Alati za analitiku velike količine podataka (big data)

Stihović, Vedran

Sveučilište Jurja Dobrile u Puli
Odjel za ekonomiju i turizam
«Dr. Mijo Mirković»

VEDRAN STIHOVIĆ

Alati za analitiku velike količine podataka (Big data analytics tools)

Diplomski rad

Pula, rujan 2015.

namjerno ostavljeno prazno

Sveučilište Jurja Dobrile u Puli
Odjel za ekonomiju i turizam
«Dr. Mijo Mirković»

VEDRAN STIHOVIĆ

Alati za analitiku velike količine podataka (Big data analytics tools)

Diplomski rad

JMBAG:
0303017130

Studijski smjer i status:
Poslovna informatika,
izvanredni student

Kolegij:
Informacijski sustavi u potpori upravljanju i odlučivanju

Mentor:
Prof.dr.sc. Vanja Bevanda

Pula, rujan 2015.

Sadržaj

1. Uvod

Velika količina podataka (Big data) relativno je nov pojam, a mnoge organizacije trenutno ne znaju iskoristiti prednosti ekstrakcije uvida i vrijednosti iz njih. Od 2008. godine više je stvari povezano na Internet od ljudi, naginjući fokus prema realizaciji pametnih gradova. Organizacije trebaju postati svjesne načina i prednosti implementacije analitike velike količine podataka, a što bi im višestruko koristilo u vremenu kada su bilijuni korisnika umreženi i aktivni na mreži, te istovremeno spojeni u sveobuhvatnu mrežnu infrastrukturu koja se popularno naziva Internet stvari (IoT)[1]. Iako se u akademskim krugovima mnogo govori o analitici velike količine podataka, ona u praksi još uvijek nije široko rasprostranjena i upotrijebljena.

Velika količina podataka i analitika spominje se unutar mnogih disciplinarnih skupina, a od kojih neke jesu: Statistika, ekonomija, sociologija, psihologija i ostalo. Ispravna upotreba i analitika velike količine podataka ima potencijal donijeti mnoge prednosti i riješiti razne poslovne i društvene probleme. Iako je analitika velike količine podataka postao ključni poslovni potencijal, obećavajući nastanak nove vrste analitike, fenomen i dalje nije potpuno istražen sa strateške i organizacijske perspektive. U nastavku su opisani pokretači velike količine podataka i karakteristike.

Informacije su u trenutku pisanja rada široko rasprostranjene u količinskom smislu zahvaljujući mnogim faktorima, a od kojih su neki: široka upotreba društvenih mreža, velika dostupnost nestrukturiranih podataka kao što su slike, video i zvuk. Nestrukturirani podaci imaju razne izvore, a od kojih su neki: senzori, kamere, ostali uređaji za nadzor, Voice over IP (VOIP) sistemi i ostalo. Veliku količinu podataka definirao je Edd Dumbill 2012. godine na sljedeći način:

[1] Engl. Internet of Things, IOT. Predstavlja koncept spajanja svih uređaja na Internet, i/ili među sobom; u nastavku rada koristit će se engl. skraćenica IoT. Izvor: Morgan (2014)

Dumbill (2012) definira veliku količinu podataka kao podatke koji nadmašuju kapacitete procesiranja uobičajenih sistema baza podataka. Podaci su preveliki, kreću se prebrzo, ili se ne uklapaju u strukturu arhitekture baze podataka. Da bi se stekla korist od takvih podataka, potrebno je pronaći alternativni način procesiranja istih.

Schneider (2013) tvrdi da velika količina podataka ima jednu ili više sljedećih karakteristika:

1. Sadržavaju veliku količinu informacija,
2. Sačinjeni su od raznih tipova datoteka i formata,
3. Stvoreni su od mnogih izvora,
4. Zadržavaju se dug period vremena,
5. Upotrijebljeni su od strane novih i inovativnih aplikacija.

Velika količina podataka, zbog svoje veličine i kompleksnosti, traži nove mogućnosti, alate i modele za upravljanje informacijama i njihovim vanjskim i unutarnjim tokovima. Transformacija velike količine podataka u strateške resurse preduvjet je za zadovoljenje budućih zahtjevnih kupčevih potreba. S druge strane, izazovi koje velika količina podataka postavlja pozivaju na promjenu poslovnih modela i ljudskih resursa.

U trenutku pisanja rada, četiri su dimenzije prepoznate kao karakteristike velike količine podataka, a Morabito (2015) ih navodi:

1. Volumen – jest prva dimenzija, a odnosi se na dostupnost izuzetno velike količine podataka (terabajti, petabajti), a koja se može pohraniti za poduzeće, a korištenjem Interneta. Primjerice, 12 terabajta Tweet poruka sa društvene mreže Twitter nastaju svakoga dana, te se mogu upotrijebiti za poboljšanje analize kupčeva stava o proizvodu.
2. Brzina – jest druga dimenzija, a odnosi se na dinamiku velike količine podataka, a specifično na vremenski osjetljivu prirodu velike količine podataka. Brzina stvaranja i korištenja takvih podataka često je gotovo u realnom vremenu.

3. Raznolikost – jest treća dimenzija, a odnosi se na vrste podataka koji su dostupni. Uz strukturirane podatke čijom se analitikom često bave informacijski sistemi u organizacijama, većina podataka koji tvore Big Data jesu polu-strukturirani ili nestrukturirani podaci. U takve podatke ubrajamo slike, video datoteke, audio datoteke, log datoteke, tekstualne datoteke i ostalo, a što je postavljeno na društvene mreže i ostala mjesta.
4. Dostupnost – jest četvrta dimenzija, a odnosi se na izuzetno velik broj kanala koji su dostupni kompaniji za rast i razvoj vlastitih podatkovnih i informacijskih resursa.
5. Istinitost – jest peta dimenzija, a koja nije široko prihvaćena. Odnosi se na kvalitetu podataka i povjerenje vezano uz dostupne podatke. U skladu sa time, ova dimenzija je relevantna za stratešku upotrebu velike količine podataka, te analitiku koju vrše kompanije.

Morabito (2015) navodi pokretače nastanka velike količine podataka – računalstvo u oblaku, društvene mreže, mobilne tehnologije i Internet of Things. U vrijeme pisanja rada, računalstvo u oblaku smatra se glavnim pokretačem velike količine podataka, jer rastuća količina podataka zahtjeva skalabilne sisteme za upravljanje bazama podataka (DBMS – DataBase Management System). S druge strane, računalstvo u oblaku često pred kompanije postavlja dvije opcije. Prva opcija uključuje plaćanje skupih komercijalnih rješenja, dok druga opcija podrazumijeva solucije različite od klasičnih tehnologija baza podataka. NoSQL (Not Only SQL) sistemi pohrane podataka su u sve većoj upotrebi, često ne zahtijevajući fiksne tablične šeme, dok istovremeno ne zadovoljavaju u potpunosti tradicionalna ACID (Atomicity, Consistency, Isolation, Durability) svojstva. MapReduce i slobodna platforma za računanje Hadoop postigli su veliku popularnost i značaj kod zadataka procesiranja, stvaranja i analize setova velike količine podataka.

Velika količina podataka, kako tvrdi Morabito (2015), odnosi se na informacijske resurse koje se može arhivirati, kojima se može upravljati i iskoristiti za donošenje

odluka, strategija i inovacija. Marketing može upotrijebiti veliku količinu podataka za povećanje kupčevog zadovoljstva, rudarenje i analizu mišljenja o proizvodu ili usluzi, a korištenjem metoda analitike društvenih mreža. Javni sektor može poboljšati detekciju prijevara spajanjem velikog broja baza podataka, a s druge strane, za transparentnost i definiranje odgovornosti za vladine i administrativne aktivnosti; što je u skladu sa inicijativama otvorenih podataka, te participacijom građana u politici. Odluke temeljene na analitici podataka smatraju se boljim odlukama od onih koje se ne temelje na činjenicama, već intuiciji pojedinaca ili stručnjaka. Za potrebe razumijevanja daljnjeg sadržaja u radu potrebno je razjasniti pojmove IaaS, SaaS i PaaS, a čiji je zajednički akronim SPI. SaaS (engl. Software as a Service) jest softverski distribucijski model pri kojem su aplikacije poslužene od strane pružatelja usluge, te omogućene korisniku za korištenje putem mreže (najčešće putem Interneta). PaaS (Platform as a Service) jest paradigma dostavljanja operativnog sustava i pripadnih usluga putem Interneta, a bez potrebe za skidanjem ili instalacijom. IaaS (Infrastructure as a Service) sadrži outsource-ing opreme korištene za podršku operacija; uključujući pohranu, sklopovlje, poslužitelje i mrežnu opremu. (Rouse 2012)

U radu će se dokazati da se upotrebom analitičkih platformi i tehnika analitike velike količine podataka mogu poboljšati performanse organizacija i organizacijskih jedinica, uključujući upravljanje i odlučivanje. Cilj rada jest usporediti i istražiti karakteristike analitičkih platformi, te mogućnosti, benefite i rizike njihove primjene u poslovanju. Dodatno, istražit će se preduvjeti i pravila prilikom uvođenja analitičkih metoda u kompaniju. Metoda deskripcije koristit će se u uvodnom dijelu rada, dok će se komparativna metoda koristiti za usporedbu sličnosti i različitosti platformi za analitiku velike količine podataka. Induktivna metoda koristit će se za donošenje suda o kvaliteti analitičkih platformi, te donošenje općeg suda o smislu i karakteristikama upotrebe analitike velike količine podataka u poslovanju. Korištenjem metode analize, detaljno će se istražiti analitičke platforme, a analitikom njihovih specifičnih komponenti i njihovih međuodnosa. Metoda kompilacije koristit će se za sakupljanje

stavova različitih autora, a vezanih za temu istraživanja. Za prikaz i usporedbu performansi analitičkih platforma i ostalih dijelova rada koristit će se statistička grafička metoda. Metoda modeliranja koristit će se za prikaz specifičnih međuodnosa komponenata obrađenih u tekstualnom dijelu rada.

Rad je podijeljen u sedam tematskih poglavlja. Prvo poglavlje jest uvod u tematiku koja će se obraditi u radu, te će se u njemu pronaći osnovne definicije primarnih pojmova koji se protežu kroz čitav rad. U drugom poglavlju bit će opisani različiti slučajevi korištenja velike količine podataka, te prednosti koje njihova primjena donosi. U trećem poglavlju bit će opisani trendovi u analitici velike količine podataka, te će se obraditi Hadoop platforma, njezine komponente, smjernice i dobra praksa pri odabiru prikladne distribucije, te će konačno biti opisan detaljan uvid u način funkcioniranja HDFS-a i MapReduce procesa. U četvrtom će poglavlju biti opisani alati i platforme za analitiku veliku količine podataka, te usporedba njihovih performansi. U ovom će se poglavlju obraditi Apache Hive, Impala, Presto, Spark SQL, Apache Drill, Hawk i ostale manje popularne platforme. Peto poglavlje je zaključak, gdje će biti iznesena konkluzija svega prethodno obrađenog u radu. Šesto poglavlje bit će popis literature, dok će sedmo poglavlje biti sažetak rada na hrvatskom i engleskom jeziku.

2. Upotreba, prednosti i rizici primjene velike količine podataka

Konkurentska prednost može se definirati kao superiornost kompanije u stvaranju profita, a u odnosu na konkurenciju. Takva prednost bila je pozicija kompanije u sektoru, i njena sposobnost da tu poziciju brani. S druge strane, konkurentska se prednost može definirati kao efikasno upravljanje resursima, a što uključuje i ljudske resurse, te smanjenje transakcijskih troškova unutar lanca vrijednosti. Uspješno upravljanje zahtjeva IT alate, a koji će pružati podršku menadžmentu u efikasnom upravljanju i donošenju odluka baziranih na činjenicama iznad intuicije. Kompanije koje navode konkurentnost kao izvor konkurentske prednosti svoj fokus često usmjeravaju na najbitnije sposobnosti; te vrijedne, rijetke i nezamjenjive resurse. Takva perspektiva cijenila je efikasnost procesa i upravljanje znanjem, te kombinaciju resursa kao najbitnije faktore, a koji su zahtijevali iskustvo. U ovom slučaju uloga IT-a bila je podrška menadžmentu u donošenju efektivnijih odluka, a na što efikasniji način. Popularizacijom e-poslovanja i „maverick“ poduzetnika, poslovanje se počelo fokusirati na fleksibilnost, a što u korijenu potječe od Schumpeterove teorije o kreativnoj destrukciji. U takvoj teoriji kompanije koje se ne prilagođavaju propadaju, a nove fleksibilne kompanije nastavljaju poslovanje. Fleksibilnost se odnosi na mnoga područja od kojih su neka: promjenjive kupčeve želje, promjene u industriji nastale dolaskom novih kompanija, nepredvidivi potezi konkurencije i ostalo. S vremenom je brzina postala glavni faktor u natjecanju sa konkurencijom, dok je istovremeno sporo donošenje odluka izgubilo značaj koji je imalo. Trendovi vezani uz veliku količinu podataka samo će povećati značaj brzine, iako predstavljaju tehnički izazov za pohranu i procesiranje. Dijeljenje infrastrukture jest novi trend koji samo jača, a što se može zamijetiti kod dijeljenih usluga i „oblaka“, a što će se možda vidjeti i kod pohrane velike količine podataka u budućnosti. (Morabito 2015)

Promjene, te njihov značaj i opseg su globalizacijom postale izuzetno nepredvidive. Prethodni autor navodi nekoliko primjera – Facebook je promijenio online trgovanje,

dok je Skype promijenio telekomunikacije. Konkurentska prednost u takvoj vrsti dinamičnog okruženja nije niti nešto što kompanija posjeduje, niti nešto što može zaštititi. Inovacija je postala ključna za uspjeh poduzeća, a što je u suštini promjena. Takvo turbulentno okruženje zahtjeva strategije umrežavanja. Dok su u prošlosti konglomerati, lobiji, ugovori i dogovori bili uobičajeni dio poslovnog života, danas su tu ulogu preuzele otvorene mreže dobavljača, zaposlenika i kupaca. Kompanije i javnost su danas međuovisni više nego ikad. Kupci, dobavljači ili samo ljubitelji proizvoda propagiraju i dijele dalje informacije o kompaniji, te joj na taj način podižu ugled. Takve aktivnosti generiraju poslovnu vrijednost besplatno. Dijeljenje na društvenim mrežama primjer je toga. Takve aktivnosti stvaraju velike količine podataka na društvenim mrežama, a koji ukazuju na socijalno ponašanje velikog broja ljudi, odnosno potencijalnih kupaca. Obradom tih podataka kompanije mogu razviti svoju strategiju na ispravan i isplativ način.

Velika količina podataka može biti upotrijebljena od strane marketinga za razvijanje strategija zadržavanja postojećih kupaca i poboljšanja ponavljajućih prodaja. Nove tehnike poboljšanja odnosa s kupcima uključuju „igre“, a koje rezultiraju povećanom lojalnošću. Poduzeća više ne moraju razdvajati tržišta u velike demografske skupine. Umjesto toga, kompanije mogu koristiti nove analitičke alate, te uz pomoć njih analizirati velike količine podataka, otkriti nove niše, ili čak dodatno, sa većom preciznošću, segmentirati već postojeće skupine u manje i bliskije. Velika količina podataka marketingu obećava masovnu prilagodbu. S obzirom da je većina objava spontano mišljenje na društvenim mrežama, takvi su podaci mnogo vrjedniji nego ispitivanja putem upitnika i slično. Analizom takvih podataka mogu se otkriti zahtjevi za nove proizvode i/ili usluge. Marketing želi predvidjeti promjene u kupčevim željama prije nego su se one dogodile, a fina analiza velike količine kupčevih mišljenja može rezultirati takvim informacijama. S obzirom da entiteti na društvenim mrežama često imaju opis sebe, svojih ukusa, prebivališta, starosti i ostalog, moguće je detaljno analizirati i povezati podatke sa mišljenjima, a posljedično generirati precizne i

konkretne skupine kupaca, a kod kojih izvori informacija nisu nepoznati. Konačno, interesne skupine se danas mogu grupirati po IP adresama, a koje otkrivaju gdje se nalazimo i ostalo. (Morabito 2015)

Po istome autoru, kombinacija izvora velike količine podataka sa ostalim modernim tehnologijama može potaknuti inovacije temeljene na dizajnu. Takve proizvode kupci ne očekuju, ali ih zavole. Primjerice, Apple nije promijenio način kako ljudi telefoniraju, već način kako koristimo telefone i kako ih doživljavamo. Može se reći da pametni telefoni danas više niti nisu telefoni, već višenamjenski uređaji, a koji se mogu okarakterizirati i kao računala. Na proizvode se više ne gleda kao na izlazne rezultate neke bezimene kompanije, već kao na perjanicu kompanije i ljudi koji su ih proizveli. Svakako se na proizvode gleda kao i na pokazatelj osobina i stavova vlasnika. Dakle, inovacije nisu samo generiranje novih outputa, već i dijeljenje zajedničkih značenja sa kupcima.

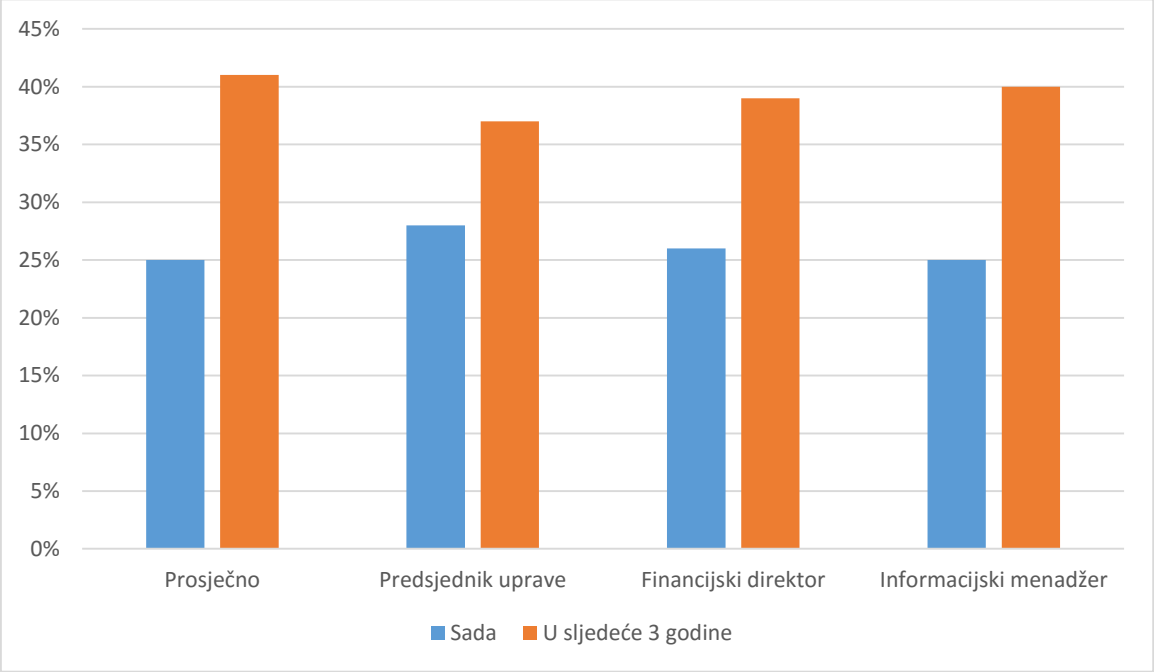
Fawcett i Provost (2013) tvrde da analitika velike količine podataka sa ciljem poboljšanja donošenja odluka nadopunjuje pristup koji se oslanja samo na intuiciju menadžera. Istraživanja pokazuju da produktivnost kompanije raste proporcionalno sa povećanjem iskorištavanja podatkovnih resursa. Dodatno, korištenjem analitike velike količine podataka povećavaju se svi ekonomski pokazatelji konkurentnosti kompanije. Unutar prikazanog konteksta, razlikuju se dvije vrste odluka: a) odluke za koje je nužno postići otkrića unutar podataka, i b) odluke koje se ponavljaju, a koje imaju korist i od najmanjeg porasta preciznosti u donošenju odluka temeljem analitike velike količine podataka. Analitika velike količine podataka vodi ka automatizaciji donošenja mnogih odluka od strane računalnih sustava. Financije i telekomunikacije prvi su prihvatili automatizaciju tog tipa. Kompanije u mnogim industrijama koriste podatkovne resurse sa ciljem povećanja konkurentske prednosti, profita i smanjenja troškova.

Neke kompanije su rudarenje podataka shvatile kao ključnu stratešku komponentu, primjerice Facebook i Twitter. Isti autori nastavljaju tvrdeći da menadžeri trebaju

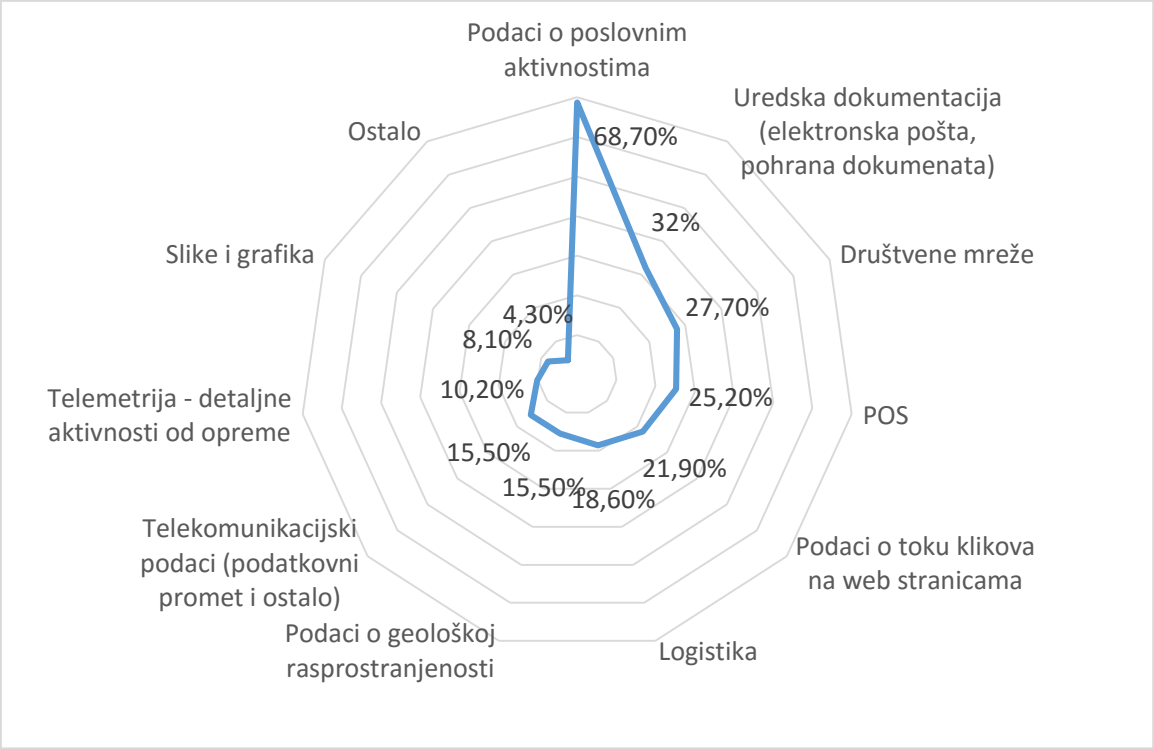
upravljati timovima i projektima za analitiku velike količine podataka, marketing treba razumjeti kampanje temeljene na analitici velike količine podataka, te da poslovne strategije moraju biti sposobne iskoristiti podatke. Projekti analitike velike količine podataka protežu se na sve poslovne jedinice. Ukoliko zaposlenici ne razumiju temelje analitičkog razmišljanja, oni neće shvatiti što se događa u kompaniji koja se oslanja na analitiku velike količine podataka. Donošenje odluka u nepripremljenom okruženju može biti gore od donošenja odluka bez provođenja analitike koju se ne razumije. Donositelji odluka moraju biti u bliskoj interakciji sa analitičarima podataka, da bi donošenje odluka od strane menadžmenta bilo što bolje. Dodatno, u kompanijama u kojima menadžment ne razumije ulogu analitičara podataka, dolazi do gubljenja vremena i nepotrebnog truda, a u najgorem slučaju do donošenja neispravnih odluka.

Kompanije koje ulažu u analitiku velike količine podataka mogu učiniti pogrešku, i posljedično izgubiti povrat na investiciju ukoliko ne znaju uključiti podatke u kompleksno donošenje odluka. Temeljni koncept glasi: Ekstrakcija korisnih znanja iz podataka, sa ciljem rješavanja poslovnih problema, može se provoditi sistematski praćenjem procesa čije su faze dobro definirane. Dodatno, smatra se da rezultati analitike podataka zahtijevaju pažljiv odabir konteksta unutar kojeg će se upotrijebiti. Odnos između poslovnih problema i analitičkih solucija često se može raščlaniti na pratljive potprobleme putem analitike očekivanih vrijednosti. Postoje razni alati za rudarenje podatka, ali poslovni problemi rijetko dolaze dovoljno pripremljeni za njihovu upotrebu, pa je zbog toga potrebno raščlaniti poslovni problem na manje dijelove. Informacijske se tehnologije mogu upotrijebiti za pronalaženje informativnih podataka unutar velike količine podataka. Smatra se da su entiteti, slični temeljem poznatih obilježja i atributa, slični i temeljem nepoznatih obilježja i atributa. Svake godine raste broj načina računalnog izračunavanja sličnosti. Dugotrajna analitika seta podataka donijet će rezultate, no rezultati ne garantiraju generalizaciju izvan promatranog seta. Potrebno je pronaći način za izbjegavanje fokusiranja unutar samo

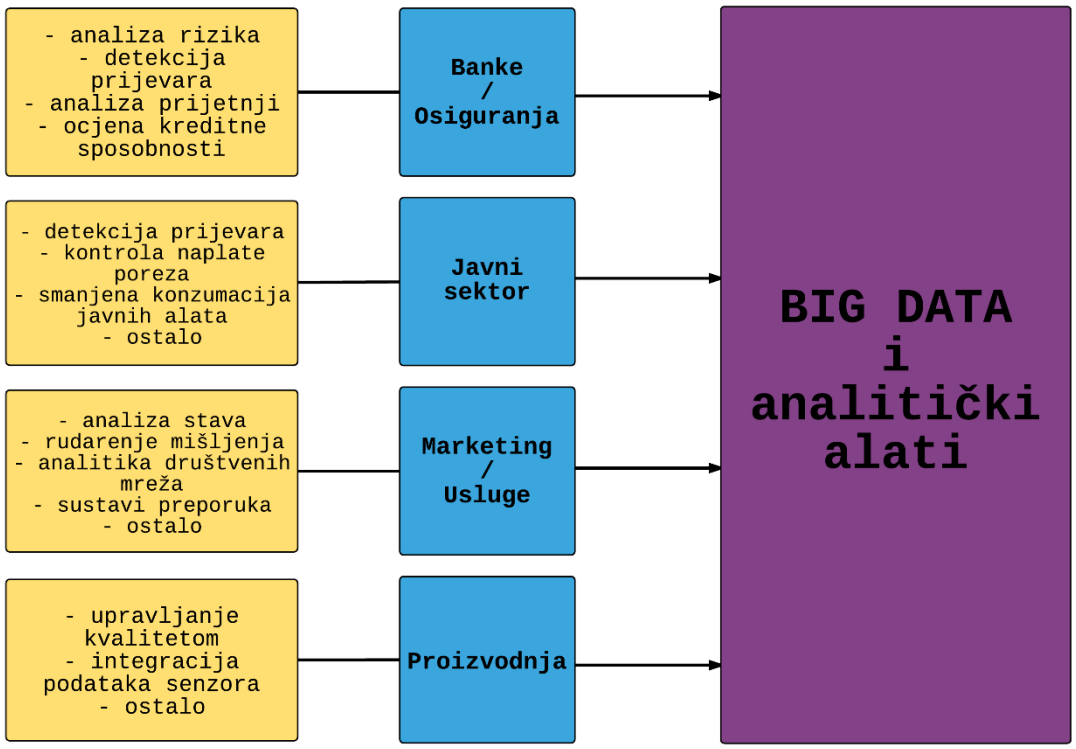
jednog seta, a za što postoje procesi, algoritmi i evaluacijske metode. Dodatno, prije donošenja brzih zaključaka potrebno je uočiti moguće skrivene faktore. Sam pronalazak korelacije među podacima ponekad nije dovoljan, jer mnoge kompanije žele saznati kako promijeniti i utjecati na uzrok takvih podataka, a sa ciljem da uzroke i ponašanja oblikuju i usmjere na sebi povoljan način. (Fawcett i Provost 2013) Konačno, 85% ispitanika, tvrdi Economist Intelligence Unit (2015), smatra da problem kod upotrebe velike količine podataka nije količina, već sposobnost analitike i reagiranja u realnom vremenu, te hardverske mogućnosti koje kompanija ima na raspolaganju.



