WEB CRAWLER TERDISTRIBUSI MENGGUNAKAN APACHE SPARK UNTUK DATA KEPARIWISATAAN PULAU LOMBOK

(*Distributed Web Crawler Using Apache Spark for Lombok Island Tourism Data***)**

Muhammad Qalbu Dary, Heri Wijayanto, I Gde Putu Wirarama.
Dept Informatics Engineering, Mataram University
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
Email: muhammadqdary@gmail.com, [heri, wirarama]@unram.ac.id

Abstract

Hadoop adalah salah satu cluster computer yang saat ini paling popular. Hadoop memiliki sebuah framework bernama MapReduce yang digunakan untuk memproses, dan menganalisis Big Data dengan cara membagi data menjadi beberapa bagian dan memprosesnya di mesin yang berbeda. Apache juga merilis Spark pada tahun 2016 yang juga merupakan framework MapReduce untuk memperbaharui Hadoop MapReduce yang konvensional. Spark memiliki kecepartan 100 kali lebih cepat dibandingkan dengan Hadoop MapReduce. Web Crawling adalah program untuk menjelajahi situs web dan mengambil data secara metodis, otomatis, dan teratur. Penelitian ini mengimplementasikan web crawling dengan sistem terdistribusi menggunakan Hadoop Cluster dan Apache Spark untuk mengumpulkan informasi-informasi mengenai potensi wisata yang ada di Pulau Lombok. Sumber data yang akan digunakan adalah beberapa website portal berita wisata yang ada di Indonesia. Pengujian dilakukan dengan membandingkan penggunaan web crawler tanpa menggunakan Spark dan web crawler menggunakan Spark dengan jenis pengujian variasi node, CPU core dan RAM yang digunakan. Hasil pengujian menunjukkan bahwa pengunaan web crawler menggunakan Spark memiliki waktu proses crawling yang lebih cepat dibandingkan web crawler tanpa menggunakan Spark. Pengunaan jumlah node yang digunakan juga berpengaruh terhadap waktu proses crawling dimana penggunaan resource manager YARN memiliki waktu proses crawling yang lebih cepat dibandingkan Spark cluster. Pada pengujian variasi CPU core dan RAM, jumlah CPU core yang digunakan sangat berpengaruh dengan waktu proses crawling sedangkan jumlah RAM hanya berpengaruh jika jumlah CPU core yang digunakan pada proses crawling lebih banyak. Dapat disimpulkan dari penelitian ini bahwa pengunaan program web crawler menggunakan Spark memiliki keunggulan waktu proses crawling lebih cepat dengan rata-rata 3.7 menit untuk setiap jumlah node yang ditambahkan dan 2.7 menit waktu proses crawling lebih cepat untuk setiap 2 CPU core yang ditambahkan.

Keywords: *Web Crawling*, Hadoop, MapReduce, Apache Spark

1. PENDAHULUAN

Pesatnya perkembangan teknologi saat ini membuat pekerjaan penggunanya menjadi lebih mudah, dimana pekerjaan yang seharusnya membutuhkan banyak waktu dapat diselesaikan dengan cepat. Salah satu teknologi yang sering digunakan manusia pada aktivitas sehari-hari adalah teknologi informasi. Banyak instansi seperti instansi pendidikan, pemerintahan, parawisata menggunakan perangkat lunak untuk memudahkan pekerjaan mereka.

Pulau Lombok merupakan salah satu pulau yang berada di Provinsi Nusa Tenggara Barat. Secara geografis pulau Lombok memiliki potensi wisata alam yang sangat besar. Lokasi wisata yang ada di pulau Lombok sangatlah beragam mulai dari pantai, pegunungan, air terjun, dan desa wisata yang menjadikan Pulau Lombok sebagai salah satu tujuan wisata untuk wisatawan asing maupun wisatawan lokal [1].

Seiring banyaknya informasi maka data yang dihasilkan sangatlah besar yang disebut dengan *Big Data*. *Big Data* merupakan sekumpulan data yang sangat besar dan mencakup segala jenis data yang terstruktur maupun data yang tidak terstruktur. Sebuah data dapat disebut dengan *Big Data* jika memenuhi kriteria "5V" yaitu, *velocity* yang merupakan kecepatan data bertambah, *volume* yang merupakan besar kapasitas data, *value* yang merupakan nilai dari data tersebut, *variety* yang merupakan variasi data yang beragam, dan *veracity*

yang merupakan kualitas dan keakuratan data [2]. Oleh sebab itu, saat ini *Big Data* banyak digunakan oleh perusahaan-perusahaan besar seperti Google, Microsoft, dan Amazon untuk meningkatkan kualitas layanan mereka.

Hadoop Merupakan salah satu *cluster computer* yang saat ini paling popular. Hadoop memiliki sebuah *framework* bernama MapReduce. MapReduce digunakan untuk memproses, dan menganalisis *Big Data* dengan cara membagi data menjadi beberapa bagian dan memprosesnya di mesin yang berbeda [3]. Apache juga merilis Spark pada tahun 2016 yang juga merupakan framework MapReduce untuk memperbaharui Hadoop MapReduce yang konvensional. Spark memiliki kecepartan 100 kali lebih cepat dibandingkan dengan Hadoop MapReduce [4].

Web crawling adalah proses membaca dan menyimpan seluruh konten yang ada pada sebuah website secara metodis, otomatis, dan teratur untuk pengumpulan data dan softwarenya disebut dengan *Web crawler*. *Web crawler* memiliki istilah lain seperti *automatic indexer*, *robot*, *web spiders* atau *robot web*. *Web crawler* adalah bot perangkat lunak. Biasanya, proses *crawling* dimulai dengan daftar URL yang akan diakses disebut *seeds*. Kemudian URL ini akan diakses satu per satu oleh *web crawler* dan dilakukan pengambilan data dengan membaca dan menyimpan konten yang ada pada website tersebut.

