TESIS – TE142599

TEMU KEMBALI INFORMASI MENGGUNAKAN ELASTICSEARCH PADA UNSTRUCTURED DATATEXT MULTIDIMENSI

RISTA NOVITASARI
2213206716

DOSEN PEMBIMBING
Mochammad Hariadi, S.T, M.Sc, Ph.D
Dr. I Ketut Eddy Purnama, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TELEMATIKA
KONSENTRASI CHIEF INFORMATION OFFICER
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

TESIS – TE142599

INFORMATION RETRIEVAL USING ELASTICSEARCH ON MULTIDIMENSIONAL UNSTRUCTURED DATATEXT

RISTA NOVITASARI
2213206716

SUPERVISOR
Mochammad Hariadi, S.T, M.Sc, Ph.D
Dr. I Ketut Eddy Purnama, ST., MT.

MASTER PROGRAM
EXPERTISE OF AREA TELEMATICS
CONCENTRATION CHIEF INFORMATION OFFICER
ELECTRICAL ENGINEERING
DEPARTEMENT FACULTY OF INDUSTRIAL TECHNOLOGY
TENTH OF NOPEMBER INSTITUE OF TECHNOLOGY
SURABAYA
2015

Tesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (MT)
di
Institut Teknologi Sepuluh Nopember

Oleh :
Rista Novitasari
NRP. 2213 206 716

Tanggal Ujian : 15 Januari 2015
Periode Wisuda : Maret 2015

Disetujui Oleh :

1. Mochammad Hariadi, S.T, M.Sc, Ph.D (Pembimbing I)
   NIP. 196912091997031002

2. Dr. I Ketut Eddy Purnama, ST., MT. (Pembimbing II)
   NIP. 196907301995121001

3. Dr. Eko Mulyanto Yuniarno, ST., MT. (Penguji)
   NIP. 196806011995121009

4. Dr. Supeno Mardi Susiki Nugroho, ST., MT. (Penguji)
   NIP. 197003131995121001

Direktur Program Pasca Sarjana,

Prof. Dr. Ir. Adi Soeprijanto, MT.
NIP. 196404051990021001

TEMU KEMBALI INFORMASI MENGGUNAKAN ELASTICSEARCH PADA UNSTRUCTURED DATATEXT MULTIDIMENSI

Nama Mahasiswa : Rista Novitasari

NRP : 2213206716

Dosen Pembimbing : 1. Moch. Hariadi, S.T, M.Sc, Ph.D

2. Dr. I Ketut Eddy Purnama, ST., MT.

ABSTRAK

Salah satu bagian penting dalam pengelolaan data karya ilmiah adalah proses pencarian informasi biasa disebut dengan temu kembali informasi (*information retrieval*). Adapun tujuan utama dari temu kembali informasi ini adalah menemukan kembali dokumen karya ilmiah yang berisi informasi yang relevan dengan apa yang diinputkan oleh pengguna pada sistem terdistribusi.

Teknik temu kembali informasi berbasis big data ini menggunakan metode *ElasticSearch* pada lingkungan komputasi terdistribusi yang menggunakan *Hadoop*. Dengan teknik ini suatu dokumen yang diinputkan akan dibuatkan index dari isi dokumen tersebut. Dari percobaan yang dilakukan didapatkan rata-rata waktu search requet yaitu 1.304 detik dan waktu index request yaitu 1.093 detik. Rata-rata waktu pencarian kata pada dokumen sangat singkat. Dari percobaan yang dilakukan didapatkan rata-rata waktu pencarian dokumen yaitu 0.0485 detik.

Kata kunci : *ElasticSearch, Hadoop, MapReduce, Big Data.*

[Halaman ini sengaja dikosongkan]

INFORMATION RETRIEVAL USING ELASTICSEARCH ON MULTIDIMENSIONAL UNSTRUCTURED DATATEXT

Name : Rista Novitasari

NRP : 2213206716

Supervisor : 1. Moch. Hariadi, S.T, M.Sc, Ph.D

2. Dr. I Ketut Eddy Purnama, ST., MT.

ABSTRACT

One important part of the scientific work of data management is the process of information retrieval commonly referred to as information retrieval (information retrieval). The main purpose of information retrieval is to rediscover scientific papers document that contains information that is relevant to what is entered by the user in a distributed system.

Information retrieval techniques based big data using methods ElasticSearch in distributed computing environments that use Hadoop. With this technique an input document will be created index of the contents of the document. Obtained from experiments conducted average search time is 1304 seconds requet and index time the request is 1093 seconds. The average search time is very brief word on the document. Obtained from experiments conducted average search time is 0.0485 seconds document.

Keyword : *ElasticSearch, Hadoop, MapReduce, Big data.*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur dipanjatkan kepada Allah SWT karena ridho dan hidayahnya penulis diberikan kemudahan untuk dapat menyelesaikan Tesis yang berjudul "Temu Kembali Informasi Menggunakan Elasticsearch Pada Unstructured Datatext Multidimensi".

Penulis juga mengucapkan terima kasih yang sedalam-dalamnya kepada Kementrian Kominfo atas bantuan beasiswa yang diberikan dalam pendidikan program magister dan tesis yang dikerjakan.Penulis.

Kepada Bapak Hariadi dan Bapak Ketut sebagai pembimbing selama di Teknik Elektro, terima kasih atas bimbingan dan nasehat-nasehatnya yang diberikanselama ini. Bapak-bapak dan Ibu Dosen Pengajar di Jurusan Teknik Elektro terutama Telematika CIO, terima kasih atas ilmu dan pengalaman yang sudah dibagikan selama kuliah. Bapak-bapak dan Ibu-ibu Staf Administrasi di Sekretariat Jurusan, terima kasih atas semua bantuannya.

Teman-teman kantor CIO 2013, terima kasih atas bantuan dan kebersamaannya selama ini, atas segala bantuan dan kebaikan pertemanannya, sudah bersedia menjadi salah satu bagian dari perjalanan hidup Penulis.

Penulis menyadari bahwa Tesis ini masih banyak kekurangan dan kelemahan. Hal ini karena keterbatasan ilmu dan kemampuan penulis sebagai hamba Allah SWT. Oleh karena itu, penulis sangat mengharapkan masukan dan saran dari pembaca yang bersifat membangun demi melengkapi kekurangan tersebut.

Terakhir penulis memohon maaf sebesar-besarnya apabila terdapat kesalahan dan kekurangan dalam penyusunan tesis ini. Penulis berharap tesis ini bermanfaat bagi pembaca yang ingin mengembangkan teknologi khususnya teknologi Telekomunikasi dan Informatika.

Surabaya, Januari 2015

Penulis

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR PUSTAKA

[1] Divya, Manda Sai. dan Goyal, Shiv Kumar. (2013), "ElasticSearch, An advanced and quick search technique to handle voluminous data". *COMPUSOFT, An international journal of advanced computer technology, 2 (6),* June-2013 (Volume-II, Issue-VI), hal. 171-175

[2] Bhadane, C., Mody, H. A., Shah, D.U. Sheth,P.R.(2014). "Use of Elastic Search for Intelligent Algorithms to Ease the Healthcare Industry". *International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307*, Volume-3, Issue-6, January 2014, hal. 222-225

[3] Hurwits, Judith., Nugent, Alan., Helper, Fern, Dr., Kaufman, Marcia., (2013), *Big Data For Dummies A Wiley Brand*. John Wiley & Sons, Inc. 111 River Street, Hoboken, NJ 07030-5774

[4] http://exploringelasticsearch.com diakses pada 3 Juli 2014

[5] http://www.elasticsearch.org diakses pada 3 Juli 2014

[6] http://www.ibm.com/big-data/us/en/ diakses pada 3 Juli 2014

[7] http://hortonworks.com/hadoop-tutorial/ diakses pada 3 Juli 2014

[8] http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html diakses pada 3 Juli 2014

[9] http://www.teradata.com/Teradata-Aster-SQL-MapReduce/ diakses pada 3 Juli 2014

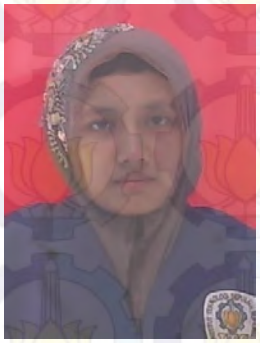
[10] http://hadoop.apache.org diakses pada 3 Juli 2014

[11] Paro, Alberto, (2013), *ElasticSearch Cookbook*, PACKT Publishing, Birmingham.

[12] Kuć, Rafał., Rogozińsk, Marek. (2013), *Mastering ElasticSearch*, PACKT Publishing, Birmingham-Mumbai.

[Halaman ini sengaja dikosongkan]

BIOGRAFI PENULIS



Penulis lahir di Kota Gresik. Penulis bersekolah di SD 3 Semen Gresik, lalu melanjutkan pendidikannya di SMP N 1 Gresik dan pendidikan SMA di SMAN 1 Gresik. Penulis melanjutkan pendidikannya ke jenjang S1 Jurusan Sistem Informasi, ITS Surabaya. Penulis bekerja sebagai PNS di Kabupaten Gresik dari awal tahun 2009 sampai sekarang. Penulis mendapat kesempatan untuk melanjutkan studi S2 dengan Beasiswa CIO dari Kementrian Kominfo pada tahun 2013 di Telematika Jurusan Teknik Elektro ITS, Surabaya. Hingga menyelesaikan pendidikan S2 pada bulan Maret 2015. Dan setelah lulus akan kembali bekerja di lingkungan pemerintah Kabupaten Gresik.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada era teknologi yang serba canggih ini banyak ditawarkan berbagai macam kemudahan. Mulai dari piranti-piranti yang sering digunakan bahkan sampai dengan informasi yang diperlukan semuanya serba lengkap.

Di bidang pendidikan sekarang ini, banyak tersedia buku-buku, artikel maupun hasil karya mahasiswa baik dalam bentuk fisik ataupun elektronik. Hal ini sangat memudahkan para pencari literatur untuk mendapatkan buku maupun literatur. Hanya dengan mengakses internet saja para siswa yang akan mencari literatur bisa mencari literatur yang sesuai dengan kriteria yang diinginkan.

Data literatur baik buku, artikel maupun karya ilmiah yang meningkat dengan pesat ini telah membuat kebutuhan akan pengelolaan data juga semakin meningkat. Begitu juga dengan penyimpanan dokumen tersebut yang membutuhkan alokasi penyimpanan yang begitu besar. Data literatur baik buku, artikel maupun karya ilmiah yang berbentuk dokumen akan sangat sulit dalam proses pencarian apabila bentuk fisik dokumen ditempatkan pada satu tempat yang sama. Hal ini akan mengakibatkan lamanya proses pencarian dokumen.

Salah satu bagian penting dalam pengelolaan data literatur karya ilmiah adalah proses pencarian informasi yang diinginkan oleh pengguna atau biasa disebut dengan temu kembali informasi (*information retrieval*). Proses pencarian informasi ini disesuaikan dengan kriteria data literatur baik buku, artikel maupun karya ilmiah. Adapun tujuan utama dari temu kembali informasi ini adalah menemukan kembali dokumen literatur karya ilmiah yang berisi informasi yang relevan dengan apa yang diinputkan oleh pengguna. Dalam penelitian ini yaitu dokumen literatur berupa karya ilmiah, dalam format word (doc), PDF dan Excel.

Sudah banyak teknik yang diusulkan untuk temu kembali informasi. Namun dari sekian teknik masih menyisakan permasalahan terkait kecepatan dan akurasi pencarian. Pada penelitian yang dilakukan Manda Sai Divya dan Shiv Kumar Goyal tahun 2013, yang berjudul "*ElasticSearch, An advanced and quick search technique to handle voluminous data*" menjelaskan mengenai temu kembali informasi dengan menggunakan ElasticSearch. Penelitian tersebut menujukkan bahwa proses pencarian dokumen berupa data yang ada di database menjadi sangat mudah dalam explorasinya, seperti melakukan pencarian pada *free-text*.

Pada penelitian C. Bhadane, H. A. Mody, D. U. Shah, P. R. Sheth, yang berjudul "*Use of Elastic Search for Intelligent Algorithms to Ease the Healthcare Industry*" (2014), menunjukkan bahwa proses pencarian dokumen dengan menggunakan *ElasticSearch* menjadi sangat mudah dan efisien. Pencarian yang juga berupa data di database Namun ketika ada penambahan data, kebutuhan *memory* yang digunakan juga meningkat.

Pada penelitian ini, penulis mengusulkan sebuah teknik temu kembali informasi berbasis big data menggunakan *ElasticSearch* pada lingkungan komputasi terdistribusi yang menggunakan *Hadoop*. *Hadoop* sendiri adalah *framework software* berbasis *Java* dan *opensource* yang berfungsi untuk mengolah data yang sangat besar secara terdistribusi dan berjalan di atas *cluster* yang terdiri dari beberapa komputer yang saling terhubung.

Dengan diusulkannya penggunaan elasticsearch pada penelitian ini, diharapkan nantinya teknik ini akan meningkatkan kinerja temu kembali informasi berbasis big data dalam hal kecepatan dan akurasi pencarian.

1.2 Perumusan Masalah

Pada penelitian ini pokok permasalahan yang akan dibahas adalah pencarian konten file data tidak terstruktur pada data terdistribusi bukan

sesuatu yang mudah dikarenakan lokasi dari file tersebut tidak terletak pada satu tempat dan isi file dapat berubah sewaktu-waktu.

1.3 Batasan Masalah

Untuk tidak meluasnya pokok pembahasan maka penelitian ini dititik beratkan pada :

1. Dokumen yang dijadikan penelitian adalah dokumen berbentuk File Word(doc), PDF dan Excel(xls).
2. Proses temu kembali informasi bersifat tidak real time.
3. Menggunakan piranti lunak Hadoop dan MapReduce.

1.4 Tujuan dan Manfaat Penelitian

1.4.1 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah meningkatkan kecepatan dan keakuratan dalam proses temu kembali informasi dengan menggunakan metode *ElasticSearch*.

1.4.2 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat untuk peningkatan kecepatan dan keakuratan dan optimasi pencarian dengan menggunakan *ElasticSearch*.

1.5 Sistematika Penulisan

Sistematika Penulisan penelitian ini terdiri atas 5 bab, setiap bab saling berhubungan antarasatu sama lain sesuai dengan urutan permasalahan yang akan dibahas. Garis besar susunan penulisannya adalah sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi latar belakang, perumusan masalah, batasan permasalahan, tujuan dan manfaat penelitian, sistematika pembahasan serta relevansi penelitian ini.

BAB II KAJIAN PUSTAKA DAN DASAR TEORI

Berisi tentang kajian teoritis mengenai konsep dasar Big Data, Framework Hadoop, HDFS, Mapreduce, unstructured data, metode elasticsearch dan metode klasifikasi data. Disamping itu melakukan studi terhadap hasil-hasil penelitian sebelumnya serta literatur pendukung lainnya.

BAB III METODOLOGI PENELITIAN

Membahas tentang proses pengambilan data yang dilakukan (akuisisi data), proses meng-inputkan data ke dalam Hadoop File System, preprocessing yang meliputi pembersihan data dan kata, stemming, stoplist, desain sistem yang dibuat, bagaimana proses pencarian dokumen, proses indexing dari dokumen yang diinputkan, hasil yang diharapkan dan bagaimana cara memvalidasi hasil yang dikeluarkan.

BAB IV HASIL PENELITIAN DAN PEMBAHASAN

Dalam bab ini dijelaskan hasil analisa hasil penelitian dan pembahasan. Analisa yang dibahas mulai dari aspek-aspek yang tercantum dalam metodologi penelitian yaitu akuisisi data. Pada bab ini dijelaskan darimana data diambil, bagaimana proses input dokumen file ke HDFS, preprocessing yang dilakukan dan pengujian pada hasil. Pengujian dilakukan dengan menghitung precision dan recall dari hasil ujicoba untuk file dokumen yang ditemukan

BAB V KESIMPULAN DAN SARAN

Berisikan kesimpulan-kesimpulan yang bisa diambil dari hasil penelitian ini serta saran-saran dan masukan untuk penelitian selanjutnya.

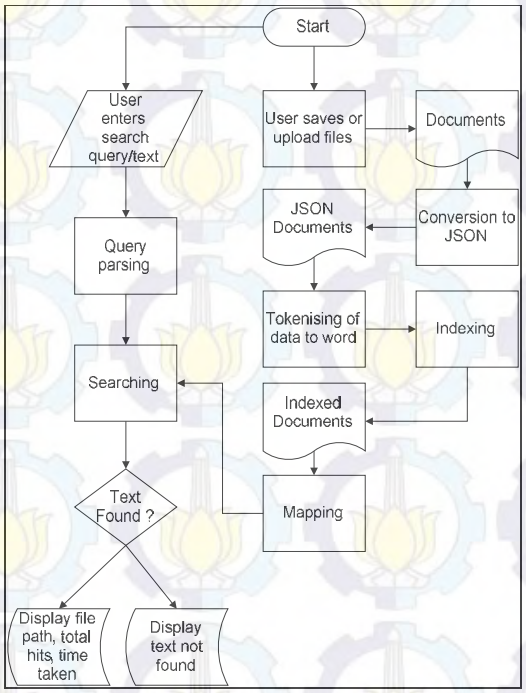
[Halaman ini sengaja dikosongkan]

BAB II
KAJIAN PUSTAKA

Dalam bab ini akan dibahas mengenai teori-teori yang dapat menunjang dan menjadi acuan dalam penelitian yang dilakukan.

2.1 Studi Literatur

Dalam penelitian yang telah dilakukan oleh Manda Sai Divya dan Shiv Kumar Goyal tahun 2013, yang berjudul "*ElasticSearch, An advanced and quick search technique to handle voluminous data*" menjelaskan mengenai temu kembali informasi dengan menggunakan *ElasticSearch*. Pada penelitian yang dilakukan tersebut menujukkan bahwa proses pencarian dokumen baik berupa data yang ada di database menjadi sangat mudah dalam explorasinya, seperti melakukan pencarian pada *free-text*. Alur dasar dari *ElasticSearch* dapat dilihat pada Gambar 2.1. Kelemahan dari penelitian ini pada *memory* komputer yang digunakan.[1]



Gambar 2.1 Alur Dasar ElasticSearch

Pada Gambar 2.1 dijelaskan alur dasar *ElasticSearch* yang merupakan penelitian yang dilakukan Manda Sai Divya dan Shiv Kumar Goyal. Alur dasar *ElasticSearch* dibagi ke dalam dua tahap. Tahapan pertama adalah tahapan dimana pengguna memasukkan data ke dalam sistem. Data yang dimasukkan diubah bentuknya menjadi dokumen. Dalam metode *ElasticSearch* yang dimaksudkan dokumen adalah data pada tabel yang berupa data teks. Dokumen tersebut lalu diubah menjadi format yang telah ditentukan oleh *ElasticSearch* yaitu dalam format *JSON* (*JavaScript Object Notation*) dan menjadi dokumen *JSON*. Dari dokumen ini lalu dipisahkan kalimat-kalimatnya menjadi kata-kata. Kata-kata tersebut yang akan dijadikan *indexing* untuk proses selanjutnya dan didapatkan indeks dari semua dokumen tersebut. Indeks ini yang akan dijadikan acuan dalam penentuan posisi mappingnya.

Tahapan kedua adalah pengguna memasukkan kata kunci yang akan dicari. Pengguna memasukkkan kata kunci yang ingin didapatkan hasilnya, oleh *ElasticSearch* akan dilakukan *query parsing*. Selain itu dilakukan proses pencarian data yang sesuai dengan kata kunci pada *database*. Proses pencarian ini mengambil data dari indeks yang telah dibuat ketika pengguna memasukkan data dan dilakukan proses *mapping* terhadap data tersebut. Apabila didapatkan hasil yang sesuai dengan kata kunci yang dimasukkan, maka akan dihasilkan suatu keluaran data dan lokasi *file*. Apabila tidak didapatkan hasilnya, maka akan didapatkan kalau hasil tidak ditemukan.

