - DD_S_CMD_FAILED – zpráva informující o neúspěchu prováděného dotazu
  - DD_S_DATA – zpráva obsahující jeden datový záznam
  - DD_S_STATISTICS – záznam s výslednými statistikami (počet zpracovaných dat apod.), posílaný zpravidla po odeslání všech datových záznamů
  - DD_S_ALL_SEND – zpráva informující o odeslání všech informací z daného slave uzlu
- DD_DATA_RECORD – položka obsahující data, náležící k danému typu zprávy

0	1	2	3
DD_CODE (uint16_t)		DD_REQUEST_TYPE (uint8_t)	SCD_SLAVE_CNT… (uint16_t)
…SCD_SLAVE_CNT (uint16_t)	SCD_SEND_CNT (uint16_t)		DD_AGGREG (char *) – *d.v.*
DD_AGGREG (char *) – *d.v.*	DD_ DATA_PATH (char *) – *d.v.*	DD_ FILTER (char *) – *d.v.*.	DD_ ORDERBY (char *) – *d.v.*

Příznak *d.v.* značí dynamickou velikost položky

Obrázek 5-1 UniRec šablona určující formát zprávy servisního kanálu

0	1	2
DD_CODE (uint16_t)		DD_DATA (char *) – *d.v.*

Příznak *d. v.* značí dynamickou velikost položky

Obrázek 5-2 UniRec šablona určující formát zprávy datového kanálu

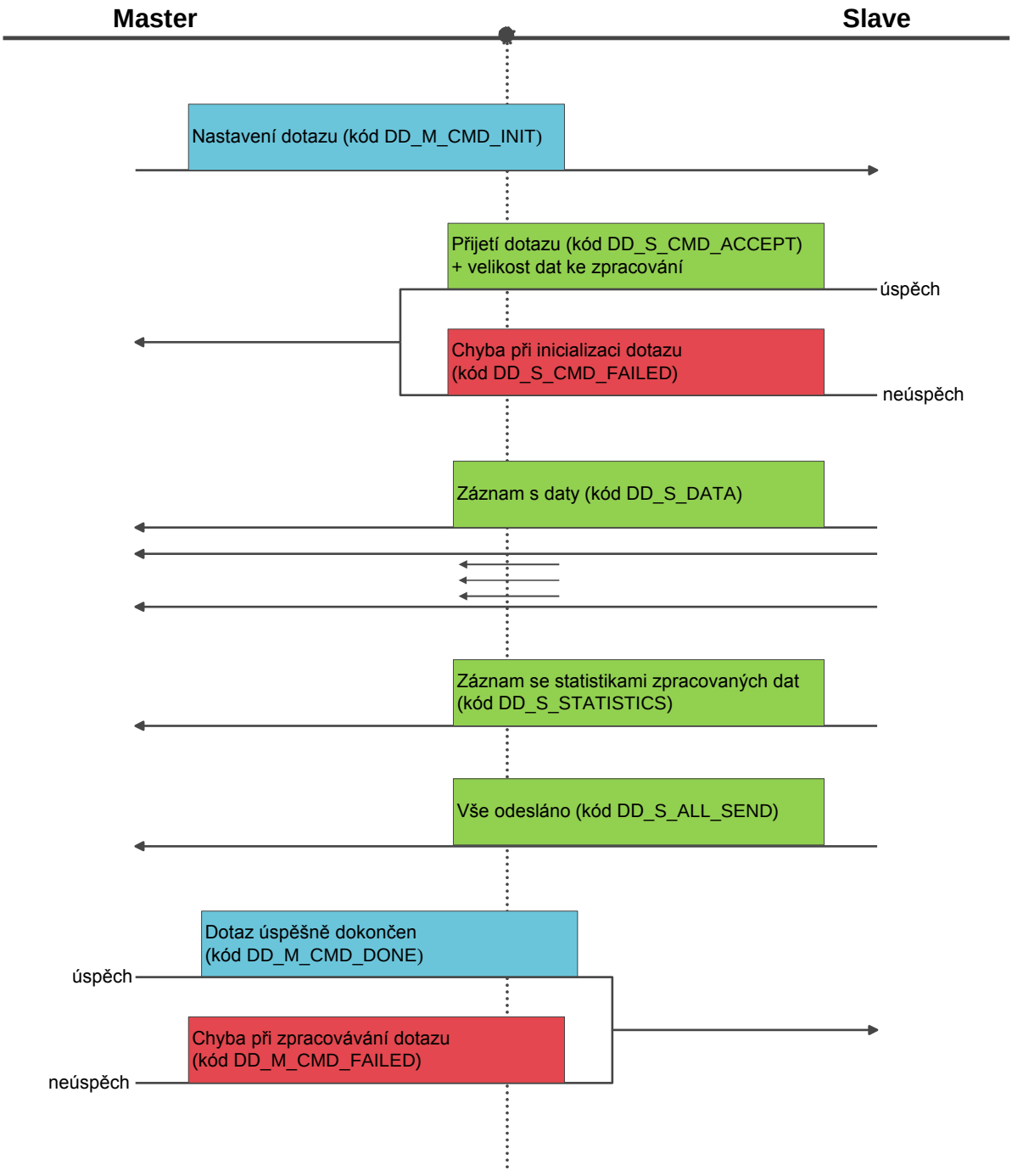
Obrázek 5-2 zachycuje strukturu a datové typy této šablony. Obrázek 5-3 zobrazuje jednoduchý komunikační protokol, prostřednictvím kterého jsou vyměňovány informace mezi master a slave uzly. V první fázi master uzel zpracuje parametry příkazového řádku, určující nastavení daného dotazu. Pokud je dotaz syntakticky nesprávný, je program ukončen s chybou a nedojde k žádné komunikaci se slave uzly. V opačném případě je vyplněna počáteční zpráva, master uzel vyčká na připojení všech slave uzlů a následně odešle všem těmto uzlům vyplněnou inicializační zprávu. Slave uzly tuto zprávu zpracují, provedou inicializaci potřebných struktur a ověří dostupnost požadovaných dat. V případě úspěchu odešlou master uzlu zprávu s kódem DD_ S_CMD_ACCEPT, obsahující v datové části počet bytů souborů ke zpracování, na základě kterého může master uzel provést odhad doby trvání dotazu. V případě neúspěchu odesílá slave uzel zprávu s kódem DD_S_CMD_FAILED. Součástí zprávy o úspěchu je i velikost dat ke zpracování v bytech. Tato informace je uživateli následně zprostředkována master uzlem pro odhad doby trvání dotazu. Pří implementaci vyhledávání požadovaných dat je brána v potaz možnost dodatečného přidávání uzlů. Tyto nově přidané uzly nebudou mít k dispozici datové soubory z doby před přidáním. V případě absence takovýchto souborů informuje slave uzel o této skutečnosti, nicméně pokračuje v provádění dotazu dále. Pokud dojde

k situaci, že ani jeden z uzlů nemá k dispozici požadovaná data, je zpracování dotazu následně ukončeno s chybou.

Pokud proběhlo zpracování počáteční zprávy úspěšně, začnou slave uzly zpracovávat zadaný dotaz. Master uzel v této fázi naslouchá na svých vstupních rozhraních a očekává data od slave uzlů. Je-li zadán dotaz typu list-flows bez agregace, jsou záznamy vypisovány hned po příchodu round-robin přístupem, jenž je předpokládán i ve fázi ukládání dat. To znamená, že je nejprve vypsán jeden záznam přijatý od slave uzlu č. 1, dále jeden záznam od slave uzlu č. 2 atd. Pokud některý z uzlů odeslal již všechna data, je tento uzel v dalším kole vynechán.

Pro dotazy typu top-n a list-flows s agregací jsou přijaty a současně spojovány všechny záznamy. Teprve až po získání dat od všech slave uzlů a jejich výslednému vyhodnocení je předán na výstup celkový výsledek dotazu. Po odeslání všech datových záznamů odesílá každý ze slave uzlů jeden záznam se statistikami o zpracovaných datech. Tyto údaje jsou na master uzlu také spojeny a připojeny na konec výstupu provedeného dotazu.

Po zprávě se statistikami je odeslána zpráva s prázdnou datovou částí typu DD_S_ALL_SEND, která momentálně nemá vyšší praktický význam (ukončení by mohlo být signalizováno i zprávou se statistikami), je zde však z důvodu ponechání prostoru pro další možná rozšíření. Po obdržení této zprávy již master uzel neočekává žádné další zprávy od daného slave uzlu. Pokud získal data od všech uzlů. Provede případné výsledné zpracování daného dotazu a vypíše hlavičku záznamů, následovanou samotnými daty. Výstup pak ukončuje souhrnná informace, obsahující počet zpracovaných dat a čas potřebný pro zpracování dotazu. Následně je slave uzlu zaslána zpráva potvrzující úspěšné dokončení dotazu, komunikující rozhraní jsou odpojena a aplikace na straně master uzlu je ukončena. Slave uzly po obdržení této zprávy uvolní alokované prostředky a přejdou znovu do počáteční fáze naslouchání a čekání na inicializační zprávu.