Untuk mempercepat proses *crawling* maka pekerjaan ini dapat dikerjakan secara paralel pada beberapa komputer atau *Web crawler* terdistribusi. *Web crawler* terdistribusi dapat menggunakan Hadoop *Cluster* untuk menjalankan proses *web crawling*. *Web crawler* terdistribusi digunakan karena sumber data yang akan diproses sangat banyak, hal ini dikarenakan setiap sumber data akan menghasilkan sumber data baru yang lain. Data baru yang berhasil didapatkan kemudian akan dilakukan proses *crawling* ulang sampai tidak ada lagi sumber data baru yang didapatkan.

Penelitian ini memfokuskan pada implementasi *web crawler* untuk mengumpulkan informasi mengenai wisata yang ada di pulau Lombok dari berbagai sumber data berupa website wisata yang ada di Indonesia untuk dilakukan proses *crawling*.

2. TINJAUAN PUSTAKA

Salah satu penelitian terkait berjudul "Implementasi *Web Crawling* Untuk Mengumpulkan Informasi Wisata Kuliner Di Bandar Lampung" ditulis oleh Hanifah dkk. Pada penelitian tersebut, proses *web crawling* dilakukan untuk mengumpulkan informasi mengenai wisata di Bandar Lampung dengan website yang dijadikan sumber data adalah TripAdvisor [5]. Penelitian ini tidak menggunakan *web crawler* terdistribusi dikarenakan sumber data yang digunakan hanya berasal dari website TripAdvisor jadi penggunaan *web crawler* terdistribusi tidak memberikan nilai tambah pada penelitian ini.

Penelitian terkait kedua berjudul "Perancangan Aplikasi *Web Crawler* untuk Menghasilkan Dokumen Teks pada Domain Tertentu" ditulis oleh Halim dkk. Pada penelitian tersebut proses *web crawling* dibuat dalam bentuk aplikasi website yang menyimpan dan membaca beberapa konten website yang telah dilakukan proses *crawling* kemudian dilakukan pengujian F-*Measure* dan *Black Box Testing* untuk mengecek hasil data yang telah didapatkan sesuai atau tidak [6]. Penelitian ini tidak menggunakan *web crawler* terdistribusi dikarenakan sumber data yang digunakan hanya berasal dari beberapa website yang sudah ditentukan. Jadi penggunaan *web crawler* terdistribusi tidak memberikan nilai tambah pada penelitian ini dikarenakan sumber data yang digunakan tidak akan bertambah selama proses *web crawling* berjalan.

Penelitian terkait ketiga berjudul "Membangun *Web Crawler* Berbasis Web Service Untuk Data *Crawling* Pada Webstie Google Play Store" ditulis oleh L. Ilmawan. Pada penelitian tersebut menggunakan program *web crawler* berbasis *web service* dengan aristektur REST (*Representational State Transfer*) yang mendukung pengunaan proses *web crawling* secara *cross platform* [7]. Penelitian ini tidak menggunakan *web crawler* terdistribusi dikarenakan sumber data yang digunakan hanya berasal dari website Google Play dan diutamakan untuk penggunaan secara *cross platform*. Jadi penggunaan *web crawler* terdistribusi tidak memberikan nilai tambah pada penelitian ini dan hanya akan mempersulit perancangan sistem.

Penelitian terkait keempat berjudul "Aplikasi Berbasis Web dengan Metode *Crawling* sebagai Cara Pengumpulan Data untuk Mengambil Keputusan" ditulis oleh Suharno dkk. Pada penelitian tersebut aplikasi *web crawler* dibuat dalam bentuk website yang menyimpan dan membaca konten website sosial media untuk dilakukan proses *web crawling* kemudian data dari hasil *web crawling* tersebut dilakukan analisis apakah informasi yang didapatkan dari proses *web crawling* benar atau tidak [8]. Penelitian ini tidak menggunakan *web crawler* terdistribusi dikarenakan sumber data yang digunakan hanya berasal dari beberapa website sosial media yang sudah ditentukan

dan aplikasi *web crawler* dibuat dalam bentuk website. Jadi penggunaan *web crawler* terdistribusi tidak memberikan nilai tambah pada penelitian ini dikarenakan hanya akan mempersulit pengimplementasian aplikasi website *web crawler* tersebut dan sumber data sudah menyediakan API untuk mempermudah proses *web crawling*.

2.1. Big Data

Big Data merupakan sebuah data yang sangat besar yang mencakup segala jenis data yang terstruktur maupun data yang tidak terstruktur. Sebuah data dapat disebut dengan *Big Data* jika memenuhi kriteria "5V" yaitu merupakan singkatan dari v*elocity*, *volume*, *value*, *variety*, dan *veracity*. *Velocity* pada *Big Data* merupakan kecepatan pertambahan data yang sangat pesat. *Volume* artinya jumlah data yang sangat besar, *value* artinya *Big Data* memiliki nilai yang tinggi, *variety* artinya variasi yang sangat beragam, dan *veracity* pada *Big Data* merupakan kualitas dan keakuratan data [2]. Oleh sebab itu, saat ini *Big Data* banyak digunakan oleh perusahaan-perusahaan besar seperti Google, Microsoft, dan Amazon untuk meningkatkan kualitas layanan mereka [9].