Pada penelitian C. Bhadane, H. A. Mody, D. U. Shah, P. R. Sheth, yang berjudul "*Use of Elastic Search for Intelligent Algorithms to Ease the Healthcare Industry*" (2014), menunjukkan bahwa proses pencarian dokumen dengan menggunakan *ElasticSearch* menjadi sangat mudah dan efisien. Namun proses tersebut memerlukan *memory* yang tidak sedikit, dan ketika ada penambahan data, kebutuhan *memory* yang digunakan juga meningkat.[3]

2.2 Big Data

Data menunjuk pada deskripsi dasar akan benda, *event*, aktivitas, dan transaksi yang terdokumentasi, terklasifikasi, dan tersimpan tetapi tidak terorganisasi untuk dapat memberikan suatu arti yang spesifik (R. Kelly Rainer, 2011). Data merupakan hal paling mendasar yang dibutuhkan yang dapat diperoleh dari proses-proses maupun sumber-sumber luar yang akan diolah menurut keinginan.

Big Data adalah istilah yang mencakup segala untuk pengumpulan data set dalam volume yang begitu besar dan kompleks sehingga menjadi sulit untuk mengolahnya menggunakan aplikasi pengolahan data tradisional.

Tantangan dalam *Big* Data meliputi analisis, akurasi, pencarian, berbagi tempat penyimpanan, transfer, visualisasi, dan pelanggaran privasi. Kecenderungan untuk set data yang lebih besar ini disebabkan oleh informasi tambahan diturunkan dari analisis set besar tunggal data terkait, dibandingkan dengan memisahkan set yang lebih kecil dengan jumlah total data yang sama, yang memungkinkan penemuan korelasi untuk tren bisnis dan sebagainya.

Keterbatasan *Big* Data karena set data yang besar di banyak daerah, termasuk meteorologi, genomik, simulasi fisika kompleks, penelitian biologi dan lingkungan, dan *e-Science* pada umumnya. Keterbatasan juga mempengaruhi pencarian, keuangan dan bisnis informatika internet. Data set tumbuh dalam ukuran sebagian karena mereka semakin sering dikumpulkan oleh perangkat informasi *mobile* di mana-mana, teknologi sensorik udara (penginderaan jarak jauh), *log software*, kamera, mikrofon, identifikasi frekuensi radio (RFID) pembaca, dan jaringan sensor nirkabel. Merupakan suatu tantangan bagi perusahaan besar adalah menentukan siapa yang harus memiliki inisiatif *big* data yang mengangkang seluruh organisasi.

Big Data sulit bekerja dengan menggunakan sebagian besar sistem manajemen *database* relasional dan statistik dan visualisasi *dekstop*. *Big* Data membutuhkan software massal paralel yang berjalan pada puluhan,

ratusan, atau bahkan ribuan *server* sebagai gantinya. Apa yang dianggap *big* data bervariasi tergantung pada kemampuan organisasi mengelola set, dan pada kemampuan aplikasi yang secara tradisional digunakan untuk memproses dan menganalisa kumpulan data. *Big* Data adalah target bergerak, apa yang dianggap sebagai *Big* hari ini tidak akan seperti begitu untuk tahun depan. Untuk beberapa organisasi, menghadapi ratusan gigabyte data untuk pertama kalinya dapat memicu kebutuhan untuk mempertimbangkan kembali pilihan manajemen data. Bagi yang lain, mungkin diperlukan waktu puluhan atau ratusan terabyte sebelum ukuran data menjadi pertimbangan yang signifikan.

Big Data didefinisikan sebagai sebuah problem domain di mana teknologi tradisional seperti relational database tidak mampu lagi untuk melayani. Definisi *Big* di sini adalah volume, *velocity* dan variasi datanya. Peningkatan volume, *velocity* dan variasi data banyak diakibatkan oleh adopsi internet. Setiap individu memproduksi konten atau paling tidak meninggalkan sidik jari digital yang berpotensial untuk digunakan untuk hal-hal baru; dari audiens targetting, rekomendasi ataupun penggunaan yang lebih tak terduga seperti *Google Translate* yang menggunakan machine learning di atas *Big Data* yang *Google* punya untuk translasi bahasa.

Untuk menghadapi *volume* yang tinggi, prinsip *Business Intelligence* mengajak kita untuk membersihkan data yang ada. Proses pembersihan ini akan membuang residu yang dianggap tidak penting. Sedangkan prinsip *Big* Data adalah untuk tidak membuang data apapun karena residu tersebut mungkin akan menjadi penting sejalannya waktu.

Sebuah perusahaan IT Gartner mendefinisikan *big* data menggunakan 3 Vs (volume data yang tinggi, *velocity*, dan *variety* informasi). Sejumlah data atau informasi dikatakan big data apabila memenuhi tiga karakteristik yang dijelaskan pada Gambar 2.2. Tiga karakteristik tersebut antara lain :

1. Volume

Jumlah data yang dihasilkan sangat penting dalam konteks ini. Ini adalah ukuran dari data yang menentukan nilai dan potensi data yang

dipertimbangkan dan apakah itu benar-benar dapat dianggap sebagai *Big* Data atau tidak. Nama *Big* Data itu sendiri mengandung istilah yang berkaitan dengan ukuran dan karenanya karakteristik.

Volume data juga terus meningkat dan belum pernah terjadi sampai setinggi ini sehingga tidak dapat diprediksi jumlah pasti dan juga ukuran dari data sekitar lebih kecil dari *petabyte* sampai *zetabyte*. Dataset big data sekitar 1 *terabyte* sampai 1 *petabyte* per perusahaan jadi jika big data digabungkan dalam sebuah organisasi / group perusahaan ukurannya mungkin bisa sampai *zetabyte* dan jika hari ini jumlah data sampai 1000 *zetabyte*, besok pasti akan lebih besar dari 1000 *zetabyte*.

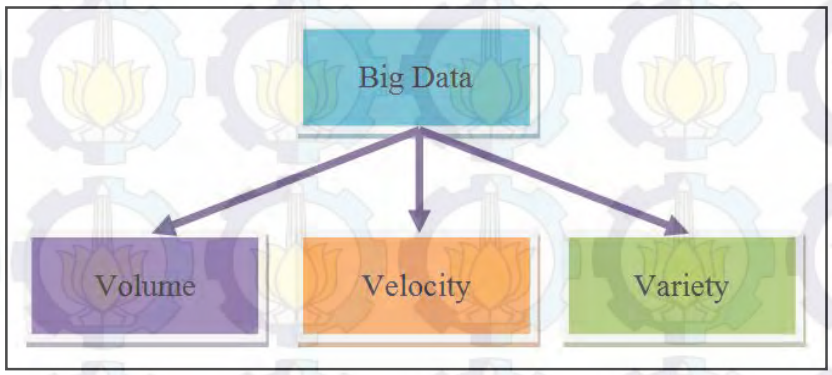
2. ***Velocity***

Aspek ini adalah kecepatan, yang mengacu pada kecepatan generasi data atau seberapa cepat data yang dihasilkan dan diproses untuk memenuhi tuntutan dan tantangan yang terbentang di depan di jalur pertumbuhan dan perkembangan.

3. ***Variety***

Aspek ini adalah keragaman yang berarti bahwa kategori yang dimiliki *Big* Data juga fakta yang sangat penting yang perlu diketahui oleh analis data. Ini membantu orang-orang yang sangat erat menganalisis data dan berkaitan dengan itu, untuk secara efektif menggunakan data tersebut untuk keuntungan mereka dan dengan demikian menegakkan pentingnya *Big* Data.

Data atau informasi yang ada memiliki variasi jenisnya, baik data terstruktur maupun tidak terstruktur. Data terstruktur adalah data yang mudah dianalisa menggunakan database relasional. Sedang data tidak terstruktur tidak bisa diolah menggunakan database relasional. *Big* data didominasi oleh data tidak terstruktur.



Gambar 2.2 Karakteristik 3V Big Data

Sedangkan menurut IBM dan beberapa sumber lainnya, data atau informasi dikatakan big data apabila memenuhi empat karakteristik. IBM menambahkan V keempat (*veracity*) dalam karakteristik big datanya. Pada Gambar 2.3 dijelaskan 4 karakteristik dari big data yang memiliki kesamaan dengan tiga karakteristik sebelumnya dan menambahkan *veracity* untuk karakteristik yang keempat. *Veracity* tinggi didefinisikan untuk memastikan data yang akurat, dan membantu seseorang untuk membuat keputusan bisnis yang lebih baik.

Gambar 2.3 Karakteristik 4 V dari Big Data (IBM)

Empat karakteristik big data menurut IBM dan beberapa sumber lainnya, data atau informasi dikatakan big data apabila memenuhi kriteria, antara lain :

1. **Volume**

   Jumlah data yang dihasilkan sangat penting dalam konteks ini. Ini adalah ukuran dari data yang menentukan nilai dan potensi data yang dipertimbangkan dan apakah itu benar-benar dapat dianggap sebagai *Big* Data atau tidak. Nama *Big* Data itu sendiri mengandung istilah yang berkaitan dengan ukuran dan karenanya karakteristik. Volume data juga terus meningkat dan belum pernah terjadi sampai setinggi ini sehingga tidak dapat diprediksi jumlah pasti dan juga ukuran dari data sekitar lebih kecil dari *petabyte* sampai *zetabyte*. Dataset *big* data sekitar 1 *terabyte* sampai 1 *petabyte* per perusahaan jadi jika *big* data digabungkan dalam sebuah organisasi / group perusahaan ukurannya mungkin bisa sampai *zetabyte* dan jika hari ini jumlah data sampai 1000 *zetabyte*, besok pasti akan lebih besar dari 1000 *zetabyte*.

2. ***Velocity***

   Aspek ini adalah kecepatan, yang mengacu pada kecepatan generasi data atau seberapa cepat data yang dihasilkan dan diproses untuk memenuhi tuntutan dan tantangan yang terbentang di depan di jalur pertumbuhan dan perkembangan.

3. ***Variety***

   Aspek ini adalah keragaman yang berarti bahwa kategori yang dimiliki *Big* Data juga fakta yang sangat penting yang perlu diketahui oleh analis data. Ini membantu orang-orang yang sangat erat menganalisis data dan berkaitan dengan itu, untuk secara efektif menggunakan data tersebut untuk keuntungan mereka dan dengan demikian menegakkan pentingnya *Big* Data. Data atau informasi yang ada memiliki variasi jenisnya, baik data terstruktur maupun tidak terstruktur. Data terstruktur adalah data yang mudah dianalisa menggunakan database relasional.

Sedang data tidak terstruktur tidak bisa diolah menggunakan *database* relasional. *Big* data didominasi oleh data tidak terstruktur.

4. Veracity

Veracity merupakan kualitas dari data yang diambil dapat sangat bervariasi. Akurasi analisis tergantung pada kebenaran dari sumber data.

Manajemen informasi dan kemampuan analisa Big data meliputi :

1. Data Management dan Warehouse

   Keuntungan kinerja database industri terkemuka di beberapa beban kerja sambil menurunkan administrasi, penyimpanan, pengembangan dan biaya server. Pemanfaatan kecepatan ekstrim dengan kemampuan yang dioptimalkan untuk analisis beban kerja seperti analisis yang mendalam, dan manfaat dari sistem beban kerja-optimal yang dapat berdiri dan berjalan dalam beberapa jam.

2. Hadoop Sistem

   Kekuatan Apache Hadoop dibawa ke perusahaan dengan akselerator aplikasi, analisis, visualisasi, perangkat pengembangan, kinerja dan fitur keamanan.

3. Streaming Computing

Secara efisien memberikan proses analitik yang real-time pada data yang terus berubah dalam gerak dan memungkinkan analisis deskriptif dan prediktif untuk mendukung keputusan real-time. Menangkap dan menganalisis semua data, sepanjang waktu, tepat pada waktunya. Dengan aliran komputasi, alokasi penyimpanan yang sedikit, menganalisis lebih banyak dan membuat keputusan yang lebih baik lebih cepat.

4. Manajemen Konten

   Mengaktifkan siklus hidup konten yang komprehensif dan manajemen dokumen dengan jenis pengendalian biaya efektif yang ada dan isi konten baru dengan skala, keamanan dan stabilitas.

5. Informasi Integrasi & Pemerintahan

   Membangun kepercayaan dalam big data dengan kemampuan untuk mengintegrasikan, memahami, mengelola dan mengatur data yang tepat di siklus hidup.

Sedangkan manfaat big data yang nyata dan signifikan, masih ada banyak tantangan. Jadi, organisasi yang berhubungan dengan volume tinggi seperti data menghadapi masalah berikut:

1. Akuisisi Data

   Ada banyak data mentah yang akan dihasilkan dari berbagai sumber data. Tantangannya adalah untuk menyaring dan mengkompres data, dan ekstrak Informasi dari itu setelah dibersihkan.

2. Penyimpanan Informasi dan Organisasi

   Setelah informasi yang ditangkap keluar dari data mentah, data model akan dibuat dan disimpan dalam perangkat penyimpanan. Untuk menyimpan dataset besar secara efektif, sistem relasional tradisional berhenti menjadi efektif skala yang tinggi. Telah ada generasi baru yang disebut database NOSQLdatabases, yang terutama digunakan untuk bekerja dengan data yang besar. NoSQL database adalah database non-relasional.

3. Pencarian dan Analisa Informasi

   Menyimpan data merupakan bagian dari membangun datawarehouse. Data ini berguna hanya jika dihitung. Big data adalah sering berisik, dinamis, dan heterogen. Informasi ini dicari, di-mining, dan dianalisis untuk pemodelan perilaku.

4. Keamanan dan privasi data

   Saat membawa data terkait dari beberapa sumber, organisasi perlu khawatir tentang keamanan data dan privasi data tersebut.

Big data menawarkan banyak tantangan teknologi untuk teknologi saat ini digunakan saat ini. Hal ini membutuhkan jumlah besar pengolahan

data dalam jangka waktu yang terbatas, yang membawa teknologi seperti massively pemrosesan paralel (MPP) teknologi dan sistem berkas terdistribusi.

2.3 Hadoop

Hadoop diciptakan oleh Doug Cutting dan Mike Cafarella pada tahun 2005. Cutting, yang bekerja di *Yahoo*! pada saat itu, bernama setelah anaknya mainan gajah. Ini pada awalnya dikembangkan untuk mendukung distribusi untuk proyek mesin pencari Nutch.

Apache Hadoop menawarkan fitur *MapReduce* yang memungkinkan kita melakukan prinsip-prinsip yang disebut di atas. *Hadoop* banyak dipakai oleh perusahaan web dan *Startup* yang kita kenal sekarang seperti Yahoo, Facebook, Foursquare dsb.

Begitu pula di sisi enterprise, vendor-vendor solusi enterprise merangkul Hadoop untuk mengatasi masalah Big Data di dalam enterprise. Microsoft (Windows Azure Hadoop), Oracle (Big Data Appliance yang mencakup solusi Hadoop dari Cloudera), SAP (Hana), EMC (GreenPlum Hadoop) adalah beberapa contoh solusi di space ini.

Apache Hadoop adalah framework open-source software untuk penyimpanan terdistribusi dan pemrosesan terdistribusi Data Big pada kelompok perangkat keras komoditas. Hal ini Hadoop Distributed File System (HDFS) membagi file ke dalam blok besar (64MB standar atau 128MB) dan mendistribusikan blok antara node di cluster. Untuk pengolahan data, Hadoop Peta / Mengurangi kode kapal (khusus Jar file) ke kelenjar yang memiliki data yang dibutuhkan, dan node kemudian memproses data secara paralel. Pendekatan ini mengambil keuntungan dari data yang lokalitas berbeda dengan arsitektur HPC konvensional yang biasanya bergantung pada sistem file paralel (menghitung dan data terpisah, namun terhubung dengan jaringan kecepatan tinggi).

Sejak 2012, istilah "Hadoop" sering merujuk tidak hanya paket Hadoop dasar melainkan untuk Ekosistem Hadoop, yang mencakup semua

paket perangkat lunak tambahan yang dapat diinstal di atas atau disamping Hadoop, seperti Apache Pig, Apache Hive, Apache HBase, Apache Spark, dan lain-lain.

Framework Apache Hadoop terdiri dari modul-modul berikut:

1. Common Hadoop

   Berisi library dan utilitas yang diperlukan oleh modul lain Hadoop.

2. Hadoop Distributed File System (HDFS)

   File sistem terdistribusi yang menyimpan data pada mesin komoditas, menyediakan bandwidth yang agregat sangat tinggi di cluster.

3. Hadoop YARN

   Platform sumber daya manajemen yang bertanggung jawab untuk mengelola sumber daya komputasi dalam kelompok dan menggunakan mereka untuk penjadwalan aplikasi pengguna.

4. Hadoop MapReduce

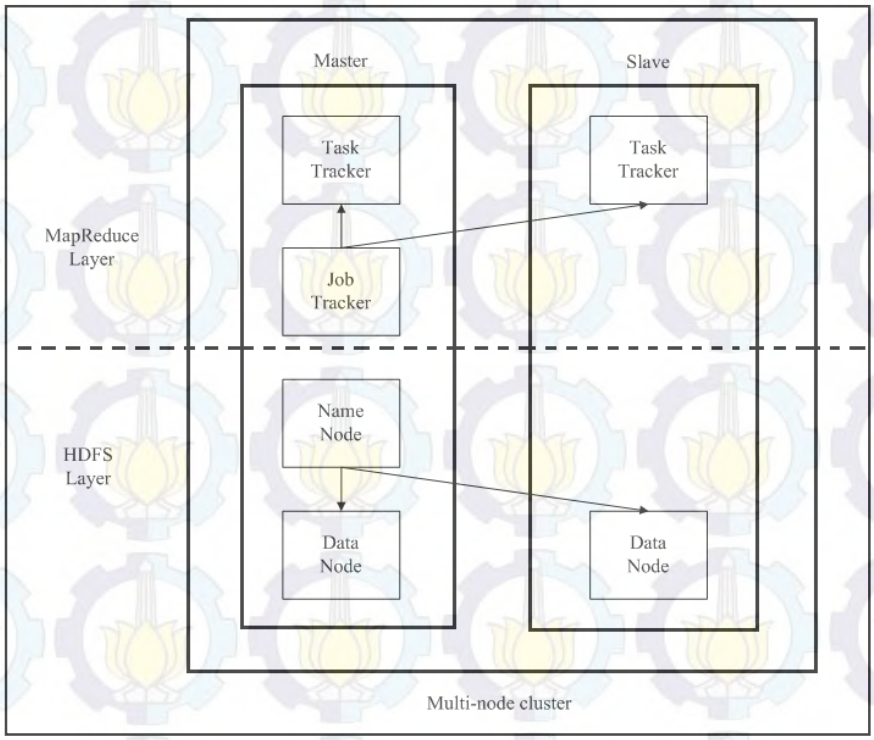
   Model pemrograman untuk pengolahan big data.

Semua modul di Hadoop dirancang dengan asumsi bahwa kegagalan hardware (mesin individu, atau rak mesin) yang umum dan dengan demikian harus ditangani secara otomatis dalam perangkat lunak oleh framework. MapReduce dan HDFS komponen Apache Hadoop yang awalnya masing-masing berasal dari Google Sistem MapReduce dan Google File (GFS).

YARN singkatan dari "Yet Another Resource Negotiator" dan ditambahkan kemudian sebagai bagian dari Hadoop 2.0. YARN memiliki kemampuan manajemen sumber daya yang ada di MapReduce dan packages sehingga mereka dapat digunakan oleh engine baru. Hal ini juga arus MapReduce untuk melakukan apa yang terbaik, memproses data. Dengan YARN, kini Anda dapat menjalankan beberapa aplikasi dalam Hadoop, semua berbagi pengelolaan sumber daya. Hingga September 2014, YARN mengelola hanya CPU (jumlah core) dan memori, tetapi manajemen sumber daya lain seperti disk, jaringan dan GPU direncanakan untuk masa depan.