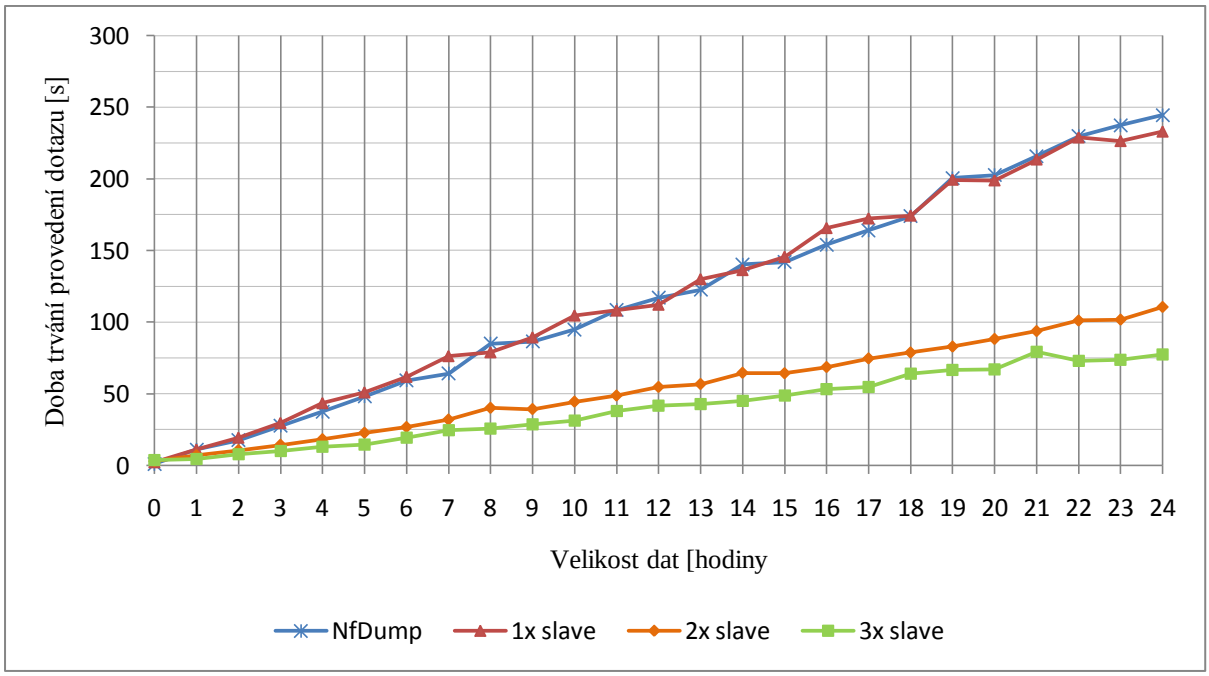


Obrázek 5-3 Komunikační protokol master a slave částí nástroje DistDump

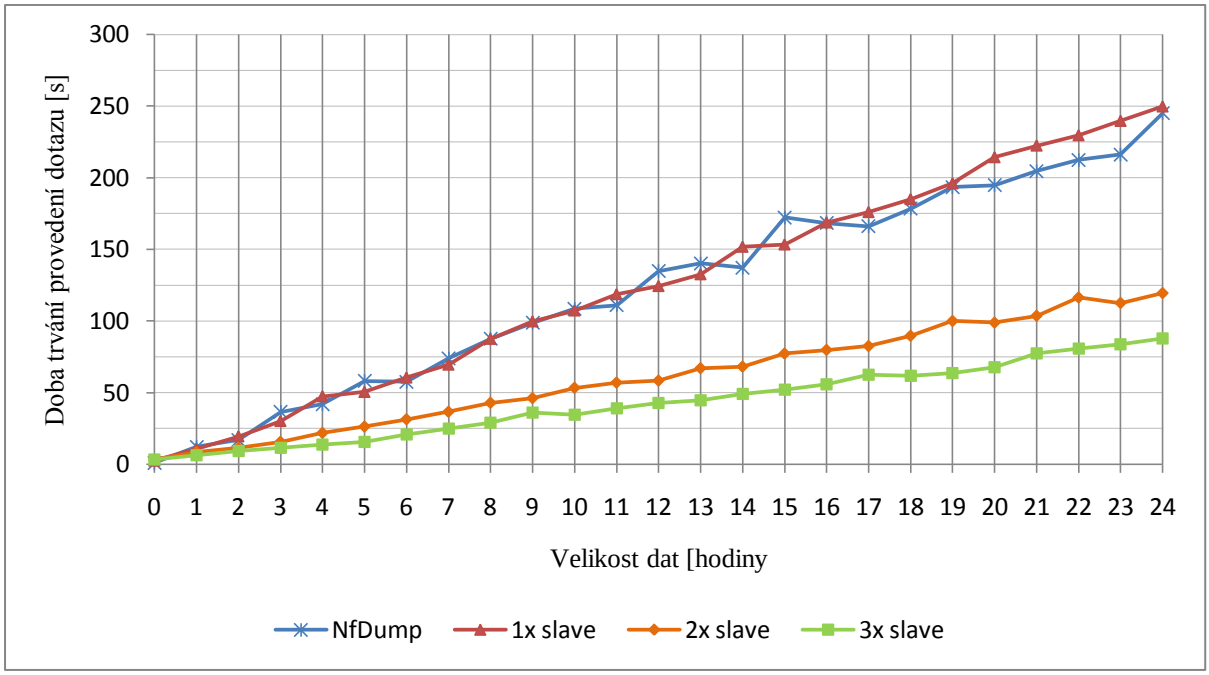
6 Vyhodnocení implementace

Pro experimenty s implementovanou aplikací mi byly poskytnuty 4 stroje se čtyřjádrovými procesory a 2GB paměti. Náplní experimentů bylo stejné testování jako u experimentů uvedených v kapitole 3.3, tj. stejné čtyři dotazy nad anonymizovanými daty odstupňovanými po jedné hodině. Tyto dotazy byly spuštěny lokálně nad nástrojem Nfdump, dále v konfiguraci 1 master s 1, 2 a 3 slave uzly nad vytvořeným programem DistDump. Výsledky těchto experimentů jsou uvedeny v následujícím textu a grafech.

Na prvních dvou grafech lze pozorovat dobu zpracování dotazu, který prochází všemi zadanými datovými soubory. Ze zobrazené doby trvání dotazů lze pozorovat, že přidání filtru k danému dotazu nemá na dobu provádění zásadní vliv. Časově nejnáročnější operací je čtení záznamů z disku, které je v obou dotazech prováděno ve stejném rozsahu. Dále je možné v grafech vidět horší průběh škálovatelnosti, kdy přínos třetího slave uzlu oproti druhému je značně menší, než přínos přidání druhého slave uzlu k samostatnému uzlu. Při zjišťování důvodu tohoto zhoršení škálovatelnosti se jako nejpravděpodobnější příčina jeví sdílené diskové úložiště, používané všemi testovacími uzly. Přestože je toto úložiště velice výkonné, může i tak docházet k určitému zpomalení při intenzivním simultánním čtení 3 strojů. Další důležitou vlastností vyvinutého programu je to, že provádění dotazu v konfiguraci s jedním slave uzlem je srovnatelné s dobou vykonání dotazu referenčním programem Nfdump. Rozdíl distribuované architektury oproti lokálně spouštěnému programu se nejvíce projeví při dotazech na velmi malé množství dat, kdy v rámci distribuované architektury trvá několik sekund (typicky 2 až 3), než se ustanoví spojení mezi komunikujícími stranami, zatímco lokální instance programu je schopna poskytnout výsledky téměř okamžitě.

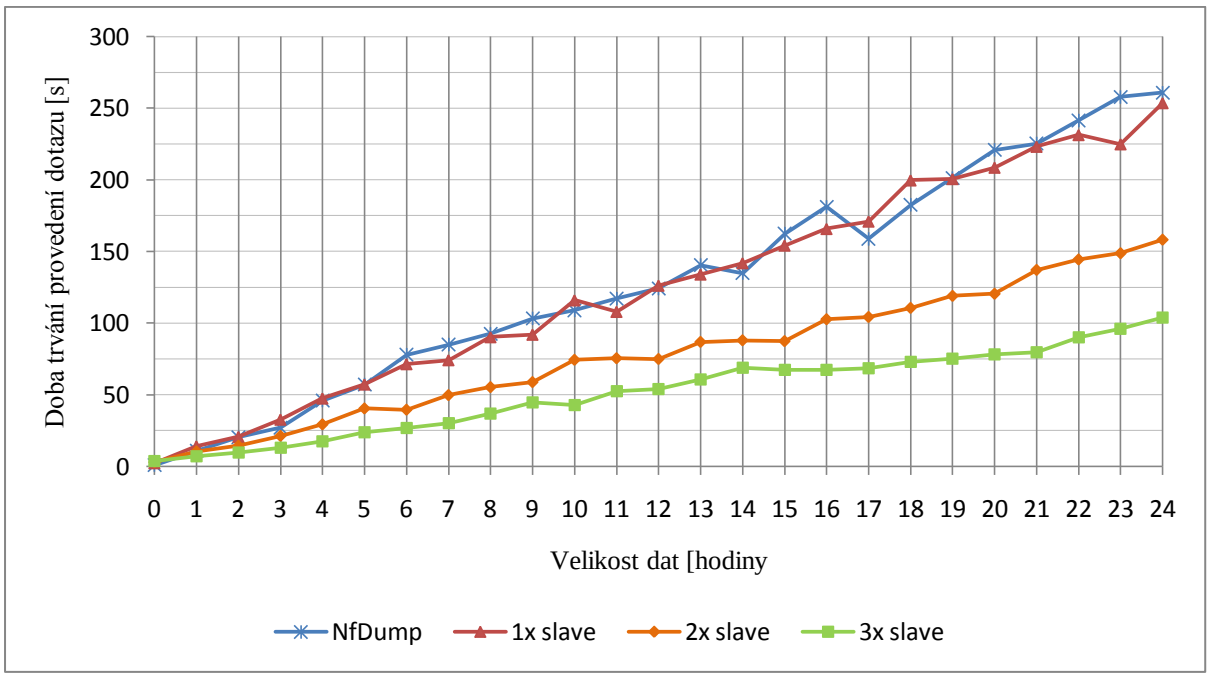


Obrázek 6-1 Dotaz č. 1: suma počtu paketů a bytů



Obrázek 6-2 Dotaz č. 2: počet všech záznamů s cílovým portem 53

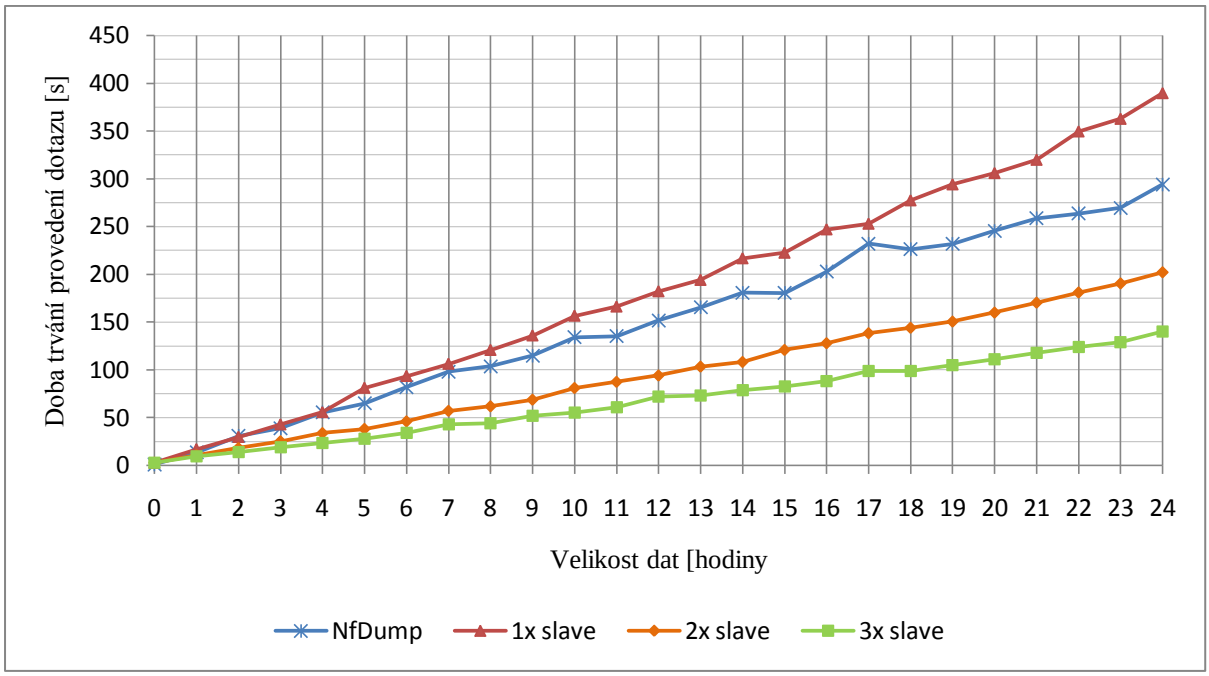
Obrázek 6-3 zobrazuje výsledky provedení dotazu, který oproti dvěma předchozím testuje více přenos dat výsledku. V předchozích dvou dotazech byly počítačovou sítí přenášeny lokálně agregované výsledky z jednotlivých slave uzlů, obsahující jediný záznam. V případě tohoto dotazu jsou přenášeny již jednotlivé záznamy toků, které jsou omezeny pouze filtrační podmínkou.