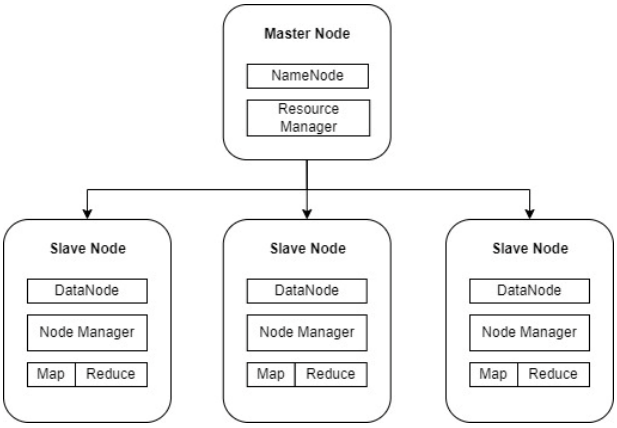
2.2. *Web Crawling*

Web crawling adalah proses membaca dan menyimpan konten penting yang ada pada sebuah website secara metodis, otomatis, dan teratur dan softwarenya disebut dengan *web crawler*. *Web crawler* memiliki istilah lain seperti *automatic indexer*, *robot*, *web spiders* atau *robot web*. *Web crawler* adalah bot perangkat lunak.

Biasanya, proses *crawling* dimulai dengan daftar URL yang akan diakses disebut *seeds*. Kemudian URL ini akan diakses satu per satu oleh *web crawler* dan dilakukan pengambilan data dengan membaca dan menyimpan konten yang ada pada website tersebut. Website yang telah dilakukan proses *web crawling* akan diidentifikasi kontennya apakah terdapat *hyperlink* didalamnya, jika ada maka akan ditambahkan ke dalam list URL yang akan dilakukan proses *web crawling* selanjutnya, ini biasanya disebut dengan *crawl frontier* [6].

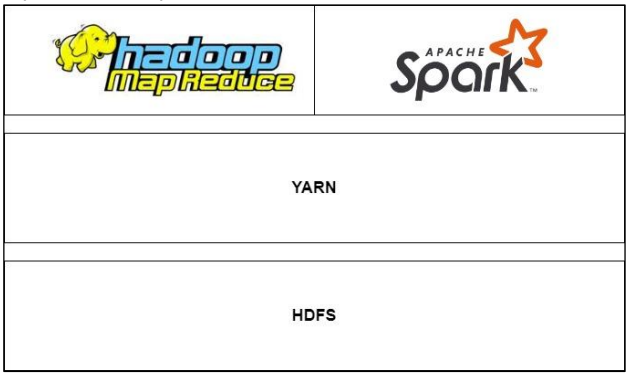
2.3. Hadoop *Cluster*

Hadoop adalah salah satu *cluster computer* yang saat ini paling populer. *Cluster computer* adalah sekelompok komputer yang saling terkoneksi untuk bersama-sama memproses data. Setiap komputer dalam *cluster* komputer biasa disebut sebagai *node*. Arsitektur dari Hadoop *Cluster* diperlihatkan pada Gambar 1.



Gambar 1. Arsitektur Hadoop *Cluster*

Hadoop *Cluster* terdiri dari 1 *Master Node* dan *n-Slave Node*. *Master Node* memiliki komponen *NameNode* dan *Resource Manager*. Kemudian setiap *Slave Node* memiliki *DataNode* dan *Node Manager* yang akan menjalankan fungsinya sebagai *Map* atau *Reduce*. Sebuah *cluster computer* bisa dipandang sebagai sebuah komputer, dimana memiliki tempat penyimpanan dan prosesor. Seperti halnya komputer desktop, *cluster computer* juga memiliki fungsi penyimpanan data dan pemrosesesan. Penyimpanan data didalam Hadoop *Cluster* di tangani oleh HDFS (Hadoop Distributed File System) yang dikelola secara terdistribusi. Untuk Pemrosesan didalam Hadoop *Cluster* menggunakan *YARN* (Yet Another Resource Manager), HDFS terdiri dari NameNode dan DataNode, *YARN* terdiri dari Resource Manager dan Node Manager [11]. Ekosistem dari Hadoop *Cluster* diperlihatkan pada Gambar 2.

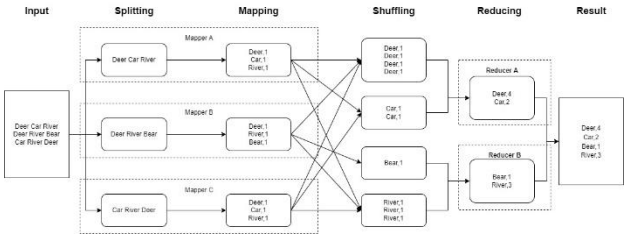


Gambar 2. Ekosistem Hadoop *Cluster*

2.4. MapReduce

MapReduce merupakan framework untuk memproses data di lingkungan komputer terdistribusi. Pemrosesan data di Hadoop *Cluster* menggunakan MapReduce Framework. MapReduce terdiri dari 2

tahap, yaitu *Map* dan *Reduce*. Fungsi *Map* dikerjakan oleh banyak *Node Mapper*. Fungsi *Reduce* dikerjakan oleh satu atau beberapa *Reducer*. Sebuah proses dibagi menjadi beberapa sub proses, setiap sub proses dikerjakan oleh 1 Mapper.Hasil dari setiap Mapper akan digabungkan menjadi hasil akhir oleh Reducer. Contoh program *Word Count* yang diperlihatkan pada Gambar 3.



Gambar 3. Contoh MapReduce program *Word Count*

Input data dibagi sesuai banyak mapper secara seimbang, setiap potongan diproses secara independen oleh setiap mapper. MapReduce menggunakan struktur data pasangan <*Key*,*Value*>. Dalam contoh program Word Count ini Mapper bertugas membentuk pasangan <*Key*,*Value*> seperti <*Deer*,1> setelah proses Mapper selesai, setiap data dengan *Key* yang sama akan dikirim ke satu Reducer tertentu. Data yang memiliki *Key* yang sama maka *Value*-nya akan ditambahkan [11].