Graf 1. Prikaz mišljenja članova kompanija, koliko smatraju da je upotreba velike količine podataka poboljšala performanse kompanije do sada, a koliko smatraju da će se poboljšati performanse u naredne tri godine. Izvor: Economist Intelligence Unit (2015)



Graf 2. Prikaz mišljenja članova kompanija, koje setove velike količine podataka smatraju najvrjednijima za organizaciju. Izvor: Economist Intelligence Unit (2015)



Slika 1. Prikaz primjene velike količine podataka u raznim slučajevima korištenja. Prilagođeno iz (Morabito 2015)

Trenutno se velika količina podataka poistovjećuje sa podacima društvenih mreža, dok s druge strane većina bitnih inovacija naginje prema Internetu stvari (IoT). Fokus je postavljen prema svijetu povezanih ljudi i objekata, mašina koje su uvijek uključene, upotrebljive i svjesne okoline putem svojih senzora. Tako u trenutku pisanja rada Google testira samostalne automobile, koji se kreću bez vozača, a korištenjem GPS-a (engl. Global Positioning System, GPS), mobilnih mreža, bežičnih mreža, ali i međukomunikacije povezanih mašina koje svojim senzorima tvore mrežu i infrastrukturu za sigurno kretanje takvih automobila, ali i njihovu svjesnost o prometu i okolini. U bliskoj budućnosti moguće je očekivati da će takav automobil na poziv, bilo putem aplikacije ili drugačije, samostalno pokupiti vlasnika na željenoj lokaciji. (Morabito 2015)

Isti autor tvrdi da u vrijeme velike količine podataka nisu pristupačni i otvoreni samo podaci i mišljenja, već i poslovne ideje. Kickstarter.com je, primjerice, zajednica koja omogućava ljudima da doniraju, naruče unaprijed ili kupe dionice firme koja im se svidi, a koja nije nužno uspješna u tom momentu, već se kategorizira u start-up skupinu. Nažalost, u vrijeme pisanja rada, nije moguće sudjelovati u zajednici sa projektima iz Republike Hrvatske. Korištenje velike količine podataka još uvijek nije suviše kreativno. Većina ljudi ima kao pokretač stare koncepte povećanja vrijednosti, želeći raditi stare stvari efikasnije i efektivnije, a korištenjem novih koncepata i analitikom velike količine podataka. Nedostatak vještina, kulturne prepreke, procesi i struktura, te raspoloživa tehnologija, glavni su problemi na koje organizacije dolaze prilikom migracije prema korištenju velike količine podataka u svojem poslovanju.

Ukoliko vlade planiraju učiniti preokret prema upotrebi velike količine podataka, morat će pronaći balans između napretka i izazova koje velika količina podataka predstavlja. S druge strane, takav fokus će prije svega biti usmjeren na pametne gradove. Europska inicijativa za pametne gradove postavila je šest faktora temeljem kojih je moguće ocijeniti spremnost gradova za uklapanje u zajednicu pametnih gradova. Nerijetko se događa da je buđet jedan od problema, te stvara napetosti između urbanih i ruralnih zona. Za gradske usluge se u takvim gradovima odgovornost polaže na mašine, partnerstva sa privatnim kompanijama, ali i na javnost.

Tablica 1. Faktori za audit kandidata za pametne gradove. Izvor: Morabito (2015)

Faktori	Opis
Pametna ekonomija	Inovativni duh i poduzetništvo, produktivnost, fleksibilnost radne snage, sposobnost preobrazbe i međunarodna povezanost
Pametna mobilnost	Lokalna i međunarodna dostupnost, dostupnost ICT infrastrukture, inovativni transportni sustavi
Pametno okruženje	Upravljanje polucijom i održivim resursima, zaštita okoliša, atraktivni prirodni uvjeti
Pametni ljudi	Cjeloživotno učenje, fleksibilnost, kreativnost, kvalifikacije, etnički i socijalni pluralitet, otvorenost uma i kozmopolitanizam
Pametan život	Zdravstveni uvjeti, osobna sigurnost, kulturne ustanove, edukacijske ustanove, kvaliteta kućanstava, turističke atrakcije i društvena kohezija
Pametna vlada	Sudjelovanje u donošenju odluka, javne i socijalne usluge, transparentnost, političke strategije i perspektive

Fokus na efikasnost i godine zapošljavanja ljudi i menadžera sa fokusom na smanjenje troškova osiromašilo je javni sektor od inovativnih ljudskih resursa, i onih iskusnih. Politika niskih plaća odbija ljude sa talentima, te čini upravljanje odnosa sa vanjskim partnerima težim, a pogotovo ako i oni slijede iste politike što manjih troškova. Javni sektor trebat će usmjeriti svoj fokus na zapošljavanje pametnih ljudi koji će sklapati uspješna partnerstva sa privatnim i javnim sektorom. Ako se vlade usmjere na stvaranje pametnih gradova, trebat će privlačiti ljude koji imaju osobine navedene u prethodnoj tablici, pod kategorijom „Pametni ljudi“. Upravljanje uspješnim partnerstvima zahtjeva nove upravljačke sposobnosti. Morabito (2015) navodi neke od njih:

1. kreiranje izvješća na otvoren i komunikativan način,
2. izgradnja povjerenja radnjama i riječima,
3. kreativno rješavanje sukoba i rješavanje problema,
4. želja za promjenama, i
5. dobrodošla međuovisnost.

Glavne želje otvorene vlade su sudjelovanje javnosti u političkim procesima i njihova uključenost u samoposlužne javne usluge. To će zahtijevati povećane razine zainteresiranosti, znanja, i zrelosti javnosti, ali i nove načine vladinog sudjelovanja. Edukacija je jedan način postizanja prethodno navedenog, dok su online participacije drugi način. Alati za analitiku velike količine podataka mogu pomoći u integraciji najboljih praksi kod rada policije na mrežnom kriminalu, a istovremeno zakone učiniti jasnijima i javnosti i policijskim službenicima.

3. Trendovi u analitici velike količine podataka

Mitchell (2014) tvrdi da tehnologije za obradu velike količine podataka napreduju velikom brzinom, te da je rizik upotrebe velike količine podataka iz dana u dan sve manji. Sa mnogim rastućim trendovima na područjima analitike i velike količine podataka, organizacije trebaju stvoriti uvjete u kojima će analitičari i znanstvenici moći eksperimentirati. Potrebno je pronaći način za testiranje i ocjenu, te uvođenje prototipa, a tako da bi se u budućnosti testirane tehnologije mogle integrirati u poslovanje. IT menadžeri i implementatori ne smiju odbijati testiranje novih tehnologija samo zato jer su još u ranoj fazi razvoja. U početku testiranje trebaju vršiti samo najstručniji analitičari, a nakon njih napredni korisnici i IT sektor trebaju donijeti odluku o uvođenju testiranih tehnologija u organizaciju. Konačno, uvođenje takvih tehnologija i sustava ne smije biti brzopleto, već precizno testirano i proračunato.

Analitika velike količine podataka u oblaku novi je trend. Hadoop, platforma i skup alata za procesiranje velike količine podataka, kreiran je da radi na grozdovima[2] fizičkih strojeva, a bit će detaljnije objašnjen u sljedećem poglavlju. Takav pristup je nešto što se danas mijenja, a usmjerava se na procesiranje podataka u oblaku, a pripadnim tehnologijama koje su u sve većem razvoju. Primjeri platformi za procesiranje u oblaku jesu Redshift i Kinesis (Amazon), BigQuery (Google), Bluemix (IBM) i ostali. Mitchell (2014) smatra da će budućnost velike količine podataka biti hibrid procesiranja u oblaku i na vlastitim strojevima.

Isti autor navodi kao primjer kompaniju Smarter Remarketer, a koja pruža usluge analitike, segmentacije i marketinga, prebacila se sa lokalnog rješenja Hadoop-a i MongoDB infrastrukture baza podataka na Redshift, Amazon-ovu uslugu u oblaku. Kompanija prikuplja podatke o online kupovini, te demografske podatke o kupcima,

[2] Hadoop grozd odnosi se na skupinu strojeva i mrežne opreme koji rade usklađeno, a sa ciljem analitike velike količine podataka.

te podatke o ponašanju u realnom vremenu. Nad takvim se podacima vrši analitika, a kao output kreiraju poruke koje kod kupca trebaju izazvati željenu reakciju na proizvod. Ponekad takav impuls vrše u realnom vremenu. Redshift platforma kompaniji je smanjila troškove, a pogotovo u situacijama kada su im potrebne velike sposobnosti izvještavanja o strukturiranim podacima. S obzirom da je platforma u obliku plaćene usluge, ona je istovremeno proširiva (fleksibilna) i jednostavna za korištenje. Novi trend koji se sve češće viđa je korištenje usluga u oblaku, jer je jeftinije koristiti virtualne strojeve, i platiti onoliko koliko je potrebno, nego imati vlastite fizičke strojeve. Na ovaj se način efektivno riješio problem popravaka, održavanja, potrošnje energije i ostalih aktivnosti vezanih za vlastitu opremu. Kompanija Intuit usmjerila se prema analitici u oblaku, a zbog potrebe za sigurnim, stabilnim i ocijenjivim okruženjem. Prijelaz planiraju izvršiti pažljivo, prebacujući podatke sa vlastitog rješenja u oblaku - Intuit Analytics Cloud, na neki eksterni. Dogovor se vrši sa kompanijama Amazon i Cloudera, a zbog potrebe za javno-privatnim oblakom, a koji će istovremeno biti visoko dostupan i siguran. Prijelaz na rješenja u oblaku neizbježan je za kompanije poput Antuit, a koje prodaju proizvode koji se izvršavaju u oblaku. S obzirom na sve niže cijene rješenja u oblaku, može se očekivati da će u bliskoj budućnosti takva rješenja biti višestruko isplativija od vlastitih rješenja.

3.1 Hadoop

Hadoop, distribuirana platforma za pohranu podataka, se s vremenom pretvara u novi operativni sustav za poslovanje, a zahvaljujući sve većem broju rješenja koja su kompatibilna sa njime. Platforma je u ovom slučaju definirana kao cjelokupnost programa, konekcija i ostalog, a što je nužno za izgradnju i pokretanje željenih aplikacija. Distribuirana analitička platforma MapReduce jedan je od poznatih primjera. Takvi sustavi mogu vršiti razne manipulacije i analitičke operacije nad

Hadoop-om. SQL, MapReduce, procesiranje tokova, procesiranje u memoriji, grafička analitika samo su neki od rješenja koja se mogu izvršavati nad Hadoop-om, a sa adekvatnim performansama. Što je više boljih rješenja, veći će broj kompanija početi koristiti Hadoop za pohranu velike količine podataka. Hadoop je jeftino rješenje za pohranu velike količine podataka, upravo zbog mogućnosti pokretanja raznih upita i operacija nad podacima koji se nalaze u njemu. Intuit razvija svoju Hadoop platformu, koristeći MapReduce, a sa ciljem omogućavanja svih vrsta interakcija sa ljudima i proizvodima. (Mitchell 2014)

Hadoop pruža ogroman kapacitet za sve vrste podataka, snažne mogućnosti procesiranja, te mogućnost podrške gotovo neograničenih paralelnih zadataka. Hadoop je open-source što znači da na njemu rade ljudi iz svih dijelova svijeta. S druge strane, sve je veći porast komercijalnih verzija Hadoop-a. Hadoop razdvaja podatke na blokove, a koji se spremaju na grozdovima pripadnog hardvera. Procesiranje velike količine podataka moguće je vršiti velikim brojem jeftinih računala, a takvim pristupom može se doći do rezultata brzo. (SAS Institute Inc. 2014)

Na web lokaciji SAS Institute Inc. (2014) navodi se temeljni razlog korištenja Hadoop-a za većinu kompanija, a to je sposobnost pohrane i procesiranja velike količine podataka, a na brz način. Što se više komputacijskih čvorova koristi, to je veća brzina procesiranja na raspolaganju. Pod takvim čvorovima smatraju se računala koja se koriste u distribuiranoj mreži. Kao što je već navedeno, podaci se ne moraju pretprocesirati prije pohrane. Podaci ali i procesiranje su zaštićeni, a zahvaljujući distribuiranoj mreži. Naime, ukoliko jedan čvor prestane raditi, zadaci se automatski prebacuju na druge čvorove. Također, čuvaju se višestruke kopije podataka, a o čemu će biti više riječi kasnije. Hadoop je besplatna platforma, a s obzirom da je sposobna koristiti hardver koji neka kompanija već posjeduje, takva platforma zaista jest jeftina. Istovremeno je i fleksibilna, a tako da se distribuirana mreža može širiti dodavanjem novih komputacijskih čvorova, a za koju nije potrebna velika količina administracije.

Povijest Hadoop platforme započinje sa širenjem World Wide Web-a (www). 90-ih godina prošlog stoljeća web je počeo značajnije rasti, dok se nakon 2000. godine počeo pojavljivati veći broj tražilica, te indeksiranje radi olakšanja pretraživanja relevantnih informacija u tekstualnom obliku. U samome početku, rezultati pretraživanja bili su vraćeni od strane čovjeka. No, s obzirom na rapidnu brzinu rasta web-a, došlo je do potrebe za automatizacijom. Tako su nastali tzv. web crawler-i, a od kojih mnogi kao istraživački projekti sa sveučilišta. U međuvremenu, start-up tražilice jačaju (Yahoo, AltaVista, i ostali). (SAS Institute Inc. 2014)

SAS Institute Inc. (2014) navodi kao jedan od takvih projekata open-source tražilicu Nutch, a koju su kreirali Doug Cutting i Mike Cafarella. Imali su ideju vraćanja rezultata pretraživanja distribucijom podataka, te kalkulacijama na više računala, a tako da više zadataka može biti izvršeno istovremeno. U to vrijeme, tražilica kompanije Google bila je u izgradnji, a bazirana na istim principima – pohrani i procesiranju podataka na distribuiran i automatiziran način, a tako da rezultati pretraživanja budu brže prikazani. 2006. godine Cutting je započeo raditi u kompaniji Yahoo, a sa sobom je ponio Nutch projekt, ali i ideje bazirane na Google-ovom projektu. Nutch projekt bio je razdvojen. Web crawler dio i dalje se nazivao Nutch, dok se dio za distribuirano procesiranje nazvao Hadoop[3]. 2008. godine, Yahoo je predstavio Hadoop kao open-source projekt. Danas, Hadoop platformu i ekosistem tehnologija održava i upravlja ne-profitna Apache Software Fondacija (ASF), globalna zajednica razvijača softvera i doprinositelja.

3.1.1 Temeljne komponente Hadoop platforme

SAS Institute Inc. (2014) navodi četiri temeljna modula u osnovnoj Hadoop platformi:

[3] Cutting-ov sin imao je igračku - slona po imenu Hadoop, a po kojoj je nastalo ime platforme.

1. Hadoop Common – biblioteke i alati koje koriste ostali Hadoop moduli,
2. Hadoop Distributed File System (HDFS) – proširivi sistem baziran na Javi[4], a koji pohranjuje podatke na distribuiranim strojevima prije nego ih organizira,
3. MapReduce – softverski model za paralelno procesiranje velikih setova podataka,
4. YARN[5] – platforma za upravljanje resursima, a za potrebe raspoređivanja i korištenja zahtjeva za resursima koji dolaze od distribuiranih aplikacija.

Način funkcioniranja nekih od komponenti biti će detaljno objašnjen u daljnjem radu. Ostale komponente koje navodi isti izvor, a detaljnije opisuje Julio (2009) jesu sljedeće:

- Pig – platforma za manipuliranje podacima koji su pohranjeni u HDFS, sadrži kompilator za MapReduce programe i visoki jezik nazvan Pig Latin. Pruža metode za ekstrakciju podataka, transformaciju i punjenje. Također, pruža jednostavne analitičke metode, bez potrebe za pisanjem MapReduce programa. MapReduce programi se jednostavno pišu, a korisnika udaljava od specifičnih detalja. Fokusira se na procesiranje podataka. (Julio 2009)
- Hive[6] – jezik za vršenje upita, sličan SQL-u. Prezentira podatke u obliku tablica. Programiranje Hive-a slično je programiranju baza podataka. Pruža sumarizaciju podataka i vršenje upita nad Hadoop-om. (Julio 2009)
- HBase – nerelacijska, distribuirana baza podataka, a koja se pokreće nad Hadoop-om. HBase tablice mogu služiti kao ulazi i kao izlazi za MapReduce poslove.
- HCatalog – sloj za upravljanje pohranom i tablicama, a koji pomaže korisnicima dijeliti i pristupati podacima. Pruža dijeljenu schemu i mehanizam tipova podataka. Pruža apstrakciju tablica, a tako da se korisnici ne moraju brinuti gdje

[4] Java je opći pojam koji se koristi za softver i pripadajuće komponente, a od kojih neke jesu 'Java Runtime Environment' (JRE), 'Java Virtual Machine' (JVM) i 'Plug-in'. Izvor: Java (2015)
[5] Skraćeno od Yet Another Resource Negotiator.
[6] Prva ga je izgradila tvrtka Facebook.

i kako su pohranjeni podaci. Kompatibilan je sa ostalim alatima za procesiranje velike količine podataka. (Julio 2009)

- Ambari – web sučelje za upravljanje, konfiguriranje i testiranje Hadoop servisa i komponenata.
- Cassandra – NoSQL distribuirani sustav baza podataka sa mogućnostima pružanja skalabilnosti, dostupnosti i performansi. Brisk solucija (DataStax) kombinira Cassandrine mogućnosti u realnom vremenu sa analitičkom snagom Hadoop-a. HDFS je u tom slučaju zamijenjen sa Apache Cassandra File System-om (CFS). (Julio 2009)
- Chukwa – sistem za sakupljanje podataka i nadziranje velikih distribuiranih sistema. Izgrađen je na HDFS-u i MapReduce-u. Pruža alate za prikazivanje, nadziranje i analiziranje log datoteka. (Julio 2009)
- Flume – softver koji sakuplja, spaja i razmješta protoke velike količine podataka u HDFS. Otporan je na zakazanja, te uključuje mehanizam pouzdanosti koji se može podesiti po volji korisnika. Dodatno, posjeduje mnoge mehanizme oporavka. (Julio 2009)
- Oozie – komponenta koja raspoređuje poslove unutar Hadoop platforme. Korisnik je u mogućnosti započeti, zaustaviti, suspendirati, nastaviti i ponovno pokrenuti poslove. Oozie jest Koordinatorski modul, specijaliziran u pokretanju tokova poslova na temelju vremenskih i podatkovnih okidača. On pokreće tokove zadataka korištenjem radnji koje izvršavaju MapReduce i Pig poslove. (Julio 2009)
- Sqoop – mehanizam konekcija i transfera koji razmješta podatke između Hadoop-a i relacijskih baza podataka. Uvoz podataka je automatiziran, a može se vršiti iz mnogih baza podataka u Hadoop. Stvara kod za korištenje u MapReduce aplikacijama. (Julio 2009)
- Spark – open-source platforma za distribuirane komputacije u grozdovima strojeva; pruža analitičke sposobnosti u memoriji.

- Solr – proširiv alat za vršenje pretraga, a koji podržava indeksiranje, centralno konfiguriranje, oporavak od katastrofa, te pouzdanost.
- Zookeeper – aplikacija visokih performansi koja koordinira distribuirane procese. Ona održava konfiguracijske informacije, imenovanje, pruža distribuiranu sinkronizaciju i grupne servise. (Julio 2009)

Moguće je odabrati i neke od komercijalnih distribucija Hadoop-a, a od kojih su poznatije Cloudera, Hortonworks i MapR. Kod takvih distribucija korisnik plaća njihovu verziju platforme, te na raspolaganju dobiva dodatne softverske komponente, alate, trening, dokumentaciju i ostale usluge.

3.1.2 Odabir Hadoop platforme i glavne smjernice

Biranje Hadoop distribucije vitalna je odluka koja ima dugotrajne posljedice na cijelu organizaciju, na načine koji se ne mogu niti predvidjeti u inicijalnom trenutku odabira. To pogotovo vrijedi u vrijeme kada je velika količina podataka rastući fenomen u poslovnom svijetu. Hadoop infrastruktura zahtjeva jednaku pažnju pri konfiguraciji i odabiru kao i ostala oprema – serveri, pohrana, baze podataka i ostalo. Hadoop okruženje će biti podvrgnuto jednakim zahtjevima kao i ostatak IT[7] portfolija. Na to će okruženje utjecati Ugovori o razini pružanja usluge[8], zaštita podataka, sigurnost, integracija sa ostalim aplikacijama, profesionalne usluge i trening. Hadoop nije jedna solucija, već platforma sa kolekcijom aplikacija. Navedeni elementi moraju raditi usklađeno da bi se maksimizirala vrijednost. Bitno je pronaći rješenje koje će se prilagoditi načinu poslovanja, umjesto zahtijevanja Hadoop tehnologije koja za određenu organizaciju nije prikladna. (Schneider 2013)

[7] Informatičke tehnologije.
[8] Engl. Service Level Agreement, SLA.

Slika 2. Glavne smjernice pri odabiru Hadoop distribucije. Izvor: Schneider (2013)

Pod performansama Schneider (2013) podrazumijeva brzinu kojom distribucija može procesirati informacije i vrijeme kada te informacije postaju dostupne za analitičke radnje. Podrazumijeva vrijeme odaziva za aplikacije u realnom vremenu i MapReduce brzinu. Skalabilnost je sposobnost proširenja u svim relevantnim dimenzijama, a kao što je broj čvorova, tablica, datoteka, i ostalo. Takva proširenja ne bi trebala stvoriti značajan administrativni teret, promjenu u aplikacijskoj logici ili izuzetne troškove. Preduvjeti koji mogu imati pozitivan utjecaj na performanse i skalabilnost navedeni su u nastavku.

Tablica 2. Preduvjeti za performanse i skalabilnost. Izvor: Schneider (2013)

Preduvjeti	Opis
Izgradnja ključnih komponenata sa jezikom poput C/C++	Hadoop je napisan u Javi. To pomaže u prenosivosti, ali uvodi i vezane probleme koji mogu umanjiti performanse, kao što je nerazdvojivo pretprocesiranje. Korištenje C/C++ je konzistentno sa gotovo svim ostalim poslovnim softverom.
Minimalni softverski slojevi	Svaki sustav profitira od smanjenja „pokretnih dijelova". U slučaju Hadoop-a i HBase-a, performanse i pouzdanost se mogu umanjiti zbog neefikasnosti koje proizlaze iz potrebe navigiranja kroz serije odvojenih slojeva kao što su HBase Master i RegionServer, Java Virtual Machine, i lokalni Linux sustav datoteka.
Jedno okruženje/platforma – za sve aplikacije koje rade sa velikom količinom podataka	Zbog nužnosti performansi, mnoge su Hadoop implementacije zahtijevale stvaranje odvojenih instanci. Mnogo je bolje da Hadoop tehnologija pruža dovoljnu propusnost za maksimalno predviđeno opterećenje. To će eliminirati ili smanjiti ponavljanje podataka, što u konačnici poboljšava skalabilnost.
Korištenje elastičnosti i skalabilnosti koju pružaju javne Cloud platforme	Administratori i graditelji sustava trebaju fleksibilnost i mogućnost pokretanja Hadoop-a unutar vlastitog

vatrozida, ali i na Cloud okruženjima, kao što su Amazon Web Services i Google Compute Engine.

S obzirom da Hadoop obično radi sa velikom količinom podataka, isti autor smatra da posao unošenja i vađenja podataka mora biti što je više moguće efikasan. Nažalost, mnoge Hadoop distribucije zahtijevaju kompleksne i zahtjevne grupne procese, a korištenjem tehnologija kao što su Flume i Scribe. Takve neefikasnosti se pogoršavaju proporcionalno sa rastom količine velikih podataka. Bolja tehnika je izložiti standardno datotečno sučelje na način da dopušta aplikacijama korištenje Hadoop grozda, a na način kao da je tradicionalni umreženi pohrambeni prostor (engl. Network Attached Storage, NAS). Aplikacijski serveri su tada u mogućnosti direktno pisati informacije u Hadoop grozd, umjesto prethodnog prenošenja na lokalne diskove. Podatkovna veza za Hadoop može biti automatski sažeta u procesu dolaska, a onda je dostupna za nasumična pisanja i čitanja od strane aplikacija, a putem višestrukih paralelnih konekcija. Takve trenutne interakcije dopuštaju donošenje odluka u realnom vremenu. Iako su projekti poput Apache Drill-a namijenjeni podršci vrlo brzog donošenja odluka, ono neće biti moguće ukoliko velika količina podataka na kojima se odluke baziraju ne dolazi u Hadoop grozd velikom brzinom.