Untuk end user, meskipun kode Java umum MapReduce, bahasa pemrograman dapat digunakan dengan Hadoop Streaming untuk menerapkan "Map" dan "Reduce" bagian dari program pengguna. [10] Apache Pig, Apache Hive, Apache Spark antara proyek-proyek terkait lainnya mengekspos antarmuka pengguna tingkat tinggi seperti Pig Latin dan varian SQL masing-masing. Kerangka Hadoop itu sendiri sebagian besar ditulis dalam bahasa pemrograman Java, dengan beberapa kode asli di C dan baris perintah utilitas ditulis sebagai shell-script.

Apache Hadoop adalah merek terdaftar dari Apache Software Foundation. Arsitektur Hadoop dapat dilihat pada Gambar 2.4.



Gambar 2.4 Arsitektur Hadoop

Pada Gambar 2.4 dijelaskan arsitektur hadoop, dimana terdapat dua layer yaitu MapReduce Layerdan HDFS Layer, :

1. MapReduce Layer

Pada MapReduce Layer terdapat arsitektur master dan slave. Dalam memproses data, secara garis besar MapReduce dapat dibagi dalam dua

proses yaitu proses Map dan proses Reduce. Kedua jenis proses ini didistribusikan atau dibagi-bagikan ke setiap komputer dalam suatu cluster (kelompok komputer yang salih terhubung) dan berjalan secara paralel tanpa saling bergantung satu dengan yang lainnya. Proses Map bertugas untuk mengumpulkan informasi dari potongan-potongan data yang terdistribusi dalam tiap komputer dalam cluster. Hasilnya diserahkan kepada proses Reduce untuk diproses lebih lanjut. Hasil proses Reduce merupakan hasil akhir yang dikirim ke pengguna.

Pada MapReduce Layer terdapat Task Tracker dan Job Tracker. Task Tracker digunakan untuk memonitoring daftar pekerjaan yang masuk. Sedangkan Job Tracker digunakan untuk memonitoring response dari seluruh pekerjaan tersebut.

2. HDFS Layer

HDFS memiliki arsitektur master / slave. Cluster HDFS terdiri dari NameNode tunggal, server master yang mengelola sistem file namespace dan mengatur akses ke file oleh klien. Selain itu, ada sejumlah DataNodes, biasanya satu per node di cluster, yang mengelola penyimpanan melekat pada node yang mereka berjalan di. HDFS memperlihatkan sistem file namespace dan memungkinkan data pengguna disimpan dalam file. Secara internal, file dibagi menjadi satu atau lebih blok dan blok ini disimpan dalam satu set DataNodes. NameNode mengeksekusi berkas sistem operasi namespace seperti membuka, menutup, dan mengganti nama file dan direktori. Hal ini juga menentukan pemetaan blok untuk DataNodes. Para DataNodes bertanggung jawab untuk melayani membaca dan menulis permintaan dari klien file sistem. Para DataNodes juga melakukan pembuatan blok, penghapusan, dan replikasi atas instruksi dari NameNode tersebut.

NameNode dan DataNode adalah bagian dari HDFS. HDFS ini biasanya berjalan pada sistem operasi GNU / Linux (OS). HDFS dibangun dengan menggunakan bahasa Java, setiap mesin yang mendukung Java dapat menjalankan NameNode atau DataNode.

Penggunaan bahasa Java yang sangat portabel berarti bahwa HDFS dapat digunakan pada berbagai mesin. Sebuah penyebaran khas memiliki mesin khusus yang berjalan hanya perangkat lunak NameNode. Setiap mesin lain dalam cluster berjalan satu contoh dari perangkat lunak DataNode. Arsitektur tidak menghalangi menjalankan beberapa DataNodes pada mesin yang sama tetapi dalam penyebaran nyata yang jarang terjadi.

Keberadaan NameNode tunggal dalam sebuah cluster sangat menyederhanakan arsitektur sistem. NameNode adalah arbiter dan repositori untuk semua metadata HDFS. Sistem ini dirancang sedemikian rupa sehingga data pengguna tidak pernah mengalir melalui NameNode tersebut.

Hadoop terdiri dari Hadoop common package, yang menyediakan filesystem dan tingkat abstraksi OS, MapReduce (baik MapReduce / MR1 atau YARN/ MR2) dan Hadoop Distributed File System (HDFS). Hadoop common Package berisi Java Archive (JAR) file yang diperlukan dan skrip yang dibutuhkan untuk memulai Hadoop. Paket ini juga menyediakan kode sumber, dokumentasi, dan bagian kontribusi yang mencakup proyek-proyek dari Hadoop Community.

Untuk penjadwalan yang efektif kerja, setiap sistem file Hadoop-kompatibel harus memberikan kesadaran lokasi: nama rak (lebih tepatnya, dari switch jaringan) di mana node pekerja adalah. Aplikasi Hadoop dapat menggunakan informasi ini untuk menjalankan pekerjaan pada node mana data tersebut, dan, gagal itu, di rak yang sama / switch, mengurangi lalu lintas backbone. HDFS menggunakan metode ini ketika replikasi data yang mencoba untuk menyimpan salinan yang berbeda dari data pada rak yang berbeda. Tujuannya adalah untuk mengurangi dampak dari kegagalan daya rak pemadaman atau switch, sehingga bahkan jika peristiwa ini terjadi, data masih dapat dibaca.

Sebuah cluster Hadoop kecil termasuk penguasa tunggal dan beberapa node pekerja. Node master terdiri dari JobTracker, TaskTracker, NameNode dan DataNode. Seorang budak atau simpul pekerja bertindak baik sebagai DataNode dan TaskTracker, meskipun mungkin untuk memiliki node pekerja data saja dan menghitung-satunya node pekerja. Ini biasanya digunakan hanya dalam aplikasi tidak standar. Hadoop memerlukan Java Runtime Environment (JRE) 1.6 atau lebih tinggi. Startup dan shutdown script standar mengharuskan Secure Shell (ssh) dibentuk antara node di cluster.

Dalam cluster yang lebih besar, yang HDFS dikelola melalui server NameNode didedikasikan untuk tuan rumah indeks sistem file, dan NameNode sekunder yang dapat menghasilkan snapshot dari struktur memori namenode itu, sehingga mencegah file sistem korupsi dan mengurangi hilangnya data. Demikian pula, server JobTracker mandiri dapat mengelola penjadwalan job. Dalam kelompok di mana mesin Hadoop MapReduce dikerahkan terhadap sistem file alternatif, NameNode, NameNode sekunder, dan arsitektur DataNode dari HDFS digantikan oleh setara file sistem khusus.

Hadoop bekerja secara langsung dengan sistem file terdistribusi yang dapat dipasang oleh sistem operasi yang mendasari hanya dengan menggunakan file: // URL; Namun, ini datang pada harga: hilangnya lokalitas. Untuk mengurangi lalu lintas jaringan, Hadoop perlu tahu mana server yang paling dekat dengan data; ini adalah informasi yang Hadoop-spesifik jembatan sistem file dapat menyediakan.

Pada bulan Mei 2011, daftar sistem file yang didukung dibundel dengan Apache Hadoop adalah :

1. HDFS: Hadoop's own rack-aware file system ini dirancang dalam skala untuk puluhan petabyte storage dan berjalan di atas sistem file dari sistem operasi yang mendasari.
2. Sistem FTP File yang menyimpan semua data pada server FTP diakses dari jarak jauh.

3. Sistem file Amazon S3. Ini ditargetkan pada kelompok host di Amazon Elastic Compute Cloud infrastruktur server-on-demand. Tidak ada rak-kesadaran dalam sistem file ini, karena semua jauh.
4. Windows Azure Storage gumpalan (WASB) sistem file. WASB, perpanjangan di atas HDFS, memungkinkan distribusi Hadoop untuk mengakses data di toko-toko gumpalan Azure tanpa memindahkan data secara permanen ke dalam cluster.

Sejumlah pihak ketiga sistem file jembatan juga telah ditulis, tidak ada yang saat ini dalam distribusi Hadoop. Namun, beberapa distribusi komersial Hadoop dengan filesystem alternatif sebagai default, terutama IBM dan MapR.

1. Pada tahun 2009 IBM membahas menjalankan Hadoop melalui IBM Umum Paralel File System. Kode sumber diterbitkan pada bulan Oktober 2009.
2. Pada bulan April 2010, ParaScale menerbitkan kode sumber untuk menjalankan Hadoop terhadap sistem file ParaScale.
3. Pada bulan April 2010, Appistry merilis sebuah driver sistem file Hadoop untuk digunakan dengan produk CloudIQ Storage sendiri.
4. Pada bulan Juni 2010, HP membahas location aware IBRIX Fusion sistem file.
5. Pada bulan Mei 2011, MapR Technologies, Inc mengumumkan ketersediaan sistem file alternatif untuk Hadoop, yang menggantikan sistem file HDFS dengan penuh random-access baca / tulis sistem file.

2.4 HDFS

Hadoop distributed file system (HDFS) adalah didistribusikan, terukur, dan portabel file sistem yang ditulis pada bahasa Java untuk kerangka Hadoop. Sebuah cluster Hadoop memiliki nominal yang namenode tunggal ditambah sekelompok datanodes, meskipun pilihan redundansi yang tersedia untuk namenode karena kekritisan nya. Setiap

datanode menyajikan blok data melalui jaringan menggunakan protokol blok khusus untuk HDFS. Sistem file menggunakan TCP / IP socket untuk komunikasi. Klien menggunakan panggilan prosedur jauh (RPC) untuk berkomunikasi antara satu sama lain.

HDFS menyimpan file besar (biasanya dalam kisaran gigabyte untuk terabyte) di beberapa mesin. Ini mencapai keandalan dengan mereplikasi data di beberapa host, dan karenanya secara teoritis tidak memerlukan penyimpanan RAID pada host (tapi untuk meningkatkan I/O kinerja beberapa konfigurasi RAID masih berguna). Dengan nilai replikasi default, data disimpan pada tiga node: dua di rak yang sama, dan satu di rak yang berbeda. Node data dapat berbicara satu sama lain untuk menyeimbangkan data, untuk memindahkan salinan sekitar, dan untuk menjaga replikasi data yang tinggi. HDFS tidak sepenuhnya POSIX-compliant, karena persyaratan untuk POSIX file sistem berbeda dari target sasaran untuk aplikasi Hadoop. Tradeoff tidak memiliki file sistem sepenuhnya POSIX-compliant meningkat kinerja untuk throughput data dan dukungan untuk operasi non-POSIX seperti Append.

HDFS memiliki kemampuan ketersediaan tinggi, seperti yang diumumkan untuk rilis 2.0 Mei 2012, membiarkan server metadata utama (NameNode) gagal atas secara manual untuk cadangan. Proyek ini juga telah mulai mengembangkan otomatis gagal-over.

File HDFS sistem termasuk namenode sekunder disebut, nama menyesatkan bahwa beberapa mungkin salah diinterpretasikan sebagai namenode cadangan untuk saat namenode utama pergi offline. Bahkan, namenode sekunder teratur menghubungkan dengan namenode primer dan membangun snapshot dari informasi direktori namenode utama, yang sistem kemudian menyimpan ke direktori lokal atau jarak jauh. Gambar-gambar checkpointed dapat digunakan untuk me-restart namenode utama gagal tanpa harus memutar ulang seluruh jurnal tindakan file sistem, maka untuk mengedit log untuk membuat struktur direktori up-to-date. Karena namenode adalah titik tunggal untuk penyimpanan dan pengelolaan

metadata, dapat menjadi hambatan untuk mendukung sejumlah besar file, terutama sejumlah besar file kecil. HDFS Federation, tambahan baru, bertujuan untuk mengatasi masalah ini sampai batas tertentu dengan memungkinkan beberapa ruang nama dilayani oleh namenodes terpisah.

Keuntungan menggunakan HDFS adalah data awareness antara job tracker dan task tracker. Job tracker pekerjaan map atau mengurangi pekerjaan untuk tugas pelacak dengan location awareness. Sebagai contoh: jika node A berisi data (x, y, z) dan node B berisi data (a, b, c), pekerjaan jadwal tracker node B untuk melakukan peta atau mengurangi tugas pada (a, b, c) dan simpul Sebuah akan dijadwalkan tampil peta atau mengurangi tugas pada (x, y, z). Hal ini akan mengurangi jumlah lalu lintas yang berjalan di atas jaringan dan mencegah transfer data yang tidak perlu. Ketika Hadoop digunakan dengan sistem file lain, keuntungan ini tidak selalu tersedia. Hal ini dapat memiliki dampak yang signifikan pada waktu pekerjaan selesai, yang telah dibuktikan ketika menjalankan pekerjaan data-intensif.

HDFS dirancang untuk file sebagian besar berubah dan mungkin tidak cocok untuk sistem yang membutuhkan bersamaan write-operasi. HDFS dapat dipasang langsung dengan Filesystem di Userspace (FUSE) sistem berkas virtual pada Linux dan beberapa sistem Unix lainnya.

Akses file dapat dicapai melalui asli Java API, Pasar Murah API untuk menghasilkan klien dalam bahasa pilih pengguna (C ++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C #, Cocoa, Smalltalk, dan OCaml), antarmuka baris perintah, melihat-lihat melalui webapp HDFS-UI melalui HTTP, atau melalui klien jaringan pihak ke-3 perpustakaan.

2.5 MapReduce

MapReduce merupakan gabungan dari kata *Map* dan *Reduce*. *Map* adalah proses yang terjadi ketika *masternode* menerima input, kemudian input tersebut dipecah menjadi beberapa sub problem yang kemudian didistribusikan ke *worker nodes*. *Worker nodes* ini akan memproses sub

problem yang diterimanya untuk kemudian apabila problem tersebut sudah diselesaikan, maka akan dikembalikan ke *masternode*. Sedangkan *Reduce* adalah ketika *Masternode* menerima jawaban dari semua subproblem dari banyak data *nodes*, menggabungkan jawaban-jawaban tersebut menjadi satu jawaban besar untuk mendapatkan penyelesaian dari permasalahan utama.

2.6 Unstructured Data

Unstructured Data atau bisa disebut data tidak terstruktur (atau informasi yang tidak terstruktur) mengacu pada informasi yang tidak memiliki model data yang telah ditentukan atau tidak terorganisir dengan cara yang telah ditentukan. Unstructured data biasanya berupa dokumen teks, tapi mungkin berisi data seperti tanggal, angka, dan juga fakta. Hal ini menyebabkan penyimpangan dan ambiguitas yang membuat sulit untuk memahami menggunakan program komputer tradisional dibandingkan dengan data yang tersimpan dalam bentuk database atau dijelaskan (semantik tag) dalam dokumen.

Pada tahun 1998, Merrill Lynch mengutip aturan praktis bahwa di suatu tempat sekitar 80-90% dari semua informasi bisnis yang berpotensi digunakan dapat berasal dalam bentuk yang tidak terstruktur. Aturan ini tidak didasarkan pada primer atau penelitian kuantitatif, tapi tetap dapat diterima oleh beberapa.

IDC dan EMC proyek menyatakan bahwa data akan tumbuh 40 zettabytes pada tahun 2020, sehingga pertumbuhan 50 kali lipat dari awal tahun 2010. Sedangkan komputer Dunia menyatakan bahwa data tidak terstruktur mungkin mencapai lebih dari 70% -80% dari semua data dalam suatu organisasi.

Teknik seperti data mining, Natural Language Processing (NLP), analisis teks, dan analisis bising-teks memberikan metode yang berbeda untuk menemukan pola dalam, atau menafsirkan, informasi ini. Teknik umum untuk teks penataan biasanya melibatkan penandaan manual dengan metadata atau bagian-of-speech tagging untuk teks lanjut penataan berbasis

pertambangan. Manajemen Informasi Unstructured Arsitektur (UIMA) memberikan kerangka umum untuk memproses informasi ini untuk mengekstrak makna dan membuat data terstruktur tentang informasi tersebut.

Software yang menciptakan struktur mesin-processable memanfaatkan linguistik, pendengaran, dan struktur visual yang melekat dalam semua bentuk komunikasi manusia. Algoritma dapat menyimpulkan struktur yang melekat ini dari teks, misalnya, dengan memeriksa morfologi kata, sintaksis kalimat, dan lainnya kecil - dan pola skala besar. Informasi Unstructured kemudian dapat diperkaya dan menandai untuk mengatasi ambiguitas dan teknik berbasis relevansi kemudian digunakan untuk memudahkan pencarian dan penemuan. Contoh "data tidak terstruktur" dapat mencakup buku, jurnal, dokumen, metadata, catatan kesehatan, audio, video, data analog, gambar, file, dan teks terstruktur seperti tubuh pesan e-mail, halaman Web, atau kata-dokumen prosesor. Sementara konten utama yang disampaikan tidak memiliki struktur yang ditetapkan, umumnya dikemas dalam objek (misalnya dalam file atau dokumen) bahwa mereka memiliki struktur dan dengan demikian campuran data terstruktur dan tidak terstruktur, namun secara kolektif ini masih disebut sebagai "data tidak terstruktur". Sebagai contoh, sebuah halaman web HTML ditandai, tapi mark-up HTML biasanya berfungsi semata-mata untuk rendering. Tidak menangkap makna atau fungsi elemen ditandai dengan cara yang mendukung proses otomatis dari isi informasi halaman. XHTML tagging tidak memungkinkan mesin pengolahan elemen, meskipun biasanya tidak menangkap atau menyampaikan makna semantik istilah tag.

Karena data tidak terstruktur biasanya terjadi pada dokumen elektronik, penggunaan konten atau manajemen dokumen sistem yang dapat mengkategorikan seluruh dokumen sering lebih dipilih daripada transfer data dan manipulasi dari dalam dokumen. Manajemen dokumen sehingga menyediakan sarana untuk menyampaikan struktur ke koleksi dokumen.

Search engine telah menjadi alat populer untuk mengindeks dan mencari melalui data tersebut, khususnya teks.

2.7 ElasticSearch

Elasticsearch merupakan sebuah *tool* yang digunakan untuk query kata-kata. Elasticsearch dapat melakukan beberapa tugas yang lainnya, tetapi itu digunakan untuk mencari teks, mengembalikan hasil teks yang dicari menjadi query atau analisis statistik dari kata-kata.[11]

Elasticsearch merupakan *standalone database server*, dituliskan dengan bahasa *Java*, yang mengambil dan menyimpan data dalam format canggih yang dioptimalkan untuk pencarian berdasarkan teks. Dengan menggunakan *elasticsearch* bekerja menjadi lebih mudah diimplementasikan ke dalam *HTTP/JSON*. *Elasticsearch* mudah terukur, mendukung *clustering* dan menjadi pilihan yang *out of the box*.

ElasticSearch dapat digunakan baik sebagai mesin pencari dan sebagai menyimpan data. Sebuah deskripsi singkat dari logika ElasticSearch membantu pengguna untuk meningkatkan kinerja dan kualitas, dan memutuskan kapan dan bagaimana berinvestasi di bidang infrastruktur untuk meningkatkan skalabilitas dan ketersediaan. beberapa rincian tentang ulangan data dan proses komunikasi dasar simpul juga dijelaskan. pada akhir bab ini protokol yang digunakan untuk mengelola ElasticSearch juga dibahas.

Konsep dan fitur-fitur dasar tentang ElasticSearch [12] :

1. **Index**

   ElasticSearch menyimpan data ke dalam satu index atau lebih. Secara analogi dari dunia SQL, index merupakan sesuatu yang menyerupai dengan database. Index ini digunakan untuk menyimpan data dan membacanya dari database ketika diperlukan untuk dibaca. ElasticSearch menggunakan beberapa kode program untuk menulis dan membaca data yang telah disimpan pada index. Index pada ElasticSearch dapat dibangun dengan menggunakan shard dan replica.