2.5. Apache Spark

Apache Spark adalah teknologi komputasi *cluster* cepat yang dirancang untuk komputasi cepat. Spark didasarkan pada Hadoop MapReduce dan memperluas model dari MapReduce untuk meningkatkan efisiensi jenis komputasi. Apache Spark dirilis pada tahun 2016 dan ditulis dengan Bahasa Pemrograman Scala. Fitur utama Apache Spark adalah komputasi *cluster* dalam memori untuk pemrosesan aplikasi yang lebih cepat. Apache Spark dirancang untuk mencakup berbagai beban kerja seperti *interactive query*, *stream processing*, *batch application*, dan *iterative algorithms* [3].

Apache Spark bekerja dengan menyimpan semua iterasi dalam memori, bukan dalam disk seperti MapReduce. sehingga Apache Spark dapat bekerja 100 kali lebih cepat dibandingkan dengan MapReduce [4].

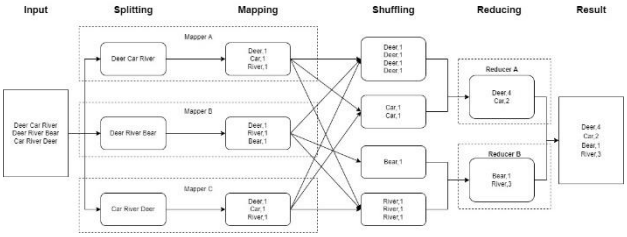
3. METODE PENELITIAN

3.1. Alat dan Bahan

Berikut merupakan alat dan bahan yang akan digunakan untuk merancang *Web Crawler* dengan sistem terdistribusi:

1. Alat Penelitian
    a. Laptop HP Omen 16
    b. 4 buah PC Intel i5
    c. Internet
2. Bahan Penelitian
    a. Sistem Operasi Windows 11 (64-bit)
    b. Sistem Operasi Ubuntu 22.04
    c. Bahasa Pemrograman Python
    d. Microsoft Word 2019
    e. Microsoft Excel 2019
    f. Oracle VM Virtualbox Manager
    g. Hadoop *Cluster*
    h. Apache Spark

3.2. Rancangan Arsitektur Sistem



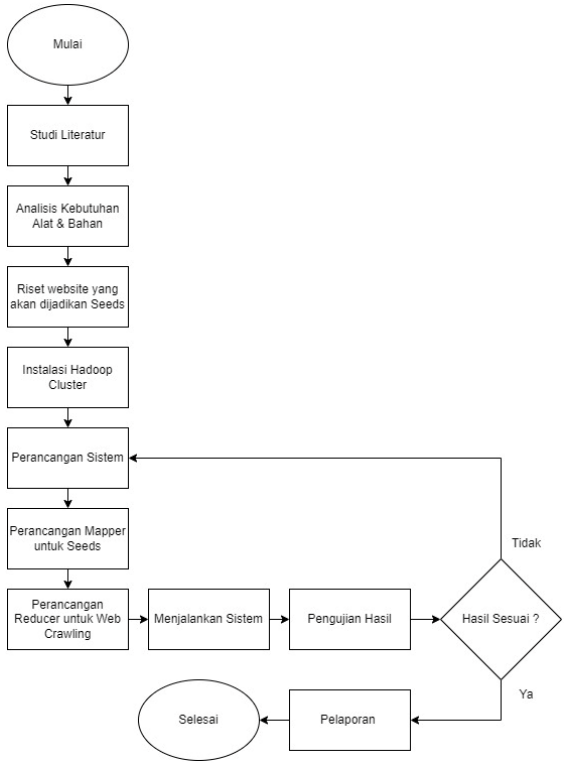
Gambar 4. Arsitektur Sistem

1. *Input*
   Input yang diberikan merupakan *seeds*/list-list URL yang akan dilakukan *crawling*, *input* terdapat bagian yaitu input awal yaitu *seeds*/list-list URL awal yang akan dilakukan *crawling*, kemudian input kedua yaitu adalah *seeds*/list-list URL baru yang didapatkan dari hasil *output crawling* yang telah dilakukan. Data *input* disimpan dengan format csv dan bahasa pemrograman yang digunakan adalah Python.
2. *Splitting*
   Pada tahap *splitting*, list URL yang ada di *seeds* dibagikan ke *mapper* yang telah dibuat dengan jumlah yang sama rata sesuai dengan banyaknya *mapper* yang ditentukan.
3. *Mapping*
   Pada tahap *mapping*, list URL yang telah dipilah pada tahap splitting dilakukan proses *web crawling* untuk mendapatkan data.
4. *Shuffling*
   Pada tahap *shuffling*, setelah proses *web crawling* pada mapper selesai maka data akan dikirimkan ke reducer dilakukan pengacakan dan menyaring data website yang sudah dilakukan *crawling* pada tahap *reducing* sebelumnya.
5. *Reducing*
   Pada tahap *reducing*, data yang sudah terkumpul pada proses *shuffling* maka akan dihitung jumlahnya, jika sudah memenuhi jumlah yang ditentukan maka data akan disimpan ke *output*, jika

belum maka data akan dilakukan pengulangan untuk proses *crawling*.

6. *Output*
   Hasil dari *web crawling* yang disimpan dengan format csv adalah informasi wisata di Pulau Lombok, dan list URL baru yang didapatkan dari proses *crawling* dilakukan *filtering* apakah sudah URL tersebut sudah dilakukan proses *crawling* sebelumnya atau tidak, jika tidak maka, URL tersebut dapat dijadikan sebagai *input* untuk dilakukan proses *crawling* hingga mencapai limit kedalaman yang telah ditentukan.

3.3. Alur Penelitian



Gambar 5. Alur Penelitian

Di dalam subbab ini dijelaskan mengenai alur penelitian yang dilakukan peneliti untuk menemukan solusi dari masalah yang ada didalam penelitian ini. Langkah pertama yaitu adalah studi literatur yang bertujuan untuk mengumpulkan informasi mengenai masalah yang diangkat di penelitian ini. Kemudian dilakukan analisis kebutuhan untuk menentukan alat dan bahan apa saja yang dibutuhkan didalam penelitian ini, Lalu dilanjutkan dengan riset website wisata indonesia yang akan dijadikan *input seeds* dari penelitian ini.

