Ukládání a zpracování dat v systému Validovaného datového úložiště

Technická zpráva FIT VUT v Brně

Ing. Vladimír Bartík, Ph.D.
Doc. Ing. Radek Burget, Ph.D.
RNDr. Marek Rychlý, Ph.D.

Technická zpráva č. FIT-TR-2021-03
Fakulta informačních technologií, Vysoké učení technické v Brně

Poslední modifikace: 30. červen 2021

Ukládání a zpracování dat v systému Validovaného datového úložiště

Ing. Vladimír Bartík, Ph.D.
Ing. Radek Burget, Ph.D.
RNDr. Marek Rychlý, Ph.D.

Vysoké učení technické v Brně, email: {bartik,burgetr,rychly}@fit.vut.cz

Abstrakt Spolehlivé a dobře škálovatelné softwarové řešení v dnešní době vyžaduje nasazení v distribuovaném prostředí, kdy se výsledný systém skládá z více samostatný počítačů komunikujících prostřednictvím sítě, jak je tomu např. u technologie cloud computing. Distribuované ukládání a zpracování dat však přináší řadu návrhových i technických problémů. V tomto dokumentu popisujeme návrh a implementaci nového distribuovaného řešení, které umožňuje ukládat dokumenty, indexovat a prohledávat jejich obsah a popis (data a metadata) a aplikovat metody dolování z dat pro získání dodatečných znalostí. Výsledný systém je základem jádra systému Validovaného datového úložiště.

1 Úvod

Cloud computing je významným technologických trendem. Informační systémy vyvinuté a nasazené pomocí technologie cloud computing poskytují svým uživatelům služby prostřednictvím rozsáhlých podnikových sítí, v případě privátních cloud computing řešení na infrastruktuře zákazníka, či v rámci sítě Internet u veřejných cloud computing systémů běžících na platformách velkých poskytovatelů. Obecně, systémy využívající cloud computing jsou snadněji udržovatelné, lépe rozšiřitelné a výkonově škálovatelné dle aktuální potřeby uživatelů. [11]

Technicky jsou cloud computing systémy realizovány jako distribuované systémy běžící na skupinách strojů (často virtualizovaných) propojených prostřednictvím soukromé či veřejné sítě. Distribuované ukládání a zpracování dat i použitá komunikační infrastruktura však přináší řadu návrhových i technických problémů, mezi které patří především netriviální plánování a alokace komunikačních, výpočetních a datových zdrojů [7], zabezpečení přístupu k výpočtům a datům a zaručení jejich důvěrnosti [1], či zajištění spolehlivosti a důvěryhodnosti dat uložených v takových systémech [23].

Výše zmiňovaným výzvám, ale i jiným praktickým problémům, jsme museli čelit během vývoje systému Validovaného datového úložiště. V tomto dokumentu popisujeme návrh a implementaci nového distribuovaného řešení, které představuje jádro zmiňovaného systému a umožňuje ukládat dokumenty, indexovat a

prohledávat jejich obsah a popis (data a metadata) a aplikovat metody dolování z dat pro získání dodatečných znalostí. Technická zpráva v dalších kapitolách představuje inovativní řešení, které se do značné míry liší od ostatních dostupných cloudových dokumentových úložišť a to nejen zaměřením (tedy poskytovanými funkcemi dle požadavků uživatelů systému), ale zejména použitými technologiemi, které jsou vhodné pro bezpečné nasazení a řízený provoz cloud computing v privátních prostředích organizace zákazníka. Z toho důvodu se tato zpráva zaměřuje především na technické požadavky a jejich konkrétní řešení.

Struktura této technické zprávy je následující. V kapitole 2 představujeme a analyzujeme technické požadavky na předmětný systém distribuovaného datového úložiště vhodného pro cloud computing s podporou distribuovaného zpracování dat a dolování. V dalších kapitolách pak postupně popisujeme návrh architektury jednotlivých částí jádra systému, t.j., distribuovaného objektového úložiště a způsobu jeho integrace do systému v kap. 3, způsob sledování událostí nad objektových úložištěm a jejich využití ke spouštění indexačních mechanismů v kap. 4, extrakci obsahu z uložených dokumentů a jejich indexaci v kap. 5, realizace vyhledávání nad v předchozím kroku vybudovanými indexy v kap. 6 a využití získaného obsahu uložených dokumentů v úlohách dolování dat v kap. 7. V závěru zprávy jsou pak dosažené výsledky diskutovány v rámci aktuálního stavu poznaní v dané oblasti v kap. 8 a stručně shrnuty v kap. 9.

2 Technické požadavky na systém

Na základě analýzy vlastností existujících cloud computing řešení a uvažovaných aplikací jsme identifikovali sadu technických požadavků, které jsou relevantní pro volbu technologií a technický návrh výsledného řešení. Tyto požadavky shrnujeme v sekci 2.1. V sekci 2.2 diskutujeme obecně jejich dopad na volbu technologií a následně v kapitole 3 představujeme návrh konkrétního technického řešení na základě těchto požadavků.

2.1 Identifikované požadavky

Návrh výsledného technického řešení je podřízen následujícím identifikovaným požadavkům na cloudové datové úložiště:

P1. Nezávislost na hostitelské platformě Vzhledem k předpokládanému nasazení řešení v cloudovém prostředí je nezbytné vyhnout se využití technologií, které jsou vázané na konkrétní hardwarovou platformu nebo operační systém. To umožní flexibilitu při výběru konkrétního cloudového řešení pro produkční nasazení.

P2. Škálovatelnost na libovolné množství výpočetních uzlů Možnost flexibilního rozšíření počtu výpočetních uzlů, na kterých je řešení nasazeno, je důležitá pro udržitelný provoz úložiště jak při vzrůstajícím objemu uložených

dat, tak i vzrůstající počet uživatelů a tím současně zpracovávaných požadavků na operace s uloženými dokumenty. Tento požadavek vylučuje využití centralizovaných technologií, které předpokládají jeden centrální výpočetní uzel, který ukládá data a/nebo zpracovává požadavky.

P3. Hierarchická organizace dokumentů Dokumenty v úložišti budou ukládány do složek, které mohou hierarchicky organizovány.

P4. Řízení přístupu na úrovni složek a/nebo souborů Pro použití systému je nutné ověření identity uživatele (autentizace). Uživatelé mohou být organizováni do skupin; jeden uživatel může být přiřazen do libovolného počtu skupin. Pro každý uložený dokument nebo složku je možno definovat přístupová práva pro jednotlivé dokumenty a složky, systém zabezpečí řízení přístupu k dokumentům a složkám na základě přidělených práv (autorizace).