Obrázek 6-3 Dotaz č. 3: výpis zvolených položek pro záznamy přenášené prostřednictvím protokolu TCP na portu 53

Poslední z testovaných dotazů pak zahrnuje největší množství přenášených dat, jelikož na zdrojová data není aplikována žádná filtrační podmínka. Jsou tedy přenášeny všechny agregované záznamy, jichž je však kvůli malým tokům také mnoho (v praxi lze předpokládat limitaci počtu řádků výstupu, čímž by bylo omezeno i množství přenášených dat). Je zde možné pozorovat mírné zhoršení doby

odezvy dotazu při konfiguraci s jedním slave uzlem oproti samostatně běžícímu nástroji Nfdump, které je způsobeno nutností přenášet všechna data po síti prostřednictvím jediného kanálu.



Obrázek 6-4 Dotaz č. 4: výpočet sumy paketů, bytů a celkového počtu záznamů pro každou zdrojovou adresu.

Z celkového pohledu ukazují výsledky provedených experimentů značný přínos distribuovaného zpracování dat o IP tocích. Vyhodnocení dotazu při použití tří výpočetních slave uzlů se blíží trojnásobnému zrychlení, což odpovídá předpokládanému výsledku. Implementovaný program se svojí výkonností při použití jediného pracovního uzlu blíží optimalizovanému nástroji Nfdump, nicméně tato konfigurace nemá pro praktické použití žádné opodstatnění a do experimentů byla zahrnuta jednak pro získání srovnání s existujícím výkonným řešením, jednak pro možnost sledování škálovatelnosti při přidávání výpočetních uzlů.

Tyto experimenty nastínily potenciál distribuovaného zpracování dat a ukázaly, že je vhodné v tomto směru pokračovat s další prací. Při dalším vývoji se bude nutné zaměřit na limity síťové komunikace s centrálním master uzlem a dále také na otázku spolehlivosti a zálohy ukládaných dat, která v rámci této práce nebyla řešena. Celkově lze však implementovaný nástroj DistDump považovat za vyhovující a výsledky provádění dotazů pomocí tohoto nástroje na distribuované architektuře za uspokojivé.

7 Závěr

Cílem této práce je poskytnout řešení distribuovaného kolektoru, který bude prostřednictvím paralelizace schopen zpracovávat masivní objemy dat o tocích. Za tímto účelem byly zkoumány různé přístupy k distribuovanému zpracování dat obecně. Jako slibný kandidát byl vybrán model MapReduce, který v posledních letech vykazuje vhodné vlastnosti pro tuto oblast. Tato práce čerpá z poznatků dalších výzkumníků, zabývajících se využitím modelu MapReduce a distribuovaného zpracování dat v oblasti zpracování a analýzy sítového provozu.

Po nastudování problematiky jsem vytvořil podrobnou specifikaci vyvíjeného řešení a zvolil několik platforem, které vykazovaly určitý potenciál v oblasti distribuovaného zpracování dat o IP tocích. Tyto platformy jsou založeny na systému Apache Hadoop. Provedené experimenty potvrdily vhodnost distribuovaného zpracování pro masivní objemy dat, nicméně testované systémy se po vyhodnocení výsledků ukázaly pro vyvíjené řešení jako nevhodné. Na základě těchto závěrů jsem se následně rozhodl pro implementaci vlastního řešení, které je založeno na stávajícím nástroji Nfdump, jenž je v oblasti zpracování síťových dat často využíván a je při tomto zpracování velmi efektivní.

Po rozhodnutí o přibližné podobě jsem v rámci této práce provedl návrh jednotlivých komponent vyvíjeného kolektoru, přičemž část pro zpracování vstupních dat a jejich uložení bude vznikat v souladu s tímto návrhem v rámci jiné bakalářské práce. Já jsem se v této práci zaměřil na část dotazovací, která se při zpracování masivních objemů dat jeví jako nejvíce kritická.

Dle vytvořeného návrhu byl následně implementován nástroj DistDump, skládající se ze dvou částí, odpovídajících navržené architektuře. Implementovaný nástroj se svým rozhraním a funkcionalitou záměrně blíží programu Nfdump, jenž je v oblasti analýzy síťového provozu běžně používán. Díky této podobnosti bude vyvíjené distribuované řešení snáze rozšiřitelné a potencionální budoucí uživatelé si na něj snáze a rychleji zvyknou.

S implementovaným řešením byly následně provedeny obdobné experimenty, jako u platforem založených na systému Hadoop. Výsledky těchto experimentů ukázaly vhodnost použití tohoto nástroje pro distribuované zpracování dat o IP tocích. Naměřená výkonnost implementovaného frameworku odpovídá vytvořené specifikaci.

V rámci této práce jsem navrhl a z části implementoval řešení pro distribuované zpracování dat o IP tocích. Toto řešení je v hodné jak pro dotazy menšího rozsahu, tak i pro zpracování masivních objemů dat. Výkonnost systému roste přidáváním dalších výpočetních uzlů téměř lineárně a poskytuje tak dobrou škálovatelnost i pro budoucí vývoj a nasazení. Celkově lze tedy výsledek této práce považovat za úspěšný a přínosný. Implementovaný nástroj je ve své první verzi docela jednoduchý a předmětem další práce budou jeho rozšíření. Tato rozšíření by se měla zaměřit na podporu dalších položek pro agregované dotazy a top-n statistiky. Dále je pak vhodné rozdělit aplikaci na více vláken a rozšířit komunikační protokol tak, aby bylo možné s větší granularitou sledovat a řídit průběh výpočtu dotazů. V neposlední řadě bude nutné řešit spolehlivost a zálohu dat, kdy se jako nejvhodnější přístup jeví zrcadlení jednotlivých slave uzlů.

Literatura

[1] *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [online]. 2013 [cit. 2015-01-16]. ISSN 2070-1721. Dostupné z: http://tools.ietf.org/html/rfc7011

[2] MORKEN, Jan Tore. *Distributed NetFlow Processing Using the Map-Reduce Model*. Trondheim, 2010. Dostupné z: http://www.diva-portal.org/smash/get/diva2:352472/FULLTEXT01.pdf. Diplomová práce. Norwegian University of Science and Technology. Vedoucí práce Svein-Olaf Hvasshovd.

[3] LEE, Yeonhee a Youngseok LEE. Toward scalable internet traffic measurement and analysis with Hadoop. *ACM SIGCOMM Computer Communication Review* [online]. 2012, vol. 43, issue 1 [cit. 2015-01-16]. DOI: 10.1145/2427036.2427038. Dostupné z: http://www.sigcomm.org/sites/default/files/ccr/papers/2013/January/2427036-2427038.pdf

[4] MARCHAL, Samuel, Xiuyan JIANG, Radu STATE a Thomas ENGEL. A Big Data Architecture for Large Scale Security Monitoring. *2014 IEEE International Congress on Big Data* [online]. 2014 [cit. 2015-01-16]. DOI: 10.1109/BigData.Congress.2014.18. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6906761

[5] DEAN, Jeffrey, Sanjay GHEMAWAT, Rainer SCHMIDT a Matthias RELLA. MapReduce. *Communications of the ACM* [online]. 2008, vol. 51, issue 1, s. 667-683 [cit. 2015-01-16]. DOI: 10.1007/978-1-4614-1415-5_26. Dostupné z: http://web.b.ebscohost.com.ezproxy.lib.vutbr.cz/

[6] KARLOFF, Howard, Siddharth SURI a Sergei VASSILVITSKII. A Model of Computation for MapReduce. In: *Symposium on Discrete Algorithms SODA '10* [online]. 2010, s. 938-948 [cit. 2015-01-16]. ISBN: 978-0-898716-98-6. Dostupné z: http://theory.stanford.edu/~sergei/papers/soda10-mrc.pdf

[7] Execution of MapReduce. In: *MapReduce: Simplified Data Processing on Large Clusters* [online]. 2008 [cit. 2015-01-16]. Dostupné z: http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0007.html

[8] Seznam položek pro IPFIXcol kolektor [online]. 2014 [cit. 2015-01-17]. Dostupné z: https://homeproj.cesnet.cz/projects/ipfixcol/repository/revisions/master/entry/base/config/ipfix-elements.xml

[9] WHITE, Tom. *Hadoop: the definitive guide*. 3rd ed. Sebastopol: O'Reilly, 2012, xxiii, 657 s. ISBN 978-1-449-31152-0.

[10] CAPRIOLO, Edward. *Programming Hive: the definitive guide*. 3rd ed. Sebastopol: O'Reilly, 2012, xvii, 328 s. ISBN 978-1-449-31933-5.

[11] KARAU, Holden. *Fast data processing with Spark*. Birmingham: Packt Publishing, 2013, 1 online zdroj (120 pages). ISBN 978-1-78216-707-5.

[12] Spark: Lightning-fast cluster computing. *Apache Spark* [online]. 2015 [cit. 2015-06-02]. Dostupné z: https://spark.apache.org/

[13] BARTOŠ, Václav, Martin ŽÁDNÍK a Tomáš ČEJKA. *CESNET Technical Report: Nemea: Framework for stream-wise analysis of network traffic*. : 1-19. Dostupné také z: http://www.cesnet.cz/wp-content/uploads/2014/02/trapnemea.pdf

[14] KOZUBÍK, Michal. *Profilování dat pomocí IPFIX mediátoru*. Brno, 2015 (bude vydáno). Bakalářská práce. FIT VUT v Brně. Vedoucí práce Jan Kořenek.

Seznam příloh

A Obsah CD

Na přiloženém disku, který je součástí této práce se nachází tento text ve formátu pdf a ve zdrojovém formátu docx. Dále cd obsahuje zdrojové kódy vytvořeného programu DistDump se soubory potřebnými pro překlad prostřednictvím systému Autotools. Spolu se zdrojovými kódy je na CD k dispozici také soubor README, obsahující návod k instalaci a použití nástroje a dokumentace vygenerovaná nástrojem Doxygen. Dokumentace i instrukce v souboru README jsou v anglickém jazyce. Zdrojové soubory spolu s instrukcemi a dokumentací jsou na disku uloženy v adresáři DistDump_src a obsaženy i jako archiv dist_dump-1.0.tar.gz. Součástí zdrojových kódů je také soubor *fields*, obsahující Unirec položky, potřebné pro správnou funkcionalitu programu. Poslední částí přiloženou na CD jsou binární soubory vytvořené aplikace, přeložené na stroji v CVT (Linux verze 3.12.38), uložené v adresáři DistDump_bin.