Isti autor tvrdi da se IT organizacije koje žele implementirati Hadoop rješenje s ciljem poboljšanja zarade obično susreću sa nekoliko opcija. Prva opcija uključuje nabavku hardvera i ostalih resursa koji će biti potrebni i na taj način utrošiti veliku količinu novca i administratorskog vremena. Druga opcija je pokušati uštedjeti što je više moguće, te kupiti limitiran set hardverske opreme i resursa, te vjerojatno propustiti potpun potencijal Hadoop platforme i velike količine podataka. Proširiva Hadoop platforma može pomoći u balansiranju odluka, te zadovoljavanju korisničkih potreba, a tako da se istovremeno ne mora utrošiti izuzetan iznos na hardver i resurse. Oprema

u takvoj situaciji postaje proširiva, te se može dodavati ili oduzimati nepotrebnu opremu po potrebi, te na takav način uštedjeti.

Kao što je već prethodno navedeno, skalabilnost je sposobnost proširenja u svim relevantnim dimenzijama, a kao što je broj čvorova, tablica, datoteka, i ostalo. Skalabilnost Hadoop instance mjeri se putem nekoliko faktora, a koje Schneider (2013) navodi sljedećim redom:

1. Datoteke – Hadoop-ova početna arhitektura se sastoji od jednog NameNode-a[9]. To ograničava Hadoop grozd na, otprilike, 100 do 150 milijuna datoteka, a taj broj ovisi i o dostupnoj memoriji za metapodatke[10]. U malim grozdovima, maksimum broja blokova na svakom čvoru dodatno ograničava broj dostupnih datoteka. Nužno je odabrati Hadoop platformu koja izbjegava zagušenje koje nastaje zbog jednog NameNode-a, te koja ima distribuiranu arhitekturu za metapodatke. Na taj se način može proširiti na datoteke i tablice čija količina može prelaziti bilijun.
2. Broj čvorova – Druga dimenzija skalabilnosti je broj fizičkih čvorova. Ovisno o zahtjevima procesiranja ili podatkovnog pohrambenog prostora, odabrana Hadoop distribucija može zahtijevati i tisuće fizičkih čvorova.
3. Kapacitet čvorova/gustoća – U slučajevima korištenja koji su pohrambeno intenzivni, potrebno se proširiti putem čvorova koji imaju veći kapacitet. Takav pristup služi smanjenju općeg broja čvorova koji su potrebni za pohranu date količine podataka.

Isti autor tvrdi da sve više organizacija koristi NoSQL rješenja za upravljanje ključnim poslovnim operacijama. Jedini način da takve nove aplikacije postignu pouzdanost i

[9] NameNode jest središnji dio HDFS-a. Sadržava stablo direktorija svih datoteka u sustavu datoteka. Prati gdje se diljem grozda čvorova datotečni podaci pohranjuju. Podatke o tim datotekama ne pohranjuje sam. Klijentske aplikacije komuniciraju sa NameNode kad god trebaju locirati datoteku, ili kada trebaju dodati/kopirati/premjestiti/obrisati datotetu. NameNode odgovara uspješnom zahtjevu vraćanjem liste relevantnih DataNode servera gdje se podaci nalaze. Izvor: Apache Wiki (2011)
[10] Metapodaci (engl. metadata) su informacije o sadržaju nekog objekta. Izvor: Tech Terms (2015)

prihvatljivost RDBMS[11] solucija jest da se podlože jednakim rigoroznim Ugovorima o razini pružanja usluge koje IT očekuje od aplikacija građenih nad relacijskim bazama podataka. Primjerice, NoSQL rješenja sa fluktuirajućim vremenima odaziva neće biti solucije kandidati za temelj poslovnih operacija koji zahtjeva brz odaziva i vrlo nisko kašnjenje.

Apache HBase je NoSQL baza podataka bazirana na ključ-vrijednost parametrima, i sagrađena nad Hadoop-om. Pruža pohranu i analitiku velike količine podataka u realnom vremenu, sa dodatnom mogućnošću vršenja MapReduce operacija koristeći Hadoop. Procjenjuje se da danas 30-40% svih Hadoop instalacija koristi HBase. Bez obzira na prednost integracije sa Hadoop-om, HBase nije široko prihvaćen zbog mnogih limita u performansama ali i njegovoj ovisnosti. Postoje inovacije koje mogu transformirati HBase aplikacije tako da budu sposobne zadovoljiti potrebe većine online aplikacija i analitike. Neke od njih navodi Schneider (2013):

- Smanjenje ukupnog broja slojeva,
- Eliminiranje potrebe za Java kolekcijama otpada[12],
- Eliminiranje potrebe ručnog odvajanja tablica unaprijed,
- Distribucija metapodataka diljem grozda, umjesto na jednom NameNode-u,
- Izbjegavanje zbijanja i vezanih I/O problema koje zbijanja izazivaju.

Schneider (2013) smatra da se od Hadoop-a može očekivati ista razina ovisnosti kao i od svih ostalih poslovnih softverskih sustava. Administratori koji administriraju ostatak IT opreme mogu administrirati i Hadoop implementacije. Kako bi se smanjio opći teret na korisnike i administratore, najuspješnija Hadoop infrastruktura bit će sposobna nositi se sa neizbježnim problemima na koje nailaze svi produkcijski sustavi. Većina takvih odgovora treba biti automatizirana tako da bi dodatno poboljšala stanje ovisnosti.

[11] Engl. relational database management system, RDBMS. Sustav koji dopušta kreiranje, ažuriranje i administriranje relacijskih baza podataka. Izvor: Rouse (2005)
[12] Engl. Java Garbage Collection, Java GC.

U nastavku će biti opisane neke osobine Hadoop platforme, a koje su izgrađene tako da platforma podnese uspješno i najstresnija okruženja. Temeljni principi koji poboljšavaju samostalnost jesu prikazani u sljedećoj tablici:

Tablica 3. Temelji arhitekture za poboljšanje samostalnosti. Izvor: Schneider (2013)

Preduvjet	Opis
Što manje odvojenih dijelova	HBase okruženje ne bi trebalo zahtijevati RegionServer-e ili HBase Master-e. Elimininranje Java virtualne mašine pomaže.
Što manje ručnih zadataka	Mnogi Hadoop administratori su opterećeni zadacima poput zbijanja, ručnog administriranjam i ručnog odvajanja unaprijed. Uklanjanje takvih odgovornosti poboljšava opću samostalnost sustava.
Integritet podataka	Može se poboljšati putem internih checksum-a, replikacija i kroz mogućnosti zaštite podataka kao što su snimke stanja (engl. snapshot).
Optimizacija poslova	Okruženje treba biti optimizirano na način da mali zadaci, kao što su trenutni upiti, ne kasne zbog preraspoređenih velikih zadataka.

Visoku dostupnost (engl. High availability, HA) Schneider (2013) definira kao sposobnost Hadoop platforme da nastavi služiti korisnicima čak i kada se suoči sa neizbježnim problemom u hardveru, softveru, mreži i ostalome što je karakteristično za distribuirana računalna okruženja takve veličine i kompleksnosti. Da bi platforma dostavila visoku dostupnost koja je kritična za produkcijske aplikacije, morat će posjedovati sva HA svojstva navedena u nastavku. Po istom autoru svojstva visoke dostupnosti jesu:

1. Visoka dostupnost je ugrađena. Ne bi trebalo biti nužno vršiti posebne korake da bi se iskoristila visoka dostupnost, ona bi trebala biti polazno svojstvo same platforme.
2. Jedan NameNode koji sadrži sve metapodatke za grozd predstavlja jednu točku neuspjeha i izloženosti za visoku dostupnost. Rješenje koje distribuira metapodatke ima HA prednosti.
3. Bitan aspekt visoke dostupnosti jest kako zakazanje utječe na MapReduce poslove. Zakazanje u poslu ili pratitelju zadataka može utjecati na sposobnost zadovoljenja Ugovora o razini pružanja usluge. Potrebno je procijeniti sadrži li MapReduce HA automatsko oporavljanje od grešaka i sposobnost nastavka rada bez potrebe za ručnim ponovnim pokretanjem.
4. NFS[13] (engl. Network File System, NFS) visoka dostupnost odnosi se na visoku propusnost i otpornost unošenja i pristupanja podacima.
5. Hadoop distribucije se razlikuju po vremenu i procesu koji se vrši za obnovu podataka u slučaju jednog ili višestrukih zakazanja.
6. Ažuriranje Hadoop platforme trebalo bi biti moguće bez potrebe za gašenjem platforme.

Mnoge organizacije donose bitne poslovne odluke koristeći Hadoop platformu. Zbog toga je potrebno zaštititi podatke koje Hadoop procesira. Replikacija i snimke stanja dokazane su tehnike i temeljni dio zaštite relacijskih podataka, a imaju ulogu u zaštiti informacija unutar Hadoop platforme. Replikacija pomaže u zaštiti Hadoop podataka od periodičnih zakazanja koja se mogu očekivati prilikom vršenja distribuiranog procesiranja velike količine podataka. Platforma treba automatski replicirati dijelove datoteka, regije tablica i metapodatke barem tri puta, a na način da sve replike ne budu pohranjene na istom mjestu. Snimke stanja nude oporavak platforme na neko stanje u vremenu, bez duplikacije podataka. Hadoop platforma bi trebala omogućiti snimkama

[13] NFS je protokol distribuiranih datotečnih sustava, proizveden od strane Sun Microsystems 1984. godine. Dopušta korisniku na klijentskom računalu pristup datotekama preko mreže, na način koji je sličan pristupu lokalnom pohrambenom prostoru. Izvor: ArchLinux Wiki (2015)

stanja dijeljenje istog pohrambenog prostora koji koriste i trenutne informacije. Takav proces ne bi trebao imati utjecaj na performanse ili skalabilnost. Također, trebalo bi biti moguće čitati datoteke i tablice direktno iz snimke stanja. Snimke stanja ne služe samo za zaštitu podataka, već se mogu koristiti i za potpomaganje procesa stvaranja novog modela. Različiti modeli mogu se pokretati nad istom snimkom stanja, a na taj se način rezultati izoliraju po promjeni modela. (Schneider 2013)

Hadoop platforma je izložena događajima koji mogu omesti poslovne operacije zbog nekoliko razloga, a koje isti autor navodi:

1. Često se pokreće na već korištenom hardveru,
2. Sadržava ogromne količine informacija,
3. Podaci su distribuirani, a mreže ponekad nemaju dovoljnu propusnost,
4. Današnje IT okruženje često je subjekt raznih napada.

Zbog toga prethodni autor smatra da je dobro vršiti „zrcaljenje“ podataka koje pomaže kod oporavka. Takva metoda podrazumijeva sinkronizaciju podataka između različitih odjela i lokacija. Takav pristup treba vršiti automatski sažete transfere podataka, a koji su promjene na razini bloka. Treba „zrcaliti“ podatke i metapodatke na način da su podaci konzistentni u svakome trenutku, tako da bi se osigurao ponovni početak rada aplikacija nedugo nakon zakazanja lokacije. Proces treba imati sljedeće karakteristike navedene u tablici:

Tablica 4. Karakteristike vršenja „zrcaljenja“ podataka. Izvor: Schneider (2013)

Efikasnost	Sigurnost	Nezahtjevnost
Delte na razini bloka (8 KB)	Konzistentnost stanja u vremenu	Rješavanje mrežnih problema na što nezamjetniji način
Automatsko sažimanje	End-to-end checksum-i[14]	Pristup kopiji podataka direktno (a ne kao cold standby[15])
Bez utjecaja na performanse		Raspoređivanje na razini spremnika podataka

U ranim danima Hadoop platforme, bilo je uobičajeno da iskusni izgraditelji sa detaljnim poznavanjem izvornog koda upravljaju višestrukim Hadoop okruženjima. To je bilo moguće jer su poznavali Hadoop na izuzetno detaljnoj razini i jer su kombinirali odgovornosti izgradnje i upravljanja. Takav pristup, gdje se uz Hadoop upravlja sa još nekoliko različitih okruženje nije prikladan za IT općenito. Sveukupni troškovi posjedovanja su jedan od glavnih faktora koji se uzima u obzir kada IT uspoređuje rješenja, a pogotovo je bitno u Hadoop okruženjima. Preporučljivo je pronaći platformu koja pruža inteligentne alate koji olakšavaju administraciju. Kako Hadoop napreduje, nove se distribucije natječu na području kvalitete svojih alata za upravljanje platformom. Alati se procjenjuju po karakteristikama administracije i nadzora. Na području administracije autor tvrdi da se alati mogu istaknuti na sljedećim područjima:

[14] Diskovi neke greške u čitanju podataka ne prijavljuju, a ponekad ne uspiju niti zapisati podatke, istovremeno tvrdeći da su podaci zapisani. End-to-end checksum može detektirati tihu korupciju podataka, a do koje može doći zbog diska ili ostalih komponenata koje prenose ili manipuliraju podacima. Izvor: IBM (2015)

[15] Metoda redundancije u kojoj se sigurnosnoj kopiji pristupa samo kada primarni sustav zakaže. Cold standby sustavi se koriste za nekritične aplikacije i sustave gdje se podaci ne mijenjaju često. Izvor: Webopedia (2015)

1. Upravljanje podacima i korisnicima na razini pohrambenog prostora,
2. Centralizirana administracija čvorova,
3. Dodavanje i uklanjanje diskova putem grafičkom korisničkog sučelja,
4. Vršenje ažuriranja koje dopušta ažuriranje softvera kroz neko vrijeme, bez ometanja usluge,
5. Automatizirani i raspoređeni administrativni zadaci,
6. Mogućnost pristupanja od strane više korisnika, sa kontrolom postavljanja podataka i poslova.

Na području nadzora isti autor tvrdi da se alati ističu na sljedećim područjima:

1. Sveobuhvatan nadzor Hadoop grozda, od aplikacijskog do hardverskog dijela, uključujući detektiranje zakazanja diskova,
2. Uzbune, alarmi, i mape temperature koje pružaju poglede na čvorove u boji i u realnom vremenu. Takvi pogledi uključuju zdravlje, memoriju, CPU i ostale metrike koje se odnose na čvorove,
3. Integracija, putem REST API, u druge komercijalne i otvorene alate, te mogućnost izgradnje vlastite kontrolne ploče,
4. Vidljivost putem standardnih alata poput Ganglia i Nagios.

Hadoop implementacije se često protežu na stotine ili tisuće čvorova. Konfiguracija, implementacija i administracija tolikog broja čvorova trebala bi biti što je više moguće automatizirana. Mnogi vodeći pružatelji Hadoop distribucija rade na automatizaciji koja je svakim danom sve bolja.

Sakupljanje velike količine podataka samo je početak. Da bi se iskoristio potpuni potencijal podataka, potrebna je Hadoop platforma koja olakšava obradu i ekstrakciju informacija na brz i siguran način. Nakon tog koraka platforma treba imati mogućnost izgradnje potpuno sposobnih aplikacija korištenjem dokazanih alata i tehnika. Povoljno je ako se postojeće aplikacije mogu lako spojiti na Hadoop podatke. (Schneider 2013)

Tablica 5. Arhitekturalni temelji za pristup podacima. Izvor: Schneider (2013)

Preduvjet	Opis
Potpun pristup Hadoop API-u	Potpuna funkcionalnost za izgradnju aplikacija, te pružanje jednostavne prenosivosti na druge Hadoop implementacije. Na taj način se izbjegava zaključavanje kod jednog prodavača.
Potpun POSIX[16] pristup datotekama (čitanje/pisanje/ažuriranje)	Veća fleksibilnost izgradnje softvera i postojeće aplikacije mogu direktno raditi sa Hadoop podacima.
Direktna kontrola graditelja nad ključnim resursima	Nema potrebe za administratorima za izgradnju stalnih ili privremenih tablica radi optimizacije aplikacijskog toka rada.
Sigurna poslovna pretraga	Aplikacije mogu sadržavati vitalne sposobnosti poput praćenja klikova, upozorenja pretrage i podešavanje relevantnosti pretraga.
Bogat izbor alata za pristup podacima	Moguće je odabrati pravu implementaciju za posao, poput: 1. Apache Flume za sakupljanje log datoteka 2. Apache Sqoop za paralelni uvoz i izvoz između Hadoop-a i relacijskih baza podataka/skladišta podataka 3. Distcp za distribuirano kopiranje između grozdova i udaljenih izvora podataka i Hadoop-a

[16] Akronim za Portable Operating System Interface. Obitelj standarda, koje je specificirao IEEE (engl. The Institute of Electrical and Electronics Engineers), za definiranje i standardiziranje API-a. Programi koji su pisani tako da se oslanjaju na POSIX standarde lako se prenose između UNIX derivata. Izvor: Harvey i Martelli (2009)

4. ODBC i JDBC za izvoz podataka iz Hadoop-a, putem Hive-a

POSIX datotečni sustav koji podržava operacije čitanja/pisanja na Hadoop i NFS pristup, otvara Hadoop mnogo široj upotrebi nego što je to slučaj kod polaznog HDFS-a. Također, olakšava vršenje zadataka koji bi inače zahtjevali mnogo kompleksnije procese. Administratori i korisnici trebali bi biti u mogućnosti pristupiti grozdu putem mreže, kao da se radi o NAS-u. Windows Explorer, Mac Finder, IDE-i[17], i standardne Linux naredbe za interakcije sa datotekama (ls, grep, tail i ostalo) trebali bi biti u mogućnosti raditi direktno sa grozdom. Kada se Hadoop grozd promatra kao dio datotečnog sustava, korisnik može prenositi podatke „drag-and-drop" metodom, ali i automatski popunjavati CLI[18] naredbe pomoću tipke Tab. Ostale aplikacije mogu koristiti Hadoop, te čitati i pisati podatke. POSIX datotečni sustav olakšava uvoz i izvoz informacija u/iz relacijskih baza podataka i skladišta podataka koristeći standardne alate, odnosno bez potrebe za specijalnim konektorima. (Schneider 2013)

Graditelji imaju godine iskustva u korištenju popularnih alata i metoda za interakciju sa relacijskim bazama podataka. Iako Hadoop uvodi novu paradigmu i koncepte, dobro je potražiti platformu koja podržava produktivnost graditelja na sljedeće načine, a koje isti autor navodi navodi:

1. Pruža komponente otvorenog koda na GitHub-u za preuzimanje i prilagodbu,
2. Pruža binarne datoteke putem Maven repozitorija za bržu izgradnju aplikacija,
3. Pruža platformu radnih tokova za bržu izgradnju aplikacija,
4. Osposobljava standardne graditeljske alate da rade direktno sa podacima koji se nalaze na grozdu,

[17] Engl. Integrated Development Environment, IDE. Aplikacija koja pruža bogat izbor alata i okruženje za programere i graditelje softvera. Izvor: Rouse (2015)
[18] Engl. Command Line Interface, CLI.

5. Dopušta postojećim aplikacijama i bibliotekama koje su napisane u bilo kojem programskom jeziku da mogu pristupati i pisati podatke na Hadoop,
6. Pruža mogućnosti interaktivnih upita poput SQL-a.

S obzirom na količinu podataka koja se pohranjuje u Hadoop-u, prethodni autor tvrdi da je nužno poduzeti proaktivne korake za zaštitu podataka. Nažalost, neke Hadoop implementacije su teške za osigurati, pa kupci izbjegavaju sigurnost i ne brinu se o njoj. Sigurnost Hadoop implementacije trebala bi obuhvatiti sljedeća područja:

1. Detaljno definirale ovlasti za datoteke, direktorije, zadatke, rasporede i administrativne radnje,
2. Liste kontrole pristupa (engl. Access Control List, ACL) za tablice i stupce,
3. Enkripcija između Hadoop grozda i svih vanjskih pristupnih točaka, ugrađeno i putem trećih strana,
4. Standardni protokoli autentikacije poput Kerberos-a, LDAP[19], Active Directory-a, NIS[20], lokalnih korisnika i grupa, i ostalih autentikacijskih i identifikacijskih sistema od trećih strana,
5. Jednostavan i siguran pristup grozdu putem ulaznog čvora, pritom blokirajući direktne interakcije sa ostalim čvorovima.

U nastavku slijedi usporedba glavnih Hadoop distribucija.

[19] Engl. Lightweight Directory Access Protocol, LDAP. Internet protokol koji koriste elektronička pošta i ostali programi za traženje informacija sa servera. Izvor: Gracion (2015)
[20] Engl. Network Information System, NIS. Dizajniran je za centralizaciju administracije UNIX sistema. Izvor: FreeBSD (2015)

Tablica 6. Usporedba razlika između glavnih Hadoop distribucija. Izvor: Schneider (2013)

	Hortonworks	Cloudera	MapR
Performanse i skalabilnost			
Unos podataka	Skupno	Skupno	Skupno i zapis u toku
Arhitektura metapodatka	Centralizirana	Centralizirana	Distribuirana
HBase performanse	Varijabilna latentnost	Varijabilna latentnost	Konzistentna niska latentnost
NoSQL aplikacije	Pretežito skupne aplikacije	Pretežito skupne aplikacije	Skupne i online/aplikacije u realnom vremenu
Samostalnost			
Visoka dostupnost	Oporavak od jednog zakazanja	Oporavak od jednog zakazanja	Samostalni oporavak za vrijeme višestrukih zakazanja
MapReduce visoka dostupnost	Ponovno pokreće zadatke	Ponovno pokreće zadatke	Nastavlja s radom bez ponovnog pokretanja
Nadogradnja	Planiran prestanak rada	Rolling nadogradnja[21]	Rolling nadogradnja
Replikacija	Podaci	Podaci	Podaci i metapodaci

[21] Čvorovi se nadograđuju redom, a rad se pritom ne prekida, već se prebacuje na čvorove koji nisu u procesu nadogradnje. Nadogradnja traje raspoređeno dok se svi čvorovi ne nadograde. Svaki se čvor nakon nadogradnje ponovno pokreće. Izvor: Oracle (2002)

Slike stanja	Konzistentnost za zatvorene datoteke	Konzistentnost za zatvorene datoteke	Konzistentnost za točku u vremenu, za sve datoteke i tablice
Oporavak od katastrofa	Nema	Raspored kopiranja datoteka (BDR[22])	Zrcaljenje
Upravljanje			
Alati za upravljanje	Ambari	Cloudera Manager	MapR Control System
Podrška za upravljanje putem standardnih OS pretraživača	Nema	Nema	Ima
Mape temperature, alarmi, upozorenja	Ima	Ima	Ima
Integracija sa REST API	Ima	Ima	Ima
Kontrola pozicioniranja podataka i zadataka	Nema	Nema	Ima
Pristup podacima			

[22] Engl. Backup and Disaster Recovery, BDR.

Pristup sustavu datoteka	HDFS, NFS - samo čitanje	HDFS, NFS - samo čitanje	HDFS, NFS – čitanje i pisanje (POSIX)
Datotečni input/output	Dodavanje	Dodavanje	Čitanje i pisanje
Sigurnost: liste za kontrolu pristupa	Ima	Ima	Ima
Autentikacija	Kerberos	Kerberos	Kerberos i nativno

Hadoop se danas koristi za rješavanje raznih problema. Neke tvrtke ga koriste kao jeftino rješenje za pohranu velike količine podataka i aktivnu arhivu podataka. Zahvaljujući relativno niskim cijenama potrebnog hardvera, Hadoop je koristan za pohranjivanje i kombiniranje podataka poput transakcija, podataka sa društvenih mreža, podataka dobivenih od senzora i mašina, znanstvenih podataka i ostalo. Jeftina pohrana dopušta spremanje podataka koji se trenutno ne smatraju kritičnima, ali možda posluže za analitiku u budućnosti. Hadoop platforma može se koristiti i za pripremanje i postavljanje podataka za potrebe skladišta podataka (engl. enterprise data warehouse, EDW), i velike analitičke platforme, koje provode naprednu analitiku, upite, izvješća i ostalo. Od Hadoop-a se očekuje da podnosi velike količine nestrukturiranih podataka i da rastereti skladište podataka od starih količina podataka. Hadoop se koristi i za pohranu velike količine podataka u jezeru podataka, a koje će biti detaljnije objašnjeno u radu. Zbog toga što je Hadoop dizajniran za upotrebu velike količine podataka, a koji mogu biti različitih oblika, sposoban je pokretati analitičke algoritme. Analitika velike količine podataka koristi organizaciji da posluje efikasnije, da otkrije nove prilike i naprednu konkurentsku prednost. Jedan od najpopularnijih slučajeva korištenja Hadoop platforme je za sustave web

preporuka. Primjerice, Facebook može prikazati ljude koje korisnik vjerojatno poznaje; LinkedIn može preporučiti poslove koji bi mogli korisnika interesirati; eBay preporučuje proizvode koji mogu korisnika interesirati temeljem proizvoda koje je promatrao u prošlosti. Takvi sustavi vrše analitiku nad velikom količinom podataka u realnom vremenu, a sa ciljem da predvide kupčeve želje i moguće interese prije nego kupac napusti web stranicu. (SAS Institute Inc. 2014)

3.1.3 HDFS i MapReduce

Lockwood (2014) definira tradicionalne probleme procesiranja kao one koji koriste više jezgri procesora (engl. Central Processing Unit, CPU), a sa ciljem da vrše više matematičkih operacija istovremeno. Takvi problemi, a koji se pokušavaju riješiti tradicionalnim metodama, po autoru dijele dva zajednička svojstva:

1. Vezani su uz CPU, a dio problema koji oduzima najviše vremena su kalkulacije koje uključuju floating point i integer aritmetiku,
2. Ulazni podaci su obično manji od stotinu gigabajta.

S druge strane, tehnološko se okruženje mijenja izuzetnom brzinom. Izvori beskrajnih podataka sve su brojniji, a počeli su rasti velikom brzinom pojavom jeftinih tvrdih diskova sa visokim kapacitetom, te naprednim podatkovnim sustavima koji ih podržavaju. Tako se dolazi do problema koji su podatkovno-intenzivni, a koje je isti autor okarakterizirao sljedećim svojstvima:

1. Ulazni su podaci mnogo veći od nekoliko stotina gigabajta, a često su i po nekoliko stotina, pa i tisuća terabajta,
2. Vezani su za ulazno/izlazne (I/O) tokove, a gdje je računalu potrebno više vremena da takve količine podataka prenese sa njihove lokacije do procesora, nego što je potrebno procesoru da ih obradi.