P5. Řešení souběžného přístupu více uživatelů Při změnách dokumentů je nezbytné se vyhnout kolizím při editaci dokumentů (změna obsahu dokumentu provedená souběžně více uživateli). Systém proto musí implementovat zamykání dokumentů pro editaci v okamžiku, kdy jiný uživatel již změny provádí.

P6. Možnost ukládání doplňujících informací (metadat) k dokumentům Kromě vlastního obsahu je každý dokument opatřen doplňujícími informacemi (metadaty), zahrnující pojmenování dokumentu, datum poslední změny apod., jejichž rozsah může být průběžně rozšiřován.

P7. Podpora uložení dokumentu ve více alternativních formátech Každý dokument může být uložen v libovolném zdrojovém formátu, ve kterém byl pořízen a současně i v libovolném množství alternativních formátů, umožňujících snazší zobrazení (např. formát PDF) nebo zpracování speciálními aplikacemi (např. textová alternativa pro indexaci obsahu).

P8. Verzování a platnost dokumentů Při aktualizaci obsahu musí systém podporovat vytváření nových verzí dokumentů a evidovat, která verze je aktuálně platná.

P9. Uchování auditní stopy Operace s dokumenty (nahrání, modifikace, zneplatnění) musí být zaznamenávány včetně času změny, autora a původního a nového stavu metadat. Systém musí umožnit revizi auditní stopy – zpřístupnění záznamu změn a vložení záznamu o kontrole do auditní stopy.

P10. Možnost fulltextového vyhledávání v obsahu uložených dokumentů Systém musí umožnit efektivní vyhledávání v uložených dokumentech na základě jejich obsahu. Za tímto účelem je třeba vytvořit a průběžně aktualizovat vyhledávací index, který bude následně využit pro vyhledání relevantních dokumentů na základě uživatelského dotazu.

P11. Podpora pokročilých aplikací pro zpracování dokumentů v cloudovém prostředí Předpokládá se využití obsahu uložených dokumentů pro následnou pokročilou analýzu v cloudovém prostředí, zejména pro získávání znalostí z dat (data mining). Systém musí poskytovat vhodné rozhraní umožňující zpřístupnění obsahu pro tyto úlohy.

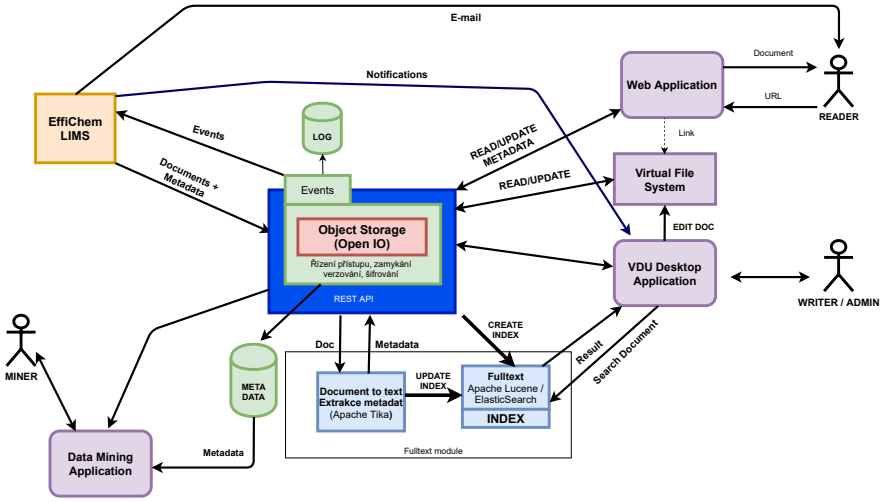
2.2 Vliv požadavků z hlediska volby technologií

Standardním řešením v oblasti síťově transparentního ukládání a sdílení souborů (dokumentů) jsou v současnosti síťové souborové systémy, zejména Common Internet File System (CIFS) Protocol[1]. Takové řešení umožňuje realizovat síťové úložiště souborů včetně řízení přístupu k jednotlivým souborům, jedná však vždy ve své podstatě o centralizované řešení, vázané na nějaký hostitelský operační systém (Windows, Linux, apod.) a proto nevhodné pro nasazení ve virtualizovaném cloudovém prostředí. Problémem je také škálovatelnost na větší množství uzlů a chybí rovněž podpora pro ukládání metadat, verzování a ukládání dokumentů v alternativních formátech. Z toho důvodu jsme navrhli zcela nové distribuované datové úložiště založené na otevřených technologiích, které vyhovuje výše uvedeným požadavkům a umožňuje nasazení ve škálovatelném distribuovaném prostředí. Provedený návrh je detailně rozebrán v následující kapitole.

3 Distribuované úložiště dat

Navržené úložiště je založeno na otevřených standardech pro softwarově definovaná úložiště objektů v cloudovém prostředí. Zejména se jedná o aplikační rozhraní *OpenStack Object Storage API*[2] (též *Swift API*) pro nízkoúrovňové operace s datovými objekty a *OpenStack Identity API*[3] (též *Keystone API*) pro řízení přístupu k úložišti. Vytvořená referenční implementace (viz. kap. 3.4) využívá pro nízkoúrovňové úložiště otevřené řešení *OpenIO*[4], díky využití uvedených otevřených standardů je však možno využít jakoukoliv jinou softwarovou implementaci, která tyto standardy podporuje. Samotné nízkoúrovňové úložiště objektů je doplněno o množství dalších modulů, které zajišťují splnění požadavků definovaných v předchozí kapitole. Celková architektura navrženého řešení je zachycena na obrázku 1.

[1] https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cifs/
[2] https://docs.openstack.org/api-ref/object-store/index.html
[3] https://docs.openstack.org/api-ref/identity/index.html
[4] https://www.openio.io/resources/openio-in-one-minute



Obrázek 1. Softwarová architektura distribuovaného úložiště dat

3.1 Vnitřní úložiště objektů

Centrální komponentou řešení je vnitřní úložiště objektů, které řídí samotné fyzické ukládání datových souborů pomocí OpenIO a zajišťuje základní funkčnost vyplývající z požadavků. Jedná se zejména o následující funkce:

Jednotná identifikace dokumentů – každý uložený dokument je označen jednoznačným identifikátorem podle standardu UUID (*universally unique identifier*, též GUID).

Správa metadat – ke každému dokumentu je možno přiřadit dodatečné údaje (metadata) zahrnující jména, zařazení do složky, datum modifikace, MIME typ, přístupová práva, doba platnosti a další. Tato metadata jsou spravována v dedikovaném úložišti nezávislém na hlavním obsahu dokumentu.

Řízení přístupu – navržené řešení umožňuje definovat *role uživatelů*, přidělovat jednotlivým rolím uživatelů přístupová práva ke složkám, autentizovat uživatele a autorizovat jejich přístup jednotlivým dokumentům. Dále je zjištěno zamykání dokumentů, které zabraňuje konfliktům při současně prováděných změnách obsahu.