Lalu dilanjutkan dengan instalasi Hadoop Cluster untuk menjalankan HDFS (Hadoop Distributed File System) yang akan digunakan untuk menyimpan data dan menjalankan sistem terdistribusi. Peneliti kemudian melanjutkan perancangan sistem sesuai dengan arsitektur rancangan sistem yang telah dibuat, kemudian melakukan perancangan untuk mapper dan reducer. Setelah perancangan selesai kemudian sistem dijalankan dan dilakukan pengujian sistem, apabila sistem tidak sesuai atau tidak berfungsi sesuai dengan peneliti harapkan maka dilakukan perancangan sistem ulang jika sistem telah sesuai maka peneliti akan melaporkan hasil dari pengujian sistem yang telah dilakukan.

3.4. Pengujian

Pengujian dilakukan untuk mengetahui apakah program *web crawler* terdistribusi tersebut sudah berjalan sesuai dengan yang diinginkan atau terdapat masalah saat program *web crawler* dijalankan. Data yang digunakan pada tahap pengujian ini adalah data prototipe yang tidak terlalu besar sehingga dapat diawasi apakah sistem sudah berjalan dengan aturan yang telah diterapkan atau tidak. Data dibaca dan disimpan dalam format csv.

Jika program *web crawler* sudah berjalan sesuai yang diinginkan maka, proses *web crawling* dapat dimulai. Pada tahap proses *web crawling* dilakukan beberapa pengujian yaitu, berapa banyak data yang dihasilkan dari proses *web crawling* dalam rentang waktu tertentu dengan waktu sebagai variabel kontrol dan jumlah data yang didapatkan sebagai variabel pengujian. Pengujian dilakukan dengan memilih beberapa rentang waktu yang telah ditentukan dan melihat seberapa banyak data yang didapatkan dalam rentang waktu tersebut.

Kemudian pengujian tersebut dilakukan dengan beberapa skenario pengujian lainnya yaitu memvariasikan jumlah *node* yang digunakan oleh Hadoop Cluster untuk melihat perbedaan hasil yang didapatkan dengan jumlah *node* sebagai variabel kontrol. Pengujian dilakukan dengan menjalankan program *web crawler* dengan memvariasikan jumlah *node* yang digunakan oleh Apache Spark dan melihat perbedaan jumlah hasil data yang didapatkan.

Memvariasikan penggunaan RAM dan CPU *core* yang digunakan oleh Apache Spark untuk melihat perbedaan hasil data yang didapatkan dengan jumlah RAM dan CPU *core* yang digunakan sebagai variabel kontrol. Pengujian dilakukan dengan menjalankan program *web crawler* dengan memvariasikan jumlah RAM dan CPU *core* yang digunakan oleh Apache Spark dan melihat perbedaan jumlah hasil data yang didapatkan.

4. HASIL DAN PEMBAHASAN

4.1. Hasil

Pada penelitian ini hasil didapatkan dengan melaukan beberapa skenario pengujian sebagai berikut.

1. Pengujian dengan variasi jumlah node menggunakan YARN dan Spark *cluster*. Pengujian akan dilakukan dengan menjalankan program *web crawler* menggunakan Spark dengan *resource manager* YARN dan Spark cluster dengan melakukan variasi jumlah *node* yand digunakan.
2. Pengujian dengan variasi jumlah CPU Core dan RAM.

   Pengujian akan dilakukan dengan menjalankan program *web crawler* menggunakan Spark dengan *single node* dan melakukan variasi jumlah CPU *core* dan RAM yang digunakan.

Skenario yang diujikan yaitu lama waktu (menit) yang dibutuhkan untuk mendapatkan data yang sudah ditentukan, dimana setiap data merupakan halaman *website* unik yang terdapat *keyword*.

4.2. Pengujian Variasi Jumlah *Node* dengan YARN

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan untuk *crawling* data dengan variasi jumlah *node* yang digunakan. Dalam setiap pengujian variasi jumlah *node*, setiap *node* di-install dalam bentuk *virtual machine* dengan spesifikasi setiap *node* sebagai berikut.

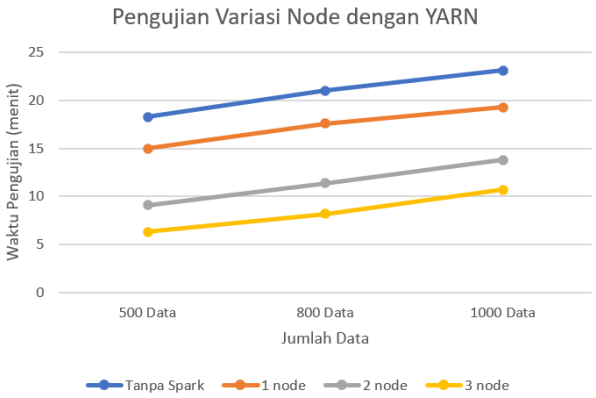
Tabel 1. Spesifikasi *node*

Komponen	Spesifikasi Komponen
OS	Ubuntu 22.04
CPU	Intel Core i5
CPU Core	3
RAM	1024 MB
Penyimpanan	80 GB

Pada pengujian ini, dilakukan variasi jumlah *node* dalam penggunaan Spark menggunakan YARN. Berikut adalah tabel dan *line graph* hasil waktu yang dibutuhkan untuk mendapatkan data hasil *crawling* menggunakan Spark dan tanpa menggunakan Spark, dengan variasi jumlah *node* yang digunakan. Jumlah data dijadikan sebagai variabel kontrol.