Kod tradicionalnih paralelnih aplikacija,

1. Glavni paralelni izvršitelj čita ulazne podatke sa diska (ponekad je moguće da više izvršitelja istovremeno koristi I/O API, a tako da kolektivno čitaju ulazne podatke),
2. Glavni izvršitelj dijeli ulazne podatke u skupine, te takve dijelove šalje svim ostalim izvršiteljima,
3. Svi paralelni izvršitelji procesiraju svoje dijelove ulaznih podataka,
4. Svi paralelni izvršitelji komuniciraju rezultate među sobom, a onda prelaze na sljedeću iteraciju kalkulacije.

Problem sa velikom količinom podataka, po prethodnom autoru, je u tome što se proces čitanja ulaznih podataka vrši serijski. Takvi su podaci od izvršitelja odvojeni nekakvom vrstom „tunela (cijevi)“, a kroz koji podaci mogu ići predefiniranom propusnošću. Kupnjom brže opreme, primjerice SSD (engl. Solid State Drive, SSD) diskova, brže mrežne opreme, više servera i ostalog, moguće je povećati propusnost i brzinu protoka ulaznih podataka. Konačno, cijena takvih kupovina ne raste linearno.

MapReduce paradigma potpuno je drugačiji način rješavanja nekog skupa problema, a koji može biti paraleliziran, te može uspješno riješiti problem nedovoljne propusnosti ulaznih podataka sa diska. Tradicionalni paralelizam donosi podatke prema procesiranju, dok MapReduce vrši obrnuto, odnosno donosi procesiranje prema podacima. Kod korištenja MapReduce-a, ulazni podaci nisu spremljeni na odvojenom mjestu pohrane visokog kapaciteta. Umjesto toga, podaci postoje u malim dijelovima, te su pohranjeni na elementima procesiranja. Lockwood (2014) tvrdi da se takva procedura odvija sljedećim koracima:

1. Nije potrebno razmještati podatke jer su preraspodijeljeni i jer se već nalaze na čvorovima, a koji su sposobni djelovati kao elementi za procesiranje,
2. Sve funkcije paralelnih izvršitelja šalju se čvorovima, a gdje se njihovi dijelovi ulaznih podataka već nalaze, te vrše kalkulacije,

3. Svi paralelni izvršitelji komuniciraju rezultate među sobom, razmještaju podatke po potrebi, a onda se usmjeravaju na sljedeći korak u kalkulaciji.

Na taj se način, tvrdi isti autor, postiže da je razmještanje podataka nužno samo onda kada paralelni izvršitelji komuniciraju rezultate među sobom. Korak u kojemu su se ulazni podaci prenosili sa pohrambenog prostora prije distribucije efektivno je uklonjen, a zbog toga što se podaci već nalaze na izvršnim resursima. Da bi se kalkulacije mogle uspješno izvoditi, nužno je da su podaci i kalkulacije neovisni jedni o drugima i ulaznim podacima. MapReduce prikladno je koristiti za trivijalne paralelne kalkulacije koje se vrše nad velikim količinama podataka.

Proces razdvajanja velikog podatka na manje, te distribucija manjih dijelova prema različitim kalkulacijskim elementima nije jednostavan zadatak. Hadoop, moguće najkorištenija map/reduce platforma, postiže zadatak koristeći Hadoop Distributed File System (HDFS). HDFS je temelj Hadoop-a jer omogućuje dijeljenje i distribuciju podataka na kalkulacijske elemente, a koji su nužni da map/reduce aplikacije budu efikasne. Komputacijski elementi se preciznije mogu nazvati komputacijski čvorovi. HDFS je sustav za datoteke u koji i iz kojeg je moguće kopirati datoteke na netradicionalan način. Mnoge komande za manipulaciju datotekama ponašaju se kao i u ostalim standardnim datotečnim sustavima. Primjerice, Linux-ov ext4, a za koji su neke od komandi ls, mkdir, rm, mv, cp, cat, tail, chmod i ostale. (Lockwood 2014)

Prema istom autoru, djeluje da HDFS sadrži datoteke kao i svaki tradicionalni datotečni sustav, u stvarnosti su datoteke raspodijeljene prema višestrukim fizičkim komputacijskim čvorovima. Kada se datoteka kopira u HDFS, ona je raspodijeljena u dijelove od kojih svaki ima 64 MB. Osim toga, datoteka je replicirana tri puta radi pouzdanosti. Svi ti dijelovi su raspodijeljeni ka raznim komputacijskim čvorovima, a na način da dio od 63 MB bude prisutan na tri neovisna čvora. Iako su datoteke fizički fragmentirane i trostruko distribuirane, korisnik datoteku u HDFS vidi kao cjelovitu, odnosno onakvu kakvu je kopirao u HDFS.

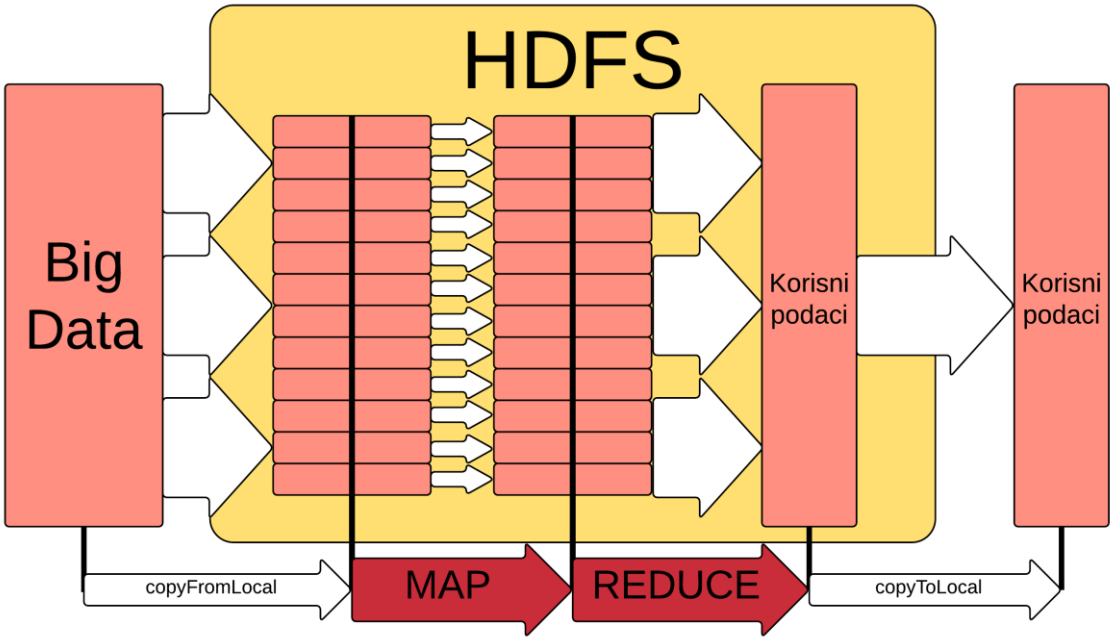
SAS Institute Inc. (2014) navodi da se datoteke u HDFS mogu unijeti korištenjem jednostavnih Java naredbi. HDFS vrši višestruke kopije blokova podataka i distribuira ih diljem čvorova. Ukoliko je nužno prenijeti velik broj datoteka, shell skripta koja pokreće višestruke –put naredbe paralelno ubrzat će proces prijenosa. U takvoj situaciji nije potrebno pisati MapReduce kod. Ako je potrebno skenirati direktorij u potrazi za novim datotekama, može se kreirati „cron job“, a koji će skenirati direktorij, te pri nailasku na nove datoteke izvršiti –put naredbu te ih automatski prenijeti na HDFS. Takav pristup je koristan za prijenos e-pošte na HDFS u definiranim intervalima. S druge strane, HDFS se može montirati kao datotečni sistem, pa se u njemu datoteke mogu kopirati, dodavati, izmjenjivati i brisati na taj način. Strukturirani podaci iz relacijskih baza podataka mogu se uvesti u HDFS, Hive i HBase korištenjem Sqoop alata. Također, sa navedenim alatom je moguće izvući podatke iz Hadoop-a, te ih izvesti u relacijsku bazu podataka i skladište podataka. Korištenjem alata Flume, moguće je neprekidno dodavati podatke iz log datoteka u Hadoop. Konačno, podaci se mogu unijeti u Hadoop i korištenjem „connector-a“ proizvedenih od trećih strana.

Dakle, HDFS fragmentira, distribuira i ponovno ujedinjuje podatke za korisnika. Dio od 64 MB i odabir repliciranja podataka tri puta polazne su postavke HDFS-a. Lockwood (2014) navodi da korisnik može te vrijednosti promijeniti:

1. Dio od 64 MB može se izmijeniti putem dfs.block.size svojstva, a koje se nalazi u hdfs-site.xml. Uobičajeno, ta se vrijednost povećava na 128 MB u produkcijskom okruženju.
2. Faktor replikacije može se izmijeniti putem dfs.replication svojstva, a koje se nalazi u hdfs-site.xml. Također, može se izmjenjivati za svaku datoteku zasebno, a specificiranjem –D dfs.replication=1 na –put naredbenom retku, ili korištenjem hadoop dfs -setrep -w 1 naredbe.

HDFS je korisna tehnologija jer pruža distribuciju i replikaciju podataka, ali i njihov automatski oporavak na datotečnom sistemu korisnika. Relativno je jednostavan za

podešavanje i nije pretežak za shvatiti. Međutim, njegovi pravi alati dolaze na vidjelo kada MapReduce poslovi bivaju izvršeni nad podacima koji su pohranjeni u HDFS. MapReduce poslovi se generalno gledano sastoje od dva koraka, map koraka i reduce koraka. Generalno, proces se odvija na način prikazan na sljedećoj slici:



Slika 2. Prikaz procesa u HDFS-u, te map i reduce koraka. Prilagođeno iz Lockwood (2014).

Lijeva polovica dijagrama opisana je prethodno, a gdje se naredba hadoop dfs –copyFromLocal koristi za premještanje velikih podataka u HDFS, a gdje se automatski replicira i distribuira na više fizičkih komputacijskih čvorova. Iako taj korak nije direktno vezan za map i reduce poslove, ulazni podaci za MapReduce moraju već postojati na HDFS-u prije nego se posao počne izvršavati.

Lockwood (2014) u Map koraku navodi sljedeće aktivnosti:

1. Pokreće određeni broj paralelnih „mappera“ diljem komputacijskih čvorova koji sadrže dijelove ulaznih podataka,
2. Za svaki dio ulaznih podataka, „mapper“ razdvaja podatke u individualne linije teksta na simbolu nove linije (\n),
3. Svaka linija teksta koja je završena sa \n predana je mapper funkciji,
4. Od mapper funkcije se očekuje da pretvori svaku liniju u nulte ili parove sa većim ključ-vrijednost parametrima, a tada emitira navedene parove ključeva-vrijednosti reduce koraku.

Dakle, map korak transformira početne ulazne podatke u nizove parova ključeva-vrijednosti, a sa očekivanjem da se takvi parovi ključeva-vrijednosti mogu smisleno analizirati u reduce koraku. Duplicirani ključevi su dopušteni u takvome modelu. Odluka da se ulazni podaci razdvajaju na poziciji znaka \n polazna je postavka, a koja pretpostavlja da su ulazni podaci samo ascii tekst datoteka. Moguće je izmijeniti kako se ulazni podaci razdvajaju prije nego se predaju mapper funkciji korištenjem InputFormats. (Lockwood 2014)

Kada su svi „mapperi“ završili transformaciju ulaznih podataka, i kad su završili emitirati sve svoje parove ključeva-vrijednosti, isti autor tvrdi da se ti parovi slažu u skladu sa svojim ključem te se predaju reduce izvršiteljima. Reduce izvršitelji dobivaju parove ključeva-vrijednosti na način da svi parovi ključeva-vrijednosti koji dijele isti ključ uvijek odlaze istom reduce izvršitelju. Može se zaključiti da reduce izvršitelj koji ima neki specifični ključ mora imati sve ostale parove ključeva-vrijednosti, a koji dijele isti ključ. Svi će zajednički ključevi biti u neprekidnom nizu parova ključeva-vrijednosti koje su reduce izvršitelji primili. Reducer funkcija tada vrši neku vrstu kalkulacije, a temeljem svih vrijednosti koje dijele zajednički ključ. Funkcija može, primjerice, izračunati zbroj svih vrijednosti za svaki ključ. Parovi ključeva-vrijednosti se tada emitiraju nazad u HDFS, a gdje je svaki ključ unikatan. Svaka od vrijednosti ključeva sada je unikatna, i rezultat je kalkulacija reducer funkcije.

Autor nastavlja tvrdeći da se izlazne vrijednosti map koraka sortiraju i distribuiraju pri predaju u reduce korak, a taj se korak još naziva i „shuffle“. Kada map izvršitelji emitiraju parove ključeva-vrijednosti, ključevi prolaze kroz Partitioner da bi se ustanovilo kojem su reduce izvršitelju poslane. Polazna funkcija Partitioner vrši hash-iranje ključeva, a tada uzima modulus tog hash-a i broj reduce izvršitelja, a sa ciljem otkrivanja koji reduce izvršitelj dobiva upravo taj par ključeva-vrijednosti. S obzirom da je hash datog ključa uvijek isti, svi parovi ključeva-vrijednosti koji dijele isti ključ dobit će istu izlaznu vrijednost od Partitioner-a. Konačno, sa istom vrijednošću završit će kod istog reduce izvršitelja. Kada su svi parovi ključ-vrijednost dodijeljeni pripadnim reduce izvršiteljima, reduce izvršitelji slažu svoje ključeve, a tako da će jedna petlja nad svim ključevima procijeniti vrijednost jednog ključa, a prije prelaska na sljedeći.

Prema tradicionalnim teorijama o bazama podataka, dizajniranje skupova podataka vrši se prije unošenja podataka. Jezero podataka, s druge strane, taj koncept okreće u potpunosti. Omogućava unošenje izvora podataka, te ubacivanje istih u veliki Hadoop spremnik, a bez pripremanja modela podataka unaprijed. Umjesto toga, na raspolaganju su alati za analitiku velike količine podataka, uz visoko definirane vrste podataka koje se nalaze u tom jezeru. Ljudi stvaraju poglede na podatke u procesu pretrage, a taj pristup je prirodan za izgradnju velikih baza podataka. S druge strane, ljudi koji koriste takve modele moraju biti stručni. Kompanija Intuit, kao dio svojeg Intuit Analitičkog Oblaka, koristi jezero podataka, a u kojem pohranjuje podatke o korisničkim nizovima tipkanja mišem, te poslovne i podatke trećih strana. Usmjerili su fokus na „demokratizaciju“ alata, a tako da bi omogućili poslovnim ljudima efektivno korištenje. Jedan od strahova pri implementaciji Hadoop platforme bilo je mišljenje da Hadoop platforma još uvijek nije prikladna za poslovnu upotrebu. Konkretnije, u poslovnim uvjetima od baze podataka se očekuje da ima kontrolu i nadzor pristupa, enkripciju, zaštitu podataka, te mogućnost praćenja podataka od izvora do destinacije, i ostalo. (Mitchell 2014)

Izazovi korištenja Hadoop-a su višestruki. Programiranje MapReduce kodova nije prikladno rješenje za sve vrste problema. Prikladno je za jednostavne upite i probleme koji se mogu razdvojiti na manje nezavisne jedinice, međutim, nije efikasno za ponavljajuće i interaktivne analitičke zadatke. Naime, MapReduce je datotečno-intenzivan. Čvorovi ne komuniciraju među sobom osim kroz slaganja i miješanja, a ponavljajući algoritmi zahtijevaju višestruke faze dok se ne izvrše. U takvom se procesu stvaraju višestruke datoteke između MapReduce faza, što je neefikasno za napredne analitičke računice. S druge strane, teško je pronaći mlade programere koji imaju dovoljno Java iskustva da bi bili produktivni sa MapReduce-om. Upravo nedostatak takvih kadrova doveo je do porasta relacijskih (SQL) tehnologija nad Hadoop-om. Puno je lakše pronaći programere sa SQL vještinama, nego one sa MapReduce vještinama. Administracija Hadoop-a, s druge strane, mnogima djeluje kao znanost. Naime, takva administracija zahtjeva poznavanje operativnih sustava, hardvera i Hadoop kernel postavki. Sigurnost Hadoop platforme svakim danom postaje sve bolja, a značajnije se poboljšala sa Kerberos protokolom autentikacije[23]. Konačno, Hadoop nema jednostavnu i potpunu skupinu alata za upravljanje podacima, filtriranje podataka i metapodatke. Alati za kvalitetu podataka i standardizaciju su najrjeđi. (SAS Institute Inc. 2014)

Mitchell (2014) tvrdi da velika količina podataka, sa mnogim atributima, omogućuje analitičarima vrlo široke mogućnosti procesiranja. Kombinacija velike količine podataka i snage računala, dopušta analitičarima istraživanje novih podataka o ponašanju, a kroz protok vremena. Takvi podaci mogu uključivati podatke o posjećenim stranicama ili lokacijama. Takvi se podaci još nazivaju „raštrkani podaci“, iz razloga što pronalazak podataka koji su od interesa uključuje analitiku velike količine podataka koji nisu relevantni. Korištenje tradicionalnih algoritama sa velikom količinom podataka pokazalo se nemogućim, a nove metode analitike pokazale su se

[23] Temelji se na „ticketima“, da bi dopustio čvorovima komunikaciju putem nesigurne mreže, a tako da bi međusobno dokazali svoj identitet na siguran način. Izvor: Zhu i Tung (2006)

mnogo uspješnijima. Problemi se formuliraju potpuno drugačije kada brzina i memorija prestaju biti ključni problemi. Moguće je pronaći najbolje analitičke varijable pogonom ogromne računalne snage oko problema. Analitika u realnom vremenu i predvidivo modeliranje, a temeljeno na Hadoop jezgri, sporo je i neisplativo. Veću brzinu za navedene zadatke postiže Apache Spark, pogon za procesiranje velike količine podataka, i pripadni jezik za postavljanje upita, Spark SQL. Spark omogućava brze i interaktivne upite, te prikaz rezultata u obliku grafova. Podaci se i dalje nalaze pohranjeni u Hadoop-u, ali performanse upita i analitike su korištenjem Spark-a znatno poboljšane.

Hadoop obećava mogućnost analitike bilo čega, a jedini uvjet je postojanje platforme koja će takvu vrstu informacija moći obuhvatiti i analizirati. Problem je što većina ljudi koji trebaju vršiti neku vrstu analitike nisu vrsni programeri niti matematičari, takvim ljudima je potreban jezik s kojim su dobro upoznati. SQL za Hadoop jedno je od prikladnih rješenja, iako u stvarnosti mnogi drugi jezici mogu biti upotrijebljeni. Alati koji podržavaju upite slične onima iz SQL-a olakšavaju poslovnim ljudima upite nad velikim količinama podataka, ukoliko su prethodno upoznati sa korištenjem SQL-a. SQL za Hadoop približava Hadoop platformu kompanijama. Tradicionalno, korisnici Hadoop-a trebali su pisati skripte koristeći Javu, JavaScript i Python. Takav pristup nije prikladan za mnoge kompanije, jer bi značilo potrebu za zapošljavanjem novih stručnjaka u programiranju i analitici. SQL za Hadoop zaista jest korisna mogućnost. Takvi alati nisu suviše novi. Apache Hive nudi strukturirani upitni jezik za Hadoop, sličan SQL-u, već duže vrijeme. Komercijalne alternative koje dolaze od kompanija Cloudera, Pivotal Software, IBM i ostalih nude više performanse, ali su i brže s vremenom. Takvo svojstvo dobro je za iterativnu analitiku, a prilikom koje analitičar postavlja jedan upit, te nakon primanja odgovora postavlja idući upit. Takva vrsta posla tradicionalno je zahtijevala izgradnju skladišta podataka. SQL za Hadoop neće zamijeniti skladišta podataka u bliskoj budućnosti, ali će ponuditi jeftiniju alternativu skupom softveru, i aplikacijama za određene vrste analitike. (Mitchell 2014)

Alternative tradicionalnim relacijskim bazama podataka nazivaju se NoSQL (skraćeno za „Not Only SQL“) baze podataka. Dobivaju sve veću popularnost kao alati kod specifičnih analitičkih zadataka i aplikacija, a trend je samo rastući. Procjenjuje se da trenutno postoji između petnaest i dvadeset otvorenih NoSQL baza podataka, a svaka od njih je specijalizirana za nešto drugo. Primjerice, NoSQL proizvod sa grafičkim mogućnostima, a kao što je ArangoDB, pruža brži i direktniji način analiziranja mreže odnosa između kupaca i prodavača nego što to pruža relacijska baza podataka. Otvorene SQL baze podataka u upotrebi su već duže vrijeme, ali trenutno su u rastućem trendu zbog sve veće potrebe za takvim vrstama analiza, a kakve one nude. NoSQL baza podataka može se koristiti, primjerice, za analitiku proizvoda, kupaca i vremena koje kupci provedu ispred proizvoda na polici. Senzori šalju tokove podataka čija količina raste eksponencijalno. NoSQL baza podataka sa parom ključeva-vrijednosti pravi je izbor za takvu situaciju, a zbog svoje specifičnosti, visokih performansi i niskih sistemskih zahtjeva. (Mitchell 2014)

Isti autor nastavlja opisujući tehnike strojnog učenja koje se temelje na neuralnim mrežama, te se nazivaju duboko učenje. Takve tehnike još su uvijek u razvitku ali pokazuju veliki potencijal za rješavanje poslovnih problema. Zahvaljujući takvim tehnikama računalo je sposobno prepoznati objekte od interesa u velikim količinama podataka, a koji mogu biti nestrukturirani, ali i binarni. Takva su računala u mogućnosti izvući veze i odnose bez nekog posebnog modela ili programskih naredbi. Primjerice, jedan algoritam dubokog učenja, procesirajući podatke sa Wikipedije, naučio je da su California i Texas obje države iz Sjedinjenih Američkih Država. Takav algoritam ne treba biti oblikovan da razumije određene koncepte, a što ga razlikuje od starih tehnika strojnog učenja. Velika količina podataka sadrži mnoge nestrukturirane i raznovrsne podatke, a nad kojima će se u budućnosti vršiti razne napredne analitičke tehnike, a kao što je duboko učenje, na načine koji trenutno još uvijek nisu dobro poznati. Mogao bi se koristiti za prepoznavanje različitih vrsta podataka, oblika, boja i objekata u video zapisima. Neuralna mreža koju je izgradio Google 2012. godine

prepoznala je postojanje mačke na fotografiji, a što je jedan od primjera dubokog učenja. Napredne tehnike strojnog učenja bit će bitne u budućnosti.

Isti autor tvrdi da su baze podataka koje se vrše u memoriji u upotrebi radi ubrzanja analitičkih procesa, te su vrlo popularne i efektivne kada su implementirane na pravi način. Mnoge kompanije već sada koriste hibridno transakcijsko/analitičko procesiranje (HTAP), a što dopušta transakcijama i analitičkim procesima da budu smješteni u istoj bazi podataka koja se nalazi u memoriji. Kod sistema gdje korisnici trebaju vidjeti iste podatke na isti način, više puta tokom dana, a gdje se podaci ne mijenjaju često i značajno, upotreba baza podataka u memoriji nije potrebna, i stvara dodatne troškove. Iako se analitika može vršiti brže koristeći transakcijsko/analitičko procesiranje, sve se transakcije moraju nalaziti u istoj bazi podataka. Većina današnjih analitičkih alata pokušava sakupiti transakcije mnogih sustava na jedno mjesto, a što predstavlja problem za transakcijsko/analitičko procesiranje. Dodatno, baza podataka koja se nalazi u memoriji dodatan je proizvod, a koji zahtjeva održavanje, zaštitu, integraciju i ostalo. Intuit je korištenjem Spark-a smanjio potrebu za bazama podataka u memoriji, tvrdeći da će radije odabrati Spark ako je u stanju zadovoljiti većinu njihovih potreba, nego baze podataka u memoriji koje mogu riješiti sve potrebe, istaknuvši da će koristiti prototip za testiranje platforme.

4. Alati za analitiku velike količine podataka

U ovom će poglavlju biti detaljno opisane najpopularnije i analitičke platforme sa najboljim performansama u trenutku pisanja rada. Bit će uspoređene njihove performanse uz pomoć različitih testova. Nakon pregleda performansi bit će opisan način njihova rada, raspoređivanja zadataka, izvršavanja i ostalo. Dodatno, bit će prikazani testovi performansi sa raznim tipovima podataka i ostalim.

Impala je trenutno vrlo popularna, i korištena više od ostatka navedenih alata, te njeno korištenje visoko preporučujem. Iako su svi navedeni alati dobri, prava usporedba je najbolja kada je izvršena na vlastitim podacima i potrebama procesiranja. Iako, Impala se trenutno smatra najboljom i najzrelijom opcijom od navedenih. Ukoliko se Impalu želi koristiti sa već aktivnim Hadoop grozdom, bit će potrebno obaviti dodatne poslove jer se Impala u većini slučajeva koristi kao CDH mogućnost. Konačno, bitno je znati da kod Impale središnji upit mora stati u memoriju, odnosno, ukoliko „group by“ upit prijeđe preko 30 GB (primjerice, raspoloživi RAM), prije izvršenja „having“ odredbe koja ga uspješno razdvaja na dijelove podataka veličine 1 MB, upit će biti neuspješan. Takva se situacija kod ostalih MPP (engl. massively parallel processing, MPP)[24] sustava ne događa. (Tariq 2013)

Aplikacije za procesiranje velike količine podataka zahtijevaju tehnike, izvore podataka i formate za pohranu. Prvi sistemi za takve poslove dali su korisniku nisko proceduralno programsko sučelje. Takvi su sistemi zahtjevali ručnu optimizaciju za postizanje željenih performansi. Zbog toga su mnogi novi sustavi ciljali na stvaranje produktivnijeg korisničkog iskustva pružanjem relacijskog sučelja velikoj količini podataka. Sistemi poput Hive, Pig, Shark i ostalih iskorištavaju prednosti deklarativnih upita za pružanje optimizacije sa većom automacijom. Popularnost

[24] Sustavi kod kojih više jedinica vrši usklađeno procesiranje zadataka istog programa. Svaki izvršitelj izvršava druge zadatke, te ima vlastiti OS i memoriju. Izvor: Techopedia (2015)

takvih programa pokazuje da korisnici preferiraju pisanje deklarativnih upita. Nažalost, relacijski pristup nije dovoljan za mnoge aplikacije koje rade sa velikom količinom podataka. Naime, mnogi izvori podataka mogu biti polu ili ne-strukturirani, a što zahtjeva prilagodbu koda. Dodatno, napredna analitika poput strojnog učenja teško se može vršiti u relacijskim sistemima. Idealna situacija je kombinacija relacijskih upita sa naprednim proceduralnim algoritmima. Nažalost, dugo vremena su te klase sustava bile razdvojene, tjerajući korisnike na biranje između relacijskih ili proceduralnih sistema.