Správa verzí – každý dokument může být uložen v několika verzích – formátech, jako např. MS Word (zdrojový dokument), PDF pro prohlížení, textová verze pro indexaci a další zpracování, apod. Jednotlivé verze jednoho dokumentu jsou vzájemně provázané pomocí metadat.

Auditní stopa – všechny operace s dokumenty (nahrání, modifikace, zneplatnění) jsou zaznamenávány v odděleném úložišti (LOG), které uchovává předmět změny, subjekt (autora změny), čas a detaily provedených změn.

Dále bylo navrženo aplikační rozhraní typu REST, které umožňuje ostatním komponentám systému komunikovat standardním způsobem (pomocí protokolu HTTP) s vnitřním úložištěm a využívat jeho funkce. Přehled koncových bodů (*endpoints*) navrženého rozhraní je uveden v tabulce 1. Pro autentizaci uživatelů je použit otevřený standard JWT[5] (*JSON Web Token*), který umožňuje realizovat bezpečnou autorizaci a autentizaci uživatelů s využitím rolí.

3.2 Klientské aplikace

Navržená architektura se síťovým REST aplikačním rozhraním umožňuje realizovat uživatelské rozhraní celého systému pomocí různých klientských aplikací, které se liší způsobem využití a použitými technologiemi. V rámci návrhu architektury jsme uvažovali zejména webové uživatelské rozhraní, dále pak desktopovou klientskou aplikaci pro MS Windows a virtuální souborový systém pro MS Windows a Linux.

Webová klientská aplikace Webová aplikace představuje primární uživatelské rozhraní systému. Poskytuje plnohodnotné rozhraní ve webovém prohlížeči umožňující procházení složek, zobrazení a editaci metadat, jakož i zobrazení a editaci vlastního obsahu dokumentů.

Předpokládáme klientskou aplikaci využívající webový prohlížeč pro komunikaci s uživatelem na jedné straně a aplikační REST rozhraní popsané v předchozí sekci pro provádění operací nad úložištěm na straně druhé. Z pohledu technologií se jedná o standardní webovou aplikaci implementovanou v jazyce JavaScript s využitím HTML a CSS pro vytvoření grafického uživatelského rozhraní a případně klientského aplikačního frameworku, jako např. Angular nebo React.js.

Způsob zobrazení a editace vlastního obsahu dokumentů závisí na jejich konkrétním formátu. Dokumenty PDF a textové dokumenty lze přímo zobrazit pomocí webového prohlížeče. Pro zobrazení a editaci kancelářských formátů lze použít komponenty třetích stran, jako např. Onlyoffice[6].

Desktopová klientská aplikace Desktopová klientská aplikace je uvažována jako alternativní uživatelské rozhraní primárně pro operační systém MS Windows. Její předností je užší integrace s lokálním prostředím a aplikacemi na klientském počítači. Vzhledem k tomu, že možnosti webových aplikací se v tomto ohledu neustále rozšiřují, nepovažujeme implementaci desktopové aplikace za perspektivní.

Virtuální souborový systém Virtuální souborový systém umožňuje zpřístupnit dokumenty uložené v distribuovaném úložišti dat lokálním aplikacím běžícím na počítači klienta. To umožňuje např. pořizovat a editovat uložené soubory přímo nástroji MS Office nebo dalšími lokálními aplikacemi. Tento koncept byl

[5] https://jwt.io/
[6] https://www.onlyoffice.com/cs/developer-edition.aspx

Tabulka 1. Koncové body REST rozhraní úložiště

Endpoint	Parametry v URL	Metoda	Popis
Složky	řízený přístup/access control; podle user/group či jednorázových tokenů		
folder/		GET	Získání seznamu složek
	folderID	GET	Získání seznamu souborů ve složce a metadat složky
	folderID	PUT	Úprava metadat složky
	folderID	POST	Vložení dokumentu nebo podsložky
	folderID	DELETE	Vyazání složky
file/			
	fileID	GET	Čtení metadat souboru
	fileID	PUT	Úprava metadat souboru
	fileID/versionID	GET	Čtení obsahu souboru
	fileID	POST	Vložení nové verze souboru
	fileID	DELETE	Vyazání souboru - všechny verze
	fileID/versionID	DELETE	Vyazání souboru - jedna verze
Fulltext	indexování a fulltextové vyhledávání		
index/			
	fileID	GET	Zaindexování souboru - převedení na text, zahrnutí do indexu
search/		GET	Vyhledání souborů podle obsahu
Události	monitorování událostí nad souborovým systémem		
events/		GET	Čtení událostí (filtrování)
subscribe/		POST	Registrace odběratele událostí (cílové URL)
Uživatelé	práva subjektů se řeší u objektů (složek/souborů) (ACL)		
user/		GET	získání seznamu uživatelů
	username	GET	čtení informací o užívateli
	username	PUT	uložení informací o uživateli
	username	POST	vložení nového uživatele (aktivace v případě existence deaktivovaného)
	username	DELETE	snazání existujícího uživatele (deaktivace)
group/		GET	získání seznamu skupin
	groupname	GET	čtení informací o skupině
	groupname	PUT	uložení informací o skupině
	groupname	POST	vložení nové skupiny (aktivace)
	groupname	DELETE	snazání existující skupiny (deaktivace)
	groupname/ users	GET	získání seznamu uživatelů ve skupině
	groupname/ users	POST	přidání uživatele do skupiny
	groupname/ users/username	DELETE	odstranění uživatle ze skupiny
audit/		GET	získání seznamu stop v audit logu (audit trail)
	trailID	GET	čtení podrobností stopy z audit logu (audit trail)
	trailID	PUT	modifikace podrobností stopy z audit logu (pouze dodatečné detaily, jako je odsouhlasení při revizi auditních stop)
token/		GET	získání aktivních přístupových tokenů (pro přímý přístup, např. z agenta virtualFS na klientovi)
		POST	získání nového přístupového tokenu s danou expirací (pro přímý přístup, např. z agenta virtualFS na klientovi): tokenID v odpovědi
	tokenID	DELETE	smazání/revokace přístupového tokenu před jeho expirací
	tokenID	GET	uplatnění tokenu, tj. v odpovědi bude fileID/versionID, kde lze získat dočasnou verzi souboru a i ji později uložit (vizte files/folders)

prostudován a implementován v rámci bakalářských prací studentů FIT VUT pro operační systémy MS Windows [9] a Linux [3].

3.3 Indexace a fulltextové vyhledávání

Funkce indexace a fulltextového vyhledávání je řešena jako samostatný modul a je podrobně popsána v kapitolách 5 a 6. Ze samotného úložiště je zde využívána zejména podpora ukládání dokumentů v alternativních formátech, což umožňuje extrahovat textový obsah z dokumentů a následně jej využít pro budování vyhledávacího indexu.