2. **Document**

   Dokumen merupakan entiti yang paling utama dari ElasticSearch. Seluruh kegunaan dari ElasticSearch dapat dilihat pada saat proses pencarian atau searching dokumen. Yang dimaksud dengan dokumen yang ada pada ElasticSearch adalah yang terdiri dari beberapa field dan setiap field memiliki nama dan memiliki satu nilai atau lebih. Dalam hal ini field juga bisa disebut dengan multi-valued. Setiap dokumen memiliki kumpulan field yang berbeda-beda, tidak ada schema atau struktur dari database. Dokumen pada ElasticSearch merupakan objek yang memiliki format JSON. Dokumen bisa juga disebut sebagai kolom beserta nilainya (data).

3. **Mapping**

   Sebelum sebuah dokumen disimpan, dokumen terlebih dahulu dianalisa. Pengguna dapat mengkonfigurasi bagaimana teks input dibagi ke dalam token, dimana token tersebut harus disaring atau melalui proses tambahan terlebih dahulu, seperti menghapus tag HTML yang tidak diperlukan. Pada ElasticSearch juga terdapat fungsi sorting (pengurutan) informasi sesuai dengan konten pada field. Pengguna dapat melakukan konfigurasi mapping tersebut untuk menghindari hal-hal yang tidak diinginkan. Ketika mapping ini terjadi proses pendefinisian tipedata dan beberapa atribut ElasticSearch seperti not-analyzed, dateFormat dan lain lain.

4. **Type**

   Setiap dokumen memiliki tipe yang telah didefinisikan. Hal ini mengijinkan bahwa pengguna dapat menyimpan dokumen dengan berbagai macam tipe ke dalam satu index dan juga memiliki mapping yang berbeda-beda untuk setiap tipe dokumen yang berbeda pula. Tipe ini akan secara otomatis dikenali ketika proses indexing.

5. **Node**

   Node merupakan instance dari ElasticSearch. Satu cluster mengandung beberapa node ElasticSearch yang sama-sama berbagi data dan beban

kerja. Satu master node ElasticSearch diperlukan di dalam cluster untuk manajemen node yang lain seperti penambahan dan penghapusan node. Perlu diingat bahwa komputer klien bisa mengakses node ElasticSearch manapun dalam cluster. Dalam istilah orang awam, node bisa juga disebut dengan satu komputer yang digunakan untuk memproses pencarian dengan menggunakan metode ElasticSearch. Apabila node yang digunakan lebih dari satu, maka penamaan node berbeda-beda antara satu dengan yang lainnya. Kebanyakan penamaan node menggunakan urutan angka

6. **Cluster**

   Cluster merupakan sekumpulan node yang bekerja sama untuk mengendalikan data yang lebih besar daripada hanya dengan menggunakan satu node. Cluster merupalakn sebuah solusi dimana kerja dari aplikasi dibagi menjadi beberapa bagian. Atau dengan kata lain, Cluster merupakan komputer server dan memiliki satu atau lebih node (client). Kerja dari aplikasi dibagi ke dalam beberapa node yang digunakan, dimana jika ada satu atau beberapa node yang tidak dapat digunakan (dalam proses upgrade) maka cluster akan mengendalikannya dan mendistribusikan kerja dari aplikasi ke dalam node-node yang masih bisa digunakan.

7. **Shard**

   ElasticSearch membagi data ke dalam beberapa index fisik yang ada. Index fisik itu disebut dengan Shard. Dapat dikatakan shard merupakan bagian-bagian potongan index. Untuk proses pembagian data index menjadi beberapa potongan index disebut dengan sharding. Pada ElasticSearch sharding ini dapat dilakukan secara otomatis dan pengguna tidak dapat mengetahui shard dari masing-masing index, tetapi hanya dapat melihat index dalam skala besar. Jumlah shard dari index harus ditentukan terlebih dahulu sebelum membuat index dan tidak dapat diubah setelahnya atau ketika digunakan. Shard dibagi menjadi dua macam :

a. **Shard Primer**

Merupakan shard utama yang digunakan untuk membagi index. Shard primer ini sangat terbatas ukurannya dengan disebabkan oleh perangkat keras yang digunakan, kompleksitas dokumen dan lain-lain.

b. **Shard Replika**

Merupakan replika atau salinan dari shard primer. Isi dan konten data yang ada pada shard replika sama persis dengan isi dan konten yang ada pada shard primer.

8. **Replica**

Pembagian data memungkinkan pengguna untuk memasukkan data yang lebih banyak ke dalam ElasticSearch untuk satu node yang menanganinya. Replika membantu node untuk mengurangi beban data yang meningkat dan ketika node tunggal tidak mampu menanganinya. Dapat dikatakan replika merupakan membuat salinan atau tiruan dari shard. Jika komputer yang mana terdapat shard dan server tersebut mengalami kerusakan, maka replika dapat digunakan dan tidak ada data yang hilang. Replika dapat ditambahkan dan dihapus kapan saja.

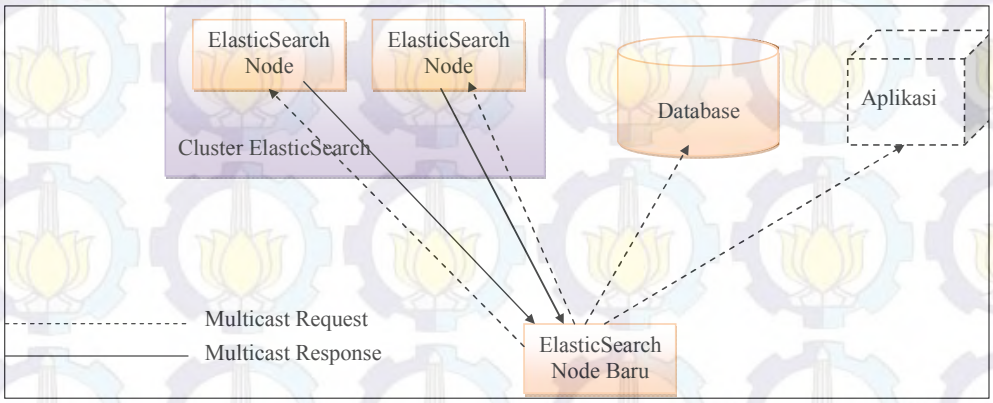
Dilihat dari segi arsitekturnya, fitur-fitur utama ElasticSearch adalah :

1. Nilai default yang wajar yang memungkinkan pengguna untuk mulai menggunakan ElasticSearch setelah proses penginstalan, tanpa ada tambahan. Hal ini termasuk built-in (misalnya, tipe field) dan konfigurasi otomatis.
2. Bekerja dengan mode default distribusi. Diasumsikan bahwa ada Node atauyang akan menjadi bagian dari cluster, dan selama percobaan penyetingan node secara otomatis bergabung dengan cluster.
3. Arsitektur Peer-to-peer tanpa titik tunggal kegagalan (SPOF). Node secara otomatis terkoneksi ke mesin lain dalam cluster untuk pertukaran data dan saling memonitoring. Hal ini mencakup replikasi shard secara otomatis.

4. Mudah dalam pengukuran baik dari segi kapasitas dan jumlah data dengan menambahkan node cluster baru.
5. ElasticSearch tidak memberikan pembatasan pada data dalam indeks. Hal ini memungkinkan pengguna untuk melakukan penyesuaian dengan model data yang ada. Seperti yang kita ketahui dalam tipe deskripsi, ElasticSearch mendukung beberapa jenis multiple data dalam indeks tunggal dan model bisnis adjustmentto termasuk hubungan antara penanganan dokumen tetapi fungsi ini agak terbatas.
6. Near Real Time (NRT) dalam proses pencarian dan pemversian. Karena sifat ElasticSearch yang mendistribusikan, tidak ada kemungkinan untuk menghindari keterlambatan dan sementara perbedaan antara data yang terletak pada node yang berbeda. ElasticSearch mencoba mengurangi masalah ini dan memberikan mekanisme tambahan sebagai versi.

2.7.1 Proses ElasticSearch

Sebelum proses ElasticSearch berjalan, perlu didefinisikan nama dari Node, Cluster dan shard yang digunakan.



Gambar 2.5 Proses ElasticSearch

Pada Gambar 2.5 dijelaskan proses ElasticSearch :

1. Ketika Node baru dimulai, proses langsung disebarkan (multicast) ke beberapa ElasticSearch Node pada ElasticSearch

Cluster. Cluster name sudah didefinisikan ketika proses konfigurasi dan terkoneksi.

2. Dalam cluster, salah satu node terpilih sebagai node master. Node ini bertanggung jawab untuk mengelola state cluster dan proses untuk menempatkan shard cluster untuk node dalam reaksi perubahan topologi cluster.
3. Ketika pengguna mengakses aplikasi, langsung segera mendapatkan respon dari ElasticSearch Node, apabila ada beberapa Node maka akan dicarikan ke dalam node-node tersebut. Di dalam node masih terdapat shard-shard yang digunakan untuk membagi index dokumen.
4. Hasil yang diminta akan ditampilkan ke dalam aplikasi.

2.7.2 Komunikasi ElasticSearch

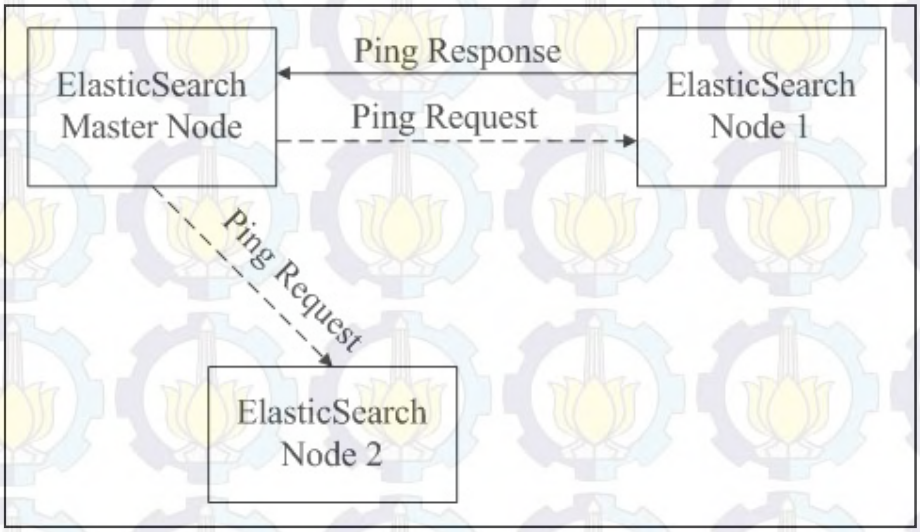
Selama bekerja pada normal cluster, node master memonitor semua node yang ada dan memeriksa apakah node tersebut bekerja. Jika salah satu dari node tidak tidak bekerja untuk waktu tertentu, maka node dianggap sebagai rusak dan proses penanganan kegagalan dimulai. Hal ini mungkin berarti rebalancing dari cluster-shard, yang ada pada node yang rusak dan untuk setiap shard tersebut node lain harus bertanggung jawab. Dengan kata lain untuk setiap kehilangan shard primer, shard primer baru dapat dipilih dari replika yang tersisa dari shard tersebut. Seluruh proses menempatkan shard baru dan replika dapat (dan biasanya harus) dikonfigurasi sesuai kebutuhan kita.

Pada Gambar 2.6 dijelaskan proses komunikasi ElasticSearch :

1. ElasticSearch terdiri dari 3 yaitu ElasticSearch Master Node, ElasticSearch Node 1 dan ElasticSearch Node 2.
2. ElasticSearch Master Node memberikan request ke ElasticSearch node 2 untuk memonitoring proses yang ada pada node 2, apabila ada respon yang disampaikan oleh node 2 maka komunikasi

dikatakan dapat berjalan dengan baik. Sebaliknya komunikasi tidak terhubung antara node 2 dengan master node.

3. Untuk selanjutnya ElasticSearch Master Node juga memonitoring ElasticSearch Node 1, hal yang dilakukan juga sama, master node memberikan request kepada node 1, dan akan mendapatkan respon dari node 1, apabila master node tidak mendapatkan respon, maka komunikasi tidak terhubung, bisa dikatakan node 1 sedang tidak aktif.



Gambar 2.6 Proses komunikasi ElasticSearch

2.7.3 Indexing Data

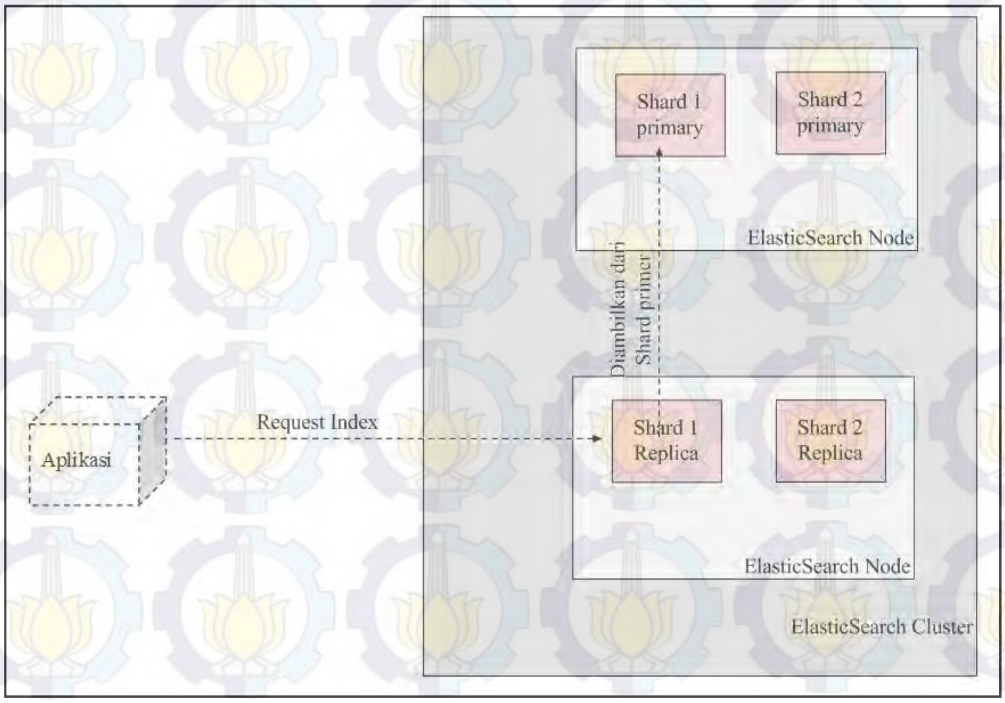
Hal yang perlu diingat adalah bahwa pengindeksan hanya terjadi pada shard primer, bukan pada replika. Jika permintaan pengindeksan akan dikirim ke node, yang tidak memiliki shard yang benar atau berisi replika, maka akan diteruskan ke shard primer.

Pada Gambar 2.7 dijelaskan konsep dasar indexing pada ElasticSearch :

1. Proses indexing dimulai ketika pengguna memasukkan data, dimana data tersebut dimasukkan ke dalam aplikasi. Pada aplikasi sudah terkonfigurasi penentuan nama node, cluster,

shard dan replika. Hal ini agar tidak terjadi tumpang tindih pada proses penamaan. Setelah pengguna memasukkan data, data tersebut akan dipisah-pisah menjadi index, dan index tersebut disimpan ke dalam shard sesuai dengan yang dibutuhkan.

2. Shard yang digunakan dibagi menjadi dua, shard primer dan shard replika, pada node yang berbeda.
3. Ketika pengguna memberikan request terhadap suatu kata, maka aplikasi akan memberikan request index ke dalam shard. Dikarenakan konfigurasi yang digunakan adalah shard replika terlebih dahulu, maka request tersebut diambilkan dari shard primer. Diperlukan koneksi dari shard replika ke shard primer.



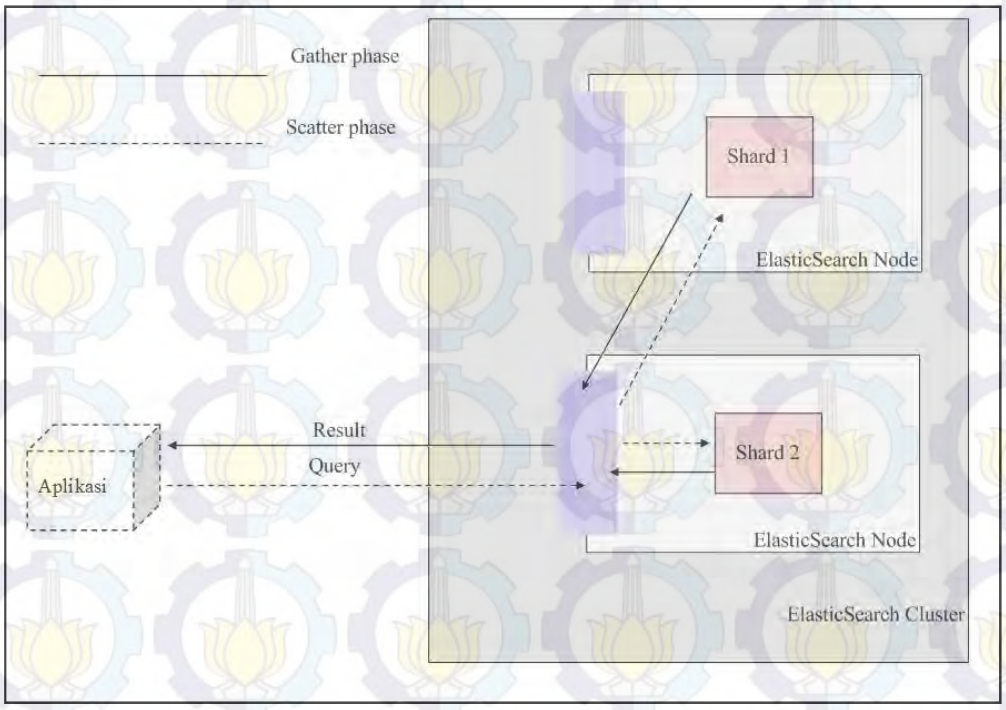
Gambar 2.7 Proses Indexing

Untuk kasus tersebut, shard replika langsung memberikan request pengambilan data pada shard primer. Dapat dikatakan shard replika tidak memiliki kekuasaan untuk memberikan respon apabila masih ada shard primer dan shard primer tersebut masih aktif dan terkoneksi dengan shard replika. Shard replika dan shard primer

dapat dimasukkan ke dalam satu node, tidak harus dimasukkan ke dalam beberapa node seperti kasus di atas.

2.7.4 Querying Data

Querying data merupakan proses utama setelah indexing data. Proses ini merupakan proses respon yang diberikan ketika pengguna memberikan request.



Gambar 2.8 Proses Querying

Pada Gambar 2.8 dijelaskan proses queri data ketika proses pencarian yang dibedakan menjadi dua tahap :

1. Scatter phase (fase pemecahan / pemisahan)

   Fase ini dimulai ketika pengguna memberikan request ke dalam aplikasi, lalu aplikasi tersebut menyampaikan request tersebut dengan memberikan queri pada cluster ElasticSearch. ElasticSearch Cluster selanjutnya menyampaikan request tersebut ke node dan shard, dari sini permintaan request ke node

langsung dibagi ke dalam beberapa shard sesuai dengan konfigurasi yang telah ditentukan. Lalu memberikan responnya.

2. Gather phase (fase penggabungan)

   Pada fase ini respon yang diberikan dari shard langsung digabungkan menjadi satu ketika respon tersebut sampai pada node. Jika ada beberapa node, maka respon dari masing-masing node tersebut digabungkan lagi sehingga menjadi respon yang utuh, lalu dilanjutkan ke dalam ElasticSearch cluster dan diterima oleh pengguna dalam keadaan utuh. Proses ini tidak pernah diketahui oleh pengguna karena terjadi di dalam sistem.

Satu atau lebih node ElasticSearch dapat diatur pada fisik atau server virtual tergantung sumber daya yang tersedia seperti RAM, CPU, dan ruang disk. Sebuah node memberikan standar menyimpan data di dalamnya dan permintaan proses dan tanggapan.