Važno je odabrati prikladnu analitičku platformu i Hadoop distribuciju, te osigurati dovoljne hardverske resurse, i potrebito softversko podešavanje sa ciljem iskorištavanja prednosti upotrebe velike količine podataka na svim raspoloživim resursima koji funkcioniraju kao cjelina. S druge strane, novi trend u analitici velike količine podataka jest plaćanje pretplate, te vršenje analitike u oblaku, na tuđem sklopovlju i softveru. Računalstvo u oblaku rastući je trend, a što sa sobom donosi povećanje Software as a Service (SaaS) usluga. Na taj način korisnici ne trebaju brinuti o hardveru, softveru i održavanju, već su u kratkom roku spremni za rad, a sve što je potrebno jest plaćati pretplatu pružatelju usluge.

Suvremeni sustavi kombiniraju jednostavnost SQL-a sa kompleksnošću sustava za analitiku velike količine podataka. Platforme se razvijaju rapidno, a performanse se poboljšavaju svakim danom. Osim poboljšanih performansi, mnoge suvremene platforme za analitiku velike količine podataka krasi izuzetna jednostavnost vršenja upita, što BI (engl. Business Inteligence) korisnicima i korisnicima upoznatima sa pisanjem SQL upita pruža mogućnost analitike na poznati način. Alati su sve bolji, tako da je moguće prikazivati analitičke rezultate na razne načine, uključujući grafove, pite, prikaze na mapi, po regiji, i na mnoge druge načine. IoT je zaslužan za rapidan rast velike količine podataka, a kada ta količina bude značajnije upotrijebljena, IoT će postati globalna stvarnost. Pametni gradovi su početak u stvaranju „pametnog svijeta“. Proces analitike velike količine podataka odvija se u nekoliko faza, a koje

Morabito (2015) dijeli na: a) definiranje ciljeva analitike, b) odabir podataka, c) procesiranje podataka, d) rudarenje podataka, e) evaluacija i interpretacija obrazaca, i f) vizualizacija i povratne informacije, a koje se onda mogu koristiti za daljnje specificiranje cijelog procesa.

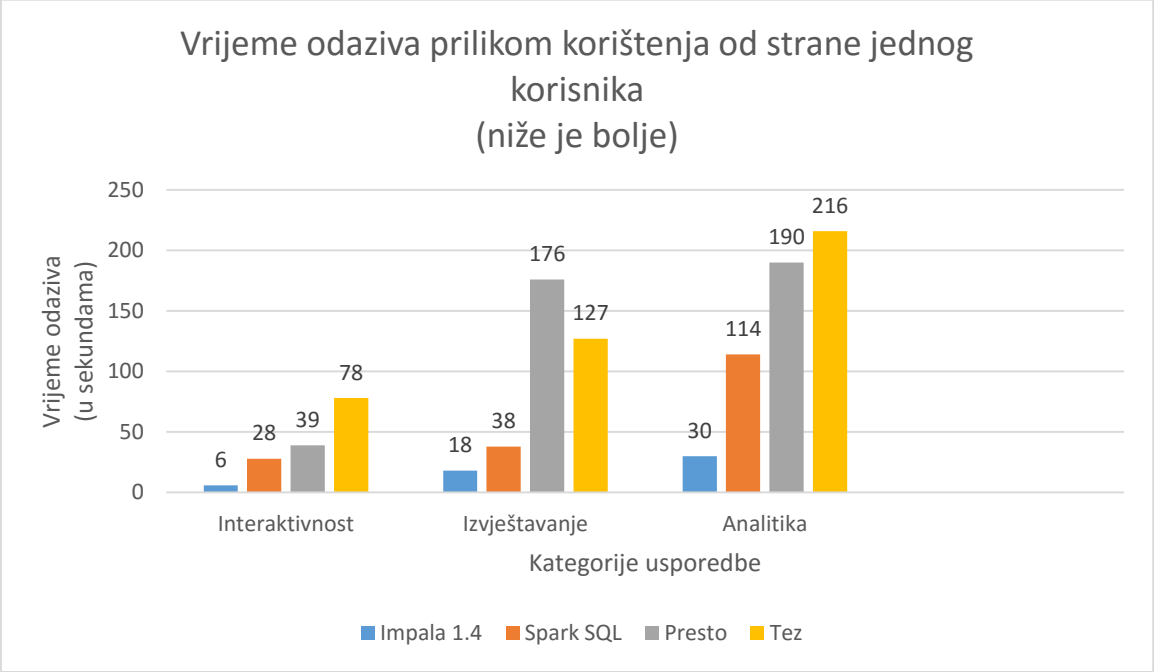
4.1 Usporedba performansi najpopularnijih analitičkih modula

Temeljem testova koje su izvršili Erickson, Kornacker, Kumar i Rorke (2014), kod upita jednog korisnika, Impala je do 13 puta brža od alternativa, i 6.7 puta brža u prosjeku. Kod upita više korisnika, Impala je brža i do 27.4 puta od alternativa, i 18 puta brža u prosjeku. Test se vršio na grozdu od točno 21 čvorova. Svaki je čvor imao 64 GB memorije, 2 procesora sa 12 jezgri (Intel Xeon CPU E5-2630L 0 na 2.00GHz), te 12 diskova od kojih svaki sa 932 GB – jedan disk za OS (engl. operating system, OS), ostatak za HDFS. U testu nije sudjelovao Shark iz razloga što ga je u međuvremenu zamijenio Hive-on-Spark. U test je ubačen i Spark SQL. Dakle, u testu su sudjelovali sljedeći sustavi:

- Impala 1.4.0: Opis u prethodnom dijelu rada,
- Hive-on-Tez: Finalna faza 18-mjesečne Stinger inicijative (ili Hive 0.13),
- Spark SQL 1.1: Novi projekt koji omogućuje developerima vršenje poziva prema SQL-u unutar reda, a za potrebe uzorkovanja, filtriranja i agregacije kao dio Spark-a, aplikacije za procesiranje podatka,
- Presto 0.74: Facebook-ov projekt za vršenje upita.

Da bi test i opterećenja Hadoop-a, te količina podataka po čvoru bili realistični, upiti su vršeni nad 15 TB podataka, raspoređeno diljem 21 čvora. Test koji se vršio bio je jednak svih prethodnim testovima. S obzirom da u svim testiranim modulima osim Impale nedostaje optimizator baziran na trošku, svi su moduli testirani sa upitima koji su konvertirani u SQL-92 „join-ove“. Isti su upiti izvršeni i nad Impalom, iako Impala

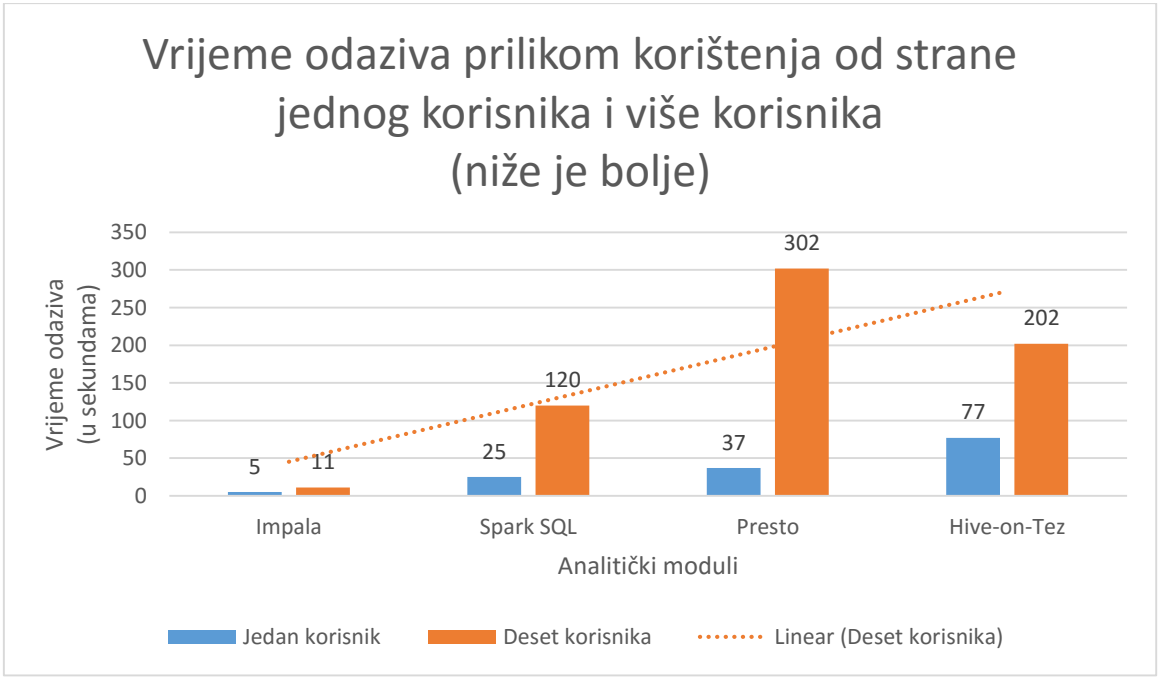
generira iste rezultate i bez navedenih modifikacija. Izabrani su najoptimalniji formati datoteka diljem svih modula, te se koristila Snappy kompresija, a sa ciljem da usporedba bude u istim uvjetima. Nadalje, svaki modul bio je zaposlen nad formatom datoteka koji osigurava najbolje moguće performanse i poštenu, te konzistentnu usporedbu: Impala na Apache Parquet, Hive-on-Tez na ORC, Presto na RCFile, i Spark SQL na Parquet. Konačno, korištene su standardne rigorozne tehnike testiranja za svaki modul, a kao što su višestruka ponavljanja, podešavanja i ostalo. (Erickson, Kornacker, Kumar i Rorke 2014)



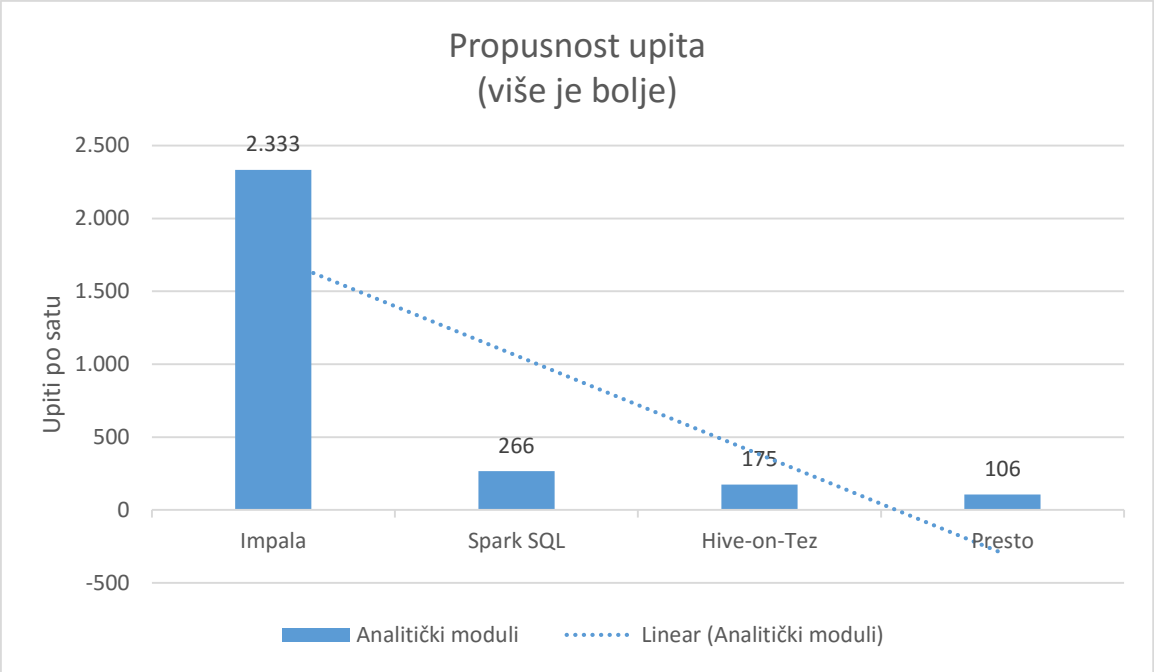
Graf 3. Rezultati prilikom korištenja od strane jednog korisnika. Iz priloženog grafa može se zaključiti da je Impala brža od svih ostalih alternativa. Njezina prednost je između 2.1 do 13.0 puta veće brzine od alternativa, dok je u prosjeku brža za 6.7 puta. Temeljem starijih testova može se zamijetiti da Impala s vremenom postaje sve brža od alternativa, a tako se od zadnjeg testa sa 4.8 puta ubrzala na 6.7 puta prosječno veću brzinu. Najveći napredak u brzini zamjećuje se pri usporedbi povijesnih podataka sa Hive-on-Tez i Presto. U odnosu na Hive-on-Tez, Impala se prosječno ubrzala sa 4.9

puta na 9 puta; dok je u odnosu na Presto, prosječna brzina porasla sa 5.3 puta na 7.5 puta. Izvor: Erickson, Kornacker, Kumar i Rorke (2014)

Kod testiranja prilikom korištenja od strane 10 korisnika rezultati su bili isti kao i kod testiranja u prošlosti. Kod Impale, prednost u performansama je sve veća što je više korisnika. Njezina prosječna prednost raste sa 6.7 puta na 18.7 puta kada se prelazi sa jednog na više korisnika. Prednost varira između 10.6 puta do 27.4 puta ovisno o usporedbi. Dakle, kod više korisnika Impalina je prednost znatno veća nego kod jednog korisnika, a što je prikazano u sljedećem grafu:



Graf 4. Rezultati prilikom korištenja od strane deset korisnika. Iz grafa se može zaključiti da se vrijeme odaziva Impale udvostručilo pod višestrukim opterećenjem, dok je kod alternativa taj odnos i do pet puta gori. Izvor: Erickson, Kornacker, Kumar i Rorke (2014)



Graf 5. Impala prednost postiže zahvaljujući svojem izuzetno efikasnom modulu za vršenje upita, a koji ukazuje na propusnost koja je od alternativa bolja između 8 puta i 22 puta, a u prosjeku je bolja za 14 puta. Izvor: Erickson, Kornacker, Kumar i Rorke (2014)

Iz prikazanih grafova može se zaključiti da Impala ima najbolje performanse, a očekuje se da će biti najkompatibilniji i najkorišteniji analitički SQL modul za Hadoop. Grafovi su pokazali rastuću prednost Impale nad konkurencijom, a pogotovo pod višestrukim opterećenjem. Dolaskom verzije Impala 2.0, modul je dodatno poboljšan, a specifikacije će biti navedene u daljnjem radu. Erickson, Kornacker, Kumar i Rorke (2014) tvrde da je Impala jedini SQL modul koji nudi:

- Nisku latentnost, te bogat SQL za BI (engl. business inteligence, BI) korisnike,
- Sposobnost procesiranja pod mnogostrukim opterećenjem,
- Efikasno korištenje resursa u zajedničkom dijeljenom okruženju (prilikom mnogostrukog korištenja),
- Otvorene formate za pristupanje bilo kojim podacima iz bilo kojeg nativnog Hadoop modula,

- Podrška za mnoge distributere.

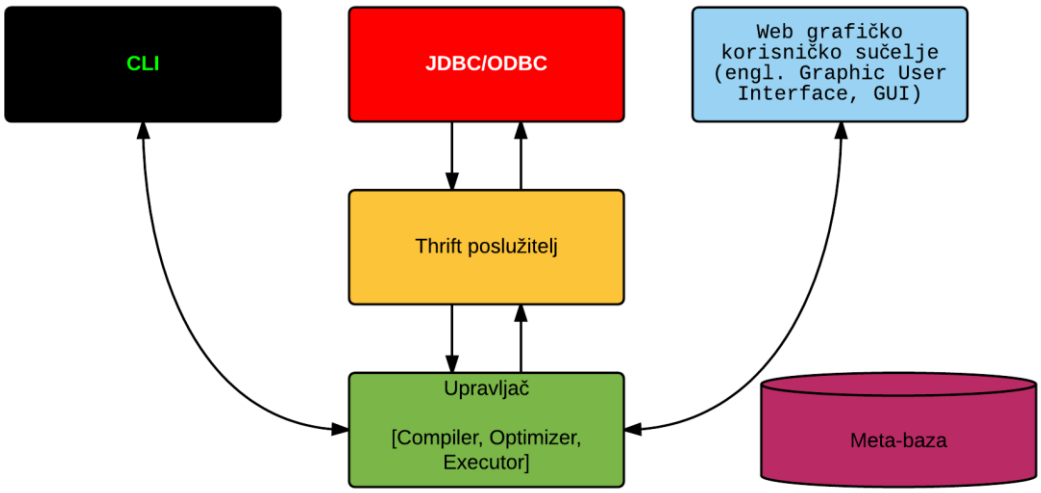
SQL jezik je „jezik podataka“, pa su posljedično mnogi alati izgrađeni tako da bi donijeli SQL Hadoop-u. (Rathbone 2014) Alati postoje raznih vrsta, od jednostavnih omotača nad MapReduce-om, do cjelovitih implementacija za skladišta podataka sagrađenih nad HDFS-om, a koja obuhvaćaju i sva ostala područja. Alata je mnogo, a bit će nabrojani redoslijedom – od jednostavnijih prema kompliciranijima. Razmotrit će se sljedeće tehnologije:

1. Apache Hive,
2. Impala,
3. Presto (Facebook),
4. Shark,
5. Apache Drill,
6. EMC/Pivotal HAWQ,
7. BigSQL od IBM-a,
8. Apache Pheonix (za HBase),
9. Apache Tajo.

4.2 Apache Hive

Apache Hive izvorna je SQL-na-Hadoop-u solucija. Java projekt je otvorenog koda, a koji pretvara SQL u serije MapReduce poslova koji se vrše nad standardnim Hadoop pratiteljima zadataka (engl. tasktrackers). Pokušava izgledati poput MySQL-a koristeći metabazu za spremanje tabličnih schema, particija i lokacija. Velikim dijelom podržava MySQL sintakse, te organizira setove podataka koristeći poznate database/table/view konvencije. (Rathbone 2014) Osnovna struktura u bazi podataka je tablica, analogna tablicama u relacijskim bazama podataka. Sve tablice u Hive bazi podataka imaju vezani HDFS direktorij. Podržane su i tablice koje su izvan Hive-a, a koje se nalaze u drugačijem sustavu datoteka poput NFS-a ili lokalnih direktorija. Podaci su distribuirani u obliku particija i spremnika (engl. bucket). Svaka tablica može imati jednu ili više particija u Hive-u. Podaci su raspodijeljeni unutar HDFS u poddirektorijima. Podaci u particijama se nadalje dijele na spremnike. Raspodjela u

spremnicima bazira se na hash vrijednosti stupca u tablici. Svaki spremnik je pohranjen kao datoteka u particijskom poddirektoriju. Tipovi podataka koje Hive podržava jesu: Integer, Floating Point, Boolean, String, Datum, Kolekcije poput mapa ili array, korisnički definirani tipovi podataka. Sučelje za vršenje upita koje sliči SQL-u, modelirano prema MySQL-u – HiveQL, podržava upite poput unija, agregacija, spajanja, select upita, i pod-upita u FROM izjavi. HiveQL podržava kreiranje novih tablica u skladu sa particijama i spremnicima, a omogućuje i unos podataka u jednu ili više tablica. Nažalost, ne podržava brisanje ili ažuriranje podataka. (Singhvi 2014)



Slika 3. Prikaz Hive arhitekture. CLI, JDBC[25]/ODBC[26] ili Web sučelje tvore vanjska sučelja do Hive platforme. Thrift poslužitelj je klijentski API koji korisniku omogućava izvršenje HiveQL izjava. Meta-baza je sistemski katalog koji bilježi informacije o informacijama koje su spremljene u HDFS, i u kojem dijelu. Upravljač je glavna

[25] JDBC (engl. Java Database Connectivity, JDBC) upravljač je softverska komponenta koja omogućuje Java aplikaciji interakciju sa bazom podataka.
[26] ODBC (engl. Open Database Connectivity, ODBC) je standardni programabilni middleware API za pristupanje Sustavu upravljanja bazom podataka (engl. DataBase Management Systems, DBMS).

komponenta koja vrši kompilaciju, optimizaciju i izvršenje HiveQL izjava. Izvor: Singhvi (2014)

Isti autor nastavlja tvrdeći da se Hive dobro uklapa u zahtjev za sučeljem niske razine u Hadoop-u. Podržava vanjske tablice što omogućava procesiranje podataka bez pohranjivanja u HDFS. Hive koristi optimizator baziran na pravilima za optimiziranje logičkih planova. Podržava particioniranje podataka na razini tablica radi poboljšanja performansi. Meta-baza znatno olakšava pretraživanje. S druge strane, Hive ne podržava ažuriranje i brisanje podataka. Podaci se prvo trebaju unijeti iz datoteke koristeći LOAD naredbu. Također, nije implementirana kontrola pristupa. Konačno, izmijenjeni pod-upiti nisu podržani.

Hive je dobar izbor za transformacije podataka, upite koji služe kao potpora odlučivanju. Hive ima dobru podršku za prozore i rollup/cube operacije. Ako se koristi Hortonworks Hadoop distribucija, Hive će biti dobar izbor; Impala je dostupna za CDH, AWS i MapR platforme. Impala je brža za upite nad poljima visoke kardinalnosti. Na neki način konkurira HBase-u. Impala je visoko korištena među analitičarima koji vrše specifične upite nad velikim setovima podataka, koji su pohranjeni u Hadoop-u. Ukoliko je opterećenje višestruko, Hive je dobar izbor jer se pokreće kao dio MapReduce/Tez aplikacijske platforme pod YARN-om. Posljedično, Hive ima dobre performanse pri višestrukim korisnicima i zapošljava cijeli grozd na kraće vrijeme. (Lipcon, Moore, Jahangir i Chaplot 2012) Pri korištenju Impale su zabilježeni slučajevi gdje su se tablice morale osvježavati pri dodavanju novih datoteka u HDFS, a što je kasnije ispravljeno, te navedeno na blogu od Cloudere:

> *„Impala 1.2 (Shipped Oct. 2013)*
>
> *Automatic metadata refresh – enables new tables and data to seamlessly be available for Impala queries as they are added without having to issue a manual refresh on on each Impala node"* Erickson i Kornacker (2014)

Potreba za ručnim osvježavanjem nije postojala pri korištenju Hive-a. Konačno, Impala je pokazala visoke zahtjeve za memorijom pri vršenju upita. (Lipcon, Moore, Jahangir i Chaplot 2012)

Hive pruža (Rathbone 2014):

- Sučelje za vršenje upita koje sliči SQL-u, modelirano prema MySQL-u i nazvano Hive-QL,
- Klijent za naredbeni redak,
- Dijeljenje metapodataka putem središnje usluge,
- JDBC upravljače,
- Višejezične Apache Thrift upravljače,
- Java API za kreiranje osobnih funkcija i transformacija.

Tablica 7. Prikaz povijesnih postignuća koja su dovela do prvog stabilnog izdanja Hive-a (v1.0). Izvor: Gates i Shanklin (2015)

Izdanje	Datum	Glavne značajke	Linije koda
0.11.0	15. svibnja 2013.	ORCFile, OLAP funkcije	132000
0.12.0	15. listopada 2013.	SQL tipovi podataka	94100
0.13.0	21. travnja 2014.	Hive on Tez, vektorizirani upiti	419300
0.14.0	12. studenog 2014.	CBO, ACID	340500
1.0	**5. veljače 2015.**	**STABILNO IZDANJE**	-

Hive je jedan od alata koji je instaliran na gotovo svim Hadoop instalacijama. Jednostavan je za podešavanje i ne zahtjeva veliku infrastrukturu za početak rada.

Jeftin je za korištenje, i ne treba ga zanemariti. S druge strane, upiti koji se vrše sa Hive-om donose poprilično spore rezultate, a zbog predradnji koje se vezuju za MapReduce poslove. Konačno, Hortonworks radi na razvoju Apache Tez-a, kao nove pozadinske podrške za Hive, a sa ciljem vraćanja bržih rezultata koji trenutno nisu mogući korištenjem MapReduce-a. (Rathbone 2014)

4.3 Impala

Rathbone (2014) tvrdi da je Cloudera Impala interaktivni SQL upitni modul za Hadoop otvorenog koda. Izgradila ga je Cloudera, jedan od najvećih prodavača na tržištu. Impala omogućuje vršenje SQL upita nad postojećim Hadoop podacima, baš poput Hive-a. Za razliku od Hive-a, ne koristi MapReduce za izvršavanje upita, već koristi vlastiti set izvršitelja koje je nužno instalirati uz komputacijske čvorove. Impala je izgrađena tako da bi se mogla integrirati sa standardnim okruženjima poslovne inteligencije, te pruža:

- Podršku za ANSI-92 SQL sintakse, te SQL-2003 analitičke funkcije,
- Podršku za standardne tipove datoteka (integer, floating point, CHAR, VARCHAR, STRING, TIMESTAMP, DECIMAL),
- HIVE-QL podršku,
- Klijent za naredbeni redak,
- ODBC i JDBC upravljače,
- Međudjelovanje sa Hive Metabazom za dijeljenje schema između platforma,
- C++ API za kreiranje funkcija i transformacija,
- Kerberos ili LDAP autentikaciju, gdje autorizacija slijedi uobičajene SQL uloge i privilegije.

Isti autor nastavlja tvrdeći da je Impala izgrađena da bi upotpunila korištenje Apache Hive-a, tako da se pri potrebi za većim brzinama pristupa podacima nudi kao dobar

izbor[27]. Da bi se u potpunosti iskoristile prednosti Impaline arhitekture potrebno je pohraniti podatke u specifičnom formatu (Parquet[28]), što može biti problematična tranzicija. Također, nužno je instalirati set izvršitelja uz komputacijske čvorove, a što rezervira resurse za sebe, odnosno, manje resursa ostaje za pratitelje zadataka. Impala trenutno podržava YARN, što olakšava implementaciju za sljedeću generaciju Hadoop grozdova.