3.4 Prototypová implementace

Jako součást návrhu úložiště vznikla prototypová implementace zahrnující následující komponenty:

- Vnitřní úložiště založené na OpenIO realizované jako samostatný softwarový kontejner pro prostředí Docker[7], který je připraven pro nasazení v cloudovém prostředí.
- Datovou vrstvu využívající objektově-relační mapování (Entity Framework) zahrnující entity pro reprezentaci uživatelů, skupin uživatelů a základních prvků souborového systému – dokumentů a složek s metadaty.
- Vzorové REST aplikační rozhraní podle Tab. 1 implementované pomocí serverové technologie .NET Core. Součástí je i autentizace uživatelů pomocí JWT.
- Aplikační logiku realizující komunikaci mezi aplikačním rozhraním a vnitřním úložištěm. Tato komunikace je implementována pomocí protokolu Swift s využitím otevřené knihovny SwiftClient[8], kterou jsme rozšířili o novou implementaci autentizačního protokolu Keystone.

Výsledná implementace demonstruje použitelnost navrženého konceptu a umožňuje připojení výše uvedených klientských aplikací.

4 Sledování a zpracování událostí nad úložištěm

Validované datové úložiště (VDÚ) navazuje na další komponenty, které potřebují reagovat na události nad úložištěm.

4.1 Typy událostí

Je třeba sledovat zejména následující události a patřičně na ně reagovat:

- přístup klientů/uživatelů k jednotlivým datovým objektům musí vést k vytvoření příslušné auditní stopy,

[7] https://www.docker.com/
[8] https://github.com/cristipufu/SwiftClient

– změna uložených dat (dokumentů) a metadat (jejich popisných vlastností) může vést k indexaci obsahu pro jeho pozdější vyhledávání (vytvoření, aktualizace, či odstranění indexu),
– změna uloženého dokumentu může vést ke generování jeho náhledu v různých formátech, dle požadavků zobrazení dokumentu (např. z dokumentu Microsoft Word se vygeneruje náhled v HTML pro pozdější zobrazení v rozhraní webového prohlížeče),
– změna verze uloženého dokumentu může vyvolat přechod k další aktivitě v příslušném workflow procesu, kterého se předmětný dokument účastní,
– určitý celkový stav úložiště může vyvolat akci správy (např. archivování a povolené odmazání starých dat při zaplnění úložiště, redistribuce kopií při změně počtu uzlů infrastruktury úložiště/škálování, aj.).

V případě výskytu takové události je třeba události publikovat a dopravit na místa její spotřeby, přičemž událost může vzniknout v různých částech úložiště na různých prvcích jeho infrastruktury a může mít jednoho či více spotřebitelů na různých místech infrastruktury i v různém čase. Z těchto důvodů je nutné publikovat a spotřebovat událost asynchronně, tedy prostřednictvím systému zasílaní zpráv (tzv. fronty zpráv, angl. „message queue“).

4.2 Fronta zpráv Beanstalk

Jednou z implementací fronty zpráv je Beanstalk[9]. Služba Beanstalk běžící na jednom čí více strojích spravuje frontu zpráv zasílaných pomocí Beanstalk protokolu přes TCP spojení s jejími vydavateli a spotřebiteli (angl. „publishers“ a „subscribers“). Zprávy jsou obecně ve formátu prostého textu (kódování ASCII), avšak v praxi se ze používá jeden ze strukturovaných textových formátů dokumentů, např. JSON či XML.

Výhodou Beanstalk [13] oproti jiným technologiím (např. Kafka, RabbitMQ či ActiveMQ) je jednoduchost a v důsledku pak minimální zpoždění (latence) zpráv. Charakteristickým rysem je pak způsob odběru zpráv, kdy spotřebitel si zprávu převezme a tím rezervuje, zpracuje ji a pak potvrdí úspěšné zpracování potvrdí, čímž se zpráva považuje za spolehlivě doručenou. Tím je možno při neúspěšném zpracování, např. v důsledku selhání spotřebitele, síťové chyby, atp., nepotvrzenou zprávu po dané době znovu nabídnout ke zpracování ostatním spotřebitelům.

4.3 Události nad objektovým úložištěm OpenIO SDS

Technologie OpenIO SDS použitá v projektu VDÚ pro objektové úložiště podporuje oznamování zpráv nad úložištěm pomocí protokolu Beanstalk. Je možno tak využít Beanstalk službu pro distribuci a doručení zpráv jednotlivým klientům úložiště. Mezi typické klienty patří např. indexační služba či služba pro záznam auditních stop, jak bylo uvedeno v kap. 4.1.

[9] https://beanstalkd.github.io/

```
{
  "job_id": "3",
  "url": {
    "account": "AUTH_d2ba552fa63c4e56a7a9c9d5a8ba5ded",
    "content": "F1F9436E6EA40500F15198A6D2009BFF",
    "version": "1588169086859764",
    "user": "vdu",
    "path": "README.md",
    "ns": "OPENIO",
    "id": "9C65AE42371A5889E04823B10D341A63"
  },
  "when": 1588169086872572,
  "parts": 1,
  "request_id": "txa4aac4b0d3ac46d2a77f8-005ea9897e",
  "part": 0,
  "data": [
    {
      "hash": "4617FCBEBB3BDC676777B0CF7CA73DB2",
      "mime-type": "text/plain;␣charset\u003dus-ascii",
      "chunk-method": "plain/nb_copy\u003d1",
      "policy": "SINGLE",
      "type": "contents_headers",
      "id": "F1F9436E6EA40500F15198A6D2009BFF",
      "size": 527
    },
    {
      "name": "README.md",
      "deleted": false,
      "header": "F1F9436E6EA40500F15198A6D2009BFF",
      "version": 1588169086859764,
      "mtime": 1588169086,
      "type": "aliases",
      "ctime": 1588166576
    }
  ],
  "event": "storage.content.new"
}
```


Obrázek 2. Zpráva zaslaná pomocí Beanstalk protokolu úložištěm OpenIO SDS v případě vzniku nového dokumentu s názvem „README.md“.

Příklad zprávy zaslané pomocí Beanstalk protokolu úložištěm OpenIO SDS v případě vzniku nového dokumentu s názvem „README.md“ uvádní obr. 2. Část „url“ v příkladu uvedené Beanstalk zprávy odkazuje na předmětný dokument v úložišti, konkrétně pak atributy „account“, „user“ a „path“, které udávají kontejner v úložišti a přístupovou cestu k dokumentu. Součástí zprávy jsou také další užitečné informace, jako např. typ dokumentu, jeho velikost a verze, či čas uložení (atributy „data.mime-type“, „data.size“ a „data.version“, či „data.mtime“). Po přijetí takové zprávy je její spotřebitel schopen si v souladu se svým oprávněním získat z úložiště předmětný dokument, dále ho použít, např. zaindexovat jeho obsah, a pak potvrdit úspěšné dokončení zpracování přijaté zprávy. Toto se hojně využívá v službě pro extrahování a indexování obsahu dokumentů, která je popsána v kap. 5.

