Model Komputasi BLAST pada Lingkungan Hadoop

Devina Adinda Hartono[1], Setyorini[2], Siti Amatullah Karimah[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[4]Divisi Digital Service PT Telekomunikasi Indonesia
[1]devinaadinda@students.telkomuniversity.ac.id, [2]setyorini@telkomuniversity.ac.id,
[3]karimahsiti@telkomuniversity.ac.id

Abstrak

Mencari kemiripan pada *sequence* DNA, RNA atau protein dalam disiplin ilmu Bioinformatika bermanfaat untuk menemukan hubungan struktur, fungsi dan evolusi antar organisme. BLAST merupakan perangkat analisa kemiripan *sequence* biologi yang membandingkan satu *sequence* terhadap kumpulan *sequence* dalam suatu basis data dengan komputasi dilakukan secara berpasangan untuk semua *sequence*. Peningkatan koleksi *sequence* dalam basis data dapat memperpanjang proses pencarian similaritasnya. Hadoop Mapreduce digunakan sebagai framework komputasi yang dapat meningkatkan performa komputasi BLAST karena pada prinsipnya operasi perbandingan berpasangan adalah saling independen sehingga bisa diparalelkan. Tugas Akhir ini mengukur tingkat efisiensi komputasi BLAST dengan memanfaatkan *framework* hadoop. Hasil penelitian menunjukan *Basic Local Alignment Search Tool* (BLAST) yang dibangun pada Hadoop berturut-turut terjadi percepatan dan cluster hadoop dengan 3 node 33x lebih cepat dibanding tanpa menggunakan Hadoop.

Kata kunci: Bioinformatika, BLAST, Sequence Alignment, Hadoop, Mapreduce

Abstract

Finds the region of similarity in DNA, RNA or protein sequence on Bioinformatics is used to find structural, functional and evolutionary relationships between organisms. BLAST is a biological sequence similarity analysis tool that compares one sequence to a collection of sequences in the database with computations are performed in pairs for all sequences. Sequence collection enhancement in the database can extend the similarity search process. Hadoop Mapreduce is used as a computational framework that can improve BLAST computing performance because in principle the pairwise comparison operation is independent so that can be paralleled. This final project measure the potential for BLAST computational efficiency by utilizing the hadoop framework. The results showed that the Basic Local Alignment Search Tool (BLAST) built on was speedup and the Hadoop cluster with 3 nodes was 33 times faster than without using Hadoop.

Keywords: Bioinformatics, BLAST, Sequence Alignment, Hadoop, Mapreduce

1. Pendahuluan

1.1. Latar Belakang

Bioinformatika merupakan perpaduan disiplin ilmu Biologi dan Informatika ntuk pencarian informasi *(retrevial)*, penyimpanan data *(storage)*, manipulasi dan distribusi informasi molekul biologi seperti DNA, RNA dan protein[1]. Menemukan kemiripan pada perbandingan *sequence*, termasuk identitas, kecocokan dan homologi memberikan petunjuk tentang fungsi genetik *sequence* yang baru[2]yang banyak digunakan oleh ahli biologi. Penyejajaran sekuen (*sequence alignment*) adalah pensejajaran dua atau lebih *sequence* sehingga memperoleh tingkat kesamaan *(sequence similarity)* untuk menemukan kesamaan paling maksimum antar residu yang membantu mendapatkan informasi kemiripan fungsi, struktur, keturunan dan analisis filogenetik.

Berdasarkan jumlah *sequence* yang dibandingkan, algoritma *alignment* dibedakan menjadi *pairwise sequence alignment* dan *multiple sequence alignment*, sedangkan berdasarkan cara alignment yang dilakukan dibedakan menjadi *global alignment* dan *local alignment.* Pada *pairwise sequence alignment* perbandingan *sequence* kueri dilakukan secara berpasangan dengan semua *sequence* dalam basis data. Pencarian pada basis data harus memenuhi kriteria antara lain *sensitivity*, *selectivity* dan *speed*[1].