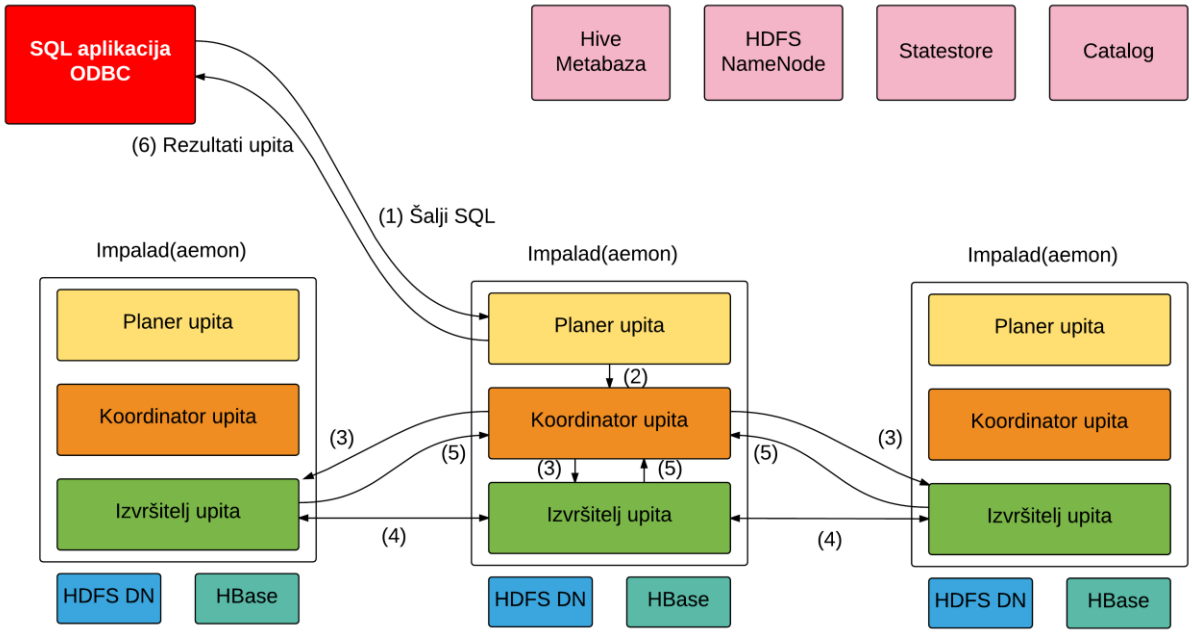
Impala kombinira SQL podršku i performanse pod višestrukim opterećenjem koje su slične performansama tradicionalnih analitičkih baza podataka, sa skalabilnošću i fleksibilnošću Hadoop platforme. Također nudi napredne sigurnosne karakteristike i proširenja za upravljanje putem Cloudera Enterprise rješenja. Impala je novi modul, napisan u C++ i Javi. Upotrebljava standardne Hadoop komponente poput HDFS, HBase, YARN i ostalo, te je sposoban čitati većinu široko rasprostranjenih formata (primjerice Parquet, RCFile i ostali). Impalina arhitektura bazira se na daemon[29] procesima koji su odgovorni za sve aspekte vršenja upita. Daemon procesi se vrše na istim računalima kao i Hadoop platforma, te ubrzavaju povrat odgovora u odnosu na ostale analitičke module. Performanse, ovisno o vrsti i obujmu opterećenja, mogu biti bolje i od nekih komercijalnih rješenja. Zbog HDFS limitacija i fokusa na brzinu i performanse vršenja upita, Imala trenutno ne podržava UPDATE i DELETE naredbe, iako se može koristiti INSERT ili CREATE TABLE AS SELECT za kopiranje podataka između tablica. U procesu se mogu ukloniti ili izmijeniti retci po potrebi. (Kornacker, Behm i ostali 2015)

Isti autori tvrde da se korištenjem HDFS API-a može dodavati podatke u tablice kopiranjem/premještanjem u direktorij tablice. Također, može se koristiti i LOAD DATA izjava. Impala podržava skupno unošenje i brisanje uklanjanjem tablične

[27] Pogotovo ako je u produkciji upogonjen Cloudera, MapR ili Amazon Hadoop grozd.
[28] Format stupčane pohrane raspoloživ svim projektima u Hadoop ekosistemu, neovisno o odabranoj platformi za procesiranje podataka, modelu podataka ili programskom jeziku. Izvor: Oracle Parquet (2014)
[29] Instanca, odnosno Impalin daemon naziva se Impalad.

particije (ALTER TABLE DROP PARTITION). Nakon unošenja podataka ili izmjena može se osvježiti statistiku o tablici korištenjem COMPUTE STATS <table> izjave. Implementacija Impale sastoji se od tri servisa. Impala daemon servis prihvaća klijentske upite, te raspoređuje njihovo vršenje u grozdu. Kada daemon upravlja izvršenjem upita, u ulozi je koordinatora upita. S obzirom da su svi daemon-i simetrični, oni mogu vršiti sve uloge. Svaki stroj u grozdu ima daemon koji na njemu vrši datanode proces[30]. Na taj način Impala može čitati blokove iz sistema datoteka, bez potrebe za mrežom. Statestore daemon je servis za objavljivanje i potpis metapodataka. Taj servis šalje metapodatke svim procesima. Treći daemon naziva se Catalog daemon, te služi kao kataloški repozitorij i početna točka za pristup metapodacima.

Cloudera Manager ima pristup navedenim servisima, ali i konfiguracijskim opcijama. Sljedeći dijagram prikazuje tok vršenja upita kod Impale.



[30] Blok server za pripadajuću HDFS implementaciju.

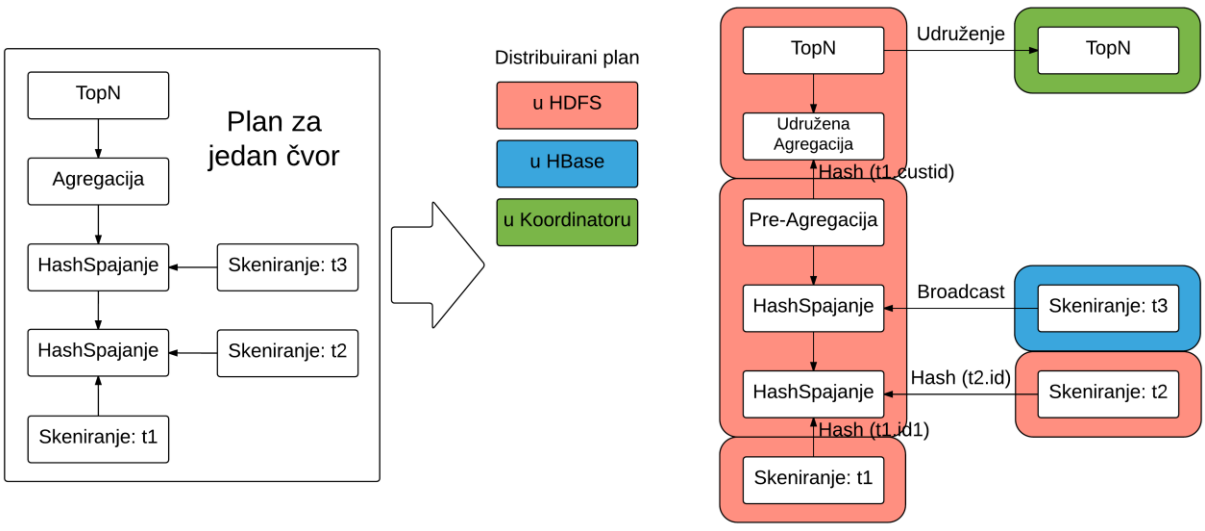
Slika 4. Prikaz radnog toka pri procesiranju upita; Impala je prikazana kao distribuirani sistem za procesiranje upita. Izvor: Kornacker, Behm i ostali (2015)

Autori nastavljaju tvrdeći da kod Impale svi čvorovi moraju biti sposobni prihvatiti i izvršiti upite. Svi čvorovi moraju imati ažuriran katalog i ažurirano stanje u grozdu, a tako da bi se zadaci mogli pravilno raspoređivati. Problem je moguće riješiti korištenjem servisa za upravljanje grozdom, iako takav pristup nije bio planiran prilikom dizajna Impale. Statestore sadrži nizove unosa, koji se sastoje kod ključa, vrijednosti i verzije. Ključ i vrijednost su nizovi bajtova, dok je verzija 64-bitni integer. Statestore ne razumije unose, a zadržava ih sve dok se servis ponovno ne uključi. Procesi koji zahtijevaju ažuriranja o unosima nazivaju se pretplatnici. Oni se registriraju kod statestore-a prilikom uključivanja te pružaju listu unosa. Statestore svakom pretplatniku šalje ažuriranje za svaki registrirani unos. Nakon registracije statestore periodički šalje ažuriranje unosa i „keepalive“ poruke. Ažuriranja se vrše slanjem promjene, odnosno delte. To je moguće zahvaljujući identifikatoru kod pretplatnika koji vraća informaciju o verziji koju pretplatnik ima. Garantira se da će se lista promjena koju vrše pretplatnici izvršiti prije sljedećeg ažuriranja. Keepalive poruke služe održanju konekcije između statestore-a i pretplatnika. Konekcija bi u nedostatku keepalive poruka istekla. Pri detekciji neuspješnog pretplatnika, statestore prestaje slati ažuriranja na toj konekciji. Neki unosi mogu biti označeni sa „transient“, što znači da će se obrisati ukoliko njihov pretplatnik vlasnik zakaže. Na taj se način održava svježina informacija za grozd. Statestore ažurira pretplatnike nejednakim redoslijedom, a metapodatke ne sprema na disk. Metapodaci se spremaju kod pretplatnika, koji mogu ažurirati statestore prilikom ponovnog paljenja.

Katalog servis pruža kataloške metapodatke daemonima putem statestore broadcast mehanizma, te vrši DDL (engl. data definition language/data description language, DDL) operacije umjesto daemona. Katalog servis vadi informacije metabaza te ih oblikuje u kompatibilnu katalošku strukturu. Takav pristup omogućuje dodavanje

novih metabaza na brz način. Sistemski se katalog može prilagođavati bez replikacije u metabazu. Servis učitava površan pogled na tablice, a metapodaci se dodaju u pozadini naknadno. Zahtjev za tablicom koja još nije učitana postaje prioritetan. (Kornacker, Behm i ostali 2015)

Prethodni autori dijele kompilaciju upita na raščlambu upita, semantičku analizu i planiranje i optimizaciju upita. Izvršivi plan upita sastoji se od nekoliko faza. Prva faza je planiranje za jedan čvor, a druga faza je planiranje paralelizacije i fragmentacije. U prvoj fazi raščlambeno se stablo prevodi u neizvršivo plansko stablo za jedan čvor. Planski čvorovi u takvome stablu su sljedeći: skeniranje HDFS/HBase, hash spajanje, cross spajanje, unija, hash agregacija, sortiranje, top-n, i analitička evaluacija. Druga faza uzima plan za jedan čvor kao input, a kao output daje distribuirani plan izvršenja. Tokom te faze donosi se strategija spajanja za svaki čvor spajanja. Podržane strategije spajanja šalju se svima i bivaju particionirane. Svi strojevi u grozdu dobivaju cijelokupnu izgradbenu stranu spajanja. Impala odlučuje o strategiji, a bira ju temeljem što manje razmjene podataka na mreži. U procesu koristi čvorove i postojeće particije inputa za ubrzanje.

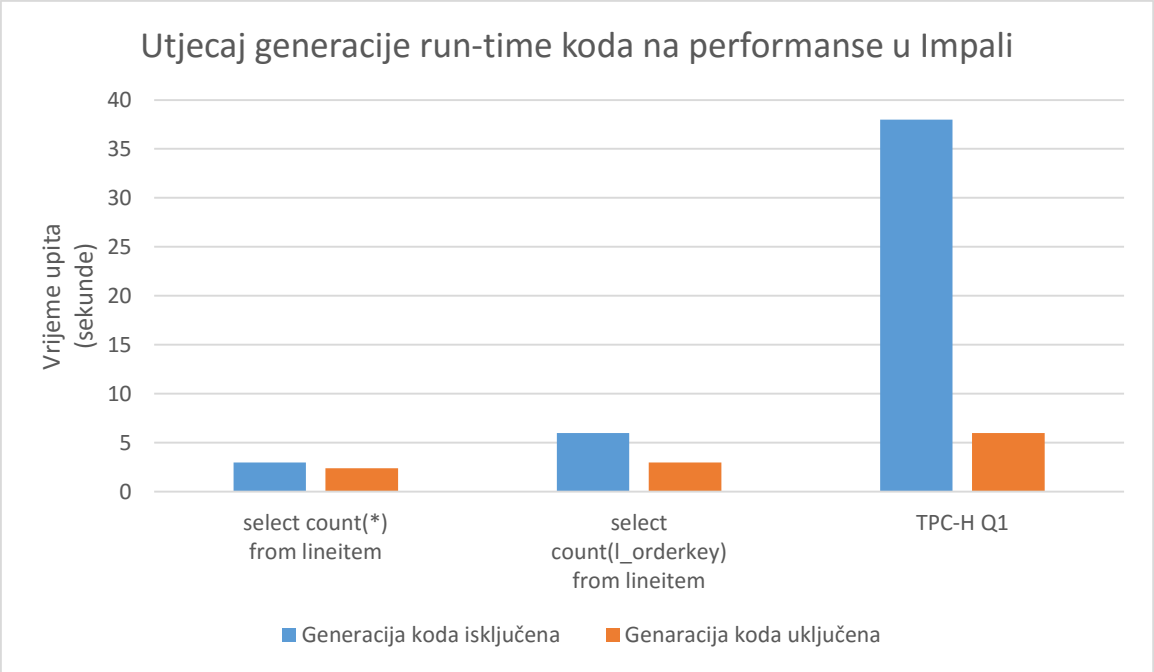


Slika 5. Primjer optimizacije upita u dvije faze. Opis dijagrama nalazi se u nastavku teksta. Izvor: Kornacker, Behm i ostali (2015)

Lijeva strana prethodnog dijagrama (slika 5.) prikazuje plan za spajanje dvije HDFS tablice (t1, t2) i jedne HBase tablice (t3), za jedan čvor. Slijedi agregacija i slaganje sa limitom (top-n). Desna strana dijagrama prikazuje distribuirani razdijeljeni plan. Pravokutnici sa zaobljenim rubovima predstavljaju granice fragmenata, a strelice predstavljaju razmjenu podataka. Tablice t1 i t2 su spojene korištenjem strategije particioniranja. Skeniranja su u vlastitim fragmentima, jer su njihovi rezultati poslani čvoru spajanja. Spajanje sa t3 je broadcast spajanje. Nakon spajanja vrši se pre-agregacija u istom fragmentu u kojem je i posljednje spajanje. Rezultati pre-agregacije su razmijenjeni putem hash-a, te agregirani još jedan put za računanje konačnog rezultata agregacije. Isti pristup se vrši nad top-n, dok je posljednji korak u koordinatoru koji rezultate šalje korisniku.

Dijelovi upita šalju se prema backend-u koji ih brzo izvršava, koristeći suvremen hardver. Rezervira malo memorije u odnosu na druge module napisane u Javi, a vrijeme izvršenja ubrzava korištenjem runtime kod generacije. Pritom se koristi LLVM[31], te kolekcije povezanih alata. LLVM je modularan i ponovno iskoristiv, te pruža JIT[32] mogućnosti kompilacija unutar aktivnih procesa. Impala kreira runtime kodove radi stvaranja funkcija, a koje su specifične po upitu, i važne za performanse. Generacija koda vrši se za funkcije sa unutarnjim petljama, a koje su ponavljajuće. Bez generacije koda javljaju se neefikasnosti. (Kornacker, Behm i ostali 2015)

[31] Engl. Low Level Virtual Machine, LLVM; jest kompajlerska biblioteka.
[32] Engl. Just-In-Time, JIT.



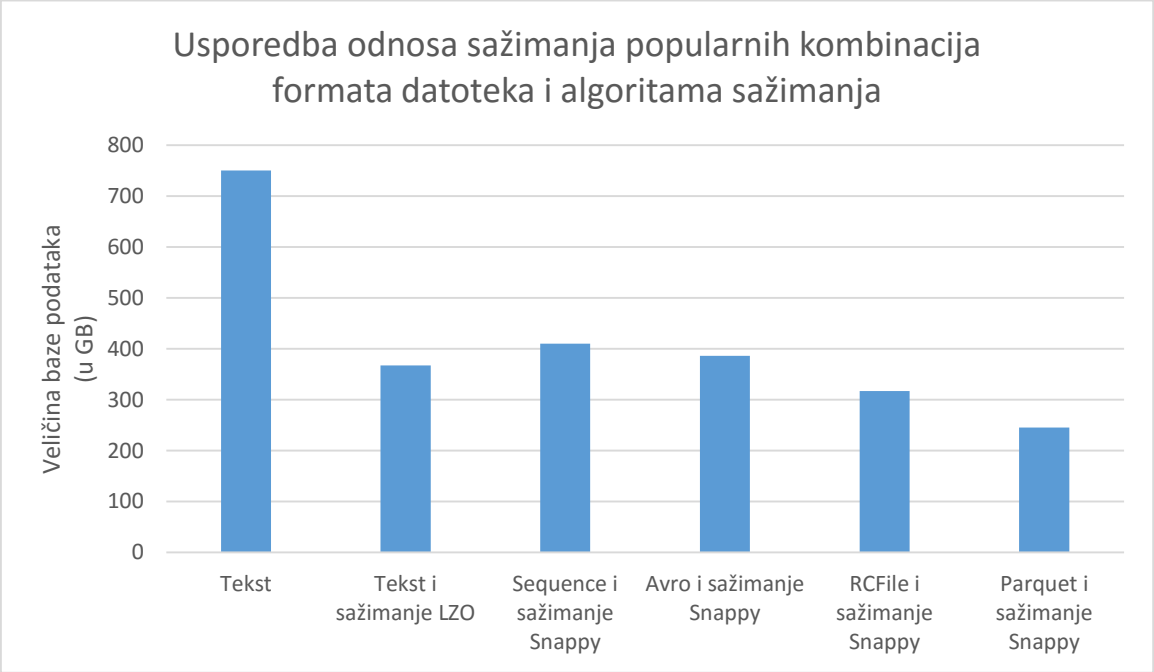
Graf 6. Prikaz utjecaja generacije run-time koda na performanse u Impali. Izvor: Kornacker, Behm i ostali (2015)

Iz grafa 6. može se iščitati izuzetan utjecaj generacije runtime koda na performanse vršenja upita. Za provedbu testa koristio se grozd sačinjen od deset čvorova. Svaki čvor imao je CPU sa 8 jezgri, 48 GB radne memorije i 12 diskova. Korištena je Avro TPC-H baza podataka, te su se vršili jednostavni agregacijski upiti. Generacijom koda izvršenje se ubrzalo do 5.7 puta, a razlika se povećavala sa kompleksnošću upita.

Short-circuit local reads je mogućnost Impale čiji je cilj skeniranje podataka iz diska i memorije velikom brzinom. Na taj se način izbjegava DataNode protokol. Impala je sposobna čitati gotovo sa istom propusnošću kao i disk, što otprilike iznosi 100 MB/sek po disku. Sa 12 diskova Impala je postigla propusnost od 1.2 GB/sek. Dodatno, HDFS keširanje omogućava Impali pristup podacima u memoriji brzinom memorije, a na taj način štedi CPU cikluse, te čini kopiranje i provjeravanje podatkovnih blokova nepotrebnim. I/O upravljačka komponenta upravlja čitanjem i pisanjem podataka sa pohrambenih uređaja u oba smjera. Izvršitelji se pridaju diskovima, a na način da

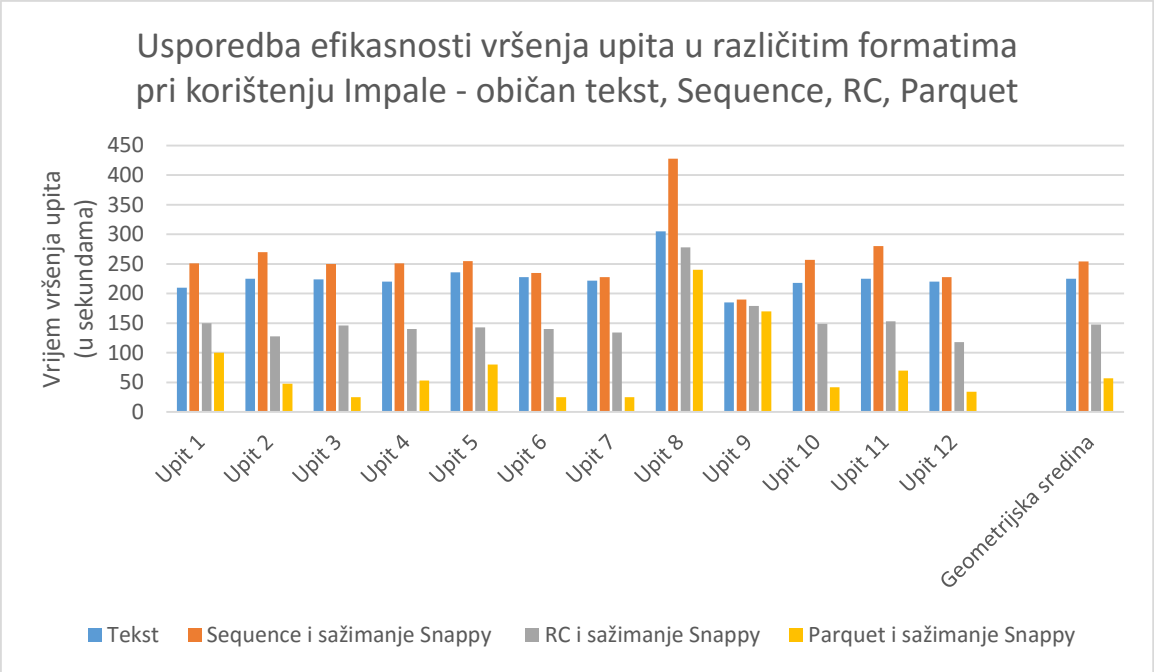
rotacijski disk dobiva jednog izvršitelja, a SSD[33] dobiva osam izvršitelja. (Kornacker, Behm i ostali 2015)

Isti autori nastavljaju tvrdeći da Impala podržava većinu popularnih formata datoteka: Avro, RC, Sequence, običan tekst, i Parquet. Najpreporučljiviji format za upotrebu jest Apache Parquet, jer nudi visoku kompresiju i efikasnost skeniranja. Formati se mogu sažimati koristeći razne algoritme, poput snappy, bz2, gzip. Dodatno, većina računalnih platformi koje se baziraju na Hadoop-u podržava procesiranje Parquet formata.[34] Parquet je optimiziran za velike blokove podataka, a podržava ugniježđene podatke. Ugniježđena polja pohranjuju se u stupcima, te se minimalno izmjenjuju zbog mogućnosti povrata ugniježđene strukture prilikom skeniranja.



Graf 7. Usporedba odnosa sažimanja popularnih kombinacija formata datoteka i algoritama sažimanja. Izvor: Kornacker, Behm i ostali (2015)

[33] Engl. Solid-State Drive, SSD.
[34] Hive, Pig, MapReduce i ostali.



Graf 8. Prikaz usporedbe efikasnosti vršenja upita u različitim formatima pri korištenju Impale - običan tekst, Sequence, RC, Parquet. Izvor: Kornacker, Behm i ostali (2015)

Graf 8. uspoređuje veličinu Lineitem tablice iz TPC-H testne baze podataka kada je spremljena u različitim kombinacijama formata i algoritama sažimanja. Najbolju kompresiju postiže Parquet sa snappy sažimanjem. Graf 8. prikazuje Impalina vremena izvršenja za različite upite iz TPC-DS testa. Baza je pritom spremljena različito – u obliku običnog teksta, Sequence, RC i Parquet formatima. Može se iščitati da Parquet postiže bolje rezultate u svim slučajevima, a u nekima ima i do 5 puta bolje rezultate.

Svaka platforma koja funkcionira poput grozda mora imati efikasno upravljanje resursima. Izazov je koordiniranje rasporeda resursa između različitih upita i platforma, a bez žrtvovanja brzine odgovora i propusnosti. YARN upravlja rasporedom resursa na Hadoop platformi bez potrebe za particioniranjem grozda. Impala je dizajnirana za rad sa tisućama upita u sekundi, a u čemu je zahtjevi za resursima i vrijeme odaziva usporavaju. Za potrebe ubrzanja navedenih aktivnosti

može se koristiti Llama[35], servis koji obuhvaća keširanje resursa, grupni raspored i djelomične promjene rasporeda resursa. Llama je samostalni daemon koji prima zahtjeve za resursima temeljene po upitu, a koji dolaze od Impalinih daemona. Dodatno, Llama prenosi odluke raspoređivanja na YARN, ukoliko zahtjevi za resursima nisu u Llama kešu. Svaki zahtjev za resursima povezan je sa bazenom resursa, koji odlučuje o optimalnom rasporedu resursa na upite sa potrebom. Ukoliko se resursi nalaze u kešu, Llama ih šalje odmah na korištenje. Na taj način Llama može izbjeći YARN-ov algoritam u prikladnim situacijama, gdje je brza alokacija moguća. Impala, za razliku od YARN-a, zahtjeva dostupnost svih resursa istovremeno, a tako da bi se upiti mogli vršiti paralelno. (Kornacker, Behm i ostali 2015)

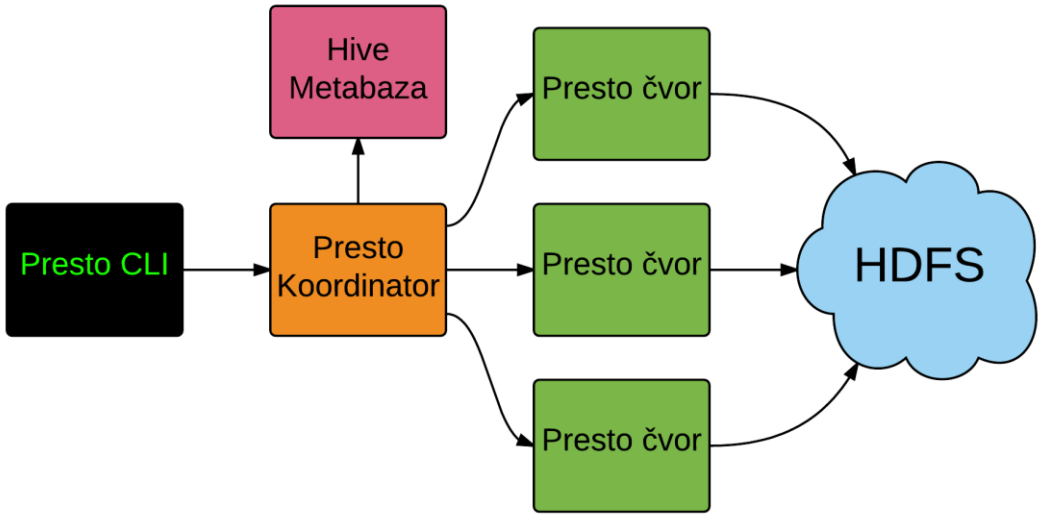
Kada su setovi podataka izuzetno veliki, nerijetko su procjene nužnih resursa netočne. Zbog toga je Impalinim upitima dopušteno procjenjivanje konzumacije resursa u toku vršenja upita. YARN ne podržava takav pristup, a zbog čega je Llama primorana vršiti nove zahtjeve za resursima prema YARN-u. Dodatno, bazeni resursa imaju limit maksimalnih zahtjeva za resursima. Zahtjevi se procjenjuju, stavljaju u niz, ili odbijaju temeljem definirane politike limita. Pristup bazenima može se kontrolirati korištenjem ACL-a, a bazenima se može upravljati kroz jednostavno sučelje Cloudera Managera. Trenutna integracija sa YARN-om nije idealna, a YARN-ov fokus na registar pojedinačne rezervacije nije prikladan za poslove koji zahtijevaju brze odgovore i visoku propusnost. (Kornacker, Behm i ostali 2015)

4.4 Presto

Presto je interaktivni SQL upitni modul za Hadoop otvorenog koda, a napisan u Javi. Sagradio ga je Facebook, a koji su izvorni kreatori Hive-a. Presto je sličan pristupom

[35] Engl. Low-Latency Application Master, LLAMA.