5 Extrakce obsahu dokumentů a jejich indexace

V systému Validovaného datového úložiště (VDÚ) je nutno nabídnout možnost fulltextového vyhledávání v obsahu uložených dokumentů. Tato požadovaná funkcionalita, společně s dalšími požadavky uvedenými v kap. 2, jako je podpora uložení dokumentu ve více alternativních formátech (např. textová varianta a PDF varianta) či podpora pokročilých aplikací pro zpracování dokumentů v prostředí cloud computing pro získávání znalostí z dat (data mining), vyžadují dodatečné a průběžné zpracování dokumentů uložených v úložišti. Pro iniciaci operací spojených s tímto zpracováním lze využít systém zasílání zpráv událostí popsaný v kap. 4.

5.1 Přijetí a zpracování zprávy o dokumentu k indexaci

Indexační služba VDÚ je napojena na Beanstalk frontu událostí publikovaných při změnách dokumentů úložištěm OpenIO SDS. Příklad zprávy o novém dokumentu k indexaci byl uveden na obr. 2 v kap. 4.3. Po přijetí takové zprávy provede indexační služba následující kroky a po jejich úspěšném konci potvrdí také úspěšné zpracování zprávy.

1. získání ve zprávě odkazovaného dokumentu z úložiště OpenIO SDS,
2. extrakce metadat a textu obsahu vč. jeho struktury ze získaného dokumentu,
3. uložení extrahovaného strukturovaného textu do úložiště OpenIO SDS v alternativních formátech k formátu původního dokumentu,
4. uložení získaných metadat a extrahovaného textu do úložiště služby pro fulltextové hledání.

5.2 Získání dokumentu a extrakce textu

Po přijetí zprávy o novém obsahu v úložišti je prvním krokem získání předmětného dokumentu a extrakce textu z jeho obsahu. Dokument je získán na základě jeho identifikace uvedené v přijaté zprávě (adresa úložiště, uživatel a cesta) a

pomocí rozhraní Swift, jak je uvedeno v kap. 3.4. Při získání dokumentu tímto způsobem musí mít služba implementující tuto funkcionalitu oprávnění k dokumentu přistupovat (vlastnit aktivní Keystone token).

Při čtení dokumentu z úložiště jsou jeho data ve formátu dokumentu (např. PDF soubor či z obrázek) přímo předávány službě zajišťující zpracování těchto dat a extrakci strukturovaného textu. V prototypové implementaci jádra VDÚ se k tomuto používá knihovna Apache Tika[10] s patřičně nakonfigurovanými detektory formátu a jeho parsery. Nástroj Apache Tika umožňuje detekovat a extrahovat text z běžných typů dokumentů (např. zmiňované PDF či dokumentu Microsoft Office). Pro extrakci textu z obrázků byl do nastroje Apache Tika zaintegrován nástroj Tesseract OCR[11] s podporou pro jazyky čeština, němčina, francouzština, italština, polština, slovenština a španělština. Oba použité nástroje, tj. Apache Tika i Tesseract OCR, mají otevřený zdrojový kód (open-source) a jsou volně k dispozici bez nutnosti získat proprietární licenci. Vzhledem k tomu, že extrakce textu i OCR mohou být u delších dokumentů značně výpočetně náročné, tyto úlohy se spouští souběžně v samostatných vláknech a neblokují tedy zbytek jádra VDÚ.

Pro zpracování specifických dokumentů laboratorních zpráv budoucích uživatelů systému VDÚ je vhodné nástroj Apache Tika rozšířit také o detektory formátu a parsery těchto dokumentů. Vzhledem k tomu, že takové laboratorní zprávy jsou většinou ve formátech tabulek či textových dokumentů v PDF, lze využít jako základ již zabudovanou podporu PDF dokumentů v nástroji Tika a tuto rozšířit o parsování strukturovaného obsahu specifického pro dané laboratorní zprávy (např. tabulku s naměřenými hodnotami v laboratoři posuzovaných ukazatelů).

Výsledkem tohoto kroku získání a extrakce předmětného dokumentu je tedy jeho strukturovaný textový obsah a soubor doprovodných metadat, které jsou pak vstupem další fáze, tj. uložení alternativního formátu dokumentu a samotná indexace obsahu dokumentu a jeho metadat, jak je popsáno v kap. 5.3 a kap. 5.

5.3 Uložení extrahovaného obsahu a alternativních formátů zdrojového dokumentu

Po získání dokumentu a po extrakci jeho obsahu a metadat jsou tyto dále využity pro uložení extrahovaného strukturovaného textu do úložiště OpenIO SDS v alternativních formátech k formátu původního dokumentu a uložení získaných metadat a extrahovaného textu do úložiště služby pro fulltextové hledání.

Jako alternativní formát pro dokumenty je navrženo používat jednoduché HTML (bez pokročilých kaskádových stylů). Tento formát umožňuje text strukturovat (nadpisy, odstavce, tabulky, seznamy, aj.), je snadno zobrazitelný jako náhled na cílových platformách klientských aplikací (webový prohlížeč či mobilní telefon/tablet) a lze ho dále využívat pro strojové zpracování (např. pro dolování dat). Výsledné HTML soubory jsou uloženy zpět do úložiště vedle zdrojových

[10] https://tika.apache.org/
[11] https://tesseract-ocr.github.io/

dokumentů, ze kterých vznikly a se kterými poté sdílí oprávnění a životní cyklus (např. při smazání zdrojového dokumentu jsou smazány i jeho alternativní formáty).

6 Prohledávání dokumentů

Jak již bylo zmíněno v kap. 5, pro indexaci extrahovaného obsahu a metadat zdrojového dokumentu je v prototypové implementaci jádra VDÚ využita služba Elasticsearch běžící jako samostatná komponenta systému VDÚ. Elasticsearch[12], poskytuje úložiště strukturovaných dat ve formátu JSON (obsahují extrahovaný obsah a metadata dokumentu) a jejich plnotextové prohledávání založené na otevřené technologii Apache Lucene[13].

Před použitím je třeba Elasticsearch nastavit, spustit a napojit na ostatní komponenty systému. Obrázek 3 popisuje mapování informací o původním dokumentu, jak byly přijaty v Beanstalk zprávě popsané v kap. 4.2, a z dokumentu extrahovaného obsahu (atr. „content“) a extrahovaných vlastností (atr. „meta“) do Elasticsearch. Výsledný index umožní plnotextové hledání podle všech položek typu „keyword“ či „text“, ale také podle struktury extrahovaných vlastností v „meta“. Kromě správy indexu nabízí služba Elasticsearch s uvedeným mapováním také rozhraní (API) pro plnotextové hledání nad typu „keyword“ či „text“. V systému VDÚ je pak komponenta Elasticsearch využívána právě k takovému plnotextovému hledání přes API Elasticsearch[14].