B Systémové požadavky

Vytvořený program byl testován na systémech Linux (Linux verze 3.10.0-229.el7.x86_64 a Linux verze 2.6.32-504.el6.x86_64). Pro přeložení zdrojových souborů je potřeba nástrojová sada autotools a překladač gcc. Dále pak knihovny libtrap, unirec a libnf. Při vývoji byly použity verze automake (GNU automake) 1.11.1, autoreconf (GNU Autoconf) 2.63, gcc (GCC) 4.4.7, libtrap 0.5.4., unirec v základní verzi a libnf 1.07.

C Manuál nástroje DistDump

Vytvoření program se skládá ze dvou částí. Část „master“ tvoří aplikace, která je vždy spouštěna při zadávání dotazu. Nastavení tohoto dotazu je určeno parametry příkazové řádky. Druhá strana aplikace, tvořená částí „slave“, je aplikace, od které se očekává stálý běh na daném slave uzlu. Použitelné parametry pro ovládání částí master i slave jsou uvedeny ve druhé sekci této přílohy.

Překlad

Pro překlad nástroje DistDump je nutné mít k dispozici knihovny libtrap, unirec a libnf. Knihovna unirec musí obsahovat položky nástroje DistDump, jenž jsou dodávány spolu se zdrojovými kódy na přiloženém CD v souboru fields. Dále jsou potřeba nástroje gcc a autotools. Před samotným překladem je nutné nastavit systémové proměnné:

- UNIREC_CFLAGS=-I<cesta_k_adresáři_unirec> (hlavičkový soubor unirec.h)
- UNIREC_LIBS=-L<cesta_k_unirec_knihovně> (knihovna unirec)

V případě, že není knihovna libtrap nainstalována v systému a je použita např. verze z repozitáře, je dále potřeba nastavit proměnné prostředí:

- LIBTRAP_CFLAGS=-I<cesta_k_adresáři_libtrap> (hlavičkový soubor trap.h)
- LIBTRAP_LIBS=-L<cesta_k_libtrap_knihovně> (knihovna unirec)

Stejně tak v případě nenainstalované knihovny libnf je potřeba nastavit:

- LIBNF_CFLAGS =-I<cesta_k_adresáři_libnf> (hlavičkový soubor libnf.h)
- LIBNF_LIBS=<cesta_k_libnf_knihovně> (knihovna libnf)

Dále stačí v adresáři souboru spustit posloupnost příkazů:

- `autoreconf -i`
- `./configure`
- `make`

V adresáři následně vzniknou dva binární soubory, dist_dump_master a dist_dump_slave, které jsou ihned připraveny k použití. Příklad překladu může být následující (znak # značí výzvu interpretu příkazů; příklad předpokládá výskyt adresářů se soubory libtrap, unirec a libnf o úroveň výše, než je aktuální adresář):

```
#export UNIREC_CFLAGS=-I../
#export UNIREC_LIBS =-L../unirec
#export LIBTRAP_CFLAGS=-I../libtrap/include
#export LIBTRAP_LIBS =-L../
#export LIBNF_CFLAGS=-I../nf-tools/libnf/c/include
#export LIBNF_LIBS =-L../nf-tools/libnf/c/src/.libs/
#autoreconf –i
#./configure
#make
```

Použití

Jednotlivé uživatelské dotazy jsou spouštěny na straně master uzlu stejným způsobem, jako jsou spouštěny dotazy nad nástrojem Nfdump.
Parametry strany master aplikace nfdump:

- `C <cesta>`: cesta ke konfiguračnímu souboru.
- `e <cesta>`: cesta pro standardní chybový výstup.

- `E <cesta>`: cesta pro standardní výstup a standardní chybový výstup.
- `v`: zapne výpis stavových informací.
- `h`: vypíše nápovědu programu.
- `r <výraz-cesty>/R <výraz-cesty>`: cesta ke zdrojovým souborům. Může být zadána jako soubor, adresář nebo rozsah souborů, oddělených znakem „:“.
- `c <n>`: nastaví limit pro výpis dotazu typu list-flows.
- `A <výraz>`: nastaví agregaci dle výrazu pro dotaz typu list-flows. Seznam podporovaných položek následuje za popisem parametrů.
- `n <n>`: nastaví limit pro výpis dotazu typu top-n (výchozí hodnota je 10, 0 znamená vše).
- `s <výraz>`: zadá dotaz na top-n statistiky pro položku definovanou výrazem. Seznam podporovaných položek následuje za popisem parametrů.
- `O <řazení>`: nastaví položku řazení pro agregované list-flows dotazy a top-n statistiky. Výchozí hodnota je "flows". Může být nastaveno na jednu z flows, pkts, bytes, bps, pps, bpp.

Položky podporované pro agregaci a top-n statistiky:

- `proto`: IP protokol,
- `srcip`: zdrojová IP adresa,
- `dstip`: cílová IP adresa,
- `srcport`: zdrojový port,
- `dstport`: cílový port,
- `srcas`: číslo zdrojového autonomního systému,
- `dstas`: číslo cílového autonomního systému,
- `inif`: číslo vstupního rozhraní,
- `outif`: číslo výstupního rozhraní,
- `srcmask`: zdrojová maska,
- `dstmask`: cílová maska,
- `srcvlan`: číslo zdrojové VLAN,
- `dstvlan`: číslo cílové VLAN,
- `insrcmac`: vstupní zdrojová MAC adresa,
- `outdstmac`: výstupní cílová MAC adresa,
- `indstmac`: vstupní cílová MAC adresa,
- `outsrcmac`: výstupní zdrojová MAC adresa,
- `srctos`: číslo zdrojového typu služby (ToS),
- `dsttos`: číslo cílového typu služby (ToS).

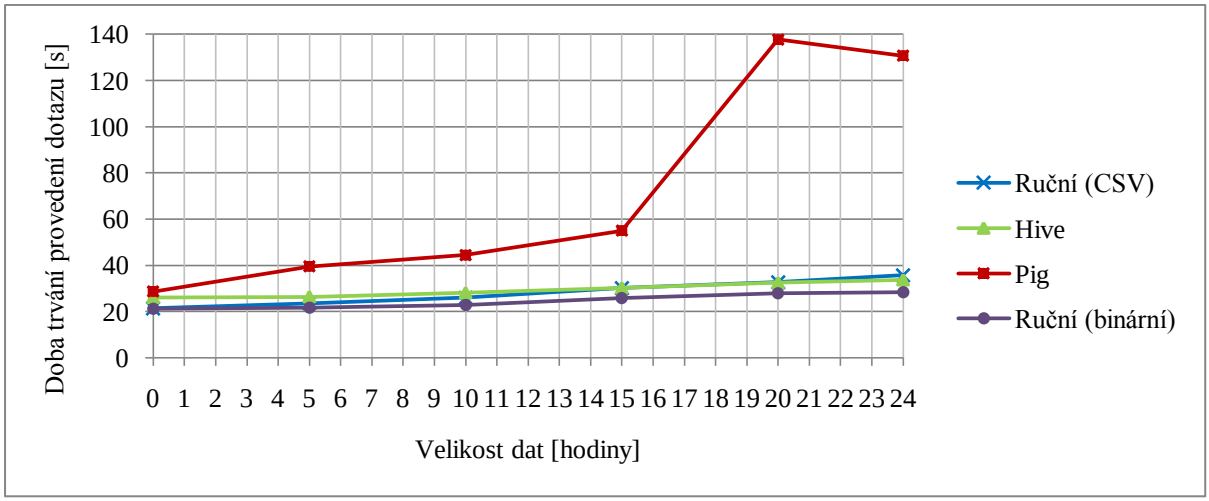
Parametry strany slave aplikace nfdump:

- `C <cesta>`: cesta ke konfiguračnímu souboru.
- `d`: spustí aplikaci jako démona systému.
- `e <cesta>`: cesta pro standardní chybový výstup.
- `E <cesta>`: cesta pro standardní výstup a standardní chybový výstup.
- `v`: zapne výpis stavových informací.
- `h`: vypíše nápovědu programu.

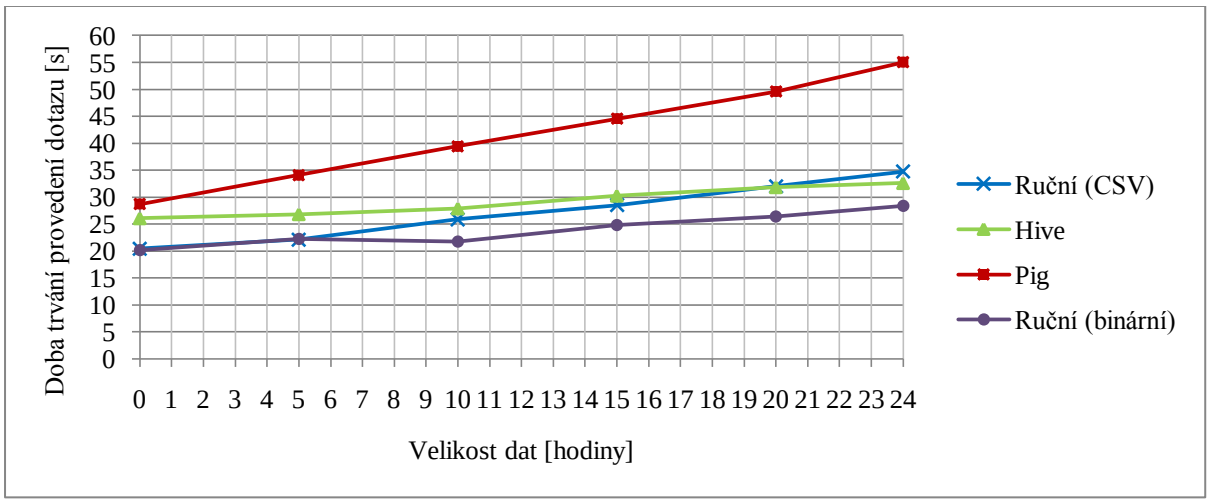
D Úplné výsledky experimentů

V této příloze jsou uvedeny úplné výsledky experimentů, zaměřených na optimalizaci platforem, založených na systému Hadoop.