Impali, pružajući interaktivno iskustvo koristeći već postojeće setove podataka koji su pohranjeni u Hadoop-u. Poput Impale, zahtjeva instalaciju na mnogim čvorovima. (Rathbone 2014) Puna instalacija uključuje koordinator i višestruke čvorove. Koordinator raščlanjuje, analizira i planira izvršenje upita, koje je klijent, poput Presto CLI, poslao. Konačno, koordinator distribuira procesiranje na čvorove. CLI se ponaša kao normalna UNIX izvršiva datoteka. CLI se može pokrenuti sa --help opcijom da bi se vidjele sve dostupne opcije. (Presto 2015)



Slika 6. Prikaz radnog toka u Prestu. Klijent šalje naredbe putem sučelja, koordinator raščlanjuje, analizira, planira i distribuira izvršenje upita na čvorove. Analitički rezultat rada čvorova u konačnici se pohranjuje. (Presto 2015)

Isti izvor tvrdi da Presto omogućuje vršenje upita nad podacima koji se nalaze na različitim lokacijama, a koje mogu biti Hive, Cassandra, relacijska baza podataka ili kupljeno skladište podataka. Jedan Presto upit može kombinirati podatke iz različitih izvora, što omogućuje vršenje analitike diljem organizacije. Presto podržava čitanje Hive podataka iz sljedećih verzija Hadoop-a: Apache Hadoop 1.x, Apache Hadoop 2.x, Cloudera CDH 4, i Cloudera CDH 5. Formati koje podržava jesu Text,

SequenceFile, RCFile, ORC. Metabaza mora biti udaljena, a s obzirom da Presto ne koristi MapReduce, on zahtjeva samo HDFS. Konektor za Cassandru potpuno je samostalan od Hive konektora, te zahtjeva samo Cassandra 2.x instalaciju. TPC-H konektor dinamički stvara podatke koji se mogu koristiti za testiranje i vršenje eksperimenata. TPC-H nema vanjskih zahtjeva. Presto cilja na analitičare koji očekuju vremena odaziva koja nisu duža od nekoliko minuta. Presto tvrdi da pozicioniran između skupih rješenja, i sporih besplatnih rješenja koja zahtijevaju moćan hardver. Facebook koristi Presto za vršenje interaktivnih upita diljem internih skladišta podataka, uključujući 300+ PB skladište. Preko 1000 zaposlenika svakoga dana koristi Presto za vršenje više od 30000 upita, skeniranjem više od petabajt podataka svakoga dana. Presto ima nekoliko osnovnih zahtjeva: a) Linux ili Mac OS X, b) Java 8, 64-bit, c) Python 2.4 ili novija verzija.

Prema prethodnom izvoru, Presto razumije SQL, međutim, on nije zamjena za bazu podataka opće namjene poput MySQL, PostgreSQL ili Oracle. Nije namijenjen za podršku OLTP-a[36]. Presto je dizajniran da bude alternativa alatima kao što su Hive ili Pig, ali nije ograničen samo na HDFS. Presto je dizajniran za vršenje analize velike količine podataka, agregiranje i iznošenje izvještaja. Presto Verifier može se koristiti za testiranje Presta sa nekom drugom bazom podataka. Također, moguće je testirati dva Presto grozda međusobno. Presto Verifier je često korišten za testiranje novih verzija sa trenutno instaliranim, a sa ciljem verifikacije kompatibilnosti i mogućih problema nakon nadogradnje. Dodatno, na raspolaganju je i Benchmark Driver, odnosno upravljač za testiranje. Pomoću njega mjere se performanse vršenja upita u Presto grozdu. SQL datoteke nalaze se u sql direktoriju, te je nužno da imaju .sql ekstenziju. Ime upita je ime datoteke koja nema ekstenziju. Upravljač za testiranje mjeri CPU vrijeme upotrijebljeno od strane Presto procesa, te CPU vrijeme upotrijebljeno od strane upita. Dodatno, upravljač ukazuje na medijan, sredinu i standardnu devijaciju vršenja upita. Izlaz upravljača za testiranje formatiran je tako da bude čitljiv i

[36] Engl. Online Transaction Processing, OLTP.

razumljiv, te u njemu mogu biti dodani novi stupci vađenjem vrijednosti iz imena tablica ili SQL datoteka. Dodatni stupci mogu se dodati i ručno, označavanjem SQL datoteka.

Presto (2015) navodi da su pravila za vršenje redova upita definirana u JSON datoteci. Pravila se odnose na broj upita koje Presto može primiti, i na broj redova upita. Nekoliko pravila definira koji upiti idu u koji red. Prvo pravili čini bob-a adminom. Drugo pravilo kaže da svi upiti koji sadrže pipeline prvo trebaju stati u korisnikov red, a potom u pipeline red. Posljednje pravilo stavlja sve upite u osobni red. Sva navedena pravila kazuju da je bob admin, a svi ostali korisnici imaju limite. Korisnici smiju imati do 5 upita istovremeno aktivno. Ne može se pokretati više od 10 pipeline upita istovremeno, te se ne može više od 100 ostalih upita vršiti u momentu.

Presto, prema informacijama sa službene stranice, podržava sljedeće konektore: Black Hole Konektor, Cassandra Konektor, Hive Konektor, JMX[37] Konektor, Kafka Konektor, MySQL Konektor, PostgreSQL Konektor, System Konektor i TPCH Konektor. Konektori su opisani u nastavku. Black Hole konektor funkcionira poput /dev/null uređaja na Unix sistemima. Metapodaci se zadržavaju u memoriji na koordinatoru. Ne funkcionira dobro uz druge koordinatore zbog različitih metapodataka. Konektor se podešava stvaranjem kataloške datoteke svojstava etc/catalog/blackhole.properties sa sljedećim sadržajem: connector.name=blackhole. Cassandra konektor dopušta vršenje upita nad podacima spremljenima u Cassandri. Cassandra se podešava stvaranjem kataloške datoteke svojstava etc/catalog/cassandra.properties sa sljedećim sadržajem: connector.name=cassandra, cassandra.contact-points=host1,host2. Hostove treba zamijeniti listom Cassandrinih čvorova koji se koriste za otkrivanje topologije grozda. Hostovi se razdvajaju zarezima. Ukoliko čvorovi ne koriste početni port 9042, bit će potrebno podesiti i cassandra.native-protocol-port. Hive konektor dopušta vršenje upita nad podacima spremljenim u Hive skladištu podataka. Presto koristi dvije komponente Hive-a:

[37] Engl. Java Management Extensions, JMX.

podatke i metapodatke (HiveQL ne koristi). Presto ima Hive konektore za više verzija Hadoop-a, preciznije: hive-hadoop1 za Apache Hadoop 1.x, hive-hadoop2 za Apache Hadoop 2.x, hive-cdh4 za Cloudera CDH 4 i hive-cdh5 za Cloudera CDH 5. Hive se podešava stvaranjem kataloške datoteke svojstava etc/catalog/hive.properties sa sljedećim sadržajem: connector.name=hive-cdh4, hive.metastore.uri=thrift://example.net:9083.[38] U većini slučajeva se HDFS klijent konfigurira automatski. JMX konektor pruža korisne mogućnosti za nadzor, pruža informacije o Java Virtualnoj Mašini i svim pripadnim softverima. JMX se podešava stvaranjem kataloške datoteke svojstava etc/catalog/jmx.properties sa sljedećim sadržajem: connector.name=jmx. Kafka konektor pruža mogućnost korištenja Apache Kafka predmeta u obliku tablica u Prestu, a gdje je svaka poruka redak. Kafka konektor se podešava stvaranjem kataloške datoteke svojstava etc/catalog/kafka.properties sa sljedećim sadržajem koji se izmjenjuje po potrebi: connector.name=kafka, kafka.table names=table1,table2, kafka.nodes=host1:port,host2:port. MySQL konektor dopušta vršenje upita i kreiranje tablica u vanjskoj MySQL bazi podataka. Može se koristiti za spajanje s podacima iz raznih izvora. MySQL konektor se podešava stvaranjem kataloške datoteke svojstava etc/catalog/mysql.properties sa sljedećim sadržajem: connector.name=mysql, connection-url=jdbc:mysql://example.net:3306, connection-user=root, connection-password=secret. PostgreSQL konektor dopušta vršenje upita i stvaranje tablica u vanjskoj PostgreSQL bazi podataka. Način konfiguriranja konektora sličan je konfiguriranju MySQL konektora. System konektor pruža informacije i mjerenja o trenutnom Presto grozdu. Informacije su dostupne korištenjem normalnih SQL upita. Katalog za njega već postoji, a zove se system. Primjerice, prikazivanje sadržaja tablice nodes može se izvršiti putem SELECT * FROM system.runtime.nodes. TPCH konektor pruža uvjete za podršku TPC Benchmark-a H. On se koristi za mjerenje performansi kompleksnih baza podataka

[38] Primjer za slučaj spajanja hive-cdh4 konektora. Example.net se mijenja u pravi host, dok se port mijenja u pravi port koji je u upotrebi. Sa uspješno podešenom izjavom spoj će se izvršiti na Hive metabazu, odnosno na Thrift servis.

koje služe kao potpora upravljanju i odlučivanju. Dodatno, s njime se može testirati Presto; naime, TPC-H može kreirati podatke uz pomoć algoritma, a za potrebe testa. Za njegovu konfiguraciju kreira se kataloška datoteka svojstava etc/catalog/tpch.properties sa sljedećim sadržajem: connector.name=tpch.

Presto pruža: (Rathbone 2014)

- Podršku za ANSI SQL sintakse (najvjerojatnije ANSI-92),
- JDBC upravljače,
- Set konektora koji se koriste za čitanje podataka iz već postojećih izvora podataka. Konektori uključuju HDFS, Cassandru i Hive,
- Međudjelovanje sa Hive Metabazom za dijeljenje schema između platforma.

Isti autor tvrdi da Presto ima iste ciljeve kao Impala. Za razliku od Impale nema podršku od najvećih prodavača, što znači da nedostaje „enterprise" podrška za instalaciju. Presto koriste mnoge velike tehnološke kompanije, što indicira na mogućnost postojanja Presto zajednice. Performase ovise o formatu podataka [RCFile[39] (engl. Record Columnar File)], kao i kod Impale. Presto se preporučuje stručnjacima koji ga znaju održavati, ali i razvijati jer je upitno koliko će vremena Facebook podržavati open source verziju.

4.5 Spark SQL

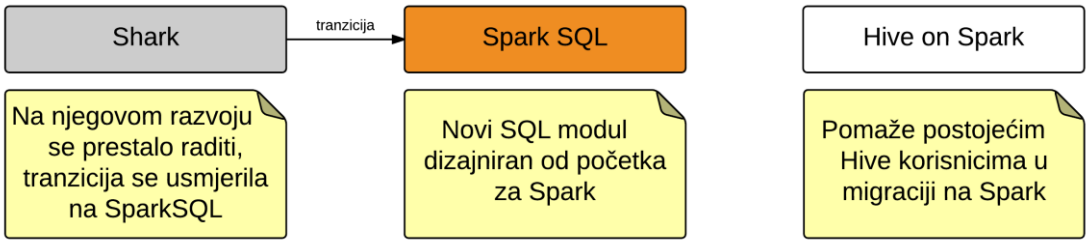
Shark je otvorena SQL platforma za vršenje upita napisana u Scali, na UC Berkeley-u. Kao prethodnici, napisana je da bi nadopunila postojeću Hive instalaciju, i vršila upite na vlastitom setu radnih čvorova, a bez korištenja MapReduce-a. Za razliku od Impale

[39] Struktura za postavljanje podataka koja procjenjuje kako pohraniti relacijske tablice na grozdovima računala. Dizajnirana je za sisteme koji koriste MapReduce platformu. Kombinacija je više komponenata: format za pohranu podataka, pristup za kompresiju podataka, i optimizacijske tehnike za čitanje podataka. Izvor: Prokopp (2014)

i Presta, Shark je izgrađen nad postojećim modulom za procesiranje podataka koji se zove Apache Spark. Shark pokušava podržati sve Hive funkcionalnosti, istovremeno nudeći velika poboljšanja u performansama. Shark ne nudi iste performanse kao Impala, ali ako je infrastruktura već pripremljena za korištenje Sparka, odabir neće biti pogreška. Naprotiv, Spark je podržan od strane mnogih velikih prodavača, a na testovima ima neznatno lošije performanse od Impale. (Rathbone 2014)

Spark je trenutno popularan, a brža je alternativa MapReduce-u. Shark pruža: (Rathbone 2014)

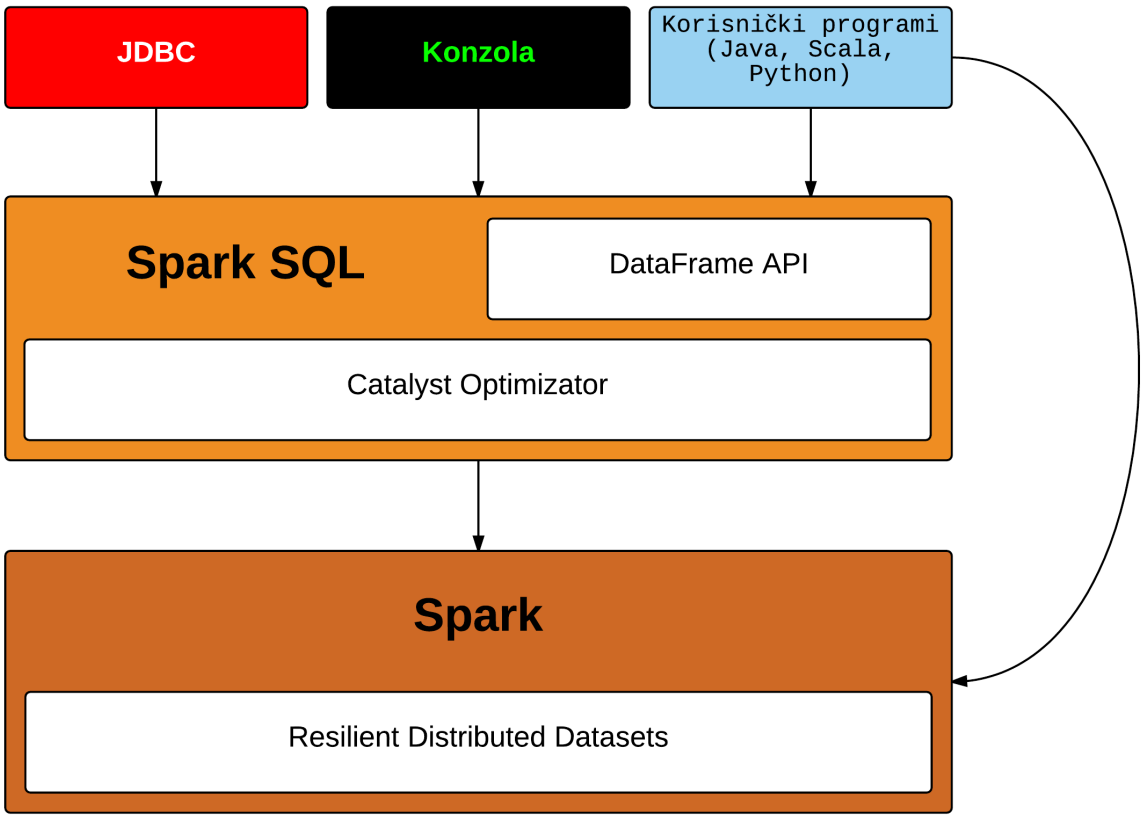
- Jezik za vršenje upita sličan SQL-u, sa podrškom za većinu Hive-QL,
- Klijent za naredbeni redak (Hive klijent),
- Međudjelovanje sa Hive Metabazom za dijeljenje schema između platforma,
- Podršku za postojeće Hive ekstenzije, poput UDF-a i SerDes-a.



Slika. 7. Prikaz migracije Sharka prema Sparku. Hive on Spark pomaže postojećim korisnicima Hive pri migraciji. Izvor: Xin (2014)

Spark SQL pruža jednostavnu nadogradnju sa Sharka 0.9, te integraciju sa generalnim Spark programima. Spark je sposoban procesirati podatke u HDFS-u, HBase-u, Cassandri i bilo koji Hadoop InputFormat. Dizajniran je za vršenje grupnog procesiranja poput MapReduce-a, tokova, interaktivnih upita i strojnog učenja. Mnoge organizacije pokreću Spark na tisućama čvorova, dok je najveći poznati grozd

sastavljen od oko 8000 čvorova. Podaci ne moraju nužno stati u memoriju, jer je Spark sposoban prenijeti višak podataka na disk, ukoliko ne stane u memoriju. Isti prijenos vrijedi i za keširane setove podataka. Spark se može pokretati na grozdu korištenjem samostalnog moda implementacije, a koji zahtjeva samo Javu na svakome čvoru, ili na Mesos i YARN upravljačima grozdova. Dodatno, Spark se može pokretati i na Amazon EC2, korištenjem EC2 sktripti za automatsku aktivaciju grozda. (Spark 2015)



Slika 8. Prikaz sučelja prema Spark SQL-u, i interakcije sa Spark-om. Spark SQL izvršava se kao biblioteka nad Spark-om. Pruža SQL sučelje, kojem se može pristupiti putem JDBC/ODBC i naredbene konzole, ali i putem DataFrame API koji je integriran u Spark-ove podržane programske jezike. Izvor: Armbrust, Xin i ostali (2015)

Armbrust, Xin i ostali (2015) tvrde da je Spark SQL izdan u svibnju 2014. godine, te je sada jedna od najrazvijenijih komponenata u Sparku. Spark SQL čini Spark dostupnim većem broju korisnika i poboljšava optimizaciju za postojeće korisnike. Spark SQL korisnicima SQL-a pruža pozivanje kompleksnih analitičkih biblioteka u Sparku, poput strojnog učenja. Putem DataFrame API-a pruža puno jaču integraciju između relacijskog i proceduralnog procesiranja, a koje može biti izvršeno na vanjskim izvorima podataka, ali i na internim distribuiranim kolekcijama. Dodatno, sadrži Catalyst – proširiv optimizator napisan u Scali. Korištenjem optimizatora mogućnosti se mogu prilagoditi potrebama moderne analize podataka. Dodatno, putem njega mogu se dodavati izvori podataka i optimizacijska pravila.

Prethodni autori tvrde da Spark pruža API za manipulaciju sa RDD[40]. RDD podržava operacije poput map, filter i reduce. Problemi koje je, sada diskontinuiran, Shark imao su sljedeći: a) Shark je mogao vršiti upite samo nad vanjski pohranjenim podacima u Hive katalogu, b) jedini način za poziv Sharka od Spark programa bio je sastavljanje SQL stringa, i c) Hive optimizator bio je prilagođen MapReduce-u i teško proširiv. Iz iskustva sa Shark-om, postavljeni su novi ciljevi za Spark SQL. Spark SQL je zamišljen tako da podržava relacijsko procesiranje unutar Spark programa – na nativnim RDD-ima, i na vanjskim izvorima podataka korištenjem API-a koji će biti jednostavan programerima. Spark SQL treba pružiti visoke performanse korištenjem postojećih DBMS tehnika, treba podržavati nove izvore podataka, uključujući polu-strukturirane podatke i vanjske baze podataka. Konačno, zamišljen je da omogući proširenja sa naprednim analitičkim algoritmima, kao što su graf-procesiranje i strojno učenje. Graf-procesiranje čini relacije između objekata i čvorova kristalno jasnima.

Prethodni autori definiraju DataFrame kao distribuiranu kolekciju redaka unutar iste šeme. On je ekvivalentan tablici u relacijskoj bazi podataka. DataFrame-ovi nadziru svoju šemu, te podržavaju relacijske operacije koje vode optimiziranom izvršenju.

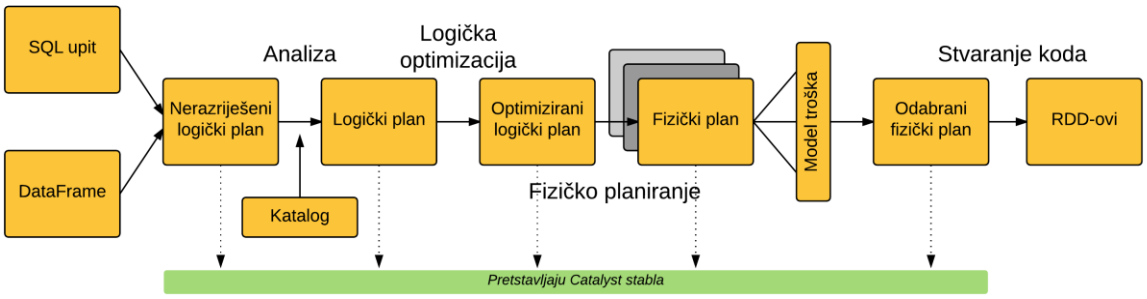
[40] Engl. Resilient Distributed Datasets, RDD. Svaki RDD jest kolekcija Java ili Python objekata, a koji su particionirani diljem grozda.

Data-Frame može se izgraditi od tablica iz sistemskog kataloga ili iz postojećih RDD-ova. DataFrame-ovi su „lijeni“, što znači da svaki DataFrame objekt predstavlja logički plan računanja seta podataka, ali se ne izvršava dok korisnik ne pozove output operaciju. Spark SQL podržava većinu SQL i korisnički kreiranih tipova podataka. Tipovi podataka mogu se ugnijezditi pri potrebama za većom kompleksnošću. Osim toga, Spark SQL podržava keširanje podatak u memoriji, koje je izuzetno korisno pri vršenju interaktivnih upita i strojnog učenja. Korisnički definirane funkcije mogu se koristiti i od strane alata za poslovnu inteligenciju.

Catalyst sadrži opće biblioteke za prikaz stabala i korištenje pravila za manipulaciju sa stablima. Nad Catalyst-om nalaze se biblioteke specifične procesiranju relacijskih upita, i set pravila vezanih uz analitiku, logičku optimizaciju, fizičko planiranje i stvaranje koda. Glavni tip podataka u Catalyst-u je stablo koje se sastoji od čvorova (objekata). Svaki čvor ima tip čvora i može imati djecu. Novi čvorovi su potklase TreeNode klase. Stablima se može upravljati putem pravila, a koja su u suštini funkcije između različitih stabala. Dodatno, pravila se nekad trebaju izvršiti više puta da bi se transformiralo kompletno stablo. Catalyst grupira pravila u skupine, te izvršava skupine dok stablo ne dosegne stadij kada se više ne izmjenjuje. (Armbrust, Xin i ostali 2015)

Prema tvrdnjama istih autora, Catalyst transformira stablo u četiri faze. U prvoj se fazi analizira logički plan da bi se razriješile reference; u drugoj se fazi vrši optimizacija logičkog plana; u trećoj se fazi vrši fizičko planiranje; i u četvrtoj fazi se stvara kod, a za potrebe kompilacije dijelova upita u Java kod. U fazi fizičkog planiranja mogu se generirati višestruki planovi, a koji se potom uspoređuju temeljem cijene. Svaka faza koristi različite stablene čvorove. Atribut je nerazriješen ukoliko nije poznat njegov tip, ili ako nije spojen sa ulaznom tablicom (ili aliasom). Catalyst pravila i Catalog objekti prate tablice u svim izvorima s ciljem razrješenja atributa. Prvo se gradi nerazriješen logički plan sa nepovezanim atributima i tipovima podataka. Nakon toga primjenjuju se sljedeća pravila: a) Traženje relacija po imenu iz kataloga, b) Mapiranje imenovanih

atributa na dati input, ukoliko su dostupna operatorova djeca, c) Procjenjivanje koji se atribut odnosi na istu vrijednost, a tako da bi im se dodijelio jedinstveni ID, d) Propagiranje i prisiljavanje tipova kroz ekspresije – sve ekspresije moraju biti razriješene, poznate i prisiljene u kompatibilne tipove.



Slika 9. Prikaz faza planiranja upita u Spark SQL-u. Catalyst stabla su obilježena. Izvor: Armbrust, Xin i ostali (2015)

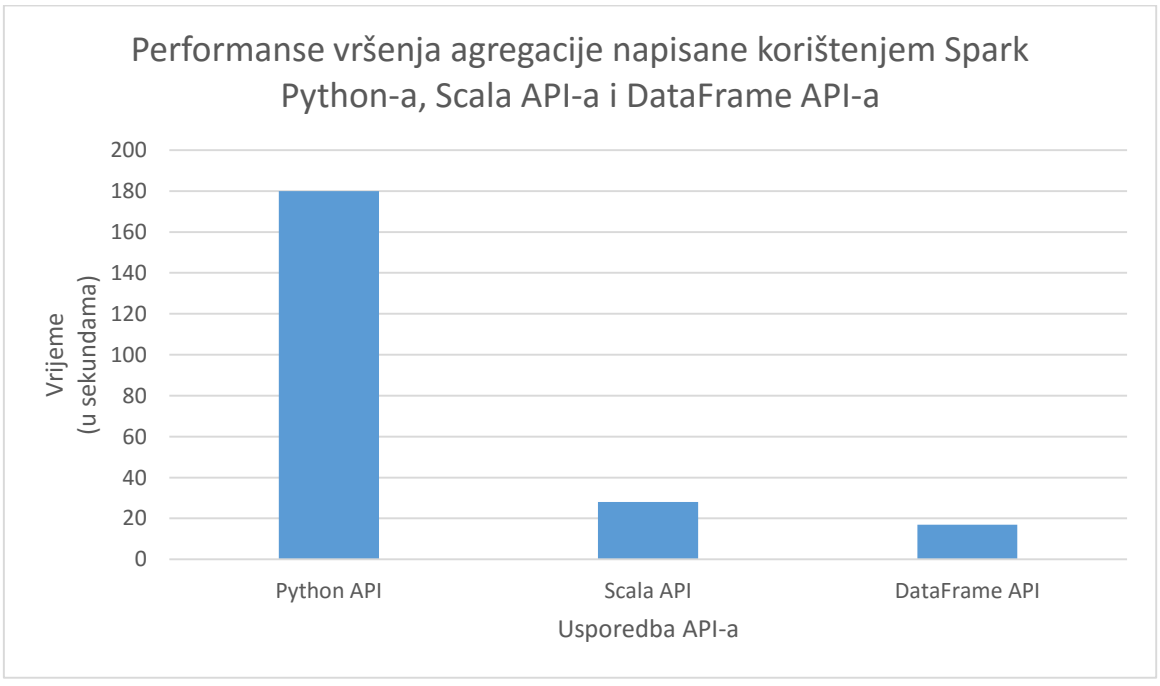
Po Armbrust-u, Xin-u i ostalima (2015) logička optimizacija podrazumijeva konstantno sažimanje, rezanje projekcije, null propagacije, pojednostavljivanje Boolean izraza i ostala pravila. U fazi fizičkog planiranja Spark SQL uzima logički plan i stvara jedan ili više fizičkih planova, korištenjem fizičkih operatora koji odgovaraju Spark izvršnom modulu. Nakon toga, odabire plan temeljem modela troška. Optimizacija bazirana na trošku tada biranju algoritama spajanja. Broadcast join se odabire kod malih relacija, korištenjem p2p[41] broadcast mogućnosti. Pravila za fizičko planiranje su oko 500 linija koda. U posljednoj fazi stvara se Java kod koji će se pokretati na svakome stroju.