Představené řešení postavené na službě Elasticsearch je stabilní a škálovatelné [6]. Služba Elasticsearch by měla být v systému nasazena jako mikro-služba, podobně jako úložiště OpenIO SDS z kap.3, a může v systému existovat v jedné či více instancích dle potřeby vyrovnat aktuální zátěž. Tyto instance, často distribuované na různé uzly clusteru, sdílí společný index.

7 Dolování dat z dokumentů

Vzhledem k tomu, že části zmíněné v předchozích kapitolách provádí extrakci dat z dokumentů do strukturované podoby, je možné tato data dále využít pro dolování dat.

V rámci systému bylo implementováno několik úloh dolování, které by pro uživatele využívající VDÚ potenciálně mohly být zajímavé. Ty zahrnují zejména regulační diagramy pro účely statistické regulace procesů, metody pro redukci dimensionality, shlukovou analýzu a predikci spojitých hodnot s využitím vybraných regresních metod.

[12] https://www.elastic.co/products/elasticsearch
[13] https://lucene.apache.org/
[14] https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-get.html

```
{
  "mappings": {
    "numeric_detection": true,
    "dynamic_date_formats": [ "strict_date_optional_time",
      "strict_date_time_no_millis",
      "strict_date_hour_minute_second",
      "strict_hour_minute_second" ],
    "dynamic": false,
    "properties": {
      "url": { "properties": {
        "account": { "type": "keyword" },
        "content": { "type": "keyword" },
        "version": { "type": "long" },
        "user": { "type": "keyword" },
        "path": { "type": "keyword" },
        "ns": { "type": "keyword" },
        "id": { "type": "keyword" }
      } },
      "aliases": { "properties": {
        "name": { "type": "keyword" },
        "deleted": { "type": "boolean" },
        "header": { "type": "keyword" },
        "version": { "type": "long" },
        "mtime": { "type": "date", "format":"epoch_second" },
        "type": { "type": "keyword" },
        "ctime": { "type": "date", "format":"epoch_second" }
      } },
      "contents_headers": { "properties": {
        "hash": { "type": "keyword" },
        "mime-type": { "type": "keyword" },
        "chunk-method": { "type": "keyword" },
        "policy": { "type": "keyword" },
        "type": { "type": "keyword" },
        "id": { "type": "keyword" },
        "size": {
          "type": "long", "meta": { "unit": "byte" } }
      } },
      "extracted": { "properties": {
        "content": {
          "type": "text", "index_options": "offsets" },
        "meta": { "type": "object", "dynamic": true }
      } }
} } }
```

Obrázek 3. Mapování informací o původním dokumentu a z něj extrahovaného obsahu (content) a vlastností (meta) do Elasticsearch.

7.1 Statistická regulace procesů

Regulační diagram slouží ke zobrazení průběhu některé z klíčových veličin procesu v čase. V regulačním diagramu je typicky označena střední hodnota (CL - Central Line), a dále horní a dolní regulační mez (UCL – Upper Control Line a LCL – Lower Control Line), které jsou zadány uživatelem nebo určeny z historických dat. Z diagramu lze zjistit, zda je chování procesu regulované nebo nepředvídatelné (mimo kontrolu).

Nejjednodušším a již velmi dlouho požívaným typem regulačního diagramu je tzv. Shewhartův regulační diagram pro analýzu jednorozměrných dat v čase, který zobrazuje průběh jedné veličiny v čase. U tohoto diagramu lze přímo jako parametr určit regulační meze UCL a LCL, které budou v diagramu zobrazeny.

Pro vícerozměrná data lze použít tzv. Hotellingův regulační diagram, který byl teoreticky navržen již v roce 1947, tento diagram umožňuje regulovat více veličin pomocí jednoho grafu. Zde již není tak jednoduché určení regulačních mezí UCL a LCL. Tyto meze je potřeba určit na základě dřívějších dat, o nichž jsme přesvědčeni, že představují regulovaný průběh procesu. Na základě takových dat je možné regulační meze určit s využitím výpočtů uvedených v [4].

7.2 Redukce dimensionality dat

Vzhledem k tomu, že data ukládaná do VDÚ jsou často velmi rozsáhlá a často obsahují vysoké množství veličin, je nutné použít redukci dimensionality, aby bylo možné data dále efektivněji využít pro dolování dat.

K redukci dimensionality je primárně využita metoda tzv. analýzy hlavních komponent (PCA - Principal Component Analysis) [14], jejíž hlavním cílem je snížení dimensionality dat s co nejmenší ztrátou informace. Takto upravená data lze efektivněji využít pro účely shlukové analýzy nebo regrese. Pokud je počet komponent, tj. atributů, které jsou výstupem metody PCA roven 2 nebo 3, je možné výstup zobrazit pomocí 2D nebo 3D grafu.

Jako možná alternativa metody PCA byla použita i moderní metoda tSNE (t-distributed stochastic neighbor embedding) [16], jejíž použití je výhodné právě, když je počet komponent velmi nízký. Při zobrazení výsledků je zřejmé, že metoda je schopna data zobrazit tak, že shluky, které se v datech nacházejí, jsou více zřetelné, a je tak větší pravděpodobnost, že budou lépe identifikovány pomocí shlukové analýzy [2].

7.3 Shluková analýza

Shluková analýza slouží k nalezení skupin záznamů v datech, tzv. shluků, pro které platí, že podobnost mezi prvky uvnitř shluku je maximální. Existuje několik typů shlukovacích metod, z nichž v rámci tohoto projekty byly využity dvě.

První použitou metodou je K-Means, která patří do skupiny metod založených na rozdělování. Tato metoda se vyznačuje tím, že je potřeba zadat požadovaný počet shluků, který se má vytvořit, a poté pracuje s centroidy shluků,

které iterativně optimalizuje, až se ustálí u optimálního řešení. Tato metoda je u našeho řešení aplikována na výsledek redukce dimensionality pomocí PCA nebo tSNE. V případě využití 2 nebo 3 komponent mohou být v grafickém zobrazení shluky barevně odlišeny.

Druhou metodou je hierarchické shlukování, jejíž průběh a výsledky jsou zobrazeny pomocí tzv. dendrogramu [10]. Shluky se u této metody sjednocují podle nejkratší vzdálenosti od jednotlivých prvků až po konečné shluky.