Dynamic programming adalah metode kuantitatif yang banyak digunakan untuk *sequence alignment* dengan hasil akurat dan reliabel namun ketika menangani *sequence* yang memiliki panjang dan jumlah yang banyak, *Dynamic Programming* terlalu lambat saat *resource* komputasi terbatas[3]. Diperlukan metode pencarian khusus untuk mempercepat proses komputasi perbandingan *sequence.* Metode heuristik merupakan salah satu solusi untuk mempercepat pencarian dengan memeriksa sebagian kecil dari kemungkinan *alignment*[1][4].

Basic Local Alignment Search Tool (BLAST) adalah versi heuristik dari *pairwise local alignment* algoritma Smith-Waterman[5]. Algoritma BLAST menggunakan teknik *seed-and-extended*[6] dalam strategi optimasi metode *word* dimana setiap *word* biasanya berisi tiga residu *sequence* protein dan 11 residu untuk *sequence* DNA sehingga sulit jika mengelola data kueri yang sangat banyak.

Keterbatasan skalabilitas membuat proses kurang efektif karena membatasi sensitivitas *alignment.* Munculnya teknologi *Big Data* seperti Hadoop dapat memberikan efisiensi komputasi sehingga kumpulan data *sequence* berukuran besar dapat diproses lebih cepat. Hadoop melakukan eksekusi secara paralel ke beberapa *worker* dengan tugas distribusi data pada HDFS dan tugas komputasi oleh MapReduce. Dengan begitu, implementasi komputasi dengan memanfaatkan *framework* hadoop berpotensi mempercepat waktu komputasi BLAST karena pada prinsipnya operasi perbandingan berpasangan adalah saling independen sehingga bisa diparalelkan.

1.2. Topik dan Batasannya

Penelitian ini menerapkan BLAST yang melakukan *sequence alignment* didalam framework Hadoop mapreduce. Dalam program HadoopBlast menerapkan *distributed cache,* untuk penyimpanan sementara program dan basis data BLAST yang dapat mengurangi jumlah pembacaan dari lokasi HDFS. Program BLAST dilakukan pada proses "mapper" dan hanya menggunakan mapper hadoop, juga dilakukan analisis performa komputasi BLAST Hadoop.

1.3. Tujuan

Tujuan dari penelitian ini adalah memodelkan aplikasi bioinformatika BLAST yang dapat melakukan *sequence alignment* dan digunakan pada framework Hadoop juga efisiensi antara *stand-alone* BLAST dan dengan menggunakan Hadoop.

1.4. Organisasi Tulisan

Jurnal ini terdiri dari lima bab utama, bab pertama merupakan Pendahuluan yaitu latar belakang, topik dan batasannya serta tujuan penelitian. Bab kedua yaitu Studi Terkait, berisi teori-teori penunjang penelitian yang dilakukan, meliputi: *sequence alignment*, metode-metode pemrosesan *sequence* biologi, BLAST dan penjelasan mengenai *framework* Hadoop (HDFS dan Mapreduce). Bab ketiga berisi Sistem yang Dibangun, menjelaskan mengenai rancangan sistem yang dibuat. Bab keempat yaitu Evaluasi, terdiri dari hasil rancangan sistem yang telah dibangun sesuai rancangan pada bab tiga. Pada bagian terakhir, bab kelima berisi kesimpulan dari hasil pengujian sistem yang dibangun serta saran untuk penelitian selanjutnya.

2. Studi Terkait

2.1. Data Genetik Protein

Protein adalah senyawa organik yang tersusun dari 20 residu asam amino[7]. Susunan dari asam amino dalam sebuah protein didefinisikan sebagai *sequence* sebuah gen yang dituliskan ke dalam kode genetik. Sebuah individual asam amino disebut residu.