Autori nastavljaju tvrdeći da se novi izvori podataka mogu definirati korištenjem API-a. Svaki izvor mora imati implementiranu createRelation funkciju. Takva funkcija

[41] Engl. Peer to Peer, p2p.

uzima set ključ-vrijednost parametara, a vraća BaseRelation objekt za tu relaciju. Izvori podataka su visoko izmjenjivi (od strane korisnika), a s lakoćom se dodaju razni tipovi i izvori podataka od kojih su neki: Avro, Parquet, JDBC i ostali. Da bi se paketi koristili, potrebno je specificirati njihovo ime u SQL izjavi, prenoseći ključ-vrijednost parove za konfiguracijske opcije. Primjerice, Avro izvor podataka može imati put do: CREATE TEMPORARY TABLE messages USING com.databricks.spark.avro OPTIONS (path "messages.avro").

Spark SQL sadrži algoritam za zaključivanje šema, za JSON i ostale polustrukturirane podatke. SparkSQL dio je API-a za Sparkovu biblioteku strojnog učenja. Spark SQL podržava federaciju upita, a koja omogućava jednom programu efikasno vršenje upita diljem različitih izvora. Sve se mogućnosti grade nad Catalyst platformom. Za prikaz setova podataka API koristi DataFrame, dok se pipeline koristi za prikaz transformacija nad podacima u obliku grafa, poput – ekstrakcija mogućnosti, normalizacije, smanjenje dimenzija, treniranje modela i ostalo. (Armbrust, Xin i ostali 2015)



Slika 10. Prikaz performansi vršenja agregacije napisane korištenjem Spark Python-a, Scala API-a i DataFrame API-a. DataFrame verzija koda brža je za 12 puta od one u Python-u. Dodatno, DataFrame verzija brža je od one u Scali više od 2 puta. Primarni razlog je generacija koda koja u DataFrame-u izbjegava skupe alokacije ključ-vrijednost parova, a što je slučaj kod Scale. Izvor: Armbrust, Xin i ostali (2015)

4.6 Apache Drill

Apache Drill je otvorena interaktivna SQL platforma za vršenje upita uz Hadoop. Podržavao ju je MapR, iako sada podržavaju i Impalu. Drill ima ciljeve poput Impale i Presta; brze interaktivne upite nad velikim setovima podataka, a poput navedenih zahtjeva instalaciju radnih čvorova[42]. Za razliku od Impale i Presta, pokušava podržati različite baze, poput HDFS, HBase i MongoDB. Fokusira se na kompleksne ugniježđene setove podataka (poput JSON). Drill pruža: (Rathbone 2014)

- SQL sa podrškom za ANSI,
- Međudjelovanje sa raznim bazama podataka i metapodataka, te datotečnim sustavima - Hive, HBase, MongoDB, MapR-DB, HDFS, MapR-FS, Amazon S3, Azure Blob Storage, Google Cloud Storage, Swift, NAS i lokalnim datotekama,
- Brze SQL upite,
- Dinamične upite na samo-deskriptivnim podacima u datotekama, a kao što su JSON, Parquet, tekst. Također, dinamični upiti podržani su i u MapR-DB/HBase tablicama,
- Podršku za ugniježđene podatke,
- Integraciju sa BI/SQL alatima korištenjem standardnih JDBC/ODBC upravljača,
- Platformu za ekstenzije za UDF-ove, te pohrambene dodatke i ostalo.

[42] Engl. Drillbits.

Apache Drill trenutno nije prikladan za produkcijsku upotrebu zbog svoje rane faze i stanja u razvoju, ali je projekt na koji definitivno treba obratiti pozornost, i pratiti njegov napredak. Apache Drill 1.1 nudi SQL window funkcije, particioniranje podataka korištenjem PARTITION BY izjave, delegiranje Hive impersonacija, te podršku za UNION i UNION ALL izjave, te bolje optimizirane planove koji uključuju UNION. Delegiranje Hive impersonacija odnosi se na pružanje autorizacije za pristup Hive metapodacima, te podacima u Hive skladištu podataka. Jedan upit može spojiti podatke iz više izvora. Optimizator automatski restrukturira plan izvršenja upita tako da bi se prilagodio mogućnostima procesiranja. Korisnik je u mogućnosti vršiti upite nad podacima bez potrebe za unošenjem podataka i stvaranjem tablica, ili transformacijom podataka prije procesiranja. Takav pristup se izvršava definiranjem puta prema direktoriju, kolekciji iz baze podataka i ostalome. (Apache Drill 2015)

Dodatno, temeljem istog izvora, JSON model podataka omogućuje vršenje upita nad složenim i ugniježđenim podacima, i nad novim strukturama. Razvojni tim Drill-a tvrdi da je on jedini stupčani upitni modul koji podržava kompleksne podatke. Stupčana reprezentacija kompleksnih podataka u memoriji omogućava postizanje stupčanih brzina, a sa fleksibilnošću JSON modela. SQL pruža poslovnim korisnicima, analitičarima i ostalim korisnicima korištenje BI alata za interakciju sa ne-relacijskim skladištima podataka, a korištenjem već prethodno spomenutih JDBC i ODBC upravljača. REST API pruža stvaranje vizualizacija u ostalim aplikacijama. Virtualni setovi podataka dopuštaju mapiranje najsloženijih oblika podataka u strukture koje su pristupačne BI korisnicima i analitičarima. Konačno, Drill pruža stupčani izvršni modul s podrškom za kompleksne podatke; kompilacije i rekompilacije vođene podacima u vremenu izvršenja; specijalizirano upravljanje memorijom koje eliminira kolekcije smeća; izvršenja svjesna raspodjele, a zbog čega se smanjuje mrežni promet; i optimizator koji preferira vršenje procesiranja u skladištu podataka.

4.7 Hawk

Hawk je closed-source proizvod tvrtke EMC Pivotal, a nudi se kao dio njihove Pivotal HD distribucije Hadoop-a. Oni tvrde da je Hawk „najbrži svjetski SQL modul za Hadoop“, te da je u izgradnji deset godina. Hawk pruža sljedeće mogućnosti: (Rathbone 2014)

- Podrška za potpune SQL sintakse, interaktivni i brzi upiti nad velikom količinom podataka,
- Međudjelovanje sa Hive, Hbase, EDW, ADW HDFS i Hive kroz Pivotal eXtension Framework (PXF),
- Međudjelovanje sa Pivotal-ovim GemFire XD, a koji je njihova „real-time “ baza podataka u memoriji, podržana sa HDFS,
- Implementacija OLAP-a.

Hawk ima smisla koristiti ako se koristi i Pivotal Hadoop distribucija, ili Hortonworks podatkovna platforma. Detaljnije, Hawk podržava istovremeno vršenje upita, analitiku i strojno učenje, te je izgrađen sa otpornošću na zakazanja za korištenje u poslovanju. Podržava razne formate – tekst, Avro, Nativne, HDFS i Parquet; a sa ugrađenom kompresijom. Mogućnosti implementacije obuhvaćaju vlastiti hardver, te virtualizaciju, odnosno IaaS[43]. (Pivotal HAWK 2015)

4.8 Ostale platforme

Ostale platforme koje su u manjoj upotrebi, te koje imaju slabije performanse od prethodno navedenih slijede u nastavku poglavlja 4.8.

[43] Engl. Infrastructure as a Service, IaaS.

4.8.1 BigSQL

IBM ima vlastitu Hadoop distribuciju po imenu Big Insights, a BigSQL se nudi kao dio navedene distribucije. BigSQL vrši upite nad podacima spremljenim u HDFS-u koristeći i MapReduce i njihove zatvorene tehnologije koje omogućuju brži povrat rezultata. BigSQL pruža: (Rathbone 2014)

- JDBC i ODBC upravljače,
- Široku SQL podršku,
- Klijent za naredbeni redak.

BigSQL će najviše odgovarati postojećim IBM-ovim kupcima.

4.8.2 Apache Phoenix

Apache Phoenix je SQL modul otvorenog koda, a napravljen za Apache HBase. Kao cilj navedeni su brzi rezultati za upite nad podacima pohranjenim u HBase putem uvrštenog JDBC upravljača. Za razliku od ostalih spomenutih modula, Phoenix nudi operacije čitanja i pisanja nad HBase podacima. Phoenix pruža: (Rathbone 2014)

- JDBC upravljač,
- Klijent za naredbeni redak,
- Alate za skupno unošenje podataka,
- Sposobnost kreiranja novih tablica, ili mapiranja na postojeće HBase podatke.

Ukoliko organizacija već koristi HBase, bit će joj jednostavno početi sa korištenjem Phoenix-a. Iako Hive može čitati podatke iz HBase, Phoenix ih može i zapisivati. Phoenix se pokazuje kao dobar analitički alat u takvom slučaju korištenja. Misija

Phoenix-a je da postane standardni način pristupanja HBase-u kroz dobro definirani i industrijski standardizirani API. U testovima performansi nad HDFS i HBase, Phoenix koristeći ključni filter postiže znatno bolje performanse od Hive-a. U testu se vršio select count (1) upit iz tablice koja ima više od 10M i 100M redaka. Podaci su 5 stupaca. Korišteno je 4 regionalna servera, sa 6 jezgri @ 3.3GHz Xeon. Dodatno, test nad HBase je pokazao da Phoenix ima značajno bolje performanse od Impale. (Apache Phoenix 2015)

4.8.3 Apache Tajo

Apache Tajo jest projekt sa ciljem izgradnje sistema naprednog skladištenja podataka nad HDFS-om. Tajo je predstavljen kao veliko skladište podataka, ali nudi i brze rezultate upita kao i prethodni moduli. Podržava vanjske tablice i Hive setove podataka (putem HCatalog-a), fokusira se na upravljanje podacima, pružanje brzog pristupa podacima, i pristup alatima za tradicionalnije ETL[44] procese. Poput Impale, Presta i ostalih modula, zahtjeva instalaciju Tajo-specifičnih izvršiteljskih procesa na čvorovima. Tajo pruža: (Rathbone 2014)

- ANSI SQL kompatibilnost,
- JDBC upravljače,
- Integraciju sa Hive metabazom radi pristupa Hive setovima podataka,
- Klijent za naredbeni redak,
- API za korisničke funkcije.

Tariq (2013) tvrdi da Hive nikad nije bio izgrađen za procesiranje u memoriji u realnom vremenu, a temelji se na MapReduce-u. Naime, Hive je izgrađen za offline skupno procesiranje. Najprikladniji je za dugotrajne poslove poput vršenja „join-a“ na

[44] Engl. Extract, Transform and Load; ETL.

izuzetno velikim setovima podataka. S druge strane, Impala, Spark ili Drill dolaze kao logičan odabir za procesiranje u realnom vremenu. Ukoliko treba vršiti upite nad ne toliko velikim setovima podataka, koji stanu u memoriju, i ukoliko se upiti trebaju vršiti u realnom vremenu, Impala, Spark i Drill se nude kao dobar odabir. Iako je sa navedenim alatima moguće vršiti upite nad velikim količinama podataka, upiti nad petabajtima podataka u realnom vremenu dovode mogućnosti alata do samog limita. Iako je moguće pročitati da kompanije imaju petabajte podatka, te da korisnici uspješno vrše upite u realnom vremenu nad njima, u stvarnosti se upiti ne vrše nad svim podacima većinu vremena. U konačnici, bitno je planirati i znati kada što koristiti. Većina spomenutih alata kreirana je sa specifičnim ciljem. Impala je izgrađena tako da iskoristi već postojeću Hive infrastrukturu, a tako da ne bi bilo potrebno krenuti iz ničega. Koristi iste metapodatke koje koristi i Hive. Impala je izgrađena sa ciljem da podržava upite u realnom vremenu nad već postojećim Hadoop skladištem podataka. S druge strane, Drill je izgrađen kao „not only Hadoop“ projekt, a tako da korisniku omogući distribuirane mogućnosti upita nad mnogim platformama velike količine podataka, uključujući MongoDB, Cassandra, Riak i Splunk. Shark je kompatibilan sa Apache Hive-om, što znači da je moguće vršiti upite sa istim HiveQL izjavama kao što bi se moglo i u Hive-u. Konačno, Shark je sposoban vratiti rezultate i do 30 puta brže od istih upita izvršenih u Hive-u.

5. Zaključak

Upotrebom analitike velike količine podataka, uz opće performanse kompanije, mogu se poboljšati performanse većine organizacijskih jedinica, uključujući upravljanje i odlučivanje u kompaniji, a što zahtjeva nekoliko preduvjeta koji su obrađeni u radu. Prikazano je da se za efikasnu upotrebu velike količine podataka u organizaciji mora omogućiti izgradnja podatkovnih resursa iz različitih vanjskih i unutarnjih izvora. Nužno je podržati pohranu velike količine podataka, te omogućiti ekstrakciju vrijednih informacija prikladnim hardverom i softverom iz strukturiranih, polu-strukturiranih i nestrukturiranih podataka. Ukazano je na važnost upoznavanja tehnika pristupa informacijama i analitici podataka u cijeloj kompaniji; te na važnost zapošljavanja stručnih kadrova i poticanja znanja na području analitike velike količine podataka. Dodatno, ukazano je na nužnost slaganja IT-a i menadžmenta u donošenju odluka o budžetu za upravljanje i inovacije na području informacijskih resursa. Što je veća integracija informacijskog sustava kompanije, to je veća mogućnost upijanja i oblikovanja informacija i znanja koji potječu iz analitike velike količine podataka.

Istražene su i uspoređene različite analitičke platforme i pripadni alati, te je zaključeno je da je u vremenu pisanja rada Impala, temeljem testiranih performansa, najbolja analitička solucija otvorenog koda. Dodatno, njezin razvoj je prilično rapidan i može se očekivati velik broj poboljšanja u kratkom vremenskom periodu. Iako su mnoge analitičke platforme izgrađene sa specifičnim ciljem, Impala se pokazala kao najbrže rješenje za većinu slučajeva korištenja, a gdje količina podataka pri upitu nije prevelika. Za izuzetno zahtjevnu analitiku ogromnih setova podataka prikladniji je Hive i njegove derivacije, iako je njegovo vrijeme odaziva dugotrajnije. Konačno, prikazano je da odabir analitičke platforme ovisi i o odabranoj Hadoop distribuciji, gdje distribucija ponekad diktira odabir analitičke platforme radi brže i lakše implementacije naprednih analitičkih sposobnosti nad već postojećim Hadoop rješenjem.

6. Literatura

[1.] Apache Drill (2015). The Apache Software Foundation, <https://spark.apache.org/documentation.html> Pristupljeno 3. srpnja 2015.

[2.] Apache Phoenix (2015). Performance. The Apache Software Foundation, <https://phoenix.apache.org/performance.html> Pristupljeno 17. srpnja 2015.

[3.] Apache Wiki (2011). Hadoop Wiki: NameNode. Apache, <http://wiki.apache.org/hadoop/NameNode> Pristupljeno 10. kolovoza 2015.

[4.] ArchLinux Wiki (2015). NFS. Archlinux, <https://wiki.archlinux.org/index.php/NFS> Pristupljeno 11. kolovoza 2015.

[5.] Armbrust M., Xin R., Lian C., Huai Y., Liu D., Bradley J.K., Meng X., Kaftan T., Franklin M.J., Ghodsi A., Zaharia M. (2015). Spark SQL: Relational Data Processing in Spark. CSAIL, <http://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf> Pristupljeno 1. srpnja 2015.

[6.] Dumbill, E. (2012). What is big data?: An introduction to the big data landscape. O'Reilly Media Inc, <http://www.springer.com/us/book/9783319106649> Pristupljeno 6. lipnja 2015.

[7.] Economist Intelligence Unit (2015). The Deciding Factor: Big Data & Decision Making. Capgemini, <https://www.capgemini.com/thought-leadership/the-deciding-factor-big-data-decision-making> Pristupljeno 24. kolovoza 2015.

[8.] Erickson J., Kornacker M. (2014). What's Next for Impala: Focus on Advanced SQL Functionality. Cloudera Inc, <http://blog.cloudera.com/blog/2014/08/whats-next-for-impala-focus-on-advanced-sql-functionality> Pristupljeno 12. kolovoza 2015.

[9.] Erickson J., Kornacker M., Kumar D., Rorke D. (2014). New Benchmarks for SQL-on-Hadoop: Impala 1.4 Widens the Performance Gap. Cloudera Inc, <http://blog.cloudera.com/blog/2014/09/new-benchmarks-for-sql-on-hadoop-impala-1-4-widens-the-performance-gap/> Pristupljeno 14. lipnja 2015.

[10.] Fawcett T., Provost F. (2013). Data Science and its relationship to Big Data and data-driven decision making. ResearchGate, <http://www.researchgate.net/publication/256439081_Data_Science_and_its_relationship_to_Big_Data_and_data-driven_decision_making> Pristupljeno 24. kolovoza 2015.

[11.] FreeBSD (2015). Network Information System (NIS). FreeBSD, <https://www.freebsd.org/doc/handbook/network-nis.html> Pristupljeno 12. kolovoza 2015.

[12.] Gates, A., Shanklin C. (2015). Announcing Hive 1.0: A Stable Moment in Time - A quick glimpse at the past, present, and future. Hortonworks Inc, <http://hortonworks.com/blog/announcing-hive-1-0-stable-moment-time/> Pristupljeno 20. lipnja 2015.

[13.] Gracion (2015). What is LDAP? Gracion Software, <http://www.gracion.com/server/whatldap.html> Pristupljeno 12. kolovoza 2015.

[14.] Harvey R., Martelli A. (2009). I never really understood: what is POSIX?. Stack Exchange Inc, <http://stackoverflow.com/questions/1780599/i-never-really-understood-what-is-posix> Pristupljeno 11. kolovoza 2015.

[15.] IBM (2015). End-to-end checksum. IBM, <https://www-01.ibm.com/support/knowledgecenter/SSFKCN_4.1.0/com.ibm.cluster.gpfs.v4r1.gpfs200.doc/bl1adv_introe2echecksum.htm> Pristupljeno 11. kolovoza 2015.

[16.] Java (2015). Java Help Center. Java, <https://java.com/en/download/help/> Pristupljeno 12. kolovoza 2015.

[17.] Julio, P. (2009). Big Data Analytics with Hadoop. LinkedIn Corporation, <http://www.slideshare.net/PhilippeJulio/hadoop-architecture> Pristupljeno 25. srpnja 2015.

[18.] Kornacker M., Behm A., Bittorf V., Bobrovytsky T., Ching C., Choi A., Erickson J., Grund M., Hecht D., Jacobs M., Joshi I., Kuff L., Kumar D., Leblang A., Li N., Pandis I., Robinson H., Rorke D., Rus S., Russell J., Tsirogiannis D., Wanderman-Milne S., Yoder M. (2015). Impala: A Modern, Open-Source SQL Engine for Hadoop.

Cloudera Inc, <http://www.cidrdb.org/cidr2015/Papers/CIDR15_Paper28.pdf> Pristupljeno 22. lipnja 2015.

[19.] Lipcon, T., Moore D., Jahangir M., Chaplot N. (2012). Does Cloudera Impala have any drawbacks when compared with Hive? Quora, <http://www.quora.com/Does-Cloudera-Impala-have-any-drawbacks-when-compared-with-Hive> Pristupljeno 17. lipnja 2015.

[20.] Lockwood, G.K. (2014). Conceptual Overview of Map-Reduce and Hadoop. <http://www.glennklockwood.com/data-intensive/hadoop/overview.html> Pristupljeno 10. lipnja 2015.

[21.] Mitchell, R.L. (2014). 8 big trends in big data analytics. Computerworld, Inc., <http://www.computerworld.com/article/2690856/8-big-trends-in-big-data-analytics.html> Pristupljeno 11. lipnja 2015.

[22.] Morabito, V. (2015). Big Data and Analytics: Strategic and Organizational Impacts. Springer, <http://www.springer.com/us/book/9783319106649> Pristupljeno 5. lipnja 2015.

[23.] Morgan, J. (2014). A Simple Explanation Of 'The Internet Of Things'. Forbes, <http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/> Pristupljeno 10. kolovoza 2015.

[24.] Oracle (2002). Oracle® Real Application Clusters Guard Installation Guide. Oracle Corporation, <http://docs.oracle.com/html/A96686_01/rollup.htm> Pristupljeno 12. kolovoza 2015.

[25.] Oracle Parquet (2014). Apache Parquet. Oracle Corporation, <http://parquet.apache.org/> Pristupljeno 12. kolovoza 2015.

[26.] Pivotal HAWK (2015). World's Most Advanced Enterprise SQL on Hadoop Analytic Engine. Pivotal Software Inc, <http://pivotal.io/big-data/datasheet/pivotal-hawq> Pristupljeno 15. srpnja 2015.

[27.] Presto (2015). Documents. Facebook Inc, <https://prestodb.io/docs/> Pristupljeno 25. lipnja 2015.

[28.] Prokopp, C. (2014). Faster Big Data on Hadoop with Hive and RCFile. Big Data Science and Cloud Computing, <http://www.semantikoz.com/blog/faster-big-data-hadoop-hive-rcfile/> Pristupljeno 12. kolovoza 2015.

[29.] Rathbone, M. (2014). 8 SQL-on-Hadoop frameworks worth checking out. Matthew Rathbone, <http://blog.matthewrathbone.com/2014/06/08/sql-engines-for-hadoop.html> Pristupljeno 19. srpnja 2015.

[30.] Rouse, M. (2005). Relational database management system (RDBMS). TechTarget, <http://searchsqlserver.techtarget.com/definition/relational-database-management-system> Pristupljeno 10. kolovoza 2015.

[31.] Rouse, M. (2012). SPI model (SaaS, PaaS, IaaS) definition. TechTarget, <http://searchcloudcomputing.techtarget.com/definition/SPI-model> Pristupljeno 23. rujna 2015.

[32.] Rouse, M. (2015). Integrated Development Environment (IDE) Definition. TechTarget, <http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment> Pristupljeno 12. kolovoza 2015.

[33.] SAS Institute Inc. (2014). Hadoop: What is it and why does it matter? SAS Institute Inc, <http://www.sas.com/en_us/insights/big-data/hadoop.html> Pristupljeno 9. lipnja 2015.

[34.] Schneider, R.D. (2013). Hadoop Buyer's Guide. Ubuntu, <http://insights.ubuntu.com/wp-content/uploads/HadoopBuyersGuide_sm.pdf> Pristupljeno 4. lipnja 2015.

[35.] Singhvi, B. (2014). Apache Hive Review. GISE - Part of Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Powai, Mumbai, <http://www.gise.cse.iitb.ac.in/wiki/images/2/26/Hive.pdf> Pristupljeno 15. lipnja 2015.

[36.] Spark (2015). Spark Documentation. The Apache Software Foundation, <https://spark.apache.org/documentation.html> Pristupljeno 29. lipnja 2015.

[37.] Tariq, M. (2013). Fast Hadoop Analytics (Cloudera Impala vs Spark/Shark vs Apache Drill). Stack Exchange Inc,

<http://stackoverflow.com/questions/17290397/fast-hadoop-analytics-cloudera-impala-vs-spark-shark-vs-apache-drill> Pristupljeno 13. lipnja 2015.

[38.] Tech Terms (2015). Metadata. Sharpened Productions, <http://techterms.com/definition/metadata> Pristupljeno 24. kolovoza 2015.

[39.] Techopedia (2015). Massively Parallel Processing (MPP). Janalta Interactive Inc, <http://www.techopedia.com/definition/2786/massively-parallel-processing-mpp> Pristupljeno 24. kolovoza 2015.

[40.] Webopedia (2015). Cold standby. QuinStreet Inc, <http://www.webopedia.com/TERM/C/cold_standby.html> Pristupljeno 11. kolovoza 2015.

[41.] Xin, R. (2014). Shark, Spark SQL, Hive on Spark, and the future of SQL on Spark. Databricks, <https://databricks.com/blog/2014/07/01/shark-spark-sql-hive-on-spark-and-the-future-of-sql-on-spark.html> Pristupljeno 27. lipnja 2015.

[42.] Zhu L., Tung B. (2006). Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). IETF, <https://tools.ietf.org/html/rfc4556> Pristupljeno 12. kolovoza 2015.

7. Sažetak (Summary)

Sažetak

Upotrebom analitike velike količine podataka, uz opće performanse kompanije, mogu se poboljšati performanse većine organizacijskih jedinica, uključujući upravljanje i odlučivanje u kompaniji. Velika količina podataka može se analizirati pomoću analitičkih platforma i pripadnih alata, a koje se razlikuju po performansama, zahtjevima i cijeni. Od upotrebe takvih podataka mogu imati koristi mnogi sektori i poslovi, ali i društvo u cjelini. Pametni gradovi logičan su slijed sve rastućeg iskorištavanja velike količine podataka, i razvoja povezanih stvari u sklopu fenomena Internet of Things.

Summary

By utilizing big data analytics, companies can increase general performance, performance of organizational units, including management and decision making. Big data can be analyzed with analytics platforms and tools which come bundled with them. They differ in their performances, requirements and prices. Many sectors, businesses and society in general can benefit from utilizing big data. Smart cities are a logical step in growing utilization of big data, and in development of connected things in a phenomenon called Internet of Things.