7.4 Regrese s využitím metody PCA

Regresní metody lze využít pro predikci hodnoty spojité veličiny. V tomto případě je potřeba mít k dispozici tzv. trénovací vzorek dat, na základě kterého je regresní model vytvořen. Vzhledem k tomu, že regrese je aplikována na výsledek redukce dimensionality metodou PCA, jedná se o postup označovaný jako PCR (principal component regression) [15]. Tento postup je výhodný zejména v situaci, kdy jsou data velmi rozsáhlá a např. metoda nejmenších čtverců u lineární regrese vykazuje velkou chybovost, typicky z důvodu multikolinearity vstupní matice. Pro samotnou validaci modelu je možné použít k-násobnou křížovou validaci nebo její speciální limitní případ, metodu Leave-One-Out.

Na základě předběžných experimentů, kdy byly testovány dostupné regresní metody, byla nakonec zvoleny celkem tři regresní metody. Tou základní je metoda lineární regrese, která se při postupu PCR typicky používá. Jako pokročilejší metody byly využity metody regrese s využitím podpůrných vektorů (SVR - Support Vector Regression) [20] a metoda založená na nejbližším sousedství [5]. Z hlediska jednoduchosti a časové složitosti je nejvýhodnější použít lineární regresi, metoda SVR však dosahuje mírně vyšší přesnosti, i když její výpočet trvá nejdéle.

8 Diskuze výsledků

V průběhu návrhu jádra Validovaného datového úložiště bylo potřeba zvolit vhodnou architekturu řešení a vhodné technologie. Následující text bude diskutovat možnosti návrhu architektury a porovná také aktuálně dostupné technologie s technologiemi zvolenými pro realizaci VDÚ.

8.1 Architektura

Jádro systém VDÚ bylo navrženo pro provoz v distribuovaném prostředí, ať už on-premise nebo v cloud computing. V takovém prostředí mohou být jednotlivé části systému nasazeny ve více instancích podle požadovaného výkonu, tj. systém je škálovatelný. Vícenásobné instance jsou pak dostupné přes jejich zástupce publikující poskytované služby přes programové rozhraní API s operacemi pro synchronní volání, nebo spolu komunikují asynchronně pomocí sběrnice zasílání zpráv. Navržená architektura toto musí respektovat a podporovat.

V realizovaném systému VDÚ byly zvoleny pro komunikaci s jádrem oba zmiňované způsoby – některé komponenty, jako je objektové úložiště, autentizační služba, či plnotextové hledání se volají přes API; jiné, jako je služba pro extrakci obsahu a indexaci dokumentů, přijímají události přes sběrnici. Každý z těchto způsobů komunikace má své výhody a nevýhody [18], avšak pro navrhovaný systém je důležité, že přístup přes API je vhodnější pro připojení klientské aplikace, kterým stačí znát přístupový bod k API, zatímco komunikace přes sběrnici je vhodná pro interakci komponent uvnitř systému bez nutné znalosti koncových bodů takové komunikace. Koncové body, tj. komunikující služby, se při komunikaci přes sběrnici mohou měnit, přesouvat, či se může měnit jejich počet dle aktuální zátěže systému.

8.2 Vlastní úložiště

Nejdůležitější komponentou jádra VDÚ je distribuované úložiště dokumentů. Vzhledem k tomu, že nebylo potřeba ukládat informace o vztazích dokumentů, osoba, aj., ale pouze samotné dokumenty, nebyl pro úložiště vzolen databázový systém, ať už relační nebo pokročilý NoSQL. Úložiště potřebovalo pouze uložit dokumenty a jejich atributy, řídit k nim přístup, monitorovat události a zajistit maximální škálovatelnost a spolehlivost. Uvedeným požadavkům vyhovují distribuované objektové úložiště [8], která navíc často implementují standardizované rozhraní, jako je S3 původně pro Amazon Simple Storage Service[15], a jsou díky tomu pro klienty vzájemně kompatibilní a tedy nahraditelná. Přesto, jsou mezi nimi patrné rozdíly jako v provedení úložiště, tak ve výkonu, který jsou schopny poskytnout [12].

Po prozkoumání dostupných možností bylo zvoleno objektové úložiště OpenIO, které je možno nasadit jako on-premise, tak provozovat jako koupenou službu v prostředí cloud computing[16]. Mezi další zvažované možnosti patřilo MinIO[17], které byl vhodné především pro on-premise nasazení ve virtualizované prostředí, a OpenStack Swift[18], které má dobrou podporu poskytovatele (firma RedHat), avšak hůře se napojuje na sběrnici zpráv pro sledování událostí a ovládání.

8.3 Extrakce, indexace a plnotextové hledání

Nástroj Apache Tika použitý pro extrakci textu z uložených dokumentů je často používanou technologií podobných projektů (např. [21,17,22]). Přestože existují i jiná, mnohdy komerční řešení[19], jejich možnosti a výkon jsou srovnatelné s otevřeným řešením Apache Tika [19]. Díky použití řešení s otevřeným zdrojovým

[15] https://aws.amazon.com/s3/
[16] https://www.openio.io/plans
[17] https://min.io/
[18] https://wiki.openstack.org/wiki/Swift
[19] např. https://xtracta.com/

kódem bylo možno rozšířit službu o extrakci dokumentů laboratorních zpráv, jejichž data jsou později využívána pro dolování. Apache Tika byla integrována do vlastního nástroje, který přijímá události z úložiště přes sběrnici správ, a proto může být spuštěn v libovolném počtu instancí a pokrýt tak zátěž systému (tj. řešení je škálovatelné). Takové vlastní implementace je vhodnější než integrace a použití speciálního rámce pro distribuované výpočty pro běh Apache Tika, jako je tomu např. v [22].

Samotná indexace a plnotextové hledání je pak zajištěno do systému integrovanou službou Elasticsearch nad otevřenou technologií Apache Lucene. I toto řešení patří mezi často používané a doporučované [6]. Jeho kombinace s objektovým úložištěm (OpenIO) a extrakcí dokumentů (Apache Tika) je však jedinečná a není nám znám jiný dostupný distribuovaný systém, který by podobný přístup používal způsobem vhodným pro použití ve VDÚ.

9 Závěr

V této zprávě jsme popsali návrh a implementaci prototypu jádra Validovaného datového úložiště (VDÚ). Jádro umožňuje uložit a opětovně načíst evidované dokumenty, generuje události monitorující aktivity nad úložiště, automaticky extrahuje vlastnosti a obsah uložených dokumentů jako prostý strukturovaný text, ukládá tyto údaje do indexů a umožňuje jejich plnotextové prohledávání a použití pro úlohy dolování z dat. Jádro je implementováno jako skupina mikroslužeb komunikujících pomocí zasílání zpráv a jasně definovaných programových rozhraní. Jádro je možno nasadit jak na lokální stroje (on-premise), tak s využitím infrastruktury cloud computing provozované uživatelem či poskytované třetí stranou (tj. private nebo public cloud computing). Při nasazení jádra na infrastrukturu provozovatele a během provozu je možno celý distribuovaný systém škálovat dle očekávané zátěže.