Tabel 2. Pengujian variasi *node* menggunakan YARN

Jenis Pengujian	Waktu yang dibutuhkan (satuan menit)		
	500 Data	800 Data	1000 Data
Tanpa Spark	18,3	21,0	23,1
1 *node*	15,0	17,7	20,3
2 *node*	9,1	11,4	14,8
3 *node*	6,3	8,2	10,7



Gambar 6. *Line graph* pengujian variasi *node* menggunakan YARN

Setiap pengujian dilakukan sebanyak 3 kali dan kemudian diambil nilai rata-ratanya dalam satuan menit. Berdasarkan hasil pengujian variasi *node* dengan YARN, pengunaan Spark pada proses *crawling* sangatlah signifikan terhadap proses waktu *crawling* dapat dilihat pada Gambar 4.1 waktu proses *crawling* menggunakan Spark dengan 1 *node*, 2 *node*, dan 3 *node* memiliki hasil yang jauh lebih cepat dibandingkan dengan tanpa menggunakan Spark. Penggunaan jumlah *node* juga memberikan peningkatan waktu proses secara signifikan hal ini dapat dilihat oleh perbedaan waktu proses *crawling* dengan jumlah *node* yang berbeda, pada penggunaan 2 jumlah *node* waktu proses *crawling* 800 data berada di rata-rata 11,4 menit sedangkan pada penggunaan 3 jumlah *node* waktu proses *crawling* 800 data berada di rata-rata 8,2 menit.

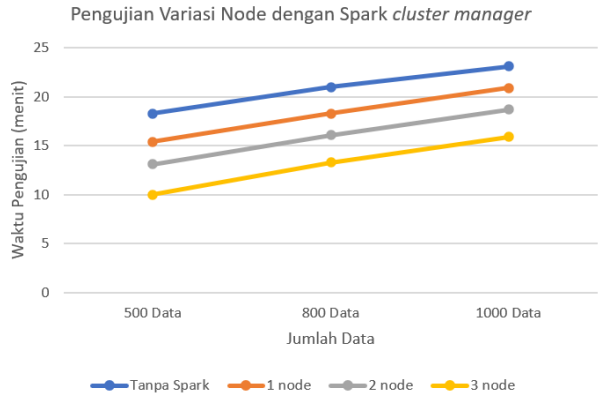
4.3. Pengujian Variasi Jumlah Node dengan Spark *cluster*

Pada pengujian ini, dilakukan variasi jumlah *node* dalam penggunaan Spark menggunakan Spark *cluster*. Berikut adalah tabel dan *line graph* hasil waktu yang dibutuhkan untuk mendapatkan data hasil *crawling* menggunakan Spark dan tanpa menggunakan Spark, dengan variasi jumlah *node* yang digunakan. Jumlah data dijadikan sebagai variabel kontrol.

Tabel 3. Pengujian variasi *node* menggunakan Spark *cluster*

Jenis Pengujian	Waktu yang dibutuhkan (satuan menit)		
	500 Data	800 Data	1000 Data

Tanpa Spark	18,3	21	23,1
1 *node*	15,4	18,3	20,9
2 *node*	13,1	16,1	18,7
3 *node*	10	13,3	15,9



Gambar 7. *Line graph* pengujian variasi *node* menggunakan Spark *cluster*

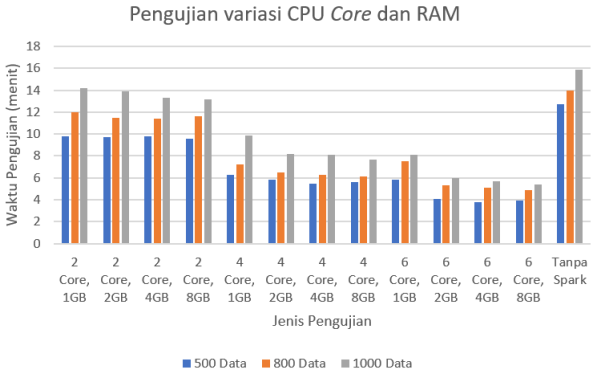
Setiap pengujian dilakukan sebanyak 3 kali dan kemudian diambil nilai rata-ratanya dalam satuan menit. Berdasarkan hasil pengujian variasi *node* dengan Spark *cluster*, sama seperti pengujian menggunakan YARN pengunaan Spark pada proses *crawling* sangatlah signifikan terhadap proses waktu *crawling* dapat dilihat pada Gambar 4.2 waktu proses *crawling* menggunakan Spark dengan 1 *node*, 2 *node*, dan 3 *node* memiliki hasil yang jauh lebih cepat dibandingkan dengan tanpa menggunakan Spark. Akan tetapi, waktu proses *crawling* menggunakan Spark *cluster* saat menggunakan *multi cluster* 2 dan 3 *node worker* membutuhkan waktu proses yang lebih lama dibandingkan dengan 2 dan 3 *node* worker menggunakan YARN.

4.4. Pengujian Variasi Jumlah CPU *Core* dan RAM

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan untuk *crawling* data dengan variasi jumlah CPU *core* dan RAM. Variabel kontrol yang digunakan pada pengujian ini adalah jumlah data yang kemudian akan dibandingkan dengan pengujian tanpa menggunakan Spark. Pengujian ini dilakukan dengan menggunakan komputer dengan spesifikasi sebagai berikut.