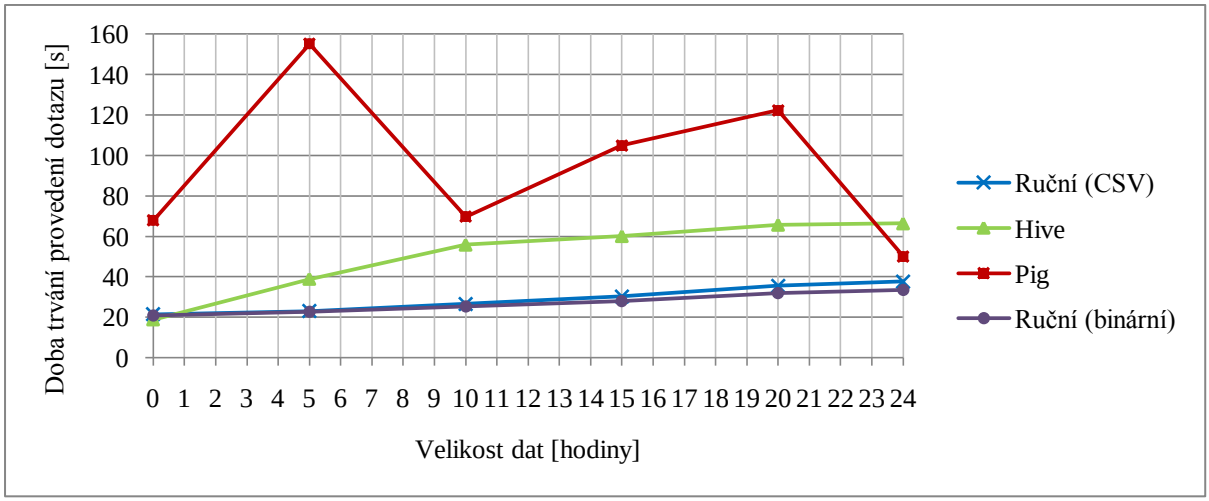
Výchozí nastavení



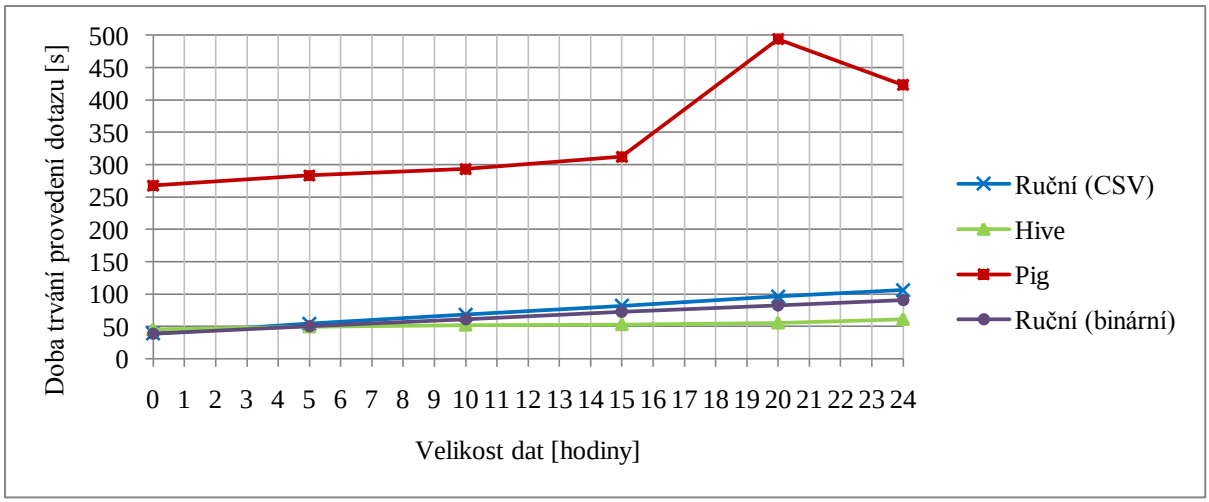
1 Dotaz č. 1: výchozí nastavení.



2 Dotaz č. 2: výchozí nastavení.

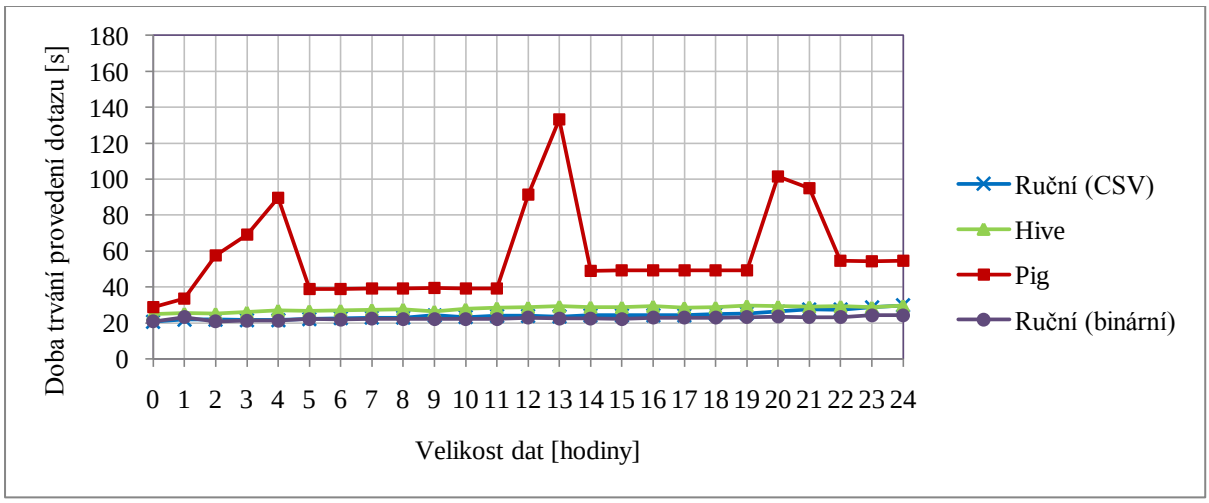


3 Dotaz č. 3: výchozí nastavení.

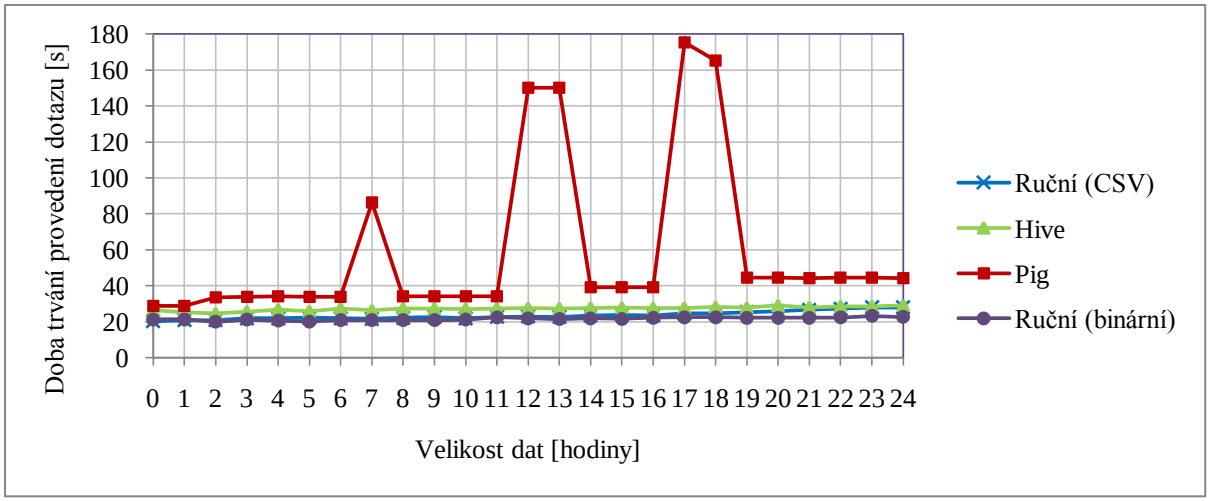


4 Dotaz č. 4: výchozí nastavení.

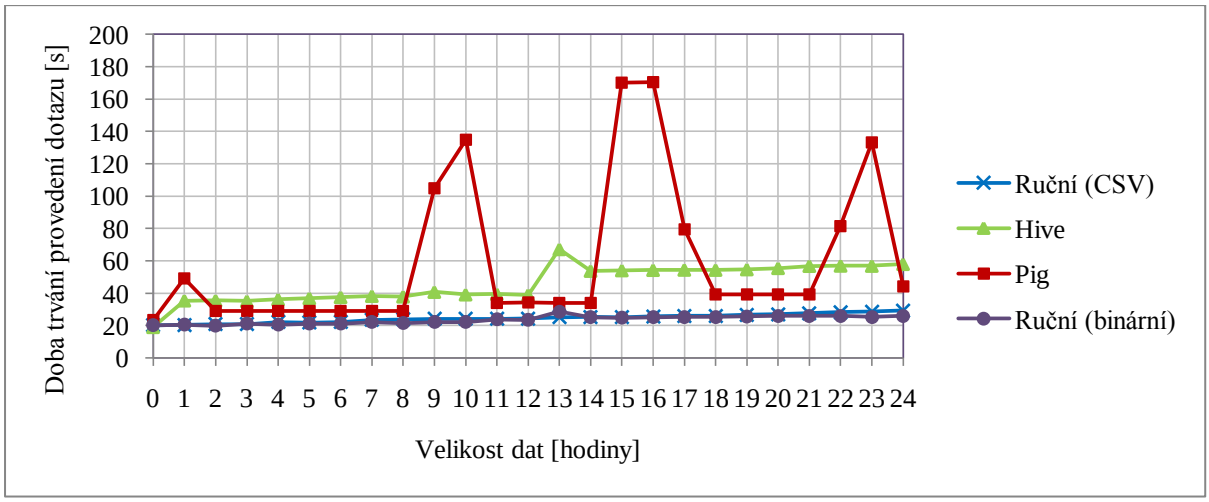
Heartbeat interval 1s



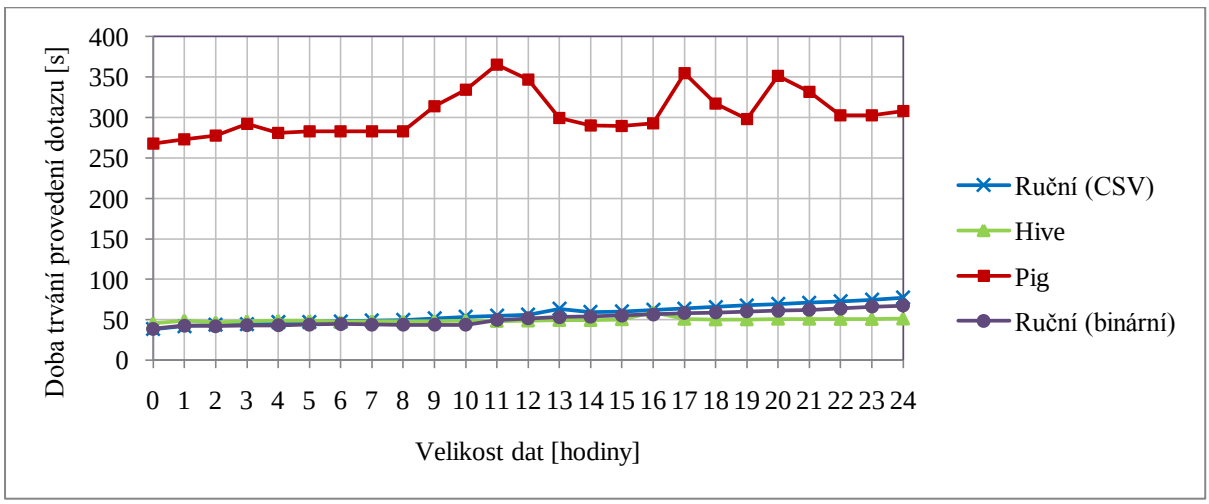
5 Dotaz č. 1: heartbeat 1s.