Penggunaan *sequence* protein memiliki keuntungan karena *sequence* DNA hanya terdiri dari 4 nukleotida sedangkan *sequence* protein mengandung 20 asam amino[1], yang berarti ada 5x lipat peningkatan kompleksitas untuk *sequence* protein sehingga jauh lebih informatif dan sensitif dalam mendeteksi homologi. *International Union of Pure and Applied Chemistry* (IUPAC) menetapkan aturan untuk standar penulisan kode genetik protein sebagai berikut:

Tabel 1 Kode IUPAC untuk Protein[8]

1-*letter code*	3-*letter code*	Nama Asam Amino
A	Ala	Alanine
C	Cys	Cysteine
D	Asp	Aspartic acid
E	Glu	Glutamic acid
F	Phe	Phenylalanine
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
K	Lys	Lysine
L	Leu	Leucine
M	Met	Methionine
N	Asn	Asparagine
P	Pro	Proline
Q	Gln	Glutamine
R	Arg	Arginine
S	Ser	Serine
T	Thr	Threonine
V	Val	Valine

W	Trp	Tryptophan
Y	Tyr	Tyrosine

2.2. Sequence Alignment

Dalam Bioinformatika, *sequence* adalah kumpulan karakter yang mendeskripsikan urutan residu asam amino atau nukleotida yang membentuk suatu DNA, RNA atau protein.

Sequence alignment adalah proses pensejajaran (pengaturan) dua atau lebih *sequence* dengan menggeser karakter-karakter pada setiap *sequence* untuk menemukan kecocokan paling maksimum antar residu[1]. Ketidakcocokan *(mismatch)* dalam *alignment* dianggap sebagai proses mutasi dan kesenjangan *(gap)* diberi tanda "-", di asosiasikan dengan proses insersi (bertambahnya kumpulan asam amino yang baru dalam *sequence*) atau delesi (hilangnya sejumlah asam amino dalam suatu *sequence*).

Pada asam amino, *pairwise sequence alignment* adalah metode utama untuk menganalisa protein komparatif[9] dengan proses *alignment* berpasangan bertujuan untuk menemukan pasangan terbaik dari dua *sequence* dan mengidentifikasi kesamaan fungsi, sifat dan informasi genetik. Berdasarkan cara *alignment* yang dilakukan[9], dibedakan menjadi *global alignment* dan *local alignment*.

2.2.1. Global Alignment

Metode ini mengatur *sequence* dengan panjang serupa yang bertujuan untuk mencari kemungkinan *alignment* terbaik di seluruh panjang *sequence.* Untuk *sequence* dengan panjang berbeda, akan mendapat hasil kurang optimal karena gagal untuk mengenali area yang sama antar *sequence*. Algoritma yang terkenal menggunakan algoritma Needleman-Wunch[10].

```
seq1   EARDF-NQYYSSIKRSGSIQ
          . : .:::::::::. . .
seq2   LPKLFIDQYYSSIKRTMG-H
```

Gambar 1. Contoh *global alignment*

2.2.2. Local Alignment

Metode ini mengatur *sequence* dengan bagian yang paling mirip tanpa memperhatikan keselarasan panjang urutan *sequence.* Dalam perhitungan untuk menemukan *alignment* yang optimal, *local alignment* bisa digunakan untuk *sequence* dengan panjang yang berbeda dalam mencari pola *sequence* protein. Algoritma yang terkenal menggunakan algoritma Smith-Waterman[5].

```
seq1   NQYYSSIKRS
          .:::::::::.
seq2   DQYYSSIKRT
```

Gambar 2. Contoh *local alignment*

2.3. Algoritma Smith-Waterman

Algoritma Smith-Waterman adalah algoritma yang melakukan proses *local alignment* dengan cara membandingkan segmen dari dua buah *sequence (pairwise sequence alignment)*[11]. Segmen merupakan panjang deretan *sequence* dari panjang keseluruhan *sequence* yang memiliki nilai kesamaan tertinggi *(high-scoring segment pair).*

Salah satu kelebihan algoritma ini yaitu dapat digunakan untuk *sequence alignment* yang sangat panjang. Semakin panjang *sequence* berarti semakin spesifik jenisnya, sebaliknya semakin pendek *sequence* maka semakin umum jenisnya. Hasil perbandingannya lebih optimal karena algoritma Smith-Waterman membandingkan setiap segmen secara *sensitive*, yaitu membandingkan setiap karakter dalam sequence. Dan *recursive*, menggunakan fungsi yang sama untuk membandingan satu karakter dengan karakter lain sehingga relatif lebih lama.