Tabel 4. Pengujian variasi CPU *core* dan RAM

Jenis Pengujian		Waktu yang dibutuhkan (satuan menit)		
CPU Core	RAM	500 Data	800 Data	1000 Data
2	1 GB	9,8	12,0	14,2
	2 GB	9,7	11,5	13,9
	4 GB	9,8	11,4	13,3
	8 GB	9,6	11,6	13,2
4	1 GB	6,3	7,2	9,9
	2 GB	5,8	6,5	8,2
	4 GB	5,5	6,3	8,1
	8 GB	5,6	6,1	7,7
6	1 GB	5,8	7,5	8,1
	2 GB	4,1	5,3	6,0
	4 GB	3,8	5,1	5,7
	8 GB	3,9	4,9	5,4
Tanpa Spark		12,7	14,0	15,9

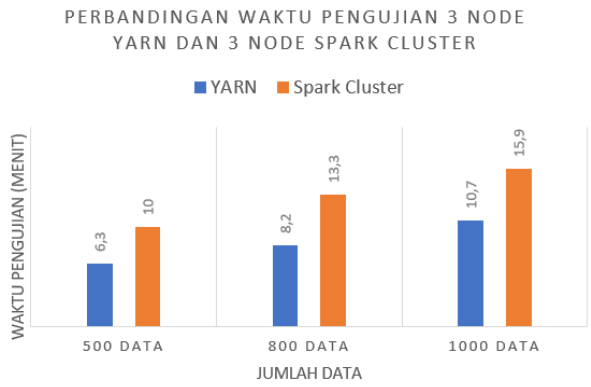


Gambar 8. *Bar graph* Pengujian variasi CPU *core* dan RAM

4.5. Pembahasan

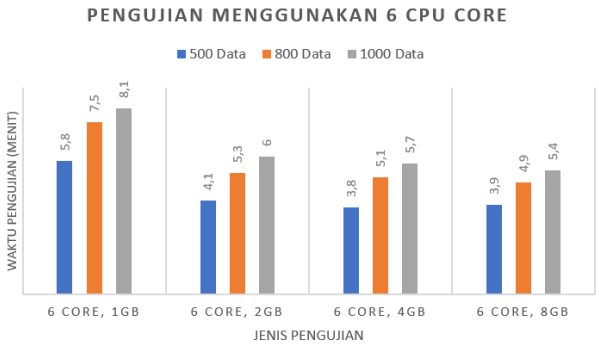
Dalam pengujian *web crawler* menggunakan Spark, ditemukan bahwa semakin banyak data yang dicrawl, maka waktu yang dibutuhkan untuk proses *crawling* akan semakin lama. Hal ini dapat dimengerti karena semakin banyak data yang harus diproses, semakin banyak waktu yang dibutuhkan untuk mengambil dan memproses informasi dari setiap halaman web.

Penggunaan Spark dalam *web crawler* terbukti memiliki pengaruh yang signifikan terhadap waktu proses *crawling*. Spark memungkinkan pemrosesan data secara terdistribusi dan paralel di beberapa node komputasi, yang dapat mengurangi waktu yang dibutuhkan untuk melakukan *crawling*. Terutama, semakin banyak jumlah node yang digunakan, maka waktu yang dibutuhkan akan semakin cepat karena pekerjaan dapat dipecah menjadi tugas-tugas yang dapat dilakukan secara paralel.



Gambar 9. *Bar graph* perbandingan waktu pengujian YARN dan Spark *cluster*

Pengujian variasi *node* dengan penggunaan Spark menggunakan resource manager YARN menunjukkan kelebihan dibandingkan penggunaan Spark *cluster*. Dalam pengujian dengan 2 atau 3 *node*. Pada Gambar 9 terlihat perbedaan waktu yang cukup signifikan pada pengujian menggunakan 3 *node*. Hal ini disebabkan oleh kemampuan YARN dalam mengelola dan membagi sumber daya secara dinamis yang lebih bagus di dalam klaster yang lebih besar, yang dapat meningkatkan efisiensi penggunaan sumber daya dan meningkatkan kinerja keseluruhan dibandingkan dengan menggunakan *cluster*.



Gambar 10. *Bar graph* perbandingan waktu pengujian 6 CPU *core*

Pada pengujian variasi jumlah CPU core dan RAM, ditemukan bahwa jumlah CPU core memiliki pengaruh yang signifikan terhadap waktu proses *crawling*. Semakin banyak jumlah CPU core yang digunakan, maka waktu proses *crawling* akan semakin cepat. Namun, dalam pengujian hanya sedikit ditemukannya relasi antara variasi jumlah RAM dengan waktu proses *crawling*. Yaitu, dimana jika penggunaan jumlah CPU core yang banyak dan jumlah RAM yang sedikit maka perbedaan waktu proses *crawling* dapat terlihat, pada Gambar 4.5 penggunaan 6 CPU *core* dengan 1GB RAM dibandingkan dengan 6 CPU *core* menggunakan 2GB, 4GB, dan 8GB memiliki perbedaan waktu proses yang cukup signifikan.

Dalam keseluruhan hasil pengujian, dapat disimpulkan bahwa penggunaan Spark dalam *web crawler* memberikan keuntungan dalam hal kecepatan proses *crawling*, terutama ketika digunakan dengan jumlah node yang lebih banyak. Penggunaan resource manager YARN juga memberikan kelebihan dalam mengelola sumber daya secara efisien. Selain itu, jumlah CPU core memiliki pengaruh yang signifikan terhadap waktu proses *crawling*, sementara RAM memiliki manfaat dalam penanganan beban kerja yang lebih besar. Hasil pengujian ini dapat menjadi acuan untuk memilih konfigurasi yang optimal dalam penggunaan Spark dalam pengembangan *web crawler*.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil pengujian yang dilakukan dapat disimpulkan *web crawler* yang menggunakan Spark memiliki keunggulan dalam hal kecepatan waktu *crawling*. Hal ini disebabkan oleh sifat terdistribusinya, yang memungkinkan *crawler* untuk bekerja secara paralel pada beberapa *node*. Dengan memanfaatkan komputasi terdistribusi, Spark dapat memproses dan mengambil data dari banyak sumber web secara efisien, meningkatkan kecepatan waktu *crawling*.

Selain itu, waktu proses *crawling* juga memiliki keterkaitan dengan jumlah CPU *core* yang digunakan. Semakin banyak jumlah CPU *core* yang tersedia, semakin cepat proses *crawling* dapat dilakukan. Dalam konteks Spark, penggunaan *multiple* CPU *core* secara efektif dapat mempercepat waktu proses *crawling*, sehingga memungkinkan pengambilan data dari web dilakukan dengan lebih efisien. Jumlah RAM yang digunakan hanya berpengaruh pada pengunaan CPU *core* yang lebih banyak jadi variasi jumlah RAM tidak begitu mempengaruhi waktu proses *crawling*.

