



TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JONNE PETTERI PIHLANEN SUOSITTELIJAJÄRJESTELMÄN RAKENTAMINEN APACHE SPARKILLA

Tiivistelmä

Tarkastaja: Timo Aaltonen

Suosittelujärjestelmät ovat jatkuvasti läsnä jokapäiväisessä elämässämme. Ne auttavat päätöksenteossa verkko-ostoksissa, suoratoistopalveluissa, sosiaalisessa mediassa tai yksinkertaisesti uutisten lukemisessa. Yksinkertaisin ja luonnollisin suosittelu tapahtuu ihmiseltä ihmiselle. Ihmiset voivat kuitenkin tehokkaasti suositella vain niitä asioita, jotka ovat itse henkilökohtaisesti kokeneet. Tällöin suosittelijajärjestelmistä tulee hyödyllisiä, sillä ne voivat mahdollisesti tarjota suosituksia tuhansista tai jopa miljoonista erilaisista tuotteista.

Suosittelijajärjestelmät eroavat toisistaan kohdistetun toimialan, käytetyn tiedon ja erityisesti suosittelualgoritmin perusteella, eli keinon, jolla suositukset tuotetaan. Tässä tekstissä keskitytään vain yhteen suosittelutekniikoiden luokkaan: *yhteisösuodatukseen*. Tätä menetelmää käytetään *Apache Spark* -sovelluskehityksessä, joka on yksi tekstin tärkeimmistä osa-alueista.

Yhteisösuodatus käyttävät suosittelijajärjestelmät perustuvat käyttäjien yhteistyöhön. Niiden tavoitteena on tunnistaa malleja käyttäjän mielenkiinnoista voidakseen tehdä suunnattuja suosituksia. Yhteisösuodatus voidaan jakaa kahteen kategoriaan: *tuotepohjaiseen- ja käyttäjäpohjaiseen yhteisösuodatukseen*. Tuotepohjaisessa suosittelussa on tarkoituksena etsiä samankaltaisia tuotteita, sillä käyttäjän ajatellaan olevan mahdollisesti kiinnostunut samankaltaisista tuotteista myös tulevaisuudessa. Käyttäjäpohjaisessa suosittelussa käyttäjän ajatellaan olevan kiinnostunut tuotteista, joita samankaltaiset käyttäjät ovat ostaneet. Tarkoituksena on siis tunnistaa samankaltaiset käyttäjät, jotta voidaan suositella näiden ostamia tuotteita.

Tekstiin liittyvän työn päämääränä on tutustua Apache Spark-sovelluskehitykseen sekä *Scala*-ohjelmointikieleen ja toteuttaa suosittelujärjestelmä näiden teknologioiden avulla siten, että suosittelua voidaan ajaa pilvipalvelussa. Pilvipalvelun ajateltiin tarjoavan laskentaresursseja käyttöön ja näin tapahtuikin. Alustaksi valikoitui ilman suurempia vertailuja Amazonin tarjoama *Amazon Web Services (AWS)*. Tästä alustasta oli hieman aiempaa kokemusta, sillä sitä on käytetty töissä ja sen osaamisesta on varmasti hyötyä jatkossakin.

AWS on Amazonin tarjoama kokoelma pilvilaskentaan (cloud computing) tarkoitettuja tai sitä avustavia palveluita. Kriteerinä oli Spark-sovelluksen ajaminen pilvipalvelussa ja Amazonin palveluista tässä autoivat *EMR (Elastic Map Reduce)* sekä *S3 (Simple Storage Service)*. EMR on hallittu klusterialusta, joka yksinkertaistaa big data -sovelluskehysten, kuten Apache Sparkin, käyttämistä AWS:n palveluissa. S3 on tietovarasto, joka on suunniteltu helpottamaan pilvilaskentaa ja se tarjoaa

yksinkertaisen rajapinnan tietovaraston hallintaan.

Työssä käytetty EMR-liukuhihna pystytettiin löytyneiden ohjeiden mukaisesti. Pysytettävän palvelun nimi on On-Demand Pipeline, mutta tosiasiaassa varatut EC2-instanssit pysyvät ajossa kellon ympäri. Tämän olisi voinut välttää laittamalla valintaruutuun valinnan, joka olisi sulkenut liukuhihnan ohjelman suorituksen jälkeen, mutta tässä työssä tavoitteena oli päästä ajalemaan silloin kun oli tehty uusi versio. Tätä ei varsinaisesti kommunikoitu missään vaiheessa, vaan asiaan havahtui vasta laskun saavuttua. Kuukauden mittaisesta EC2 (m3.xlarge) -instanssien ajamisesta olisi tullut maksettavaa reilut 1000 dollaria.

Apache Spark on sovelluskehys, joka mahdollistaa hajautettujen ohjelmien rakentamisen. Hajautetussa ohjelmassa suoritus voidaan jakaa useiden käsittelysolmujen kesken. Jotkin suosittelemat voidaan mallintaa hajautettuna ohjelmalla, jossa kaksi matriisia, käyttäjät ja tuotteet, prosessoidaan iteratiivisella algoritmilla, joka mahdollistaa ohjelman suorittamisen rinnakkain. Spark-ohjelmia voidaan kirjoittaa Java-, Scala-, Python- sekä R-ohjelmointikielillä. Jokainen Spark-sovellus koostuu driver-ohjelmasta sekä yhdestä tai useammasta executor-ohjelmasta. Driver on ohjelma, joka ajaa käyttäjän pääohjelmaa ja hajauttaa laskennan klusteriin. Executor on yksi kone klusterissa.

Spark voidaan esitellä kuvailemalla sen edeltäjää, MapReduce:a, ja sen tarjoamia etuja. MapReduce tarjosi yksinkertaisen mallin ohjelmien kirjoittamiseen ja pystyi suorittamaan kirjoitettua ohjelmaa rinnakkain sadoilla tietokoneilla. MapReduce skaalautuu lähes lineaarisesti datan koon kasvaessa. Suoritusaikaa hallitaan lisäämällä lisää tietokoneita suorittamaan tehtävää.

Apache Spark säilyttää MapReduce:n lineaarisen skaalautuvuuden ja vikasietokyvyn mutta laajentaa sitä muutamalla merkittävällä tavalla. MapReducessa map- ja reduce-tehtävien väliset tulokset täytyy kirjoittaa levyille kun taas Spark kykenee välittämään tulokset suoraan liukuhihnan (pipeline) seuraavalle vaiheelle. Lisäksi Spark esittelee muistissa tapahtuvan prosessoinnin tarjoamalla abstraktion nimeltä *Resilient Distributed Dataset (RDD)*. RDD tarjoaa kehittäjälle mahdollisuuden materialisoida minkä tahansa askeleen liukuhihnassa ja tallentaa sen muistiin. Tämä tarkoittaa sitä, että tulevien askelten ei tarvitse laskea aiempia tuloksia uudelleen ja tällöin on mahdollista jatkaa juuri käyttäjän haluamasta askeleesta. Aiemmin tämänkaltaista ominaisuutta ei ole ollut saatavilla hajautetun laskennan järjestelmissä.

Apache Spark on rakennettu Scala-ohjelmointikielellä. Scala on monikäyttöinen, moniparadigmainen ohjelmointikieli, joka tarjoaa tuen funktionaaliselle ohjelmoinnille sekä vahvan tyyppityksen. Funktionaalista ohjelmointia varten Scalasta löytyy tuki funktionaalisen ohjelmoinnin käsitteille, kuten muuttumattomat (immutable) tietorakenteet ja funktiot ensimmäisen luokan kansalaisina. Olio-ohjelmointia varten Scalasta löytyy tuki käsitteille kuten luokat, oliot ja piirre (trait). Scala tukee myös kapselointia, perintää, moniperintää ja muita tärkeitä olio-ohjelmoinnin konsepteja. Scala on staattisesti tyyppitetty kieli ja sillä kirjoitetut ohjelmat käännetään Scala-kääntäjää käyttäen. Scala on JVM-perustainen (Java Virtual Machine, Java-virtuaalikone) kieli, joten Scala kääntäjä kääntää sovelluksen Java-tavukoodiksi, joka voidaan ajaa missä tahansa Java-virtuaalikoneessa. Tavukooditasolla Scala ohjelmaa ei voida erottaa Java sovelluksesta. Scalan ollessa JVM-perustainen, Scala on täysin yhteensopiva Javan kanssa ja näin ollen Java-kirjastoja voidaan käyttää suoraan Scala-koodissa. Tästä syystä Scala-sovellukset hyötyvät suuresta Java-koodin määrästä. Vaikka Scala tukee sekä olio- että funktionaalista ohjelmointia, funktionaalista ohjelmointia suositaan.

Saadut tulokset eivät ole palveluiden kuten Netflix tasolla, mutta ei sitä varmaan kannattanut odottaakaan. Mielenkiintoista oli se, kuinka "huonoja" saadut suositukset olivat. Yllättäviä ja uusia kylläkin, mutta mikään elokuvista ei kuulosta mielekkäältä. Tässä tosin voikin piillä juuri hyvän suosittelun raja, sillä luultavasti ihmisen muodostama mielipide pelkän nimen perusteella saattaa johtaa elokuvan hylkäämiseen. Ihminen ei välttämättä ole täysin objektiivinen valitsemaan sitä, onko jokin suositeltu elokuva katsomisen arvoinen. Elokuvan julkaisuvuosi, ohjaajan tunnettuus, näyttelijät ja jopa kansikuva herättävät mielipiteitä, jotka saattavat johtaa elokuvan hylkäämiseen.