2.8 Temu Kembali Informasi

Sistem temu kembali informasi berasal dari kata Information Retrieval System (IRS). Temu kembali informasi adalah sebuah media layanan bagi pengguna untuk memperoleh informasi atau sumber informasi yang dibutuhkan oleh pengguna. Sistem temu kembali informasi merupakan sistem informasi yang berfungsi untuk menemukan informasi yang relevan dengan kebutuhan pemakai. Sistem temu kembali informasi berfungsi sebagai perantara kebutuhan informasi pengguna dengan sumber informasi yang tersedia. Pengertian yang sama mengenai sistem temu kembali informasi menurut Sulistyo-Basuki sistem temu kembali informasi adalah kegiatan yang bertujuan untuk menyediakan dan memasok informasi bagi pemakai sebagai jawaban atas permintaan atau berdasarkan kebutuhan pemakai. Dapat dinyatakan bahwa sistem temu kembali informasi memiliki fungsi dalam menyediakan kebutuhan informasi sesuai dengan kebutuhan dan permintaan penggunanya.

Ada beberapa definisi dalam sistem temu kembali informasi menurut para ahli di bidang ilmu perpustakaan dan informasi, yaitu sebagai berikut:

1. Menurut Kochen, kata retreve yang dikaitkan dengan IR (Information retrieval) yaitu upaya membantu pengguna sistem komputer menemukan dokumen yang dicari. Lebih spesifik lagi, kemampuan komputer tersebut dikaitakan dengan recall (mengingat). Pendit (2008) menambahkan, dalam bahasa indonesia kata retrieve diterjemahkan menjadi temu kembali. Jadi kata Information Retrieval diterjemahkan sebagai temu kembali informasi.
2. Menurut Odlis menjelaskan bahwa Temu Kembali informasi (IR) adalah proses, metode, dan prosedur yang digunakan untuk menyeleksi informasi yang relevan yang tersimpan dalam database. Dalam bidang perpustakaan dan arsip, temu kembali informasi biasanya untuk dokumen yang diketahui atau untuk informasi mengenai subyek tertentu, dan file biasanya katalog atau indeks, atau penyimpanan informasi berbasis komputer dan sistem pencarian, seperti katalog online atau Database bibliografi. Dalam merancang sistem tersebut, keseimbangan harus dicapai antara kecepatan, akurasi, biaya, kenyamanan, dan efektivitas.
3. Menurut wikipedia dijelaskan bahwa Temu Kembali Informasi (Information Retrieval) digunakan untuk menemukan kembali informasi-informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis. Salah satu aplikasi umum dari temu kembali informasi adalah search-engine atau mesin pencarian yang terdapat pada jaringan internet. Pengguna dapat mencari halaman-halaman Web yang dibutuhkannya melalui mesin tersebut, misalnya google.
4. Menurut Mooers (1948) dijelaskan bahwa Information Retrieval sendiri adalah seni dan ilmu dalam mencari informasi pada dokumen, mencari untuk dokumen mereka sendiri, mencari untuk metadata dengan gambaran berbentuk dokumen, atau mencari dalam database, apakah itu

hubungan database yang berdiri sendiri atau hiperteks jaringan database seperti internet atau intranet, untuk teks, suara, gambar atau data. Mooers (1951) juga menjelaskan bahwa Information Retrieval adalah bidang di persimpangan ilmu informasi dan ilmu komputer. Berkutat dengan pengindeksan dan pengambilan informasi dari sumber informasi heterogen dan sebagian besar-tekstual. Istilah ini diciptakan oleh Mooers pada tahun 1951, yang menganjurkan bahwa diterapkan ke "aspek intelektual" deskripsi informasi dan sistem untuk pencarian.

5. Menurut Hougthon (1977) dijelaskan bahwa sistem temu kembali informasi adalah penelusuran yang merupakan interaksi antara pemakai dengan sistem dan pernyataan kebutuhan pengguna diekspresikan sebagai suatu istilah tertentu
6. Menurut Lancaster (1979) dijelaskan bahwa temu kembali informasi tidak menginformasikan semua isi dari subjek yang dimiliki koleksi tersebut tetapi hanya memberikan informasi keberadaan pustaka yang mempunyai hubungan subjek seperti yang dicari oleh pengguna.
7. Menurut Salton (1983) dijelaskan secara sederhana menjelaskan bahwa temu kembali informasi merupakan suatu sistem yang menyimpan informasi dan menemukan kembali informasi tersebut.
8. Menurut Sulistyo-Basuki (1991) dijelaskan bahwa temu kembali informasi sebagai kegiatan yang bertujuan untuk menyediakan dan memasok informasi bagi pemakai sebagai jawaban atas permintaan atau berdasarkan kebutuhan pemakai.
9. Menurut Baeza-Bates dan Riberto-Neto (1999) dijelaskan bahwa temu kembali informasi berkaitan dengan representasi, penyimpanan, dan akses terhadap dokumen representasi dokumen.
10. Menurut Zaenab (2002) dijelaskan bahwa temu kembali informasi informasi merupakan suatu proses pencarian dokumen dengan menggunakan istilah-istilah bahasa pencarian untuk mendefinisikan dokumen sesuai dengan subjek yang diinginkan.

11. Menurut Hasugian (2003) dijelaskan bahwa temu kembali informasi pada dasarnya adalah suatu proses untuk mengidentifikasi, kemudian memanggil (retrieval) suatu dokumen dari suatu simpanan (file), sebagai jawaban atas permintaan informasi.
12. Menurut Harter (1986) dijelaskan bahwa Sistem temu-kembali informasi (Information Retrieval System/IRS) adalah perangkat yang menghubungkan antara pemakai potensial dengan koleksi atau kumpulan informasi.

2.9 Validasi Output

Untuk memvalidasi hasil yang ditunjukkan dari proses pencarian dengan menggunakan metode ElasticSearch, maka diperlukan suatu cara untuk menghitung beberapa atribut dari proses tersebut. Selain itu perlu juga dihitung kecepatan dari setiap percobaan yang dilakukan. Hal ini dilakukan agar dapat dikertahui kecepatan dari proses tersebut.

2.9.1 Precision

Precision ialah perbandingan jumlah dokumen relevan yang didapatkan sistem dengan jumlah seluruh dokumen yang terambil oleh sistem baik relevan maupun tidak relevan. Precision mengevaluasi kemampuan sistem temu kembali informasi untuk menemukan kembali data top-ranked yang paling relevan, dan didefinisikan sebagai persentase data yang dikembalikan yang benar-benar relevan terhadap query pengguna. Precision merupakan proporsi dari suatu set yang diperoleh yang relevan.

$$\text{precision} = \frac{\text{Jumlah dokumen relevan dengan query dan terambil.}}{\text{Jumlah seluruh dokumen yang terambil}}$$

2.9.2 Recall

Recall ialah perbandingan jumlah dokumen relevan yang didapatkan sistem dengan jumlah seluruh dokumen relevan yang ada dalam koleksi dokumen (terambil ataupun tak terambil sistem).

$$\text{recall} = \frac{\text{Jumlah dokumen relevan dengan query dan terambil.}}{\text{Jumlah seluruh dokumen relevan dalam koleksi dokumen}}$$

2.9.3 F-Measure

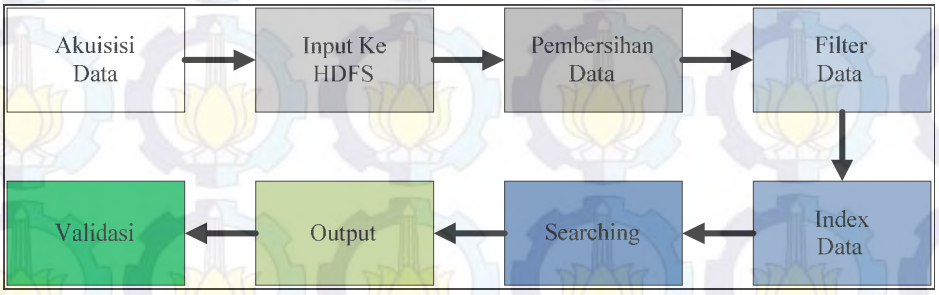
Dari hasil perpaduan atau kombinasi antara Recall dan Precision disebut harmonic mean, biasa disebut F-measure yang mana dapat dihitung dengan cara

$$\text{F-Measure} = 2 \times \frac{\text{Precision x Recall}}{\text{Precision + Recall}}$$

BAB III
METODOLOGI PENELITIAN

Pada Bab ini akan dijelaskan metodologi penelitian yang dilakukan pada ini. Pada bab ini metode ElasticSearch digunakan ketika proses indexing dan searching. Tahapan-tahapan metodologi yang dilakukan dalam penelitian ini ditunjukan sebagai berikut.

3.1 Metodologi Penelitian



Gambar 3.1 Metodologi Penelitian

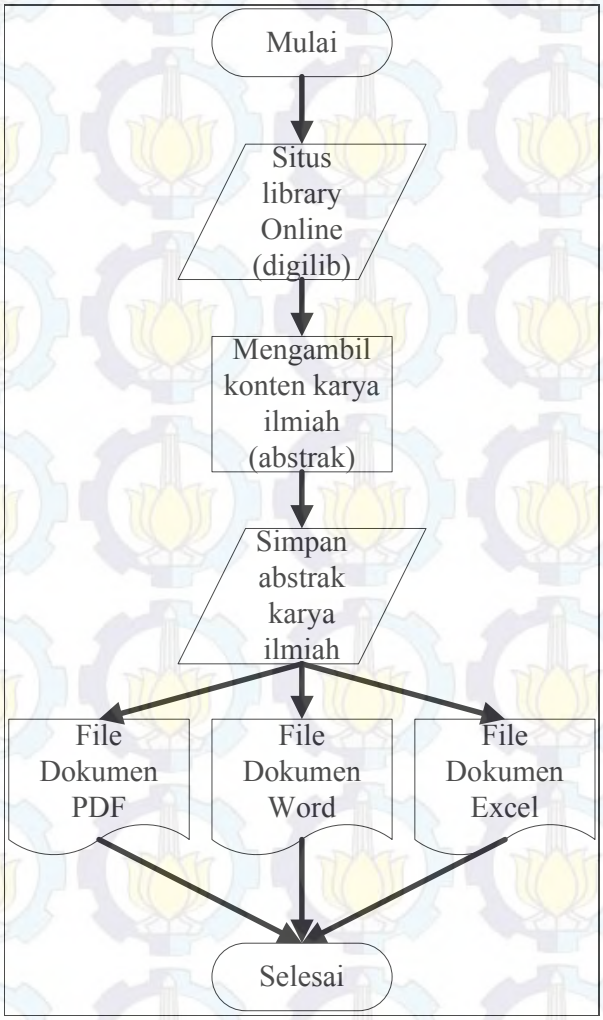
Pada Gambar 3.1 menggambarkan metodologi penelitian yang dilakukan. Tahapan-tahapan yang dilakukan adalah Akuisisi Data, Input ke HDFS, Pembersihan Data, Filter Data, Index Data, Searching, Output, dan Validasi. Pemaparan dari masing-masing tahapan tersebut ada pada penjelasan berikut ini.

3.1.1 Akuisisi Data

Tahapan ini adalah tahapan yang pertama. Pada tahapan ini dilakukan pengumpulan data. Data yang akan digunakan dalam penelitian ini adalah data literatur karya ilmiah. Dari data karya ilmiah yang dijadikan dasar untuk proses pencarian adalah data abstrak. Data abstrak karya ilmiah diambil dari beberapa situs library perguruan

tinggi di internet. Jenis data yang diambil dengan format sebagai berikut :

1. Word dengan ekstensi doc
2. Excel dengan ekstensi xls
3. *Portable Document Format* dengan ekstensi pdf



Gambar 3.2 Diagram alur proses akuisisi data

Alur proses akuisisi data pada Gambar 3.2 dapat dijelaskan sebagai berikut :

1. Hal pertama yang dilakukan dalam proses pengambilan data atau biasa disebut dengan akuisisi data adalah mengunjungi situs-situs

e-library (digilib) di beberapa perguruan tinggi negeri maupun perguruan tinggi swasta.

2. Mengambil konten karya ilmiah yang berupa abstrak. Abstrak diambil baik abstrak tesis, tugas akhir maupun abstrak disertasi. Apabila ada file yang bisa diunduh maka file tersebut diunduh dan disesuaikan dengan kebutuhan yaitu mengambil abstraknya saja.
3. Simpan abstrak karya ilmiah dari digilib ke dalam komputer.
4. Abstrak karya ilmiah tersebut disimpan dalam bentuk word dengan ekstensi .doc, bentuk pdf dan bentuk excel dengan ekstensi .xls.
5. File disimpan di dalam komputer.

Pada beberapa situs website *e-library* kadangkala hanya ditampilkan file PDF saja yang dapat diunduh, sehingga perlu adanya perubahan dari file PDF menjadi file word atau excel. Atau bisa juga data diambil dari e-library yang langsung menampilkan hasil abstrak tanpa perlu mendownloadnya. Pada kondisi tersebut akuisisi data dilakukan dengan cara menyalin abstrak tersebut lalu mengubahnya ke dalam bentuk file yang diperlukan seperti word, pdf dan excel. Untuk file excel didapatkan dengan cara menyalin abstrak yang ada pada halaman web, kemudian dimasukkan ke dalam file word lalu dijadikan file PDF. Dari file PDF tersebut, dikonversikan menjadi file excel. Hal ini dilakukan agar setiap file excel yang diuplod memiliki perbedaan dalam penataan kolom-kolomnya dan didapatkan hasil yang lebih untructured. Ketika abstrak yang disalin langsung dimasukkan ke dalam file excel, maka akan terjadi persamaan penentuan kolom mana yang akan digunakan, dan variasinya tidak banyak.

Sumber-sumber pengambilan data abstrak karya ilmiah antara lain:

a. Library ITS
b. Library Undip
c. Library ITB
d. Library Ubaya
e. Library UI

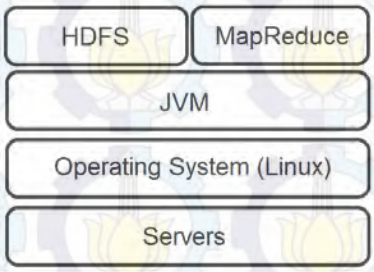
Pada Tabel 3.1 ditunjukkan beberapa data yang telah diakuisisi dari situs-situes e-library yang sumbernya disebutkan di atas. Pada Tabel 3.1 tersebut dicontohkan datanya berupa file dokumen word (doc), dokumen pdf, dan file dokumen excel (xls). Data yang dicontohkan hanya diambil sebagian dari isinya Isi dari masing-masing file dapat dirata-rata sampai 2 atau 3 paragraf.

Tabel 3. 1 Akuisisi data

No	Isi Abstrak Karya Ilmiah	Nama File	Tipe File
1	penentuan faktor prioritas mahasiswa dalam memilih telepon seluler merk blackberry dengan fuzzy ahp hanien nia h shega rita rahmawati hasbi yasin mahasiswa jurusan statistika fsm universitas diponegoro ….	PENENTUAN FAKTOR PRIORITAS MAHASISWA DALAM MEMILIH TELEPON SELULER MERK BLACKBERRY DENGAN FUZZY AHP.pdf	pdf
2	interval konfidensi spline kuadrat dengan pendekatan pivotal quantityrowan daflix syaranamual i nyoman budiantaram ahasiswa magister jurusan statistika its do sen jurusan statistika its abstrak ….	INTERVAL KONFIDENSI SPLINE KUADRAT DENGAN PENDEKATAN PIVOTAL QUANTITY.pdf	pdf
3	memanusiawikan warga bantaran kali belajar dari kasus penanganan warga kali code yogyakarta abdul malik persoalan lingkungan perkotaan ….	Memanusiawikan_Warga_Bantaran_Kali.xls	xls
4	model regresi data tahan hidup tersensor tipe iii berdistribusi eksponensial winda faati kartika triastuti wuryandari program studi statistika jurusan matematika fmipa universitas diponegoro ….	MODEL REGRESI DATA TAHAN HIDUP TERSENSOR TIPE III BERDISTRIBUSI EKSPONENSIAL.xls	xls
5	kajian tingkat penghidupan berkelanjutan sustainable livelihood di kawasan dieng kasus di dua desa kecamatan kejajar …	Kajian Tingkat Penghidupan Berkelanjutan.doc	doc
Dst….			

3.1.2 Input ke HDFS

Setelah data diakuisisi, data tersebut akan di-inputkan ke dalam HDFS (Hadoop File System) yang ada di Hadoop. Arsitektur HDFS seperti pada Gambar 3.3.



Gambar 3.3 Arsitektur HDFS

Pada Gambar 3.3 arsitektur HDFS (Hadoop File System) dijabarkan sebagai berikut :

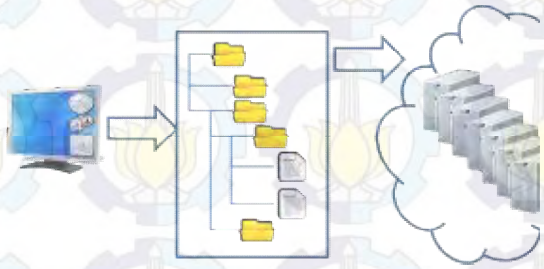
1. HDFS sudah dimasukkan ke dalam server. Server yang digunakan bisa 1 komputer atau terdiri dari beberapa komputer yang disebut dengan *NameNode*.
2. Server yang berjalan hampir semuanya menggunakan sistem operasi Linux. Linux yang digunakan tidak dibatasi.
3. HDFS berbasiskan JVM (Java Virtual Machine).
4. HDFS dan MapReduce berjalan berdampingan. Dimana HDFS digunakan untuk menyimpan file-file yang dimasukkan sedangkan MapReduce untuk memprosesnya.

HDFS menyimpan suatu data dengan membaginya menjadi potongan-potongan data. Potongan data tersebut memiliki ukuran tertentu. Antara satu potongan dengan potongan data yang lain memiliki ukuran yang sama. Ukuran potongan data ini dapat ditentukan sesuai dengan kebutuhan dan sesuai keinginan. Potongan-potongan data ini kemudian disimpan tersebar dalam komputer-komputer yang membentuk clusternya. Potongan-potongan data tersebut dalam HDFS disebut block.

Walaupun data disimpan secara tersebar, pengguna yang mengakses data tidak mengetahui hal itu, data tetap terlihat seperti halnya kita mengakses file pada satu komputer. File yang secara fisik disimpan tersebar dalam banyak komputer itu diperlakukan layaknya memperlakukan file dalam satu komputer.