Navržené a implementované řešení odpovídá technickým a uživatelským požadavkům pro VDÚ. Díky použitým technologiím a specifickému způsobu jejich propojení se jedná se o unikátní a inovativní řešení. Velký potenciál má především část zaměřená na dolování dat z extrahovaného obsahu uložených dokumentů – zde se očekává budoucí vývoj dle aktuálních potřeb zadavatele i uživatelů systému.

Tato zpráva vznikala v průběhu projektu „Validované datové úložiště“ (VDÚ), MPO ČR, TRIO, číslo projektu FV40176, v letech 2019 až 2021. Zpráva vznikala průběžně během řešení příslušných etap a byla aktualizována na konci projektu tak, aby odpovídala finálnímu stavu návrhu a implementace. Zpráva může obsahovat důvěrné údaje podléhající obchodnímu tajemství a její distribuce či kopírování proto vyžaduje souhlas autorů zprávy.

Reference

1. Almutairi, A.; Sarfraz, M.; Basalamah, S.; aj.: A Distributed Access Control Architecture for Cloud Computing. *IEEE Software*, ročník 29, č. 2, 2012: s. 36–44, doi:10.1109/MS.2011.153.

2. Anowar, F.; Sadaoui, S.; Selim, B.: Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE). *Computer Science Review*, ročník 40, 02 2021, doi:10.1016/j.cosrev.2021.100378.
3. Bernard, J.: *Aplikace pro řízený přístup ke vzdáleným dokumentům pro GNU/Linux*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2021, vedoucí práce Rychlý Marek.
URL https://www.fit.vut.cz/study/thesis/23590/
4. Bersimis, S.; Psarakis, S.; Panaretos, J.: Multivariate Statistical Process Control Charts: An Overview. *Quality and Reliability Engineering International*, ročník 23, 08 2007: s. 517 – 543, doi:10.1002/qre.829.
5. Burba, F.; Ferraty, F.; Vieu, P.: k-Nearest Neighbour method in functional nonparametric regression. *Journal of Nonparametric Statistics*, ročník 21, 05 2009: s. 453–469, doi:10.1080/10485250802668909.
6. Divya, M. S.; Goyal, S. K.: ElasticSearch: An advanced and quick search technique to handle voluminous data. *Compusoft*, ročník 2, č. 6, 2013: str. 171.
7. Endo, P. T.; de Almeida Palhares, A. V.; Pereira, N. N.; aj.: Resource allocation for distributed cloud: concepts and research challenges. *IEEE Network*, ročník 25, č. 4, 2011: s. 42–46, doi:10.1109/MNET.2011.5958007.
8. Factor, M.; Meth, K.; Naor, D.; aj.: Object storage: the future building block for storage systems. In *2005 IEEE International Symposium on Mass Storage Systems and Technology*, 2005, s. 119–123, doi:10.1109/LGDI.2005.1612479.
9. Feranec, A.: *Application for Controlled Access to Remote Documents for Microsoft Windows*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2021, vedoucí práce Rychlý Marek.
URL https://www.fit.vut.cz/study/thesis/23591/
10. Fernández, A.; Gomez, S.: Solving Non-Uniqueness in Agglomerative Hierarchical Clustering Using Multidendrograms. *Journal of Classification*, ročník 25, 02 2008: s. 43–65, doi:10.1007/s00357-008-9004-x.
11. Furht, B.; Escalante, A.; aj.: *Handbook of cloud computing*, ročník 3. Springer, 2010.
12. Gadban, F.; Kunkel, J.: Analyzing the Performance of the S3 Object Storage API for HPC Workloads. *Applied Sciences*, ročník 11, č. 18, 2021, ISSN 2076-3417, doi:10.3390/app11188540.
URL https://www.mdpi.com/2076-3417/11/18/8540
13. Hemmings, M.: *Thin client collaborative visualizations using the distributed cloud*. Dizertační práce, 2016.
14. Jolliffe, I.: Principal component analysis. 2nd ed. *http://lst-iiep.iiep-unesco.org/cgi-bin/wwwi32.exe/[in=epidoc1.in]/?t2000=017716/(100)*, ročník 98, 10 2005, doi:10.1002/0470013192.bsa501.
15. Kelechi, C.: Regression and Principal Component Analyses: a Comparison Using Few Regressors. *American Journal of Mathematics and Statistics*, ročník 2, 01 2012: s. 1–5, doi:10.5923/j.ajms.20120201.01.
16. van der Maaten, L.; Hinton, G.: Viualizing data using t-SNE. *Journal of Machine Learning Research*, ročník 9, 11 2008: s. 2579–2605.

17. Mattmann, C. A.; Oh, J.-H.; Palsulich, T.; aj.: DRAT: An Unobtrusive, Scalable Approach to Large Scale Software License Analysis. In *2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, 2015, s. 97–101, doi:10.1109/ASEW.2015.14.
18. Selim, M. R.; Endo, T.; Goto, Y.; aj.: A Comparative Study Between Soft System Bus and Traditional Middlewares. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, editace R. Meersman; Z. Tari; P. Herrero, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ISBN 978-3-540-48276-5, s. 1264–1273.
19. Skluzacek, T. J.; Wong, R.; Li, Z.; aj.: A Serverless Framework for Distributed Bulk Metadata Extraction. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '21, New York, NY, USA: Association for Computing Machinery, 2020, ISBN 9781450382175, str. 7–18, doi:10.1145/3431379.3460636.
URL https://doi.org/10.1145/3431379.3460636
20. Smola, A.; Schölkopf, B.: A tutorial on support vector regression. *Statistics and Computing*, ročník 14, 08 2004: s. 199–222, doi:10.1023/B%3ASTCO.0000035301.49549.88.
21. Totaro, G.; Bernaschi, M.; Carbone, G.; aj.: ISODAC: A high performance solution for indexing and searching heterogeneous data. *Journal of Systems and Software*, ročník 118, 2016: s. 115–133, ISSN 0164-1212, doi:https://doi.org/10.1016/j.jss.2015.11.043.
URL
https://www.sciencedirect.com/science/article/pii/S0164121215002678
22. Verma, R.; Mattmann, C.: Extending Spark Analytics through Tika-Based Information Extraction and Retrieval. In *2015 IEEE International Conference on Information Reuse and Integration*, 2015, s. 215–218, doi:10.1109/IRI.2015.43.
23. Wang, Q.; Wang, C.; Ren, K.; aj.: Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems*, ročník 22, č. 5, 2011: s. 847–859, doi:10.1109/TPDS.2010.183.