6 Dotaz č. 2: heartbeat 1s.

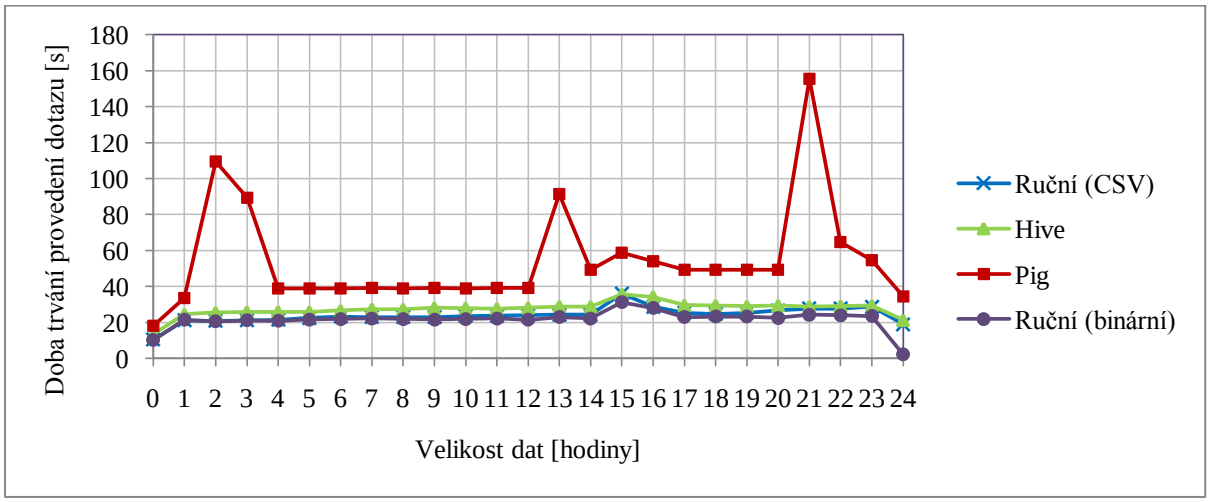


7 Dotaz č. 3: heartbeat 1s.

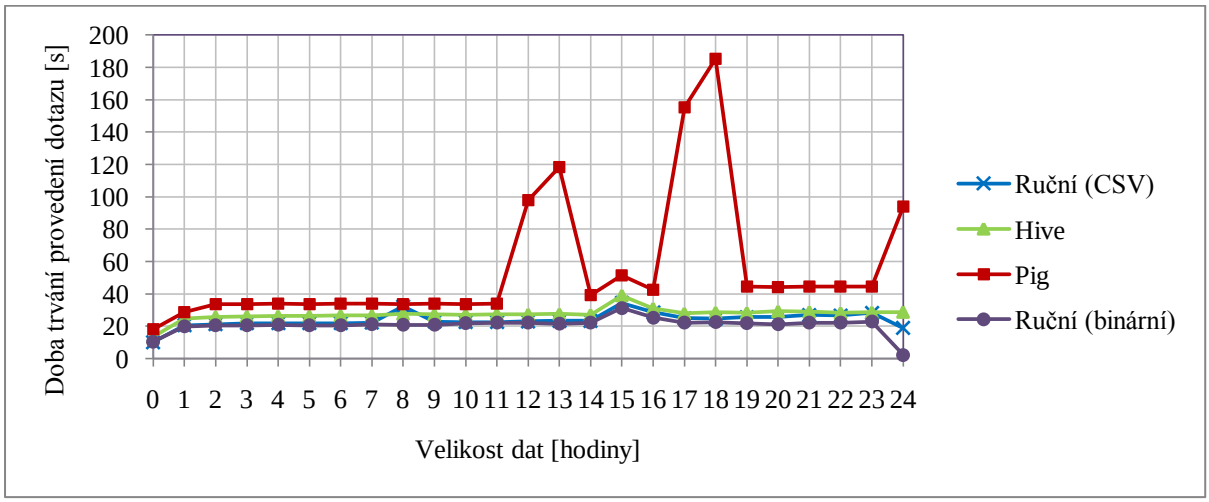


8 Dotaz č. 4: heartbeat 1s.

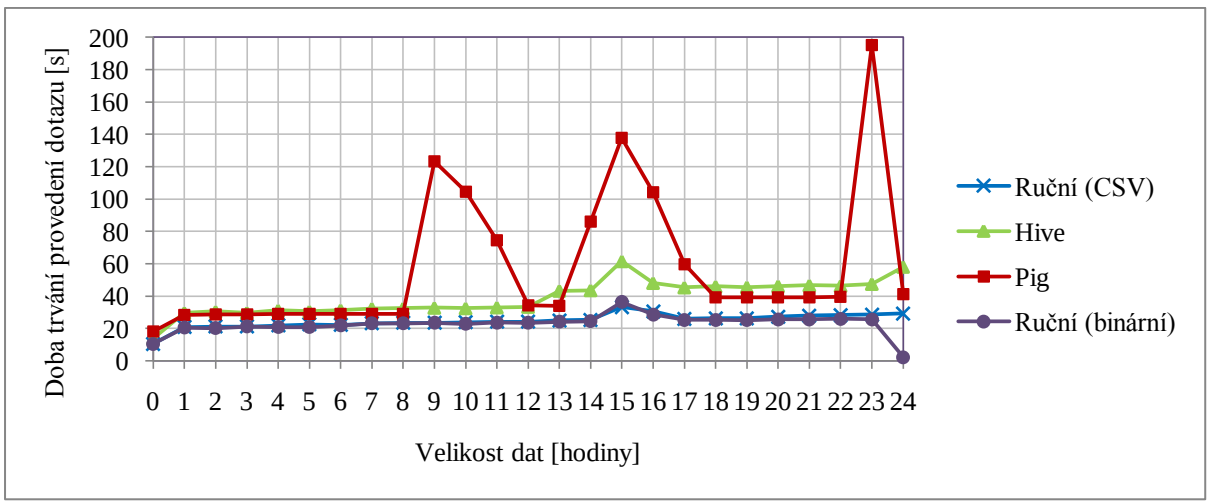
Zapnutý über mód



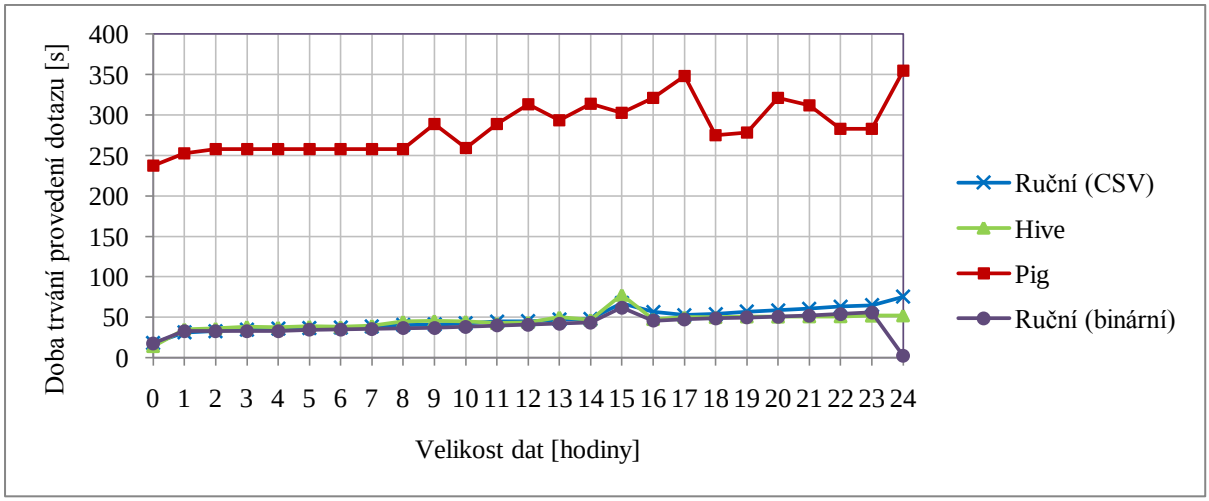
9 Dotaz č. 1: zapnutý über mód.