Pada Gambar 3.4 dijabarkan pengaksesan file dari HDFS:

1. Pengguna mengakses file dengan user interface yang ditampilkan di layar monitor.
2. Permintaan akses file diambilkan dari HDFS. HDFS terdiri dari beberapa DataNode yang menyimpan masing-masing file dokumen.
3. File diambilkan dari potongan-potongan yang ada dalam HDFS

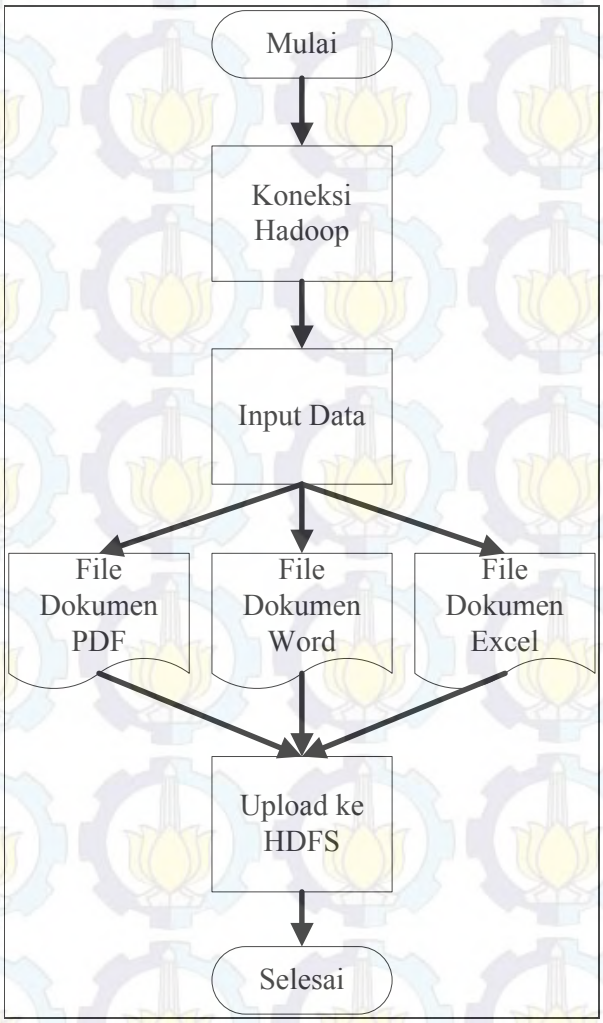


Gambar 3.4 Pengaksesan File ke HDFS

Pada Gambar 3.5 dijelaskan diagram alur input data ke HDFS:

1. Input data ke HDFS dimulai dengan membuat koneksi dengan Hadoop. Apabila koneksi telah terhubung maka bisa dilakukan proses input. Proses koneksi ini dapat dilihat pada command prompt yang ada di komputer.
2. Input data ke HDFS. Memasukkan file dokumen ke dalam HDFS dengan mengetikkan beberapa perintah ke dalam command prompt.
3. Data yang diupload berupa file dokumen Word, Excel, dan PDF.
4. Proses Upload dilakukan dengan mengetikkan perintah-perintah pada command prompt yang ada dan sampai semua proses upload

data selesai dilakukan. Proses upload data ini langsung diambil pada satu folder untuk masing-masing jenis file dokumen.



Gambar 3.5 Diagram Alur proses input ke HDFS

Agar antara sistem dengan HDFS terhubung, maka diperlukan penghubung atau koneksi antara hadoop HDFS dengan sistem. Hal ini bertujuan agar data yang ada pada sistem dan HDFS tersinkronisasi. Koneksi ini diperlukan ketika proses input data dari sistem. Data pada sistem dengan data yang ada pada HDFS sama.

3.1.3 Preprocessing

3.1.3.1 Pembersihan Data

Data yang telah didapatkan akan dilakukan pembersihan data, tahapan ini termasuk ke dalam bagian dari Data Preprocessing. Pembersihan ini dilakukan setelah data diupload ke dalam sistem. Pada proses pembersihan data dilakukan untuk membuang data yang tidak konsisten yang mengandung noise dan banyak kekeliruan dibetulkan misalnya data dengan huruf ganda. Pada proses ini tanda baca juga dihilangkan, kecuali tanda baca untuk kata ulang, pembersihan dari huruf ganda dan pembersihan dari angka. Hal ini dilakukan agar index yang didapatkan bersih dari noise-noise yang ada. Pada proses ini juga dilakukan penyeragaman huruf. Huruf yang digunakan untuk membuat index adalah huruf kecil semua.

Tabel 3. 2 Pembersihan data

No	Data Mentah	Data Bersih	Tipe File
1	**PENENTUAN FAKTOR PRIORITAS MAHASISWA DALAM MEMILIH TELEPON SELULER MERK BLACKBERRY DENGAN FUZZY AHP** **Hanien Nia H Shega, Rita Rahmawati, Hasbi Yasin**[3] [1] Mahasiswa Jurusan Statistika FSM Universitas Diponegoro…….	penentuan faktor prioritas mahasiswa dalam memilih telepon seluler merk blackberry dengan fuzzy ahp hanien nia h shega rita rahmawati hasbi yasin mahasiswa jurusan statistika fsm universitas diponegoro ….	pdf
2	**INTERVAL KONFIDENSI SPLINE KUADRAT DENGAN PENDEKATAN *PIVOTAL QUANTITY***	interval konfidensi spline kuadrat dengan pendekatan pivotal quantityro wan daflix syaranamual i nyoman budi antaramahasiswa magister jurusan	pdf

Tabel 3.2 Lanjutan

No	Data Mentah	Data Bersih	Tipe File
	Rowan Daflix Syaran amual[1], I Nyoman Budiantara[2] [1)] Mahasiswa Ma gister Jurusan Statistika ITS [2)] Dosen Jurusan Statistika ITS **Abstrak** ….	statistika its dosen jurusan statistika its abstrak ….	
3	**MEMANUSIAWIK AN WARGA BANTARAN KA LI** [1] Belajar dari Kasus Pen anganan Warga Kali Code Yog yakarta Abdul Malik [2] **ABSTRAK** *Persoalan lingkungan….*	memanusiawikan warga bantaran kali belajar dari kasus penanganan warga kali code yogyakarta abdul malik persoalan lingkungan perkotaan ….	xls
4	**MODEL REGRESI DATA T AHAN HIDUP TERSENSOR TIPE II I**	model regresi data tahan hidup tersensor tipe iii berdistribusi eksponensial winda faati kartika	xls

Tabel 3.2 Lanjutan

No	Data Mentah	Data Bersih	Tipe File
	BERDISTRI BUSI EKSPONENSIAL **Winda Faati Karti ka[1], Triastuti Wuryandari[2]** [1, 2)] Program Studi Statistika Jurusan M atematika FMIPA Universitas Diponeg oro….	triastuti wuryandari program studi statistika jurusan matematika fmipa universitas diponegoro ….	
5	**Kajian Tingkat Penghidupan Berk elanjutan (*Sustainable Livelihood)* Di Kawasan Dieng (Kasus Di Dua Desa Kecamatan Kejaja r……**	kajian tingkat penghidupan berkelanjutan sustainable livelihood di kawasan dieng kasus di dua desa kecamatan kejajar …	doc

Pada Tabel 3.2 dijabarkan cara memproses data dengan pembersihan data :

1. Kolom Data Mentah berisikan data abstrak karya ilmiah asli yang ada pada file dokumen yang diupload ke dalam sistem. Data mentah ini belum terjadi proses pembersihan data.
2. Kolom Data Bersih berisikan data isi abstrak yang sudah dibersihkan dari tanda baca, huruf kapital, angka, cetak tebal huruf dan juga paragraf kata.
3. Kolom Tipe File berisikan jenis file dari data yang sudah diupload ke dalam sistem

3.1.3.2 Filtering Data

Proses untuk memilih atau menyaring data pada abstrak karya ilmiah yang berbahasa Indonesia saja dan ditentukan kata-kata yang jarang digunakan. Untuk kata sambung ataupun kata hubung dihilangkan. Hal ini dilakukan agar kata yang dicari dan index yang dibentuk lebih spesifik.

Pada tabel 3.3 dijabarkan cara memproses data dengan filtering data :

1. Kolom Data Bersih merupakan data asli yang sudah dibersihkan pada proses pembersihan data.
2. Kolom Filter Data berisikan data yang telah dilakukan proses filtering. Pada proses filter ini kata hubung dihilangkan agar index yang didapatkan lebih spesifik.
3. Kolom Tipe File berisikan jenis file dari data yang sudah diupload ke dalam sistem

Tabel 3. 3 Filtering data

No	Data Bersih	Filter Data	Tipe File
1	penentuan faktor prioritas mahasiswa dalam memilih telepon seluler merk blackberry dengan fuzzy ahp hanien nia h shega rita rahmawati hasbi yasin mahasiswa jurusan statistika fsm universitas diponegoro ….	penentuan faktor prioritas mahasiswa memilih telepon seluler merk blackberry fuzzy ahp hanien nia h shega rita rahmawati hasbi yasin mahasiswa jurusan statistika fsm universitas diponegoro ….	pdf
2	interval konfidensi spline kuadrat dengan pendekatan pivotal quantit yrowan daflix syaranamual i nyo man budiantaramahasiswa magist er jurusan statistika its dosen	interval konfidensi spline ku adrat pendekatan pivotal quantityrowan daflix syaran amual i nyoman budiantara mahasiswa magister jurusa	pdf

Tabel 3.3 Lanjutan

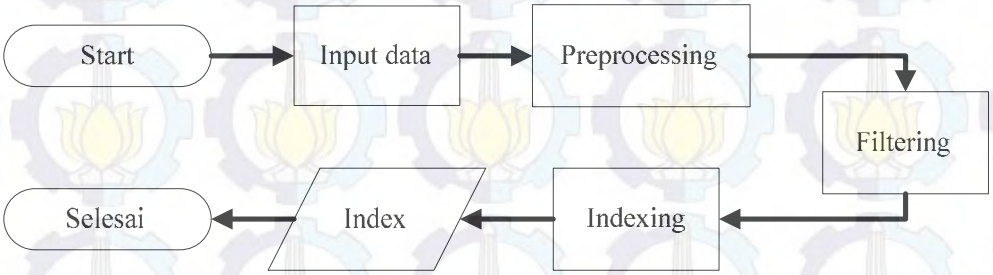
No	Data Bersih	Filter Data	Tipe File
	jurusan statistika its abstrak ….	statistika its dosen jurusan st atistika its abstrak ….	
3	memanusiawikan warga bantaran kali belajar dari kasus penanganan warga kali code yogyakarta abdul malik persoalan lingkungan perkotaan ….	memanusiawikan warga bantaran kali belajar kasus penanganan warga kali code yogyakarta abdul malik persoalan lingkungan perkotaan ….	xls
4	model regresi data tahan hidup tersensor tipe iii berdistribusi eksponensial winda faati kartika triastuti wuryandari program studi statistika jurusan matematika fmipa universitas diponegoro ….	model regresi data tahan hidup tersensor tipe iii berdistribusi eksponensial winda faati kartika triastuti wuryandari program studi statistika jurusan matematika fmipa universitas diponegoro ….	xls
5	kajian tingkat penghidupan berkelanjutan sustainable livelihood di kawasan dieng kasus di dua desa kecamatan kejajar …	kajian tingkat penghidupan berkelanjutan sustainable livelihood kawasan dieng kasus dua desa kecamatan kejajar …	doc

3.1.3.3 Indexing Data

Pada tahapan ini dilakukan proses pengindexan data. Dimana data yang dilakukan proses index yaitu dipisahkan dari kata dasar dan kata-kata penghubung serta data telah di filter. Indexing ini akan memudahkan dalam pencarian file dokumen yang diperlukan.

Pada Gambar 3.6 diejlaskan diagram alur proses indexing data :

1. Input data berupa dokumen dengan ekstensi doc, xls dan pdf
2. Setelah data diinput dilakukan preprocessing yaitu pembersihan data. Data dibersihkan dari tanda baca, huruf ganda.
3. Filtering digunakan untuk menyaring kata-kata yang jarang digunakan dan kata sambung dihilangkan.
4. Dari kata-kata yang sudah dilakukan preprocessing dan dilakukan filtering dibuatkan indexnya.
5. Dari proses indexing dihasilkan index yang bersih yang digunakan untuk memudahkan proses pencarian.



Gambar 3.6 Diagram Alur proses indexing

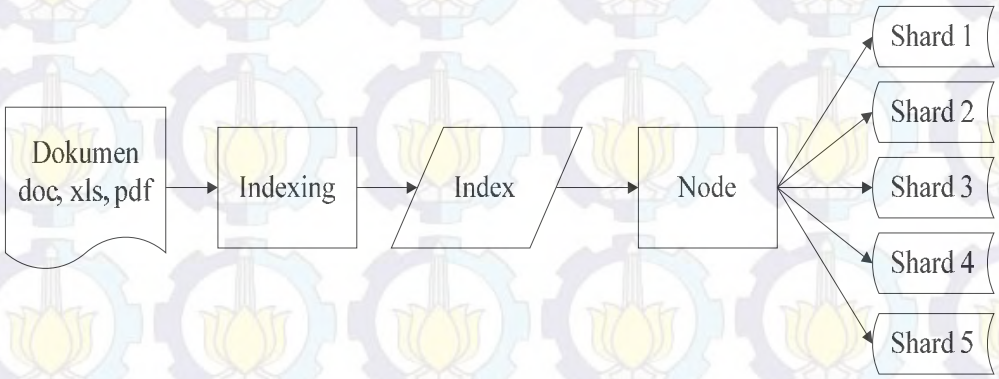
Pada metode elasticsearch ini hasil index dapat dibagi-bagi ke dalam beberapa bagian sesuai dengan konfigurasi yang telah ditentukan. Pada penelitian ini, ditentukan jumlah node yang digunakan adalah satu, jumlah cluster yang digunakan juga satu dan jumlah shard yang digunakan adalah 5.

Pada Gambar 3.7 dijelaskan alur pembagian index ke dalam beberapa shard :

1. Alur pertama adalah ketika dokumen dengan ekstensi doc, xls dan pdf telah dimasukkan ke dalam sistem dan

telah dilakukan preprocessing dan filtering data. Data ini yang akan digunakan dalam proses indexing.

2. Proses indexing dilakukan sehingga didapatkan index dari masing-masing dokumen yang telah dimasukkan.
3. Index tersebut akan dimasukkan ke dalam node yang telah disediakan. Node disini disesuaikan dengan jumlah komputer yang digunakan.
4. Dari node ini, apabila telah dikonfigurasikan jumlah shard maka index disebarkan sesuai dengan jumlah shard yang digunakan. Dalam penelitian ini shard yang digunakan sebanyak lima buah. Sehingga index tersebut disebarkan ke dalam lima shard tersebut. Pengguna tidak mengetahui bagaimana hasil pembagian index tersebut.



Gambar 3.7 Alur penempatan index

3.1.4 Searching

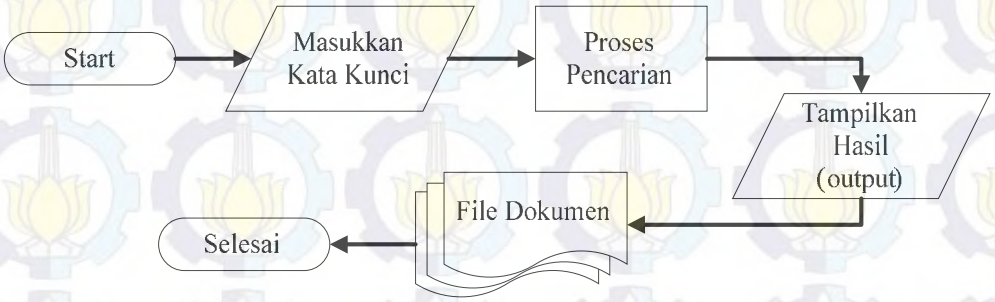
Tahapan ini dilakukan setelah proses index, dimana user memasukkan inputan untuk mencari kata-kata pada abstrak yang diperlukan. File dokumen-dokumen abstrak sudah diinputkan terlebih dahulu dan disimpan ke dalam HDFS.

Pada Gambar 3.8 dijelaskan alur proses searching :

1. Proses searching dimulai ketika pengguna mengakses sistem yang dikembangkan dan memasukkan kata kunci akan hal yang akan

dicari pada sistem. Kata kunci yang dimasukkan disesuaikan dengan kebutuhan dan keinginan pengguna. Kata kunci tersebut bisa terdiri dari satu kata, dua kata atau lebih.

2. Proses pencarian dilakukan ketika user memberikan request kata kunci tersebut. Proses pencarian dimulai dengan mencari ke dalam index sesuai dengan kata kunci yang dimasukkan. Hasil yang dikeluarkan oleh index pasti sesuai, hampir sesuai dan tidak sesuai dengan maksud dari pengguna.
3. Semua hasil tersebut ditampilkan pada sistem yang dibangun.
4. Hasil yang ditampilkan berupa file dokumen dan lokasinya yang sesuai dengan kata kunci yang dicari.



Gambar 3.8 Alur proses searching

Pada proses searching dilakukan queri untuk mencapai hasil yang masksimal. Queri ini dimulai ketika pengguna memberikan request ke dalam aplikasi, lalu aplikasi tersebut menyampaikan request tersebut dengan memberikan queri pada cluster ElasticSearch. ElasticSearch Cluster selanjutnya menyampaikan request tersebut ke node dan shard, dari sini permintaan request ke node langsung dibagi ke dalam beberapa shard sesuai dengan konfigurasi yang telah ditentukan. Lalu memberikan responnya.

Selanjutnya respon yang diberikan dari shard langsung digabungkan menjadi satu ketika respon tersebut sampai pada node. Jika ada beberapa node, maka respon dari masing-masing node

tersebut digabungkan lagi sehingga menjadi respon yang utuh, lalu dilanjutkan ke dalam ElasticSearch cluster dan diterima oleh pengguna dalam keadaan utuh. Proses ini tidak pernah diketahui oleh pengguna karena terjadi di dalam sistem.

3.1.5 Output

Merupakan hasil yang ditampilkan ketika telah dilakukan proses searching. Hasil yang diharapkan adalah nama file dokumen yang dicari. Apakah di dalam dokumen tersebut sesuai dengan kata kunci yang dicari atau tidak. Apabila ada pada hasil akan diurutkan sesuai dengan kata kunci yang dimasukkan. Hasil dari proses searching ini tidak hanya satu atau dua dokumen saja. Hasilnya bisa banyak dokumen yang ditemukan.

3.1.6 Validasi

Pada tahapan ini dilakukan validasi terhadap output data yang dicari. Data tersebut diukur nilai keakuratan dan kecepatan pencarian.

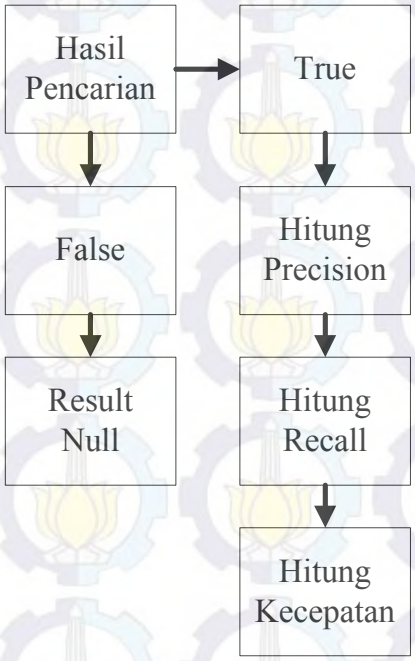
Pada Gambar 3.9 dijelaskan tahapan validasi data yang dilakukan setelah proses pencarian dan didapatkan hasilnya :

1. Jika hasil didapatkan true dan hasilnya sesuai dengan keinginan pengguna, maka akan dilakukan perhitungan presisi, recall dan kecepatan. Untuk perhitungan kecepatan secara otomatis pada sistem akan menampilkan waktu yang dibutuhkan untuk proses pencarian tersebut.

   Precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Precision merupakan perbandingan jumlah dokumen relevan yang didapatkan sistem dengan jumlah seluruh dokumen yang terambil oleh sistem baik relevan maupun tidak relevan. Sedangkan recall adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Recall merupakan

perbandingan jumlah dokumen relevan yang didapatkan sistem dengan jumlah seluruh dokumen relevan yang ada dalam koleksi dokumen (terambil ataupun tak terambil sistem).

2. Jika hasil didapatkan false, maka tidak didapatkan hasil dari proses pencarian karena kata kunci yang dimasukkan tidak terdapat pada file index.