Dalam penggunaan Spark, penggunaan master YARN lebih disarankan daripada master Spark *cluster*. Dengan menggunakan YARN, *web crawler* dapat mengoptimalkan penggunaan sumber daya komputasi, mengatasi beban kerja yang berat, dan memanfaatkan fitur-fitur seperti pembagian sumber daya yang dinamis serta pengelolaan kegagalan secara otomatis

Secara keseluruhan, menggunakan Spark dalam pengembangan *web crawler* memberikan keunggulan dalam hal kecepatan waktu *crawling* berkat sifat terdistribusinya dan keterkaitan dengan jumlah CPU core yang digunakan. Penggunaan master YARN dalam Spark juga memberikan manfaat tambahan berupa skalabilitas, pengelolaan sumber daya yang dinamis,

dan penanganan kegagalan yang efisien. Oleh karena itu, disarankan untuk memanfaatkan Spark dengan master YARN dalam pengembangan *web crawler* untuk mencapai kecepatan dan efisiensi yang optimal.

5.2. Saran

Apabila penelitian ini akan diperluas dalam penelitian lebih lanjut di masa depan, berikut adalah beberapa saran yang dapat dipertimbangkan sebagai referensi pengembangan sistem selanjutnya:

1. Pertimbangkan untuk mengembangkan fungsionalitas sistem dengan menambahkan fitur-fitur baru yang dapat meningkatkan kemampuan dan kinerja *web crawler*. Misalnya, pengembangan kemampuan ekstraksi teks dari halaman web, deteksi perubahan pada halaman web, atau integrasi dengan teknologi terkini seperti machine learning untuk analisis data yang lebih canggih.
2. Fokuskan pada peningkatan skalabilitas dan efisiensi sistem. Pertimbangkan untuk mengoptimalkan algoritma *crawling*, atau menggunakan teknologi *cluster manager* lain seperti Kubernetes.
3. Membuat *user interface* untuk memudahkan pengguna dalam mengkonfigurasi dan mengontrol proses *crawling*. Pertimbangkan untuk mengembangkan *user interface* web interaktif yang intuitif, visualisasi data yang menarik, atau kemampuan *monitoring* dan notifikasi yang lebih baik.
4. Jika sistem *web crawler* akan digunakan dalam konteks yang sensitif atau terhadap data pribadi, pertimbangkan untuk mengintegrasikan fitur keamanan yang lebih kuat. Mengikuti kebijakan dan batasan yang ditetapkan oleh pemilik situs web yang di-*crawl*.
5. Lakukan optimasi kinerja sistem dengan menganalisis dan memperbaiki bottleneck yang ada.

DAFTAR PUSTAKA

[1] G. Das, D. Gunopulos, N. Koudas, and D. Tsirogiannis, “Answering Top-k Queries Using Views,” 2006.

[2] A. V. Putu and B. S. Latif, “Dampak Perkembangan Pariwisata Pulau Lombok terhadap Pengembangan Bandar Udara Internasional Lombok The Impact of Lombok Island Tourism Development on the Development of Lombok International Airport,” Yogyakarta, Mar. 2020. doi: 10.30865/json.v3i3.3905.

[3] R. T. Aldisa, P. Maulana, and M. A. Abdullah, “Penerapan Big Data Analytic Terhadap Strategi Pemasaran Job Portal di Indonesia dengan Karakteristik Big Data 5V,” Jurnal Sistem Komputer dan Informatika (JSON), vol. 3, no. 3, p. 267, Mar. 2022, doi: 10.30865/json.v3i3.3905.

[4] P. Akas, T. Taqwin, A. B. Osmond, and R. Latuconsina, “Implementasi Metode MapReduce Pada Big Data Berbasis Hadoop Distributed File System,” e-Proceeding of Engineering Telkom University, vol. 5, no. 1, pp. 1013–1020, Mar. 2018.

[5] Apache, “Apache Spark,” 2016. https://Spark.apache.org/ (accessed Sep. 25, 2022).

[6] R. Hanifah and I. S. Nurhasanah, “Implementasi Web Crawling Untuk Mengumpulkan Informasi Wisata Kuliner Di Bandar Lampung,” vol. 5, no. 5, pp. 531–536, 2014, doi: 10.25126/jtiik20185842.

[7] A. Halim, R. D. Nyoto, and N. Safriadi, “Perancangan Aplikasi Web Crawler untuk Menghasilkan Dokumen Teks pada Domain Tertentu,” Jurnal Sistem dan Teknologi Informasi (JUSTIN) UNTAN, vol. 5, no. 2, 2017.

[8] L. Ilmawan, “Membangun Web Crawler Berbasis Web Service Untuk Data Crawling Pada Website Google Play Store,” ILKOM Jurnal Ilmiah, vol. 10, no. 2, Aug. 2018, doi: https://doi.org/10.33096/ilkom.v10i2.282.215-224.

[9] F. A. Suharno and L. Listiyoko, Aplikasi Berbasis Web dengan Metode Crawling sebagai Cara Pengumpulan Data untuk Mengambil Keputusan. 2018.

[10] R. Rusdiah, Big Data Analytics Ecosystem & Solution Dengan Apache Hadoop, 1st ed. Jakarta: Perkumpulan Basis Data Indonesia (ABDI), 2019.

[11] C. Ramadhani, Dasar Algoritma dan Struktur Data dengan Bahasa JAVA. , 1st ed., vol. 1. Yogyakarta: Andi Offset, 2015.

[12] T. White, Hadoop: The Definitive Guide, 3rd ed. O’Reilly Media, Inc., 2012.