2.4. Basic Local Alignment Search Tool (BLAST)

Basic Local Alignment Search Tool (BLAST) adalah perangkat analisa kemiripan *sequence* biologi dengan membandingkan satu *sequence* terhadap kumpulan *sequence* dalam suatu basis data, versi heuristik dari *pairwise local alignment* algoritma Smith-Waterman[5].

Perangkat ini dikembangkan oleh Stephen Altschul untuk NCBI (*National Center Biotechnology Information*) pada tahun 1990 dan menjadi salah satu program yang paling populer untuk analisis *sequence*[12]. Tujuannya untuk menemukan urutan residu kueri dengan skor segmen tertinggi atau mencapai HSP (*high-scoring segment pair*) dan mengurangi kendala waktu *alignment*.

Algoritma BLAST menggunakan teknik *seed-and-extended*[13] dimana pada satu *sequence* berisi kombinasi string yang berbeda-beda, potongan karakter string ini disebut *word.*

Seeding adalah proses pencarian kecocokan *word* yang mendekati atau identik yang selanjutnya akan dihitung skor berdasarkan identitas[14], asumsi kedua *sequence* terkait harus memiliki setidaknya satu *word* yang

cocok. Proses *extend* dengan memperluas daerah *similarity word* untuk *alignment* yang lebih panjang sehingga ditemukan skor segmen tertinggi (HSP) dari jumlah skor residu yang selaras.

Berikut cara kerja algoritma BLAST:

1. Daftar k-*letter word* dari *sequence* kueri

   Membuat daftar setiap kemungkinan *word* pada kueri, panjang setiap *word* berisi 3 residu untuk *sequence* protein dan 11 residu untuk *sequence* DNA atau didefinisikan oleh pengguna. Langkah ini disebut *"seeding".* Setelah itu, memeriksa setiap residu yang akan digunakan dalam mencari *word* pada basis data BLAST.

Gambar 3. Contoh *sequence* kueri

2. Kemungkinan word yang cocok.

   Asumsi salah satu *word* menemukan kecocokan, cari pada basis data BLAST kemunculan word. Langkah ini untuk mengidentifikasi basis data *sequence* berisi *word-word* yang cocok.

Query	PYN	PYN	PYN	PYN	...
Database	PYN	PFN	PFQ	PFE	...

Gambar 4. Daftar word pada kueri dan basis data[1]

3. Menghitung jumlah skor kecocokan *(match score)* berdasarkan Matriks BLOSUM62

   Pencocokan *word* dari masing-masing pasangan residu dengan subtitusi matriks menggunakan BLOSUM62. Untuk mengurangi jumlah kemungkinan *word* yang cocok digunakan ambang batas, word di anggap cocok jika ≥ ambang batas.

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	**9**	-1	-1	-3	0	-3	-3	-3	-4	-3	-3	-3	-3	-1	-1	-1	-1	-2	-2	-2
S	-1	**4**	1	-1	1	0	1	0	0	0	-1	-1	0	-1	-2	-2	-2	-2	-2	-3
T	-1	1	**4**	1	-1	1	0	1	0	0	0	-1	0	-1	-2	-2	-2	-2	-2	-3
P	-3	-1	1	**7**	-1	-2	-1	-1	-1	-1	-2	-2	-1	-2	-3	-3	-2	-4	-3	-4
A	0	1	-1	-1	**4**	0	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-2	-3
G	-3	0	1	-2	0	**6**	-2	-1	-2	-2	-2	-2	-2	-3	-4	-4	0	-3	-3	-2
N	-3	1	0	-2	-2	0	**6**	1	0	0	-1	0	0	-2	-3	-3	-3	-3	-2	-4
D	-3	0	1	-1	-2	-1	1	**6**	2	0	-1	-2	-1	-3	-3	-4	-3	-3	-3	-4
E	-4	0	0	-1	-1	-2	0	2	**5**	2	0	0	1	-2	-3	-3	-3	-3	-2	-3
Q	-3	0	0	-1	-1	-2	0	0	2	**5**	0	1	1	0	-3	-2	-2	-3	-1	-2
H	-3	-1	0	-2	-2	-2	1	1	0	0	**8**	0	-1	-2	-3	-3	-2	-1	2	-2
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	**5**	2	-1	-3	-2	-3	-3	-2	-3
K	-3	0	0	-1	-1	-2	0	-1	1	1	-1	2	**5**	-1	-3	-2	-3	-3	-2	-3
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	**5**	1	2	-2	0	-1	-1
I	-1	-2	-2	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	**4**	2	1	0	-1	-3
L	-1	-2	-2	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	**4**	3	0	-1	-2
V	-1	-2	-2	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	**4**	-1	-1	-3
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	**6**	3	1
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	**7**	2
W	-2	-3	-3	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	**11**

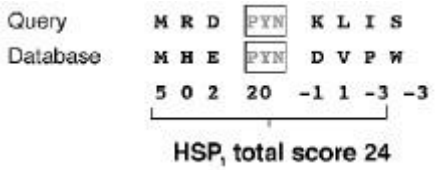
Gambar 5. Matriks BLOSUM62

Query	PYN	PYN	PYN	PYN	...
Database	PYN	PFN	PFQ	PFE	...
Sum of score	20	16	10	10	...

Gambar 6. Skor kecocokan[1]

4. Memindai basis data *sequence* yang sesuai (*match word)* dan perluas *alignment* di kedua arah.

   Memperluas alignment di kedua arah *sequence* dan menghitung skor dengan subtitusi matriks BLOSUM62 berlanjut sampai skor *alignment* turun dibawah ambang batas T (*threshold* untuk protein adalah 22 dan untuk DNA adalah 20)[1]. Perluasan berlanjut hingga skor *alignment* turun di bawah ambang batas karena *mismatch*.



Gambar 7 Perluas *alignment* dan hitung skor[1]

5. Laporan hasil skor
   Pasangan segmen sejajar yang berdekatan yang dihasilkan tanpa *(gap)* disebut *High-scoring Segment Pair* (HSP). Skor HSP tertinggi ditunjukan sebagai laporan akhir atau juga biasa disebut *maximum scoring pairs*.

Output BLAST memberikan daftar kecocokan *pairwise alignment* yang diberi peringkat berdasarkan signifikansi statistik. Skor signifikansi membantu membedakan *sequence* yang terkait secara evolusioner[14]. Indikator statistik *E-value* adalah jumlah *hits* yang diharapkan dengan skor parameter untuk memperhitungkan jumlah total *sequence alignment* yang dilakukan sebanding dengan ukuran basis data. *E-value* ditentukan oleh rumus berikut:

$$E = \square \times \square \times P$$

Dimana $\square$ adalah jumlah total residu dalam basis data, $\square$ adalah jumlah residu dalam *sequence* kueri, dan P adalah probabilitas HSP *alignment.* Jika $E > 10$, maka *sequence* tidak terkait atau terkait dengan hubungan yang sangat jauh dibawah batas metode saat ini[1].

Indikator statistik lainnya pada output BLAST adalah *bit-score,* yaitu mengukur *sequence similarity* yang tidak bergantung pada panjang sequence kueri dan ukuran basis data dan dinormalisasi[14]. Berikut rumus *bit score* ($\square'$*):*

$$\square' = \frac{\lambda \times \square - \ln K}{\ln 2}$$

Dimana λ adalah konstanta distribusi Gumble, $\square$ adalah *alignment* skor, dan K adalah konstanta skor matriks penilaian. Semakin tinggi *bit score* maka semakin baik *sequence similarity* nya.

2.5. Hadoop

Hadoop adalah perangkat lunak *open source* yang dikembangkan oleh Apache untuk pengolahan Big Data[15]. Proses penyimpanan lokal dan eksekusi program dilakukan secara terdistribusi dengan memecah data untuk dapat diproses secara paralel ke beberapa *worker* dengan tugas distribusi data pada HDFS dan tugas komputasi oleh MapReduce. Setiap *node* di hadoop dapat bertindak sebagai *master* atau *slave/worker*[16]. Master mengindeks lokasi data yang disimpan di *datanode* (HDFS *slave)* dan menkordinasikan pemrosesan data yang dijalankan di *tasktracker* (Mapreduce *slave).*

2.5.1. Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) adalah *file system* berbasis Java yang dirancang untuk menyimpan file dengan ukuran yang sangat besar [17] hingga berukuran petabytes. Data yang disimpan akan dipecah menjadi potongan-potongan data yang disebut *Block.* Selanjutnya *block file* direplikasi untuk toleransi kesalahan *(fault tolerance)* yang memiliki lebih dari satu salinan atau disebut *replication factor*. Ukuran *block* dan *replication factor* dapat dikonfigurasi per file untuk menentukan jumlah replikasi yang dapat ditentukan pada waktu pembuatan file.

Namenode (HDFS master) membuat semua keputusan terkait replikasi block, yang secara berkala menerima heartbeat dan block report dari datanode. Apabila dalam 10 menit namenode tidak menerima heartbeat dari datanode maka datanode dianggap rusak atau tidak berfungsi sehingga setiap permintaan *read* ataupun *write* dialihkan ke node lain[17].

2.5.2. Map Reduce

Mapreduce adalah *framework* Hadoop untuk komputasi paralel seperti mempartisi data input, penjadwalan, pemantauan dan komunikasi antar mesin yang diperlukan untuk *remote execution.* Komputasi dalam Mapreduce dibagi menjadi dua fase yaitu Map dan Reduce.

Proses pertama adalah melakukan input split, yaitu input data di pecah ke beberapa bagian *(shard)* sejumlah dengan banyaknya slave/worker yang di *assign.* Fase Map melakukan *read* dari *input shard*, melakukan parsing data (sesuai kode yg dibuat berdasarkan kebutuhan) dan menghasilkan *record key value pair*. Output *key value pair* dari mapper tadi menjadi input untuk fase reduce. Pada reducer, dilakukan *sorting*

key lalu seluruh *key* yang sama akan digabungkan dan mengurutkan *value*nya. Hasil *key value pair* yang baru ini yang akan menjadi file output ke pengguna.

Salah satu fasilitas yang disediakan oleh mapreduce adalah *distributed cache*. *Distributed cache* digunakan untuk menyimpan file cache yang dibutuhkan oleh aplikasi[18]. File yang dapat di cache antara lain, *read-only file, archives* dan *jar file*. Cache file ini dicopy kesetiap *worker* sebelum eksekusi dengan melakukan konfigurasi pada *driver class.* Tujuannya adalah untuk mengurangi jumlah pembacaan atau akses ke HDFS.

2.6. Speed Up

Speed up mengukur kinerja dari 2 sistem yang memproses masalah yang sama atau secara teknis peningkatan kecepatan eksekusi yang dijalankan pada arsitektur serupa dengan sumber daya berbeda. Hukum Amdahl adalah konsep dasar *speedup* yang berfokus pada pemrosesan parallel namun dapat digunakan untuk menunjukan performa setelah peningkatan resource.

Untuk merancang algoritma parallel umumnya menggunakan pendekatan *data parallelism, task parallelism* atau kombinasi antara *data parallelism* dan *task parallelism*. Pendekatan ini pada dasarnya untuk mengetahui optimasi pada saat menjalankan algoritma BLAST pada lingkungan Hadoop (*stand-alone* dan *cluster*). Parameter yang di ukur untuk mengetahui kinerja BLAST Hadoop adalah *speed up* yang dihitung menggunakan rumus:

$$\text{Speed up} = \frac{T_1}{T_p}$$

Dimana T_1 adalah waktu eksekusi pada program dengan algoritma sekuensial dan T_p adalah waktu eksekusi pada program dengan algoritma paralel[19].

3. Sistem yang Dibangun

3.1. Spesifikasi Sistem

Spesifikasi *hardware* pada *physical machine* yang digunakan,yaitu:

Tabel 2 Spesifikasi Hardware

Komponen	Spesifikasi
CPU	Intel Core i5-6200U