Gambar 3.9 Validasi Data

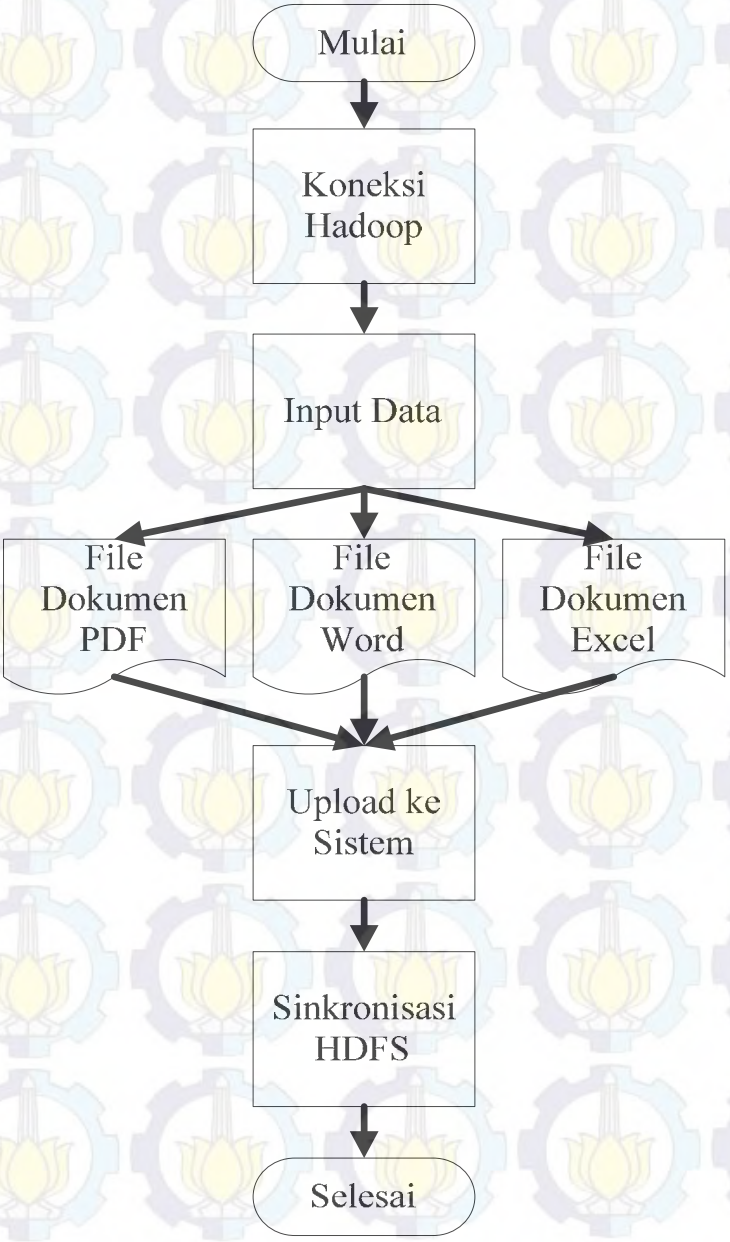
3.2 Desain Sistem

Sistem yang dibangun untuk melakukan pencarian konteks kata dalam file dokumen ini dibagi menjadi 2 desain. Desain yang pertama merupakan desain untuk mengupload data pada sistem, yang kemudian akan dilakukan sinkronisasi antara data pada sistem dengan data yang ada pada HDFS. Desain yang kedua adalah pencarian atau searching merupakan desain untuk user agar dapat memasukkan kata kunci yang akan dicari.

3.2.1 Input File

Desain ini dibuat pada sistem untuk proses upload file dokumen yang ada. Data yang diupload pada sistem harus sama dengan data

yang ada pada HDFS. Yang dimaksudkan di sini adalah ketika kita melakukan upload data ke dalam sistem yang dibangun, maka data tersebut akan dimasukkan dalam HDFS. Sistem yang dibangun harus secara otomatis terkoneksi dengan HDFS.



Gambar 3.10 Desain Input File

Pada Gambar 3.10 dijelaskan alur dari upload data ke sistem :

1. Sebelum pengguna melakukan upload data, terlebih dahulu pada sistem dibuat koneksi ke hadoop. Hal ini agar data yang dimasukkan ke dalam sistem diarahkan ke HDFS.
2. Pengguna memasukkan inputan yang berupa file dokumen ke dalam sistem. Pada sistem didesain antar muka berbasis web untuk memudahkan pengguna dalam memasukkan inputan data.
3. File yang dimasukkan harus file dokumen word dengan ekstensi doc, file dokumen excel dengan ekstensi xls dan file PDF dengan ekstensi pdf. File ini berisi abstraksi dari karya ilmiah yang disesuaikan dengan sumber pengambilan data.
4. Setelah koneksi berhasil dan data berupa file dokumen sudah ada lalu dimasukkan ke dalam sistem. Pada sistem data tersebut disimpan pada alokasi tempat yang sudah ditentukan. Alokasi tempat tersebut sudah dikonfigurasikan ketika menentukan nama dari node, cluster dan shard.
5. Agar data yang ada sama, maka dilakukan sinkronisasi dengan data yang ada pada HDFS. Sinkronisasi otomatis terjadi ketika ada perubahan data baik ada data baru yang diinputkan ataupun data lama yang diupdate.

3.2.2 Searching

Desain ini merupakan desain yang kedua. Pencarian atau searching merupakan desain untuk user agar dapat memasukkan kata kunci yang akan dicari. Pada desain ini juga menggunakan antar muka berbasis web untuk memudahkan pengguna dalam proses pencarian data.

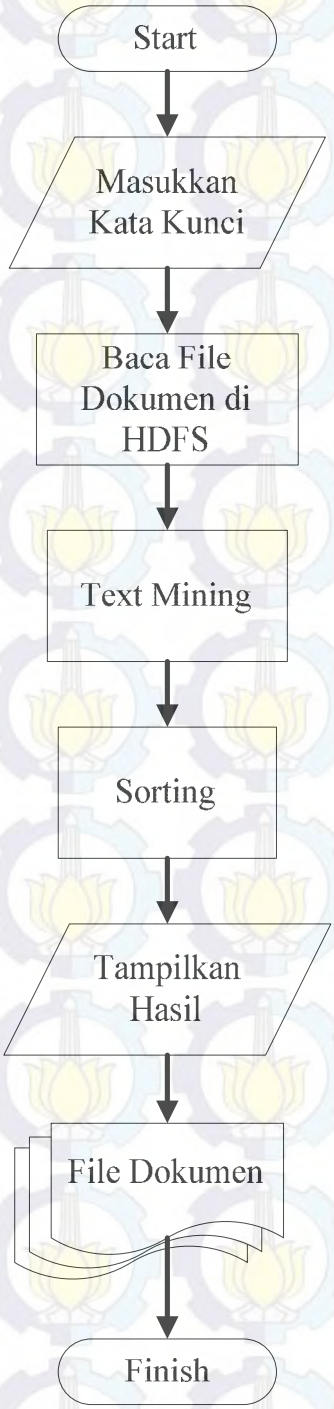
Antar muka pada proses pencarian ini dikembangkan secara sederhana. Hanya ada ruang untuk memasukkan kata kunci disertai dengan tombol cari. Ketika memasukkan kata kunci, secara otomatis

akan diberikan rujukan kata-kata yang sudah ada pada index data. Apabila sesuai dengan kata kunci yang ingin dimasukkan bisa langsung dipilih dan di sesuaikan dengan keinginan pengguna, atau bisa juga dengan memilih kata-kata yang lain yang ada pada daftar sugesti kata-kata.

Pada gambar 3.10 sistem pencarian dimulai dari :

1. Pengguna memasukkan kata kunci. Kata kunci yang dimasukkan berupa satu kata atau dua kata sesuai dengan kebutuhan dan keinginan pengguna.
2. Kata kunci itu akan dicarikan dalam file dokumen yang ada di HDFS. File dokumen tersebut sudah memiliki index dari masing-masing kata yang terdapat dalam konteks file tersebut.
3. Pada tahap ini dilakukan text mining. Text Mining didesinisikan sebagai proses menambang data yang berupa teks dengan sumber data dari dokumen yang telah diupload, dan tujuannya adalah mencari kata-kata yang dimasukkan dalam sistem yang mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen dengan kata kunci yang dimasukkan.
4. Kata kunci tersebut akan disamakan dengan index yang ada. Apabila ditemukan kata pada index tersebut yang sesuai dengan kata kunci yang dimasukkan maka akan di sorting hasilnya. Sorting dilakukan untuk mencocokkan kata yang ada pada index dan diurutkan sesuai dengan kata kunci yang dimasukkan pengguna. Sorting merupakan pengurutan hasilnya.
5. Tampilan hasil menampilkan dokumen-dokumen yang ditemukan sesuai dengan kata kunci yang dimasukkan dalam inputan. User dapat menentukan dokumen mana yang akan diambil. Karena ditampilkan semua hasil tidak hanya satu dokumen saja.
6. Hasil yang ditampilkan berupa file-file dokumen. File dokumen dapat berupa word, excel dan pdf. Apabila tidak ada dokumen yang sesuai dengan kata kunci yang dimasukkan, maka tidak akan

ditampilkan hasilnya. Sistem akan memberitahukan bahwa dokumen tidak ada pada sistem.



Gambar 3.11 Desain Sistem Pencarian

[Halaman ini sengaja dikosongkan]

BAB IV
HASIL DAN PEMBAHASAN

Pada Bab ini akan dijabarkan proses pengujian yang dilakukan. Pengujian ini dengan menggunakan beberapa skenario dan data yang digunakan juga bervariasi. Hal ini dilakukan agar dapat dicapai hasil yang maksimal.

4.1 Pengujian

Akan dilakukan delapan skenario percobaan pengujian validasi hasil dari pencarian. Pada pengujian ini juga akan dilakukan evaluasi pengukuran keberhasilan sistem berdasarkan parameter precision, recall dan waktu komputasi. Pada Tabel 4.1 disebutkan kata-kata yang akan digunakan dalam proses pencarian. Kata-kata tersebut diambil sepuluh jenis kata.

Tabel 4. 1 Daftar Kata Kunci

No.	Kata Kunci
1	statistika
2	kata kunci
3	universitas diponegoro
4	abstrak
5	pengujian
6	klasifikasi
7	uji coba
8	desa
9	kabupaten
10	jawa

Dari kata-kata kunci yang akan dicari, disebutkan pula beberapa skenario pengujian terhadap data. Data yang digunakan sudah diperiksa apakah ada data file dokumen yang ganda atau tidak. Pada penelitian ini, data yang

digunakan tidak memiliki data ganda, data abstrak karya ilmiah berbeda antara satu dengan yang lain. Persebaran jenis file dokumen tidak ditentukan secara pasti.

Tabel 4. 2 Skenario Pengujian Data

Percobaan	Jumlah Dokumen
1	10 Dokumen
2	30 Dokumen
3	50 Dokumen
4	100 Dokumen
5	150 Dokumen
6	200 Dokumen
7	250 Dokumen
8	300 Dokumen

Pada Tabel 4.2 dijelaskan skenario pengujian data. Pengujian data dilakukan sebanyak delapan kali dengan jumlah dokumen yang berbeda-beda untuk tiap percobaan yang dilakukan. Percobaan pertama dilakukan dengan menggunakan sepuluh dokumen yang jenis filenya tidak ditentukan batasannya. Percobaan ke dua dilakukan dengan menggunakan 30 dokumen. Percobaan ke tiga dilakukan dengan menggunakan 50 dokumen. Percobaan ke empat dilakukan dengan menggunakan 100 dokumen. Percobaan ke lima dilakukan dengan menggunakan 150 dokumen. Percobaan ke enam dilakukan dengan menggunakan 200 dokumen. Percobaan ke tujuh dilakukan dengan menggunakan 250 dokumen. Percobaan ke delapan dilakukan dengan menggunakan 300 dokumen.

Pengukuran nilainya berdasarkan perhitungan precision, recall dan F-Measure serta kecepatan dari proses pencarian itu sendiri.

$$\text{precision} = \frac{\text{Jumlah dokumen relevan dengan query dan terambil.}}{\text{Jumlah seluruh dokumen yang terambil}}$$

$$\text{recall} = \frac{\text{Jumlah dokumen relevan dengan query dan terambil.}}{\text{Jumlah seluruh dokumen relevan dalam koleksi dokumen}}$$

$$\text{F-Measure} = 2 \times \frac{\text{Precision x Recall}}{\text{Precision + Recall}}$$

4.1.1 Pengujian Dengan Data 10 Dokumen

Pengujian pertama ini dengan menggunakan 10 dokumen yang dibagi menjadi 3 file dokumen berekstensi xls, 3 file berekstensi doc, dan 4 file berekstensi pdf. Kata kunci yang dimasukkan ke dalam sistem untuk dilakukan pencarian sesuai dengan kata kunci yang ada pada Tabel 4.1.

Tabel 4. 3 Pengujian 10 Dokumen

No	Kata Kunci	Search Request (detik)	Index Request (detik)	Waktu (detik)
1	statistika	0.667	0.900	0.067
2	kata kunci	0.917	0.900	0.064
3	universitas diponegoro	1.433	0.900	0.045
4	abstrak	1.433	0.900	0.072
5	pengujian	1.183	0.900	0.038
6	klasifikasi	1.267	0.900	0.052
7	uji coba	1.183	0.900	0.028
8	desa	1.350	0.900	0.052
9	kabupaten	1.433	0.900	0.035
10	jawa	1.167	0.750	0.040

Pada Tabel 4.3 dijelaskan proses searching, indexing dan waktu komputasi untuk melakukan suatu pencarian dengan kata kunci yang telah disebutkan. Dari hasil didapatkan untuk kata kunci statistika proses request search sebesar 0.667 detik (667 milidetik), proses request indexnya sebesar 0.900 detik (900 milidetik) dan waktu komputasi sebesar 0.067 detik (67 milidetik). Untuk kata kunci yang lainnya nilai perhitungannya sama seperti untuk kata kunci statistika. Dari masing-masing kata kunci didapatkan hasil yang berbeda-beda,

hal ini dikarenakan konten yang ada dalam file dokumen abstraksi berbeda-beda pula

Tabel 4. 4 Akurasi Pengujian 10 Dokumen

No	Kata Kunci	Precision	Recall	F-Measure
1	statistika	0.667	0.333	0.444
2	kata kunci	0.500	0.500	0.500
3	universitas diponegoro	0.667	0.667	0.667
4	abstrak	0.750	0.600	0.667
5	pengujian	0.750	0.750	0.750
6	klasifikasi	0.667	0.667	0.667
7	uji coba	0.667	0.571	0.615
8	desa	0.500	0.500	0.500
9	kabupaten	0.500	0.500	0.500
10	jawa	0.500	0.500	0.500

Pada Tabel 4.4 dihitung akurasi pengujian 10 dokumen untuk setiap kata kuncinya. Untuk kata kunci statistika didapatkan nilai presisi sebesar 0.667, nilai recall sebesar 0.333 dan nilai F-Measure sebesar 0.444.

4.1.2 Pengujian Dengan Data 30 Dokumen

Pengujian kedua ini dengan menggunakan 30 dokumen yang dibagi menjadi 10 file dokumen berekstensi xls, 10 file berekstensi doc, dan 10 file berekstensi pdf.

Pada Tabel 4.5 dijelaskan hasil proses searching, indexing dan waktu komputasi untuk melakukan suatu pencarian dengan kata kunci yang telah disebutkan. Dari hasil didapatkan untuk kata kunci universitas diponegoro proses request search sebesar 0.717 detik (717 milidetik), proses request indexnya sebesar 0.900 detik (900 milidetik) dan waktu komputasi sebesar 0.038 detik (38 milidetik). Hasil untuk

pengujian 30 dokumen ada yang memiliki perbedaan untuk beberapa kata kunci, hal tersebut karena jumlah dokumen tersebut juga lebih banyak.

Tabel 4. 5 Pengujian 30 Dokumen

No	Kata Kunci	Search Request (detik)	Index Request (detik)	Waktu (detik)
1	statistika	0.717	0.900	0.038
2	kata kunci	1.183	0.950	0.052
3	universitas diponegoro	1.267	0.900	0.028
4	abstrak	1.350	0.950	0.052
5	pengujian	1.517	0.950	0.035
6	klasifikasi	1.100	1.100	0.040
7	uji coba	1.183	1.100	0.067
8	desa	1.450	0.900	0.064
9	kabupaten	1.350	0.900	0.045
10	jawa	1.433	0.950	0.072

Tabel 4. 6 Akurasi Pengujian 30 Dokumen

No	Kata Kunci	Precision	Recall	F-Measure
1	statistika	0.400	0.333	0.364
2	kata kunci	0.667	0.500	0.571
3	universitas diponegoro	0.667	0.444	0.533
4	abstrak	0.750	1.000	0.857
5	pengujian	0.750	0.200	0.316
6	klasifikasi	0.667	0.400	0.500
7	uji coba	0.667	0.400	0.500
8	desa	0.500	0.250	0.333
9	kabupaten	0.500	0.333	0.400
10	jawa	0.500	0.500	0.500

Pada Tabel 4.6 didapatkan akurasi pengujian untuk 30 dokumen untuk setiap kata kuncinya. Untuk kata kunci universitas diponegoro didapatkan nilai presisi sebesar 0.667, nilai recall sebesar 0.444 dan nilai F-Measure sebesar 0.533.

4.1.3 Pengujian Dengan Data 50 Dokumen

Pengujian ke tiga ini dengan menggunakan 50 dokumen yang dibagi menjadi 20 file dokumen berekstensi xls, 10 file berekstensi doc, dan 20 file berekstensi pdf.

Tabel 4. 7 Pengujian 50 Dokumen

No	Kata Kunci	Search Request (detik)	Index Request (detik)	Waktu (detik)
1	statistika	1.583	0.950	0.037
2	kata kunci	1.100	0.987	0.052
3	universitas diponegoro	1.767	0.865	0.055
4	abstrak	1.183	0.750	0.057
5	pengujian	1.183	1.100	0.048
6	klasifikasi	1.183	1.130	0.026
7	uji coba	1.600	0.950	0.044
8	desa	1.350	0.900	0.081
9	kabupaten	1.733	0.983	0.031
10	jawa	1.150	0.900	0.036

Pada Tabel 4.7 dijelaskan hasil proses searching, indexing dan waktu komputasi untuk melakukan suatu pencarian dengan kata kunci yang telah disebutkan. Dari hasil didapatkan untuk kata kunci abstrak proses request search sebesar 1.183 detik, proses request indexnya sebesar 0.750 detik (750 milidetik) dan waktu komputasi sebesar 0.057 detik (57 milidetik). Hasil untuk pengujian 50 dokumen ada yang memiliki perbedaan untuk beberapa kata kunci, hal tersebut

karena jumlah dokumen tersebut juga lebih banyak dari pengujian sebelumnya.

Tabel 4. 8 Akurasi Pengujian 50 Dokumen

No	Kata Kunci	Precision	Recall	F-Measure
1	statistika	0.200	0.200	0.200
2	kata kunci	0.074	0.074	0.074
3	universitas diponegoro	0.133	0.133	0.133
4	abstrak	0.120	0.120	0.120
5	pengujian	0.231	0.231	0.231
6	klasifikasi	0.667	0.143	0.235
7	uji coba	0.667	0.400	0.500
8	desa	0.500	0.200	0.286
9	kabupaten	0.500	0.250	0.333
10	jawa	0.500	0.333	0.400

Pada Tabel 4.8 didapatkan akurasi pengujian sebanyak 50 dokumen untuk setiap kata kuncinya. Untuk kata kunci abstrak didapatkan nilai presisi sebesar 0.120, nilai recall sebesar 0.120 dan nilai F-Measure sebesar 0.120.

4.1.4 Pengujian Dengan Data 100 Dokumen

Pengujian pertama ini dengan menggunakan 100 dokumen yang dibagi menjadi 30 file dokumen berekstensi xls, 30 file berekstensi doc, dan 40 file berekstensi pdf.

Pada Tabel 4.9 dijelaskan hasil proses searching, indexing dan waktu komputasi untuk melakukan suatu pencarian dengan kata kunci yang telah disebutkan. Dari hasil didapatkan untuk kata kunci pengujian proses request search sebesar 1.383 detik (1383 milidetik), proses request indexnya sebesar 1.200 detik (1200 milidetik) dan waktu komputasi sebesar 0.054 detik (54 milidetik). Hasil untuk

pengujian 100 dokumen ada yang memiliki perbedaan untuk beberapa kata kunci, hal tersebut karena jumlah dokumen tersebut juga lebih banyak dari pengujian sebelumnya.