10 Dotaz č. 2: zapnutý über mód.

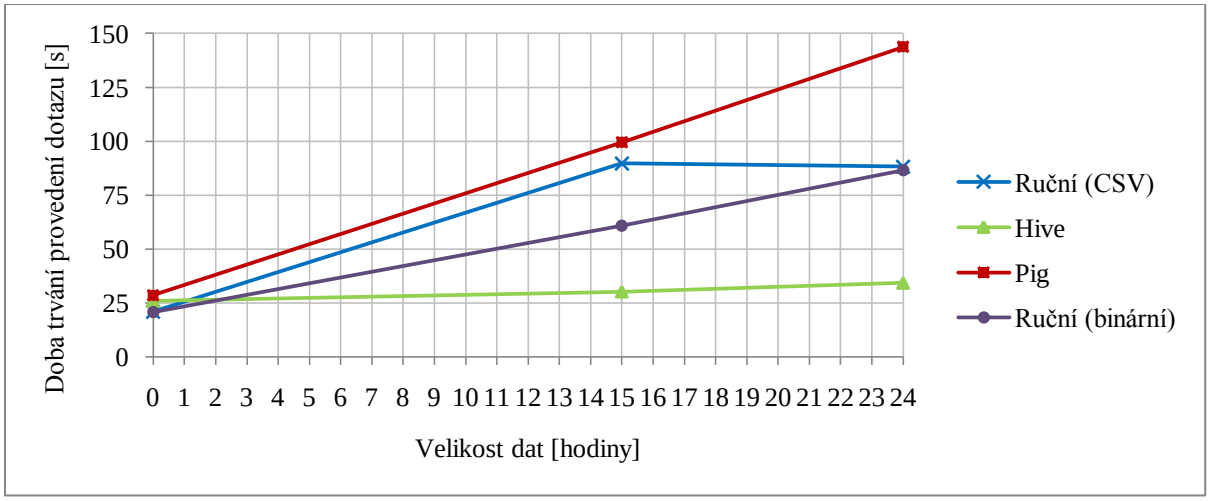


11 Dotaz č. 3: zapnutý über mód.

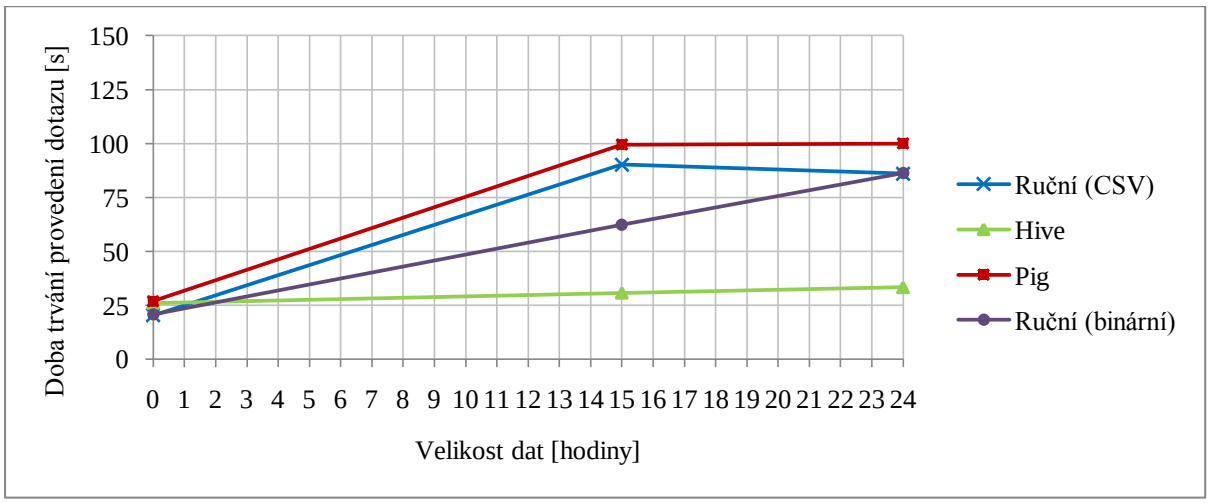


12 Dotaz č. 4: zapnutý über mód.

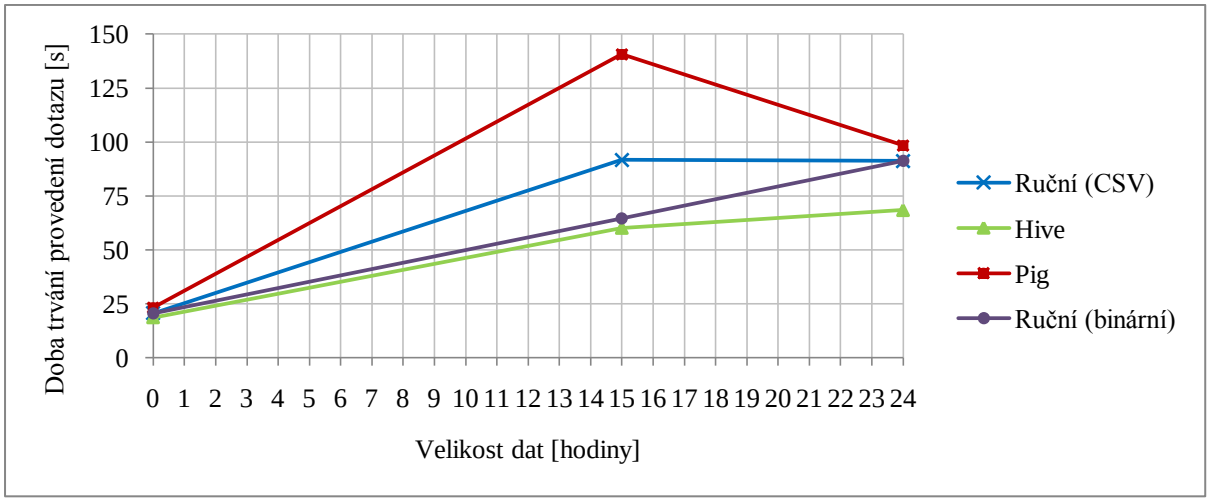
Replikační faktor 1



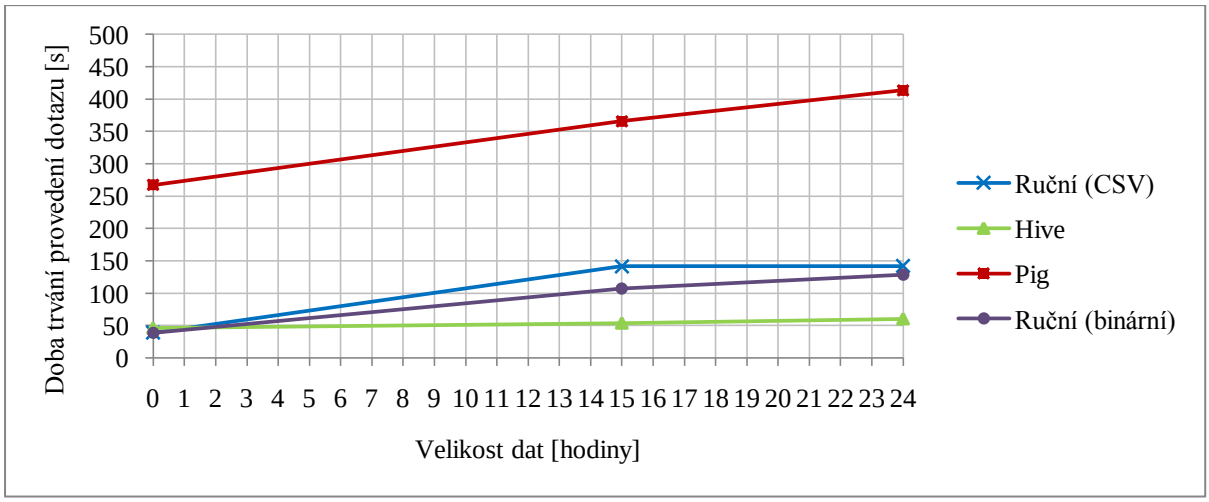
13 Dotaz č. 1: replikační faktor 1.



14 Dotaz č. 2: replikační faktor 1.

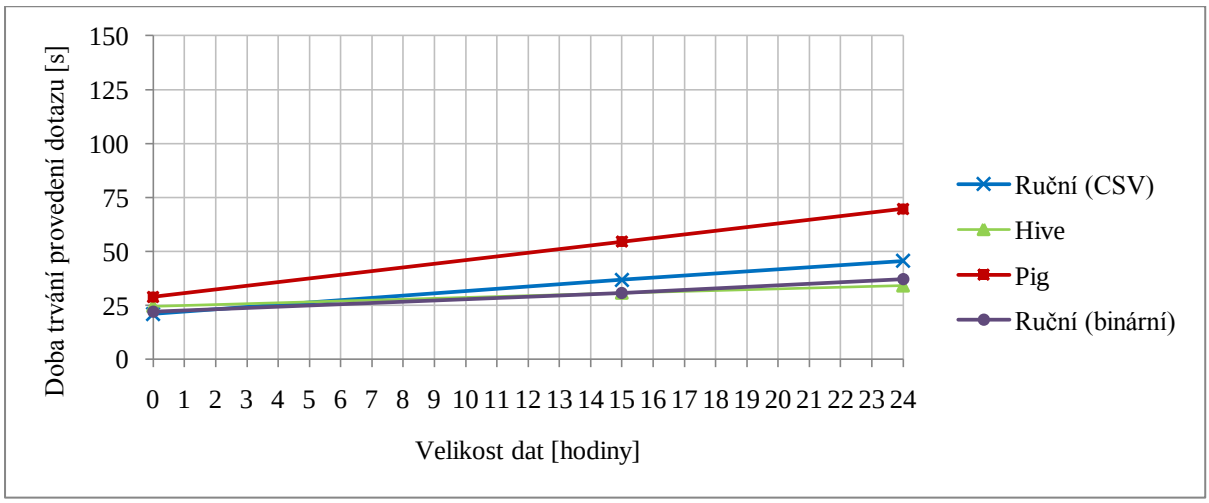


15 Dotaz č. 3: replikační faktor 1.

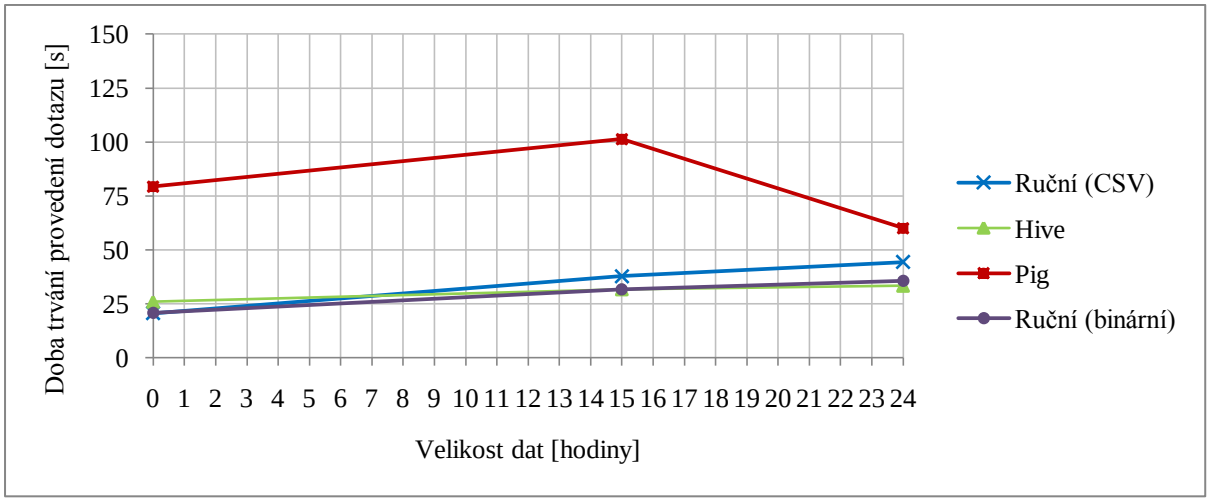


16 Dotaz č. 4: replikační faktor 1.

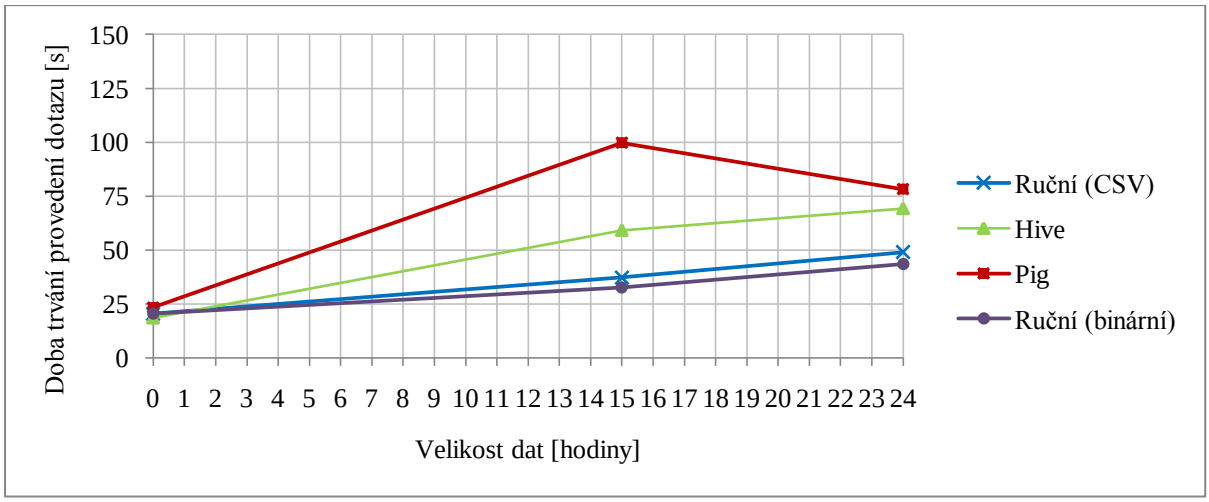
Replikační faktor 2



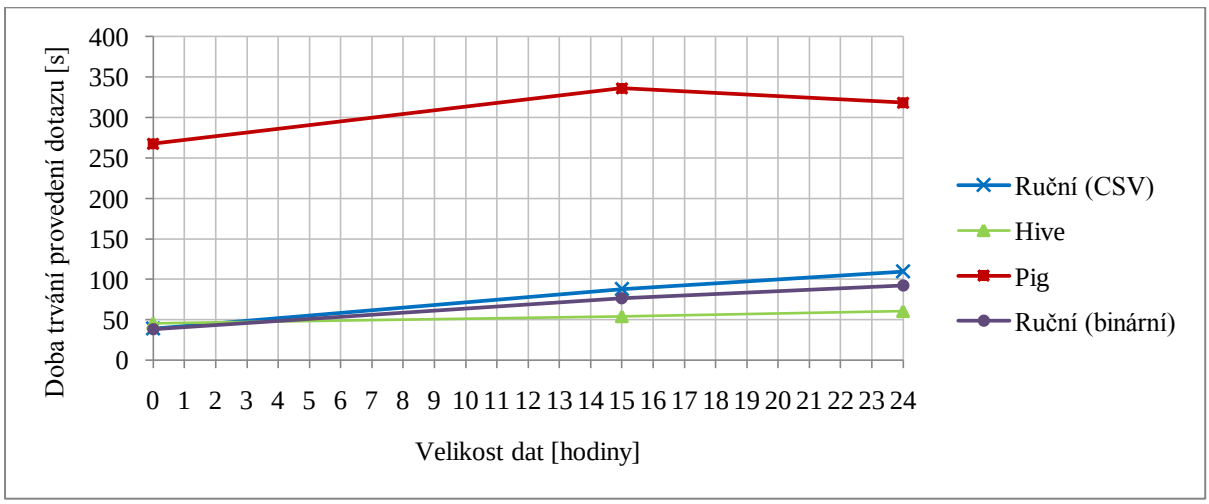
17 Dotaz č. 1: replikační faktor 2.



18 Dotaz č. 2: replikační faktor 2.

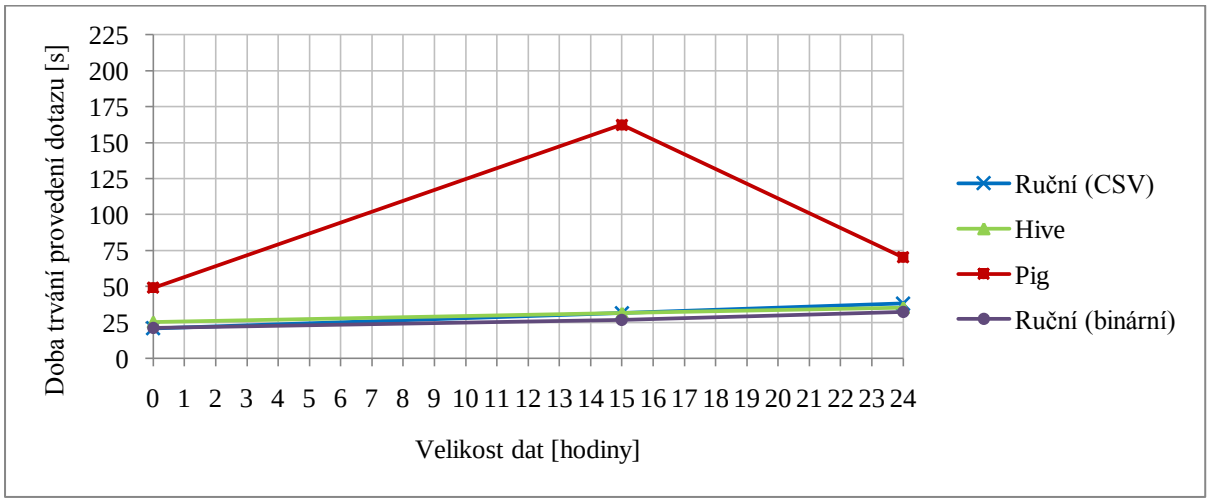


19 Dotaz č. 3: replikační faktor 2.

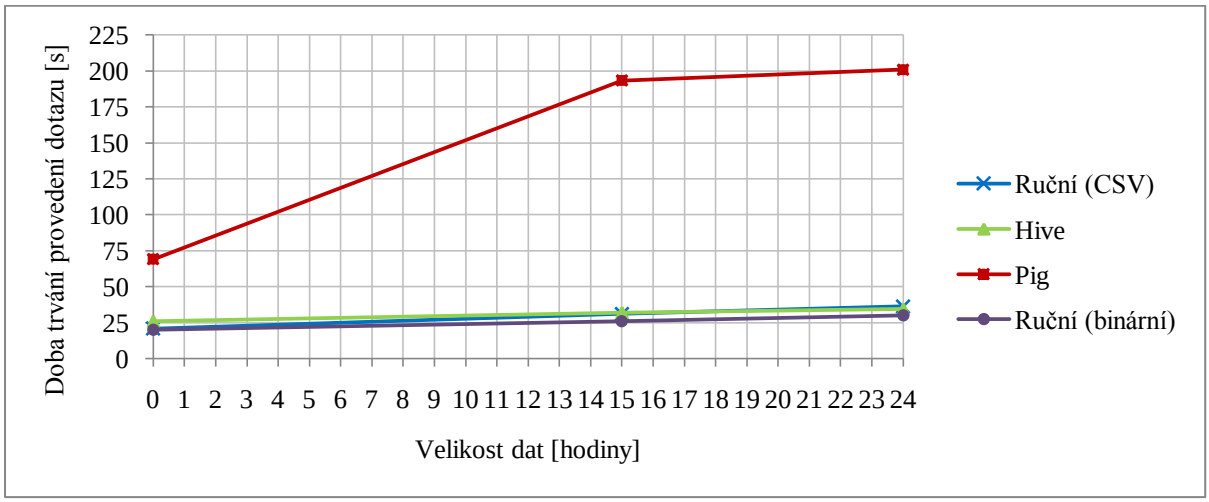


20 Dotaz č. 4: replikační faktor 2.

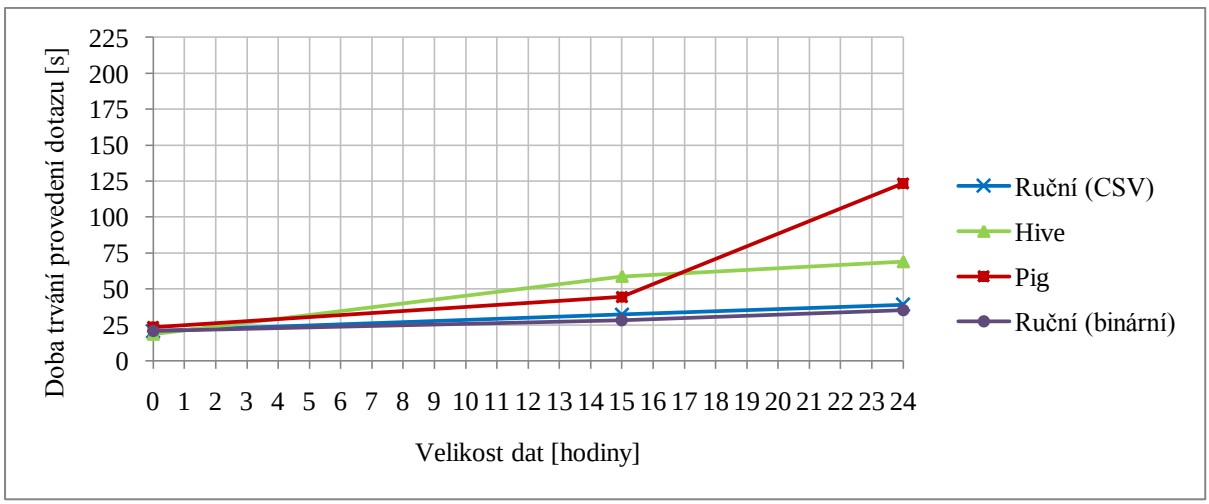
Replikační faktor 6



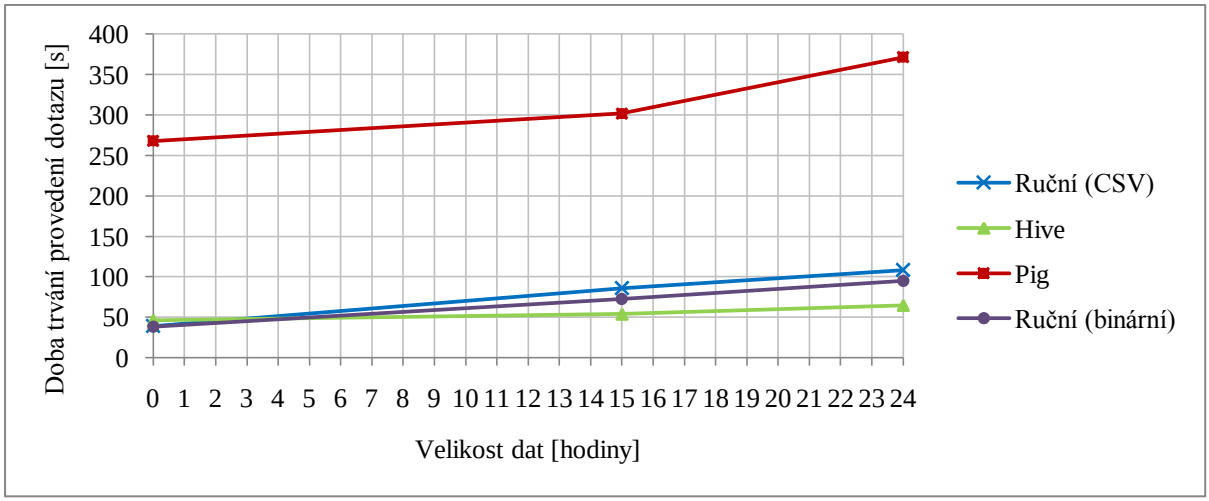
21 Dotaz č. 1: replikační faktor 6.



22 Dotaz č. 2: replikační faktor 6.



23 Dotaz č. 3: replikační faktor 6.



24 Dotaz č. 4: replikační faktor 6.