I. IKERGAZTE
NAZIOARTEKO IKERKETA EUSKARAZ

2015eko maiatzaren 13, 14 eta 15
Durango, Euskal Herria

ANTOLATZAILEA:
Udako Euskal Unibertsitatea (UEU)

INGENIARITZA ETA ARKITEKTURA

Testu kopuru handiak prozesatzeko big data teknikak

Z. Beloki, X. Artola eta *A. Soroa*

589-598 or.
https://dx.doi.org/10.26876/ikergazte.i.81

Testu kopuru handiak prozesatzeko *big data* teknikak

Beloki Z. eta **Artola X.** eta **Soroa A.**

IXA Taldea
Euskal Herriko Unibertsitatea (UPV/EHU)
Donostia, Euskal Herria.

Laburpena

Eskura dauzkagun datu kopuru erraldoiak prozesatzeko, zaharkituta gelditu dira XXI. mendearen hasieran erabiltzen ziren prozesaketa-teknikak eta algoritmoak. Gaur egun sistema banatuak erabiltzen dira, prozesaketa makina batean baino gehiagotan eginez. Gauza berbera gertatzen da hizkuntzaren prozesamenduan ere. Corpusak edo testu-bilduma handiak prozesatzeko, makina bat baino gehiagoko inguruneak beharrezkoak bihurtu dira dagoeneko. Lan honetan, testu-dokumentu kopuru handiak ingurune banatuetan prozesatzeko teknikak aztertuko ditugu. Horretarako, makina birtualetan oinarritutako sistema bat eraiki dugu, Storm konputazio banatuko frameworka erabiliz. Esperimentu batzuk ere aurkeztu ditugu, eta hainbat ezarpenekin lortutako errendimenduaren hobekuntzak.

Hitz gakoak: Big data, hizkuntzaren prozesamendua, sistema banatuak

Abstract

Processing techniques and algorithms used at the beginning of the 21th century to process massive data sets have become obsolete. Nowadays, distributed systems are used to performing the processing in several computers simultaneously. In the Natural Language Processing field, clusters of several computers are already necessary to process large quantities of text. In this work we analyze an architecture to perform distributed processing of text. The architecture relies on virtual machines and is based on the Storm distributed processing framework. We describe some experiments and show the performance gain obtained in diverse settings.

Keywords: Big data, natural language processing, distributed systems

1 Sarrera eta motibazioa

Egunetik egunera horrenbeste informazio berri sortzen den garai hauetan, ezinezkoa da gizakiak, makinen laguntzarik gabe, informazio hori guztia prozesatzea. Lanbide eta arlo askotan beharrezkoa ere bada, ordea, munduan gertatzen denaz jabetu eta gai desberdinen inguruan egunean mantentzea. Informazioa eskuratzea geroz eta errazagoa da. Zailena, eskura daukagun informazio horri etekina ateratzea da.

XX. mendean aurrerapen handiak egin ziren arlo horretan, ordenagailuen gizarteratzeak informazioa automatikoki prozesatzea ahalbidetu baitzuen. Horrela, datu kopuru handiak programa baten bidez prozesatu eta emaitza gizakiak errazago ulertzeko moduko formatu batean jaso zitekeen. Horren aplikazioak, datuen kalkulu estatistikoak egitetik, gertakizunen portaera edo patroiak erauzteraino zabaltzen ziren.

Azken urteetan esponentzialki hazi da munduan zehar dabilen informazio kopurua; Interneti esker, batik bat. Eta ez da informazioa eskuratzea erraztu zaigulako soilik, sortzen den informazio eta datu kopurua bera ere asko hazi baita. Sare sozialak dira horren adibide, bertan publikatzen den guztia informazio oso baliagarria baita erakunde edo enpresa askorentzat. XXI. mendean esparru berri bat sortu da informatikaren barruan, informazio kopuru handiei etekina ateratzean datzana. *Big data* deritzogu kontzeptu berri honi. Esate baterako, sare sozialetan idazten diren mezu multzo erraldoiak prozesatuz,

gaixotasunen agerraldiak antzeman daitezke, jendeak, bere mezuetan, nabari dituen sintomak aipatzen dituen neurrian.

Azkeneko hamarkadan aurrerapauso handiak eman dira arlo honetan. Asko garatu dira informazio kopuru handiak prozesatzeko teknikak. Zeregin honetan bete-betean sartuta daude mundu osoko hainbat ikerkuntza talde, herrialdeetako gobernu asko eta IBM, Google, Microsoft eta Amazon bezalako enpresa erraldoiak ere. Diru asko inbertitzen ari dira, beraz, *big data*ko algoritmo eta baliabideen ikerkuntzan.

Hizkuntzaren prozesamenduaren (HP) arloan ere gaurkotasun handiko gaia da *big data*ren kontzeptua. HPn testuak prozesatzen dira, informazio linguistikoaz baliatuz hainbat aplikazio lortzeko. Itzultzaile automatikoena da HParen aplikazio garrantzitsuenetako bat. Testu bat automatikoki hizkuntza batetik bestera itzultzen da, gizakiaren parte-hartzerik gabe. Beste aplikazio bat galdera-erantzun sistemena da, eta, hauetan, sistemak, erabiltzaile batek idatzitako galderei erantzuten die automatikoki. Euskal Herrian bertan ere, bertsoak automatikoki sortu eta botako dituen robot bertsolari bat sortzeko lanean ari dira (Astigarraga *et al.*, 2013). Aplikazio horiek gauzatzeko, ordea, gehienetan beharrezkoa izaten da corpusen erabilera. Corpusak testu-bilduma erraldoiak dira. Adibide gisa ZT corpusa [1] daukagu, zientzia eta teknologiaren alorreko euskarazko testu-bilduma. Baita Lexikoaren Behatokia corpusa [2] ere, hainbat euskal komunikabidetako artikuluekin osatutako bilduma. Corpus bat HP tresnekin prozesatzean, testuaren gaineko analisi linguistikoak lortzen dira, lehen aipatutakoak bezalako aplikazioak garatzeko beharrezkoak. Esaterako, morfosintaxi-analisi batek, besteak beste, hitz baten kategoria zein den esaten du, hau da, izena den, aditza, determinatzailea edo beste zerbait. Lematizazio-analisi batek, bestalde, hitz baten lema zein den definituko du. Lema hitz baten hiztegi-forma da. Adibidez, "etorriko" hitzaren lema "etorri" da, eta "politarekin" hitzarena "polit". Aipatutakoez gain, analisi sintaktikoa, sentimentuen analisia, rol semantikoena eta beste hainbat motatakoak ere egiten dira. Laburbilduz, corpus bat HP tresnekin prozesatzean, testu zati bakoitzaren gaineko anotazio konplexuekin osatutako testu-bilduma erraldoi bat lortzen dugu. Informazio hori lehen aipatutako itzulpen automatikoaren eta antzeko beste sistema askoren oinarria izango da.

Big data kontzeptura itzuliz, corpusak gero eta handiagoak diren honetan, beharrezkoa bihurtzen ari da corpus osoak prozesatu ahal izateko teknika berriak diseinatzea. Esate baterako, NewsReader (Agerri *et al.*, 2014a) proiektuan, egunean ehunka mila dokumentu prozesatzea aurreikusten da. Kontuan hartuz dokumentu bakoitza hamabost HP tresnak prozesatu behar dutela, eta dokumentu bakoitza prozesatzeak bost minututik gora hartzen dituela, urtebete inguru beharko litzateke sistema arrunt batekin 100.000 dokumentu prozesatzeko.

*Big data*n erabiliena den eredua prozesua zati txikitan zatitu eta hainbat makinatan banatuta exekutatzean datza. Hau da, prozesatu beharreko informazio kopurua handiegia denez ordenagailu bakar batean exekutatzeko, oso azkarra den ordenagailu bat bilatu ordez, arruntak eta merkeak diren ordenagailu (nodo) askoz osatutako sistema (cluster) bat lortzea izaten da helburua. Horrela, prozesu bat zatitu eta clusterreko nodoetan zehar banatuta paraleloan exekutatzeko gai diren algoritmoak ditugu aztergai. Sistema horiei sistema banatuak deritze, prozesaketa nodoetan banatuta egiten baita. Hauen ezaugarri garrantzitsuenetako bat eskalagarritasun horizontala da. Eskalagarria izateko, sisteman zenbat eta ordenagailu gehiago txertatu, eraginkortasunak ere proportzionalki igo behar du.

2 Arloaren egoera eta ikerketaren helburuak

Datu kopuru masiboen prozesaketa modu eraginkorrean eta eskalagarrian egitea HPn gaurkotasun handia daukan erronka da. Prozesatzen diren testuak, askotan, web-orriak edo egunkarietako artikuluak bezalako datu ez-egituratu moduan egon ohi direnez, dokumentu arteko korreferentzia (Mayfield *et al.*, 2009) edota gertakarien detekzioa (Ritter *et al.*, 2012) bezalako atazetan, maiz, milioika dokumentu prozesatu behar izaten dira denbora mugatuan. Adibidez, Singh *et al.* (2011) autoreek azken 20 urteetako berrien artikuluez osatutako corpus bat prozesatu dute. McCreadie *et al.* (2013) autoreek segundoko ehunka mila txio prozesatzeko ahalmena daukan gertaeren detekzioa egiteko framework banatu bat aurkeztu zuten. HPko algoritmoak testu kopuru handiekin exekutatzeko proposamen bat ere aurkeztu zuten Goyal *et al.* (2009) autoreek.

[1] `http://www.ztcorpusa.net/aurkezpena.htm`
[2] `http://lexikoarenbehatokia.euskaltzaindia.net/aurkezpena.htm`

Yu eta Chen autoreek (2013) informazio semantikoaren prozesaketa banatua lainoan egitearen arteaen egoera aztertzen dute. Bertan aipatzen duten bezala, makina birtualak (MB) erabiltzen dira, askotan, horrelako zereginetarako. MBen erabilera nahiko zabala da prozesaketa banatua burutzeko, batez ere, prozesaketa lainoan egiten denean. Makina birtual bat konputagailu baten software bidezko inplementazio bat da, konputagailu erreal batean exekutatuko dena. MB bat konputagailu batean exekutatzean, konputagailu horrek exekutatutako MBaren itxura hartzen du eta MBan instalatutako softwarea eta ezarpenak dira une horretatik aurrera konputagailuan erabilgarri edukiko ditugunak. Oso erabilgarria da nork bere prozesuak lainoan dauden superkonputagailuetan exekutatzeko, normalean ez baita konputagailu horietarako sarbiderik izaten beharrezko softwarea etab. instalatzeko. Horregatik, erabiltzaileak beharrezko prozesaketa egiteko prest daukan MB bat eraikitzen du, lainoan dauden konputagailuetan oso modu sinplean exekutatuz. Amazon bezalako enpresa handiek superkonputagailuak eskaintzen dituzte, bakoitzak bere prozesuak bertan exekuta ditzan prezio baten truke.

Prozesaketa astunak modu eskalagarrian egiteko sistemen beharra gaur egun oso ohikoa denez, framework bat baino gehiago garatu dira zeregin hori errazteko. Antzeko arazoei aurre egin behar izaten dieten programa gehienek zati handi bat oso antzekoa izaten dute. Kodearen berrerabilgarritasuna bermatzeko, zati komun hori identifikatu eta bereizi egiten da askotan, gainontzeko aplikazioek berrerabiliko duten modulu independente bat eskainiz. Modulu horiei framework deitzen diegu. Beharbada, Hadoop [3] da konputazio banatua egiteko dagoen frameworkik erabiliena. Hadoopek MapReduce (Dean eta Ghemawat, 2008) algoritmoa erabiltzen du prozesaketa banatua egiteko. MapReduce algoritmoa Googlek sortu zuen, programa sinpleak datu kopuru izugarriekin exekutatzeko beharrari aurre egiteko. Hadoopen ezaugarri nagusietako bat batch-prozesaketara zuzenduta egotea da. Batch-prozesaketan, datu multzo oso bat bidaltzen da prozesatzera, prozesaketak hasiera eta bukaera finko bat dituelarik. Beste eredua streaming-prozesaketa da, non programa etengabe martxan dagoen datu berriak noiz iritsiko zain, iritsi ahala prozesatuz. Streaming motako prozesaketak egiteko Hadoop sistema ez da oso naturala. Hutsune hori betetzeko, Spark (Zaharia *et al.*, 2010) frameworka garatu zuten, Hadoopen antzeko funtzionalitateak biltzen dituena baina streaming-eredua modu naturalagoan lantzen duena.

Hadoop eta Spark sistemen beste alternatiba bat Storm [4] da. Hau ere streaming-prozesaketara zuzenduta dago, eta prozesuak hainbat nodoz osatutako cluster batean banatzeaz arduratzen da. Twitterreko txioen prozesaketan, adibidez, oso erabilia da sistema hau.

Azken hau da gure sistemaren oinarri izateko aukeratu duguna. Gure helburua dokumentu kopuru erraldoiak modu eraginkor batean prozesatuko dituen streaming-sistema banatu eta eskalagarri bat egitea da. Streaming-sistema izango den arren, prozesaketa batch moduan egitea ere eskaini beharko luke. Sistema banatua izango dela diogu, prozesaketa hainbat konputagailutan zehar banatuta egingo delako. Azkenik, eskalagarria izan behar du, milioika dokumentu prozesatzen direnean clusterrera konputagailu berriak gehitzeak aski izan behar bailuke sistema hobetzeko, programa bera aldatu behar izan gabe.

Dokumentuak dagoeneko inplementatuta dauden 15 HP moduluz osatutako analisi-kate batekin prozesatuko ditugu. Garrantzitsua da argi uztea gure helburua modulu horiek bere horretan uztea dela, aldatu gabe. Hau da, dagoenarekin moldatuko gara prozesaketa banatua egiteko. Puntu hau garrantzitsua da, ondoren, analisi-katera modulu berriren bat gehitu behar bada, moduluarekin arazorik ez izateko. Gure sistema oso orokorra izango da, beraz.

Horrez gain, makina birtualak erabiliko ditugu sistema eraikitzeko. Horrela, beharrezko softwarea behin instalatu eta konfiguratuko dugu, eta hortik aurrera, makina birtualak edozein makina fisikotan ezarri eta exekutatu ahal izango ditugu. Gainera, sistema hau publikoki zabalduko dugunez, edonorentzat errazagoa izango da sistema erabiltzea, aurrez sortutako makina birtualak edukita.

Azkenik, analisi-tresnen kateak ere konfiguragarria izan behar du. Hasiera batean HPko tresna jakin batzuk ezarriko ditugun arren, tresna jakin bat kendu edo berri bat txertatzeko prozesuak sinplea eta erraza izan behar du.

[3] `http://hadoop.apache.org/`
[4] `https://storm.apache.org/`

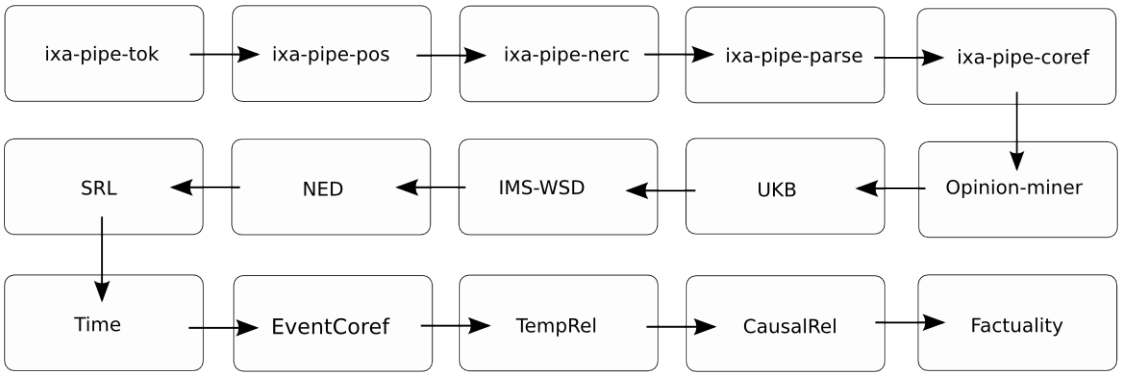
3 Ikerketaren muina

3.1 Analisi-katea

Testu bat prozesatzen, normalean, testua hainbat programak exekutatzen dute, batak bestearen atzetik. Programa horiei hizkuntzaren prozesamendurako modulu (HP modulu) deitzen diegu. Normalean, HP moduluek kate bat osatzen dute, modulu bakoitzaren irteera hurrengoaren sarrera izanik. Horrela, kateko lehenbiziko moduluak jatorrizko testua hartuko du, eta emaitza hurrengo moduluari pasatuko dio. Modulu bakoitzak testuari anotazioak eransten dizkio, informazio linguistikoa gehituz. Adibidez, tokenizatzailea, testua tokenetan banatzen duen HP modulua da. Tresna hau, normalean, analisi-kateko lehenbiziko modulua izaten da, eta jatorrizko testua jasotzen du sarrera gisa. Ondoren, testuko tokenak bereizten ditu, hau da, hitz eta puntuazio-ikur guztiak identifikatzen ditu, eta bakoitzarentzat anotazio bat sortzen du. Hurrengo moduluak tokenizatzailearen anotazioak erabiliko ditu bere zeregina burutzeko, testuari bigarren anotazio-geruza bat erantsiz. Azkeneko moduluaren emaitza analisi-kate osoak prozesatutako testu anotatua izango da. Informazio hori behar-beharrezkoa izango da, ondoren, hizkuntzarekin lotutako aplikazio funtzionalak lortzeko.

Proiektu honetan erabili dugun analisi-katea 15 HP moduluz osatuta dago. 1. irudian ikus daiteke katearen parte diren moduluak zein diren. Lehenengo bostek, *IXA pipes* (Agerri *et al.*, 2014b) familia osatzen dute. Jarraian, batzuk sakonago aipatzearren, *IXA pipes* osatzen duten moduluak aipatuko ditugu, bakoitzak zein funtzio betetzen duen azalduz:

1 Irudia: Analisi-katea osatzen duten HP moduluak



- **ixa-pipe-tok** testua tokenetan banatzen duen modulua da. Tresna hau, normalean, analisi-kateko lehenbiziko modulua izaten da eta jatorrizko testua jasotzen du sarrera gisa. Ondoren, testuko tokenak bereizten ditu, hau da, hitz eta puntuazio-ikur guztiak identifikatzen ditu, eta bakoitzarentzat anotazio bat sortzen du.
- **ixa-pipe-pos** moduluak, hitzen kategoriak identifikatzeaz gain, lematizazioa egiten du. Adibidez, "lagun" hitzaren kategoria izena da, eta "etorri"-rena, aditza. Lema, berriz, hitzaren zati nagusia da. Adibidez, "lagunaren" hitzaren lema "lagun" da, eta "etortzea"-rena, "etorri". Modulu honek tokenizazio mailako anotazioak behar ditu bere lana burutzeko. Horregatik, ohikoa da kateko bigarren modulua izatea, tokenizatzailearen jarraian.
- **ixa-pipe-nerc** izendun entitateak identifikatzen dituen tresna da. Izendun entitateak lau multzotan sailkatzen ditu: pertsona-izenak, toki-izenak, erakunde-izenak eta bestelakoak. Modulu honek beharrezkoa du ixa-pipe-pos tresnaren emaitza.
- **ixa-pipe-parse** moduluak analisi sintaktikoa egiten du. Horretarako, honek ere, tokenak eta beren kategoria eta lemak behar ditu. Beraz, ixa-pipe-pos tresnaren atzetik egongo da analisi-katean, baina ixa-pipe-nerc tresnarekiko posizioak ez du garrantziarik, ez baitute elkarrekiko dependentziarik.
- **ixa-pipe-coref** moduluak korreferentzia ebazten du, hau da, testuko hitzen arteko erreferentziak identifikatu eta ebazten ditu. Adibidez, *"sendagilearengana joan eta hark esandakoa egin"* esaldian,

"hark" hitzak sendagileari erreferentzia egiten dio. Modulu honek, ixa-pipe-nerc moduluaren irteera behar du korreferentziak behar bezala identifikatu eta ebazteko.

Noski, etorkizunean analisi-kate hori osatu edo laburtu egin nahi izango dugu. Ez hori bakarrik, sistema osoak edozein analisi-katerekin funtzionatu behar luke. Hori dela eta, modulu bat kentzeak edo gehitzeak ez luke arazorik eman behar. Guk, hori bete dadin, aurrebaldintza bat ezarri dugu modulu bat gure sistemarekin bateragarria izan dadin: modulu horrek jaso behar duen sarrerak eta sortuko duen irteerak formatu jakin batean kodetuta egon behar dute. Aukeratu dugun formatua NAF (NLP Annotation Format) (Fokkens *et al.*, 2014) da.

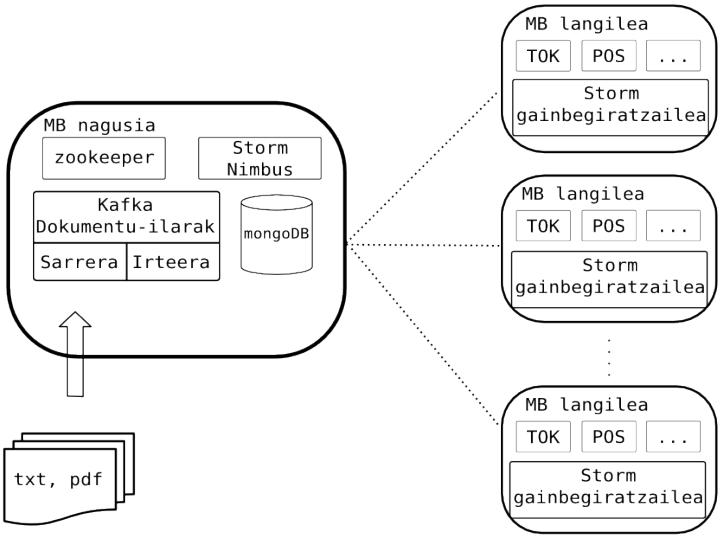
3.2 Testuen prozesaketa

HP moduluekin osatutako analisi-kate bat exekutatzeko ohiko bidea linealki egitea da. Konputagailu batean modulu guztiak instalatu, eta prozesatu beharreko dokumentuak, banan-banan, modulu guztietatik pasatuko dira. Hori eskalagarri egiteko, sistema banatu batean exekutatzea da egokiena. Horrela, prozesaketa hainbat konputagailutan banatuta egingo da, zenbat eta konputagailu gehiago jarriz, orduan eta azkarrago. Hori da lan honetarako hartu dugun bidea.

3.2.1 Sistemaren arkitektura

Sistema banatua osatzeko hainbat teknologia erabili behar izan ditugu. Sistemak, funtzionala izan dadin, atal ugari eta askotarikoak dauzka: prozesaketa banaturako frameworka, sarrera-irteerako dokumentuen ilarak, tarteko informazioa gordetzeko datu-basea, clusterra kudeatzeko frameworka... 2. irudian sistemaren arkitektura orokorra ikus daiteke.

2 Irudia: Dokumentuen prozesaketa banatua egiteko sistemaren arkitektura



Sistema antolatzeko makina birtualak (MB) erabili ditugu. Bi MB mota bereizten ditugu: nagusia eta langileak. MB nagusiak sistema kudeatzen du, eta bertan daude instalatuta datu-basea, prozesu kontrolatzaileak, sarrera-irteerako ilarak etab. MB langileek, aldiz, HP modulu guztiak dauzkate instalatuta, eta bertan egiten da testuen prozesaketa-lan guztia. Noski, MB nagusi bakarra dago sistema osoan, baina MB langileak nahi adina egon daitezke. Eskura dauzkagun makina fisikoak izango dira MB kopuruaren muga markatuko dutenak. Hasiera batean, CPU bakoitzeko, asko jota, MB langile bat ezarri ahal izango dugu. Kontuan izan beharreko beste gauza bat da MB langile bakoitzak 10 GB-ko memoria behar duela.

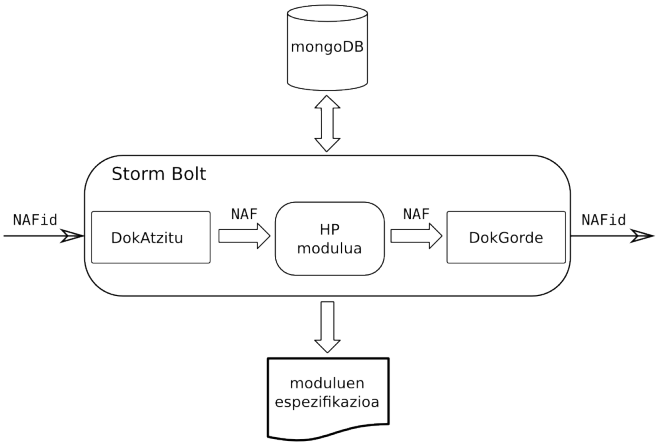
Jarraian, sisteman parte hartzen duten atal garrantzitsuenak laburbilduko ditugu:

- **Storm**: Streaming-prozesaketa banatua egiteko sistema. Honi esker, gure HP moduluak cluster oso-

ko nodoetan zehar banatuko ditugu, prozesaketa paraleloa ahalbidetuz. Adibidez, nodo guztietan modulu guztiak ezartzeko eska diezaiokegu Stormi. Ondoren, dokumentuak prozesatzera bidaltzen ditugunean, Stormek erabakiko du dokumentu bakoitza zein nodotan prozesatuko den. Horrez gain, erroreekiko tolerantzia, dokumentu guztiak prozesatuak izango direnaren bermea eta beste hainbat abantaila eskaintzen dizkigu. Storm erabiliz sistema bat garatzeko, lehenbizi Storm topologia bat definitu behar da. Besterik gabe, ezarriko diren modulu guztiak grafo batean adieraziko dira, moduluak elkarren arteko dependentzien arabera lotuz. Topologiako nodo bakoitzari *bolt* deitzen zaio.

- **MongoDB datu-basea**: Moduluen arteko informazio-trukearen ondorioz clusterreko nodoen artean sor daitekeen datu-trafiko handia dela-eta, datuen transferentzia optimizatzeko irtenbide bat diseinatu dugu. Modulu bakoitzak ez dio hurrengo moduluari sortutako NAF dokumentu osoa bidaliko. Aldiz, bere irteera aldi baterako datu-basean gordeko du, sortu berri dituen anotazioak bakarrik. Ondoren, hurrengo moduluak, datu-basetik, behar-beharrezkoak dituen anotazioak bakarrik jasoko ditu. Horrela, modulu batek hurrengoari bidali beharreko informazio bakarra dokumentu-identifikadorea da, hark datu-basetik informazio egokia atzi dezan. 3. irudian hobeto ikusten da azaldutakoa. NoSQL datu-base bat erabili dugu, MongoDB motakoa hain zuzen. Besteak beste, eraginkortasuna eta ingurune banatuetara egokitzeko gaitasuna dira haren ezaugarri nagusietakoak.
- **Sarrera-irteerako dokumentuen ilarak**: Dokumentuak prozesatzera streaming moduan bidaliko dira. Edozein unetan, erabiltzaileak, dokumentu bat edo batzuk bidali ahal izango ditu programa edo web baten bidez. Sisteman dokumentu bat jasotzean, memorian dagoen ilara batean gordeko da sistema lanpetuta dagoen bitartean, libratzen denean bertatik jasotzeko. Gauza berbera gertatzen da prozesatutako dokumentuekin, irteerako ilara batean gordeko baitira, edozein unetan erabiltzaileak bertatik jasotzeko. Ilarak Kafka sistema erabiliz inplementatu ditugu.

3 Irudia: MongoDBren erabilera sisteman



3.2.2 Topologiak

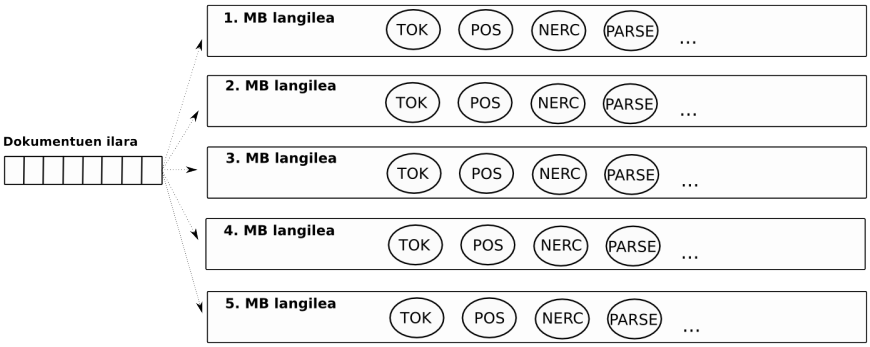
Storm sistemaren laguntza izango dugu prozesaketa banatua egiteko, baina, horretarako, topologia bat definitu beharra daukagu lehenik. Stormen topologia egokia definitzean egongo da prozesuen banaketa egokia lortzeko gakoa. Gure sisteman bi topologia definituko ditugu, eta erabiltzaileak bien artean aukeratuko du, beharraren arabera. Izan ere, sistemak bi prozesaketa mota onartuko ditu: batch- eta streaming-prozesaketak. Bi prozesaketa motetarako topologia desberdinak definituko ditugu, bi kasuetan eraginkortasuna optimoa izan dadin.

- **Batch-prozesaketa**n, prozesatuko diren dokumentu guztiak batera bidaliko dira prozesatzera, eta, beraz, dokumentu guztiak prozesatzeko behar den denbora da garrantzitsuena: horri esaten zaio throughputa. Kasu honetan, clusterreko MB langile bakoitzean analisi-kate oso bat ezartzen dugu, eta dokumentuak makinetan banatzen dira. MB bakoitzean, dokumentu bat prozesatzez bukatzen den unean hurrengo dokumentua prozesatzen hasiko da, baina inoiz ez dira makina berean bi dokumentu

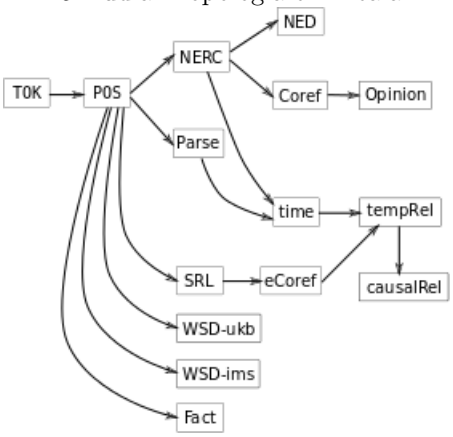
aldi berean prozesatzen egongo. Horrela, makina guztiak lanpetuta edukitzea lortzen dugu, baina makinetako bakoitza gainkargatu gabe. Eredu honetan, dokumentuen banaketa egokian dago gakoa, makina guztiek lan-karga antzekoa izan dezaten. Eredu hau argiago ikus daiteke 4. irudian.

- **Streaming-prozesaketa**n, dokumentuak denboran zehar iritsiko dira sistemara, banaka edo multzo txikiagotan. Beraz, *throughput*a ere garrantzitsua izango den arren, dokumentu bakoitza prozesatzeko behar den denbora (latentzia) minimizatzea izango dugu helburu. Horretarako, dokumentuak nodoetan zehar banatu ordez, topologia ez-lineal bat definituko dugu, 5. irudian ikus daitekeen bezala. Horrela, dokumentu bakoitza ez da makina bakar batean exekutatuko. Aldiz, dokumentu bakoitza makina desberdinetan prozesatuko da aldi berean, baina makina bakoitzean modulu desberdin bat aplikatuz. Modulu batzuk ezin dira aldi berean exekutatu, batak bestearen irteera jaso behar baitu lana ondo burutzeko. Beste kasu batzuetan, aldiz, elkarren artean dependentziarik ez dutenez, paraleloan exekuta daitezke. Esaterako, NERC, parse eta SRL moduluak paraleloan exekuta daitezke, hirurek behar dituzten datuak modulu berberaren (POS) irteera baitira. NERC eta NED moduluak, berriz, ezin izango dira paraleloan exekutatu, NED moduluak NERCen emaitza behar baitu prozesatzeko. Bestalde, orokorrean streaming-egoera batean sistemaren lan-karga txikiagoa izango denez, ez dugu makina bakoitzean modulu guztien kopia bana ezarriko. Moduluen prozesatze-denborak neurtuta, denbora horien arabera proportzionalki zehaztuko da cluster osoan jarriko den modulu bakoitzaren kopurua.

4 Irudia: Batch-prozesaketaren eskema



5 Irudia: Topologia ez-lineala



3.2.3 Esperimentuak eta emaitzak

Bi prozesaketa motak kontuan hartuz, bi esperimentu egin ditugu. Lehenik, batch moduan dokumentu multzo bat prozesatu dugu, denborak eta errendimendua neurtuz. Streaming moduan ere beste dokumentu multzo bat prozesatu dugu, latentzia neurtuz.

1 Taula: Batch-prozesaketaren esperimentuaren emaitzak. Igarotako denbora, prozesaketaren hasieratik bukatu arte pasatu den denbora erreala da. Prozesatze-denbora, aldiz, nodo bakoitzaren denboren batura da. Azken hori, paralelizaziorik gabeko ingurune batean behar izango litzatekeen denboraren parekotzat hartzen dugu. Denborak minututan daude.

Dok. kop.	Igarotako denb.	Throughputa	Prozesatze-denb.	Latentzia (dok/esaldi/token)
2.000	1.819,47	1,10	15.130,78	7,14 / 0,21 / $8{,}40\times10^{-3}$
1.000	719,4	1,39	6.519,12	6,66 / 0,18 / $7{,}48\times10^{-3}$

2 Taula: Streaming-prozesaketaren esperimentuaren emaitzak. Denborak minututan daude.

Ezarpena	Dok. kop.	Prozesatze-denb.	Latentzia (dok/esaldi/token)
Lineala	1.000	4.620,90	5,00 / 0,14 / $5{,}78\times10^{-3}$
Ez-lineala	1.000	5.421,33	2,78 / 0,08 / $3{,}13\times10^{-3}$

Batch-prozesaketan 1000 eta 2000 dokumentuko bi multzo prozesatu ditugu, 6 MBko ingurune batean (nagusi bat + bost langile). MB bakoitzari bi CPU eta 10 GB RAM esleitu zaizkio; beraz, 10 CPU dauzkagunez erabilgarri, Stormen hari kopurua edo paralelizazio maila 10 balioarekin definitu dugu. 1. taulan ikus daitezke lortutako emaitzak.

Emaitzei erreparatuta, minutuko 1,1 dokumentu prozesatzen direla ikus daiteke. Kontuan izanik latentzia 7,14 minutukoa dela, hau da, dokumentu bakoitza makina bakar batean prozesatzeko minutu kopuru hori behar dela, garbi dago paralelizazioari esker denbora asko aurreztu dugula. Hala ere, 10 CPU izanik, prozesatze-denbora igarotako denbora baino 8,32 aldiz handiagoa dela ikus daiteke, hau da, dena CPU bakar batean exekutatuz gero, 10 CPUrekin baino 8,32 aldiz denbora gehiago behar izango lukeela. Horrek esan nahi du paralelizazioa ez dela optimoa izan, hala balitz denborak 10 aldiz handiagoa izan beharko bailuke. Badirudi arazoa dokumentuak makinen artean banatzeko eran dagoela, Kafka sistemak hala behartuta, makina bakoitzak bere sarrerako dokumentuen ilara propioa baitauka. Ondorioz, dokumentuen banaketa prozesaketa hasi aurretik egin behar denez, inoiz ez da optimoa izango, nahiz eta dokumentu kopuru berbera bidali ilara guztietara, dokumentuen tamaina eta beste hainbat parametro ez baitira berdinak izango.

Streaming moduan egindako esperimentua MB nagusi bat eta 5 MB langileko ingurune batean egin dugu. MB bakoitzak 2 CPU dituenez, oraingoan ere paralelizazio maila 10ekoa da, 10 hari egongo baitira aldi berean exekutatzen. 1.000 dokumentu prozesatu ditugu. Egoera erreal bat simulatzearren, dokumentu guztiak batera bidali ordez, Poissonen banaketa erabili dugu, batezbesteko 0.5 dokumentu/min balioarekin. Horri esker, gutxi gorabehera bi minuturo dokumentu bat bidaliko da prozesatzera, baina baliteke denbora-tartea zenbaitetan txikiagoa edo handiagoa izatea ere. Bi minutuko balioa aukeratu dugu, batch-esperimentuetan ikusi baitugu balio horrekin makina ez dela ez gehiegi, ez gutxiegi, kargatuko. Helburua, sistema beti dokumenturen bat edo beste prozesatzen egotea da, baina topologia ez-linealaren araberako moduluen arteko paralelizazioa bermatzeko adina nodo libre egonik. Emaitzak 2. taulan daude ikusgai.

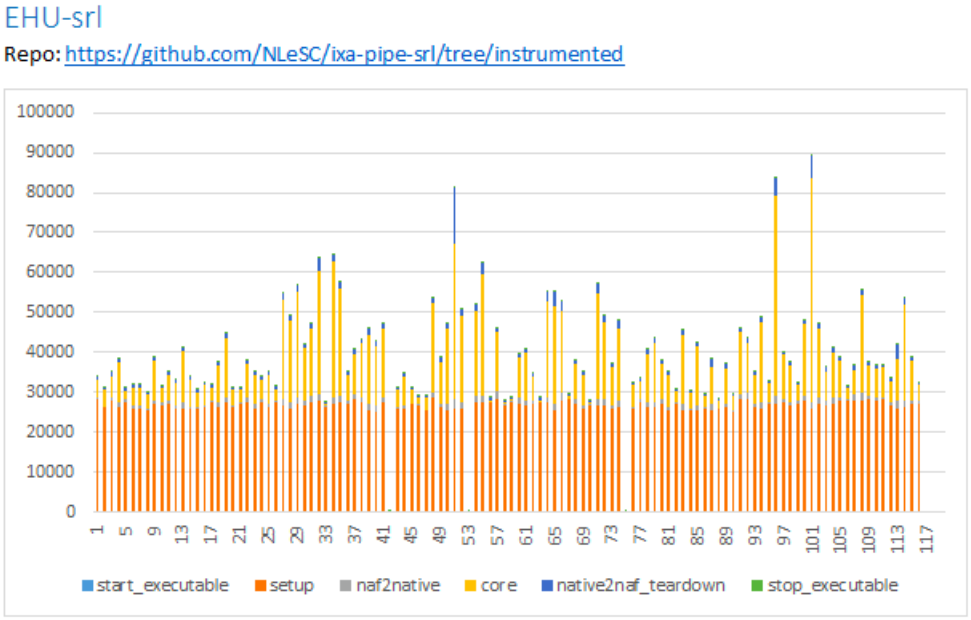
Kasu honetan, topologia ez-linealarekin lortu dugun latentzia, linealarekin lortutakoa baino % 44 txikiagoa da. Hobekuntza handia den arren, topologian 15 fase (modulu bakoitzeko, bat) edukitzetik 6 edukitzera pasatu garenez (5. irudia), handiagoa izan zitekeela pentsa zitekeen hasiera batean. Arazoaren iturria moduluen arteko exekuzio-denboren desberdintasunean dago. SRL moduluak gainontzekoek baino askoz ere denbora gehiago behar duenez prozesatzen, ezin dugu lortu modulu horren prozesatze-denbora baino laburragoa izatea dokumentu osoaren prozesatze-denbora. Latentzia hobetzeko modu bakarra moduluaren prozesaketa zatitzea litzateke, modulu bakoitzak dokumentu osoaren prozesaketa makina bakar batean egin ordez, dokumentua zatitan banatu eta zati bakoitza nodo desberdinetan banatuz. Teknika hau 4. atalean azaldu dugu hobeto.

4 Ondorioak eta etorkizuneko lanak

Lan honetan testu kopuru handiak denbora laburrean eta modu eskalagarrian prozesatzeko sistema bat garatu dugu. Horretarako, teknologia ugari erabili behar izan dugu: Kafka ilarak kudeatzeko, MongoDB datu-basea inplementatzeko, Storm prozesaketa banatua egiteko etab.

Sistemak batch- eta streaming-inguruneetan behar bezala joka dezan, hainbat erronkari egin diegu aurre. Batetik, batch-prozesaketan throughputa igotzea izan dugu helburu, hau da, denbora-unitateko prozesatutako dokumentu kopurua. Streaming-prozesaketan, aldiz, dokumentu bakoitza ahalik eta azkarren prozesatzea izan dugu helburu. Bi kasuetan, sistemak nahiko ondo eskalatzen duela frogatu dugu. Lehenbizikoan, % 15 inguruko galera behatu dugu, Kafkak ez baitigu sistema osoan ilara bakarra ezartzeko aukerarik eskaini, eta, ondorioz, dokumentuen banaketa ez baita optimoa izan clusterreko nodoen artean. Horri aurre egiteko, ilara-sistema aldatzea ikusten dugu irtenbiderik egokiena. Bigarren kasuan, latentziaren hobekuntzan muga bat aurkitu dugu. HP moduluen prozesaketa-denboren artean desoreka handia dago, eta ondorioz, topologia ez-linealei ezin izan diegu nahi bezainbesteko etekinik atera. Hala ere, latentzian irabazten ez dena, throughputean irabazten da, sistema orekatuz.

6 Irudia: SRL moduluaren exekuzio-denborak. Laranjaz, hasieraketa-denbora. Horiz, prozesatze-denbora.



Etorkizunerako bi lan nagusi aurreikusten ditugu. Batetik, sarrerako dokumentuen granularitate desberdinak probatu nahi genituzke, dokumentuen latentzia hobetze aldera. Latentziaren muga-denbora gehien behar duen moduluak markatzen duenez, modulu bakoitzaren prozesaketa ere modu banatuan egitea dugu helburu. Horretarako, dokumentu osoa unitate gisa prozesatu ordez, dokumentua esalditan edo paragrafotan zatituko genuke, zati guztiak nodo desberdinetan prozesatuz. Horrela, zenbat eta konputagailu gehiago eduki sisteman, latentzia proportzionalki murriztuko genuke, sistema eskalagarriago eginez.

Bestalde, moduluen izaera ez da gure sistemara behar bezala egokitzen. Modulu askok denbora gehiago behar dute hasieraketa egiten, dokumentu bat prozesatzen baino. Moduluak prozesaketa bakoitzean hasieratu behar direnez, sistema moduluak hasieratzen dabil etengabe. Hori ekidin liteke moduluek zerbitzu bezala funtzionatuko balute, behin bakarrik hasieratuz eta dokumentu berrien zain geldituz. Beraz, hori egitea da hurrengo lanetako bat. Horrekin denborak asko hobetzea espero dugu; izan ere, oraingoz denboren hobekuntza mugatzen duen moduluak denbora asko behar baitu hasieratzeko (6. irudia).

Erreferentziak

Agerri, Rodrigo, Eneko Agirre, Itziar Aldabe, Begoña Altuna, Zuhaitz Beloki, Egoitz Laparra, Maddalen López De Lacalle, German Rigau, Aitor Soroa, eta Rubén Urizar. 2014a. Newsreader project. *Procesamiento del Lenguaje Natural* 53.155–158.

——, Josu Bermudez, eta German Rigau. 2014b. Ixa pipeline: Efficient and ready to use multilingual

nlp tools. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014)*, 26–31.

Astigarraga, Aitzol, Manex Agirrezabal, Elena Lazkano, Ekaitz Jauregi, eta Basilio Sierra. 2013. Bertsobot: the first minstrel robot. In *Human System Interaction (HSI), 2013 The 6th International Conference on*, 129–136. IEEE.

Dean, Jeffrey, eta Sanjay Ghemawat. 2008. Mapreduce: simplified data processing on large clusters. *Communications of the ACM* 51.107–113.

Fokkens, Antske, Aitor Soroa, Zuhaitz Beloki, Niels Ockeloen, German Rigau, Willem Robert van Hage, eta Piek Vossen. 2014. NAF and GAF: Linking linguistic annotations. In *Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, p. 9.

Goyal, Amit, Hal Daumé III, eta Suresh Venkatasubramanian. 2009. Streaming for large scale nlp: Language modeling. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 512–520. Association for Computational Linguistics.

Mayfield, James, David Alexander, Bonnie J. Dorr, Jason Eisner, Tamer Elsayed, Tim Finin, Clayton Fink, Marjorie Freedman, Nikesh Garera, Paul McNamee, Saif Mohammad, Douglas W. Oard, Christine D. Piatko, Asad B. Sayeed, Zareen Syed, Ralph M. Weischedel, Tan Xu, eta David Yarowsky. 2009. Cross-Document Coreference Resolution: A Key Technology for Learning by Reading. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, 65–70. AAAI.

McCreadie, Richard, Craig Macdonald, Iadh Ounis, Miles Osborne, eta Sasa Petrovic. 2013. Scalable distributed event detection for twitter. In *Proceedings of IEEE International Conference on Big Data*.

Ritter, Alan, Mausam, Oren Etzioni, eta Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, 1104–1112, New York, NY, USA. ACM.

Singh, Sameer, Amarnag Subramanya, Fernando Pereira, eta Andrew McCallum. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Association for Computational Linguistics: Human Language Technologies (ACL HLT)*.

Yu, Wei, eta Junpeng Chen. 2013. The state-of-the-art in web-scale semantic information processing for cloud computing. *arXiv preprint arXiv:1305.4228* .

Zaharia, Matei, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, eta Ion Stoica. 2010. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 10–10.

5 Eskerrak

Lan hau NewsReader (FP7-ICT 2011-9-316404) proiektuaren babesean egin da. Zuhaitz Belokiren lana Euskal Herriko Unibertsitateko Euskararen arloko Errektoreordetzak esleitutako doktoretza egiteko laguntzarekin finantzatua izan da.