Tabel 4. 9 Pengujian 100 Dokumen

No	Kata Kunci	Search Request (detik)	Index Request (detik)	Waktu (detik)
1	statistika	0.950	1.200	0.056
2	kata kunci	1.333	1.200	0.036
3	universitas diponegoro	1.100	1.200	0.050
4	abstrak	1.183	1.200	0.060
5	pengujian	1.383	1.000	0.054
6	klasifikasi	2.233	1.200	0.059
7	uji coba	1.400	1.200	0.049
8	desa	2.083	1.200	0.061
9	kabupaten	1.283	1.200	0.031
10	jawa	1.050	1.200	0.053

Tabel 4. 10 Akurasi Pengujian 100 Dokumen

No	Kata Kunci	Precision	Recall	F-Measure
1	statistika	0.938	0.857	0.896
2	kata kunci	0.857	0.800	0.828
3	universitas diponegoro	0.818	0.794	0.806
4	abstrak	0.789	0.750	0.769
5	pengujian	0.533	0.400	0.457
6	klasifikasi	0.733	0.611	0.667
7	uji coba	0.700	0.467	0.560
8	desa	0.667	0.500	0.571
9	kabupaten	0.429	0.333	0.375
10	jawa	0.400	0.286	0.333

Pada Tabel 4.10 didapatkan akurasi pengujian dari data sebanyak 100 dokumen untuk setiap kata kuncinya. Untuk kata kunci pengujian didapatkan nilai presisi sebesar 0.533, nilai recall sebesar 0.400 dan nilai F-Measure sebesar 0.457.

4.1.5 Pengujian Dengan Data 150 Dokumen

Pengujian ke lima ini dengan menggunakan 150 dokumen yang dibagi menjadi 50 file dokumen berekstensi xls, 50 file berekstensi doc, dan 50 file berekstensi pdf.

Tabel 4. 11 Pengujian 150 Dokumen

No	Kata Kunci	Search Request (detik)	Index Request (detik)	Waktu (detik)
1	statistika	0.917	1.000	0.081
2	kata kunci	1.100	1.200	0.052
3	universitas diponegoro	1.100	1.200	0.037
4	abstrak	1.100	1.200	0.060
5	pengujian	1.100	1.200	0.048
6	klasifikasi	1.100	1.200	0.031
7	uji coba	1.283	1.200	0.026
8	desa	1.167	1.000	0.042
9	kabupaten	1.183	1.200	0.030
10	jawa	1.050	1.200	0.044

Pada Tabel 4.11 dijelaskan hasil proses searching, indexing dan waktu komputasi untuk melakukan suatu pencarian dengan kata kunci yang telah disebutkan. Dari hasil didapatkan untuk kata kunci klasifikasi proses request search sebesar 1.100 detik (1100 milidetik), proses request indexnya sebesar 1.200 detik (1200 milidetik) dan waktu komputasi sebesar 0.031 detik (31 milidetik). Hasil untuk pengujian 150 dokumen ada yang memiliki perbedaan untuk beberapa

kata kunci, hal tersebut karena jumlah dokumen tersebut juga lebih banyak dari pengujian sebelumnya.

Tabel 4. 12 Akurasi Pengujian 150 Dokumen

No	Kata Kunci	Precision	Recall	F-Measure
1	statistika	0.938	0.600	0.732
2	kata kunci	0.686	0.533	0.600
3	universitas diponegoro	0.756	0.540	0.630
4	abstrak	0.704	0.543	0.613
5	pengujian	0.536	0.484	0.508
6	klasifikasi	0.560	0.483	0.519
7	uji coba	0.783	0.600	0.679
8	desa	0.727	0.533	0.615
9	kabupaten	0.563	0.500	0.529
10	jawa	0.462	0.353	0.400

Pada Tabel 4.12 didapatkan akurasi pengujian dari data sebanyak 150 dokumen untuk setiap kata kuncinya. Untuk kata kunci klasifikasi didapatkan nilai presisi sebesar 0.560, nilai recall sebesar 0.483 dan nilai F-Measure sebesar 0.519.

4.1.6 Pengujian Dengan Data 200 Dokumen

Pengujian pertama ini dengan menggunakan 200 dokumen yang dibagi menjadi 80 file dokumen berekstensi xls, 60 file berekstensi doc, dan 60 file berekstensi pdf.

Pada Tabel 4.13 dijelaskan hasil proses searching, indexing dan waktu komputasi untuk melakukan suatu pencarian dengan kata kunci yang telah disebutkan. Dari hasil didapatkan untuk kata kunci uji coba, proses request search sebesar 0.917 detik (917 milidetik), proses request indexnya sebesar 1.300 detik (1300 milidetik) dan waktu komputasi sebesar 0.055 detik (55 milidetik). Hasil untuk pengujian

200 dokumen ada yang memiliki perbedaan untuk beberapa kata kunci, hal tersebut karena jumlah dokumen tersebut juga lebih banyak dari pengujian sebelumnya.

Tabel 4. 13 Pengujian 200 Dokumen

No	Kata Kunci	Search Request (detik)	Index Request (detik)	Waktu (detik)
1	statistika	2.317	0.517	0.036
2	kata kunci	1.100	1.200	0.030
3	universitas diponegoro	0.917	1.233	0.055
4	abstrak	2.017	1.267	0.044
5	pengujian	1.100	1.067	0.057
6	klasifikasi	1.100	1.250	0.054
7	uji coba	0.917	1.300	0.055
8	desa	1.283	1.317	0.037
9	kabupaten	1.100	1.267	0.050
10	jawa	1.183	1.200	0.054

Tabel 4. 14 Akurasi Pengujian 200 Dokumen

No	Kata Kunci	Precision	Recall	F-Measure
1	statistika	0.500	0.337	0.403
2	kata kunci	0.444	0.308	0.364
3	universitas diponegoro	0.472	0.378	0.420
4	abstrak	0.388	0.362	0.374
5	pengujian	0.531	0.378	0.442
6	klasifikasi	0.556	0.385	0.455
7	uji coba	0.826	0.633	0.717
8	desa	0.692	0.529	0.600
9	kabupaten	0.667	0.600	0.632
10	jawa	0.733	0.579	0.647

Pada Tabel 4.14 didapatkan akurasi pengujian dari data sebanyak 200 dokumen untuk setiap kata kuncinya. Untuk kata kunci uji coba didapatkan nilai presisi sebesar 0.826, nilai recall sebesar 0.633 dan nilai F-Measure sebesar 0.717.

4.1.7 Pengujian Dengan Data 250 Dokumen

Pengujian ke tujuh ini dengan menggunakan 250 dokumen yang dibagi menjadi 90 file dokumen berekstensi xls, 75 file berekstensi doc, dan 85 file berekstensi pdf.

Tabel 4. 15 Pengujian 250 Dokumen

No	Kata Kunci	Search Request (detik)	Index Request (detik)	Waktu (detik)
1	statistika	0.917	1.000	0.057
2	kata kunci	1.283	1.200	0.048
3	universitas diponegoro	1.100	1.283	0.030
4	abstrak	1.100	1.300	0.059
5	pengujian	1.100	1.300	0.045
6	klasifikasi	1.350	1.100	0.051
7	uji coba	2.183	1.267	0.049
8	desa	2.167	1.267	0.076
9	kabupaten	1.617	1.200	0.104
10	jawa	1.750	1.200	0.047

Pada Tabel 4.15 dijelaskan hasil proses searching, indexing dan waktu komputasi untuk melakukan suatu pencarian dengan kata kunci yang telah disebutkan. Dari hasil didapatkan untuk kata kunci desa, proses request search sebesar 2.167 detik (2167 milidetik), proses request indexnya sebesar 1.267 detik (1267 milidetik) dan waktu komputasi sebesar 0.076 detik (76 milidetik).

Tabel 4. 16 Akurasi Pengujian 250 Dokumen

No	Kata Kunci	Precision	Recall	F-Measure
1	statistika	0.600	0.462	0.522
2	kata kunci	0.474	0.439	0.456
3	universitas diponegoro	0.471	0.412	0.440
4	abstrak	0.535	0.366	0.434
5	pengujian	0.444	0.369	0.403
6	klasifikasi	0.474	0.321	0.383
7	uji coba	0.750	0.553	0.636
8	desa	0.632	0.500	0.558
9	kabupaten	0.682	0.600	0.638
10	jawa	0.667	0.522	0.585

Pada Tabel 4.16 didapatkan akurasi pengujian dari data sebanyak 250 dokumen untuk setiap kata kuncinya. Untuk kata kunci desa didapatkan nilai presisi sebesar 0.632, nilai recall sebesar 0.500 dan nilai F-Measure sebesar 0.558.

4.1.8 Pengujian Dengan Data 300 Dokumen

Pengujian pertama ini dengan menggunakan 300 dokumen yang dibagi menjadi 100 file dokumen berekstensi xls, 100 file berekstensi doc, dan 100 file berekstensi pdf.

Pada Tabel 4.17 dijelaskan hasil proses searching, indexing dan waktu komputasi untuk melakukan suatu pencarian dengan kata kunci yang telah disebutkan. Dari hasil didapatkan untuk kata kunci kabupaten, proses request search sebesar 1.100 detik (1100 milidetik), proses request indexnya sebesar 1.283 detik (1283 milidetik) dan waktu komputasi sebesar 0.036 detik (36 milidetik). Sedangkan untuk kata kunci jawa didapatkan hasil proses request search sebesar 1.267 detik (1267 milidetik), proses request indexnya sebesar 1.267 detik

(1267 milidetik) dan waktu komputasi sebesar 0.055 detik (55 milidetik).

Tabel 4. 17 Pengujian 300 Dokumen

No	Kata Kunci	Search Request (detik)	Index Request (detik)	Waktu (detik)
1	statistika	0.917	1.100	0.059
2	kata kunci	1.267	1.267	0.057
3	universitas diponegoro	1.100	1.267	0.030
4	abstrak	1.450	1.200	0.048
5	pengujian	2.350	1.200	0.051
6	klasifikasi	1.750	1.317	0.037
7	uji coba	1.100	1.083	0.044
8	desa	0.917	1.317	0.030
9	kabupaten	1.100	1.283	0.036
10	jawa	1.267	1.267	0.055

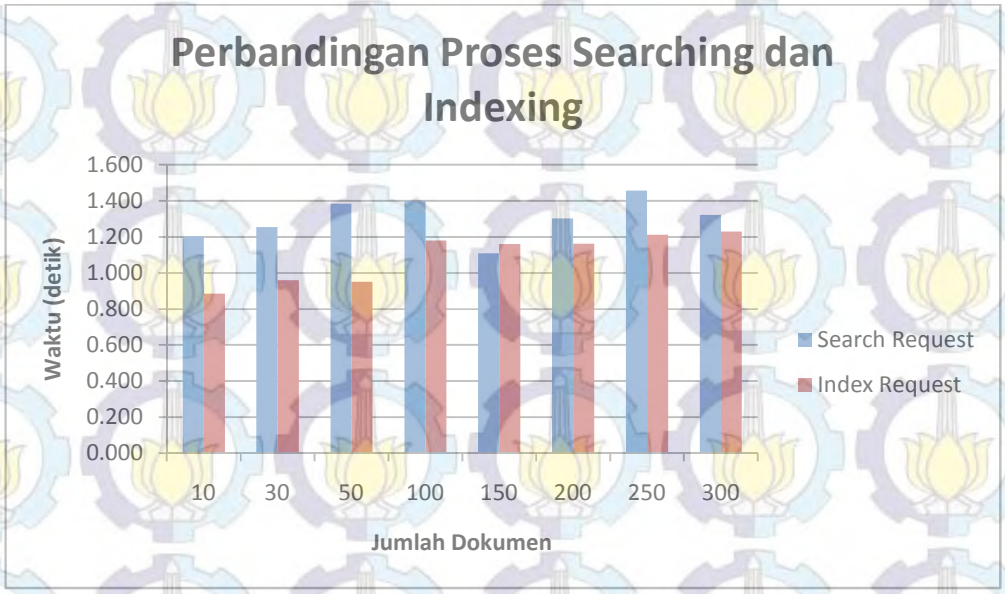
Tabel 4. 18 Akurasi Pengujian 50 Dokumen

No	Kata Kunci	Precision	Recall	F-Measure
1	statistika	0.667	0.333	0.444
2	kata kunci	0.500	0.500	0.500
3	universitas diponegoro	0.667	0.667	0.667
4	abstrak	0.750	0.600	0.667
5	pengujian	0.750	0.750	0.750
6	klasifikasi	0.667	0.667	0.667
7	uji coba	0.667	0.571	0.615
8	desa	0.500	0.500	0.500
9	kabupaten	0.500	0.500	0.500
10	jawa	0.500	0.500	0.500

Pada Tabel 4.18 didapatkan akurasi pengujian dari data sebanyak 300 dokumen untuk setiap kata kuncinya. Untuk kata kunci kabupaten didapatkan nilai presisi sebesar 0.500, nilai recall sebesar 0.500 dan nilai F-Measure sebesar 0.500. sedangkan untuk kata jawa didapatkan nilai yang hampir sama yaitu presisi sebesar 0.500, nilai recall sebesar 0.500 dan nilai F-Measure sebesar 0.500.

Dari berbagai macam percobaan yang telah dilakukan didapatkan rata-rata dari proses searching dan indexing. Pada Gambar 4.1 didapatkan hasil proses searching request yang lebih tinggi dibandingkan dengan proses index request. Proses searching request lebih lama dibandingkan dengan proses index request dikarenakan ketika proses searching dilakukan pencarian dulu terhadap node yang ada, lalu dicarikan pada index.

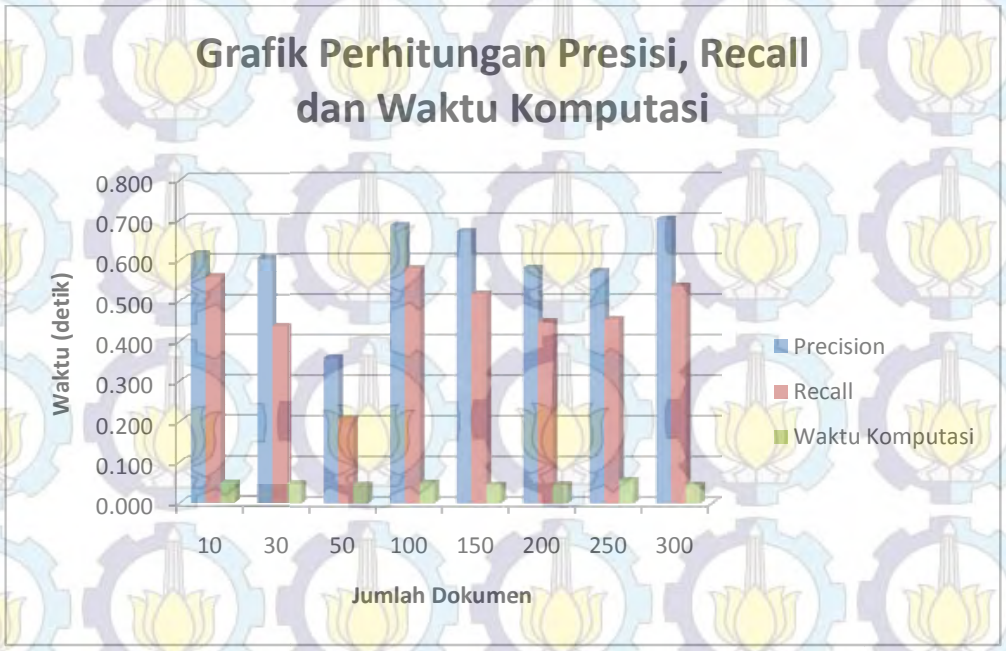
Gambar 4. 1 Hasil Perbandingan Percobaan



Pada Gambar 4.2 dijelaskan grafik perbandingan perhitungan presisi, recall dan waktu komputasi. Dari jumlah dokumen yang dilakukan ujicoba, didapatkan nilai precision yang tinggi dibandingkan nilai recall dan waktu komputasi. Waktu komputasi memiliki nilai yang paling rendah dikarenakan

kecepatan proses yang dilakukan ketika pencarian sangat cepat. Hal ini membuktikan metode ElasticSearch dapat digunakan untuk data yang sangat besar.

Gambar 4. 2 Hasil Perbandingan Percobaan



BAB V
PENUTUP

5.1 Kesimpulan

Dari percobaan yang telah dilakukan, didapatkan rata-rata waktu search requet yaitu 1.304 detik dan waktu index request yaitu 1.093 detik. Untuk rata-rata waktu pencarian kata pada dokumen sangat singkat. Dari percobaan yang dilakukandidapatkan rata-rata waktu pencarian dokumen yaitu 0.0485 detik.

Dengan menggunakan metode ElasticSearch dapat digunakan untuk proses dijadikan salah satu solusi untuk menangani big data yang terdiri dari data tidak terstruktur yang besar yaitu denganmenambahkan klasterisasi dalam indexing informasi.

5.2 Penelitian Selanjutnya

Di dalam penelitian ini terdapat beberapa kelebihan dan kekurangan sehingga membutuhkan perbaikan untuk semakin mengembangkan penelitian ini menjadi lebih baik. Adapun perbaikan yang diberikan adalah sebagai berikut:

1. Dari segi klasterisasi bisa menggunakan optimasi dari algoritma yang bersangkutan. Hal ini bertujuan untuk menghasilkan klaster dan waktu komputasi yang lebih baik dan lebih cepat. Perlu dicoba untuk algoritma klasterisasi yang lain sehingga dapat dilihat optimasinya.
2. Dari indexing informasi bisa juga menggunakan bahasa lain selain bahasa indonesia. Hal ini bertujuan agar proses temu kembali informasi berbasis big data berlaku secara umum.
3. Pada penelitian ini menggunakan single node sebagai ujicobanya, bisa dilakukan penelitian untuk ujicoba dengan menggunakan beberapa node, agar dapat dihasilkan sistem terdistribusi yang lebih besar.

4. Pada proses pencarian dan penyimpanan dokumen ini hanya terbatas pada dokumen word(doc), PDF, dan Excel(xls). Diharapkan pengembangan berikutnya tidak hanya berupa dokumen tersebut saja, namun bisa bermacam file ppt, docx, xlsx, dan lain-lain.

BAB V
PENUTUP

5.1 Kesimpulan

Dari percobaan yang telah dilakukan, didapatkan rata-rata waktu search requet yaitu 1.304 detik dan waktu index request yaitu 1.093 detik. Untuk rata-rata waktu pencarian kata pada dokumen sangat singkat. Dari percobaan yang dilakukandidapatkan rata-rata waktu pencarian dokumen yaitu 0.0485 detik.

Dengan menggunakan metode ElasticSearch dapat digunakan untuk proses dijadikan salah satu solusi untuk menangani big data yang terdiri dari data tidak terstruktur yang besar yaitu denganmenambahkan klasterisasi dalam indexing informasi.

5.2 Penelitian Selanjutnya

Di dalam penelitian ini terdapat beberapa kelebihan dan kekurangan sehingga membutuhkan perbaikan untuk semakin mengembangkan penelitian ini menjadi lebih baik. Adapun perbaikan yang diberikan adalah sebagai berikut:

1. Dari segi klasterisasi bisa menggunakan optimasi dari algoritma yang bersangkutan. Hal ini bertujuan untuk menghasilkan klaster dan waktu komputasi yang lebih baik dan lebih cepat. Perlu dicoba untuk algoritma klasterisasi yang lain sehingga dapat dilihat optimasinya.
2. Dari indexing informasi bisa juga menggunakan bahasa lain selain bahasa indonesia. Hal ini bertujuan agar proses temu kembali informasi berbasis big data berlaku secara umum.
3. Pada penelitian ini menggunakan single node sebagai ujicobanya, bisa dilakukan penelitian untuk ujicoba dengan menggunakan beberapa node, agar dapat dihasilkan sistem terdistribusi yang lebih besar.

4. Pada proses pencarian dan penyimpanan dokumen ini hanya terbatas pada dokumen word(doc), PDF, dan Excel(xls). Diharapkan pengembangan berikutnya tidak hanya berupa dokumen tersebut saja, namun bisa bermacam file ppt, docx, xlsx, dan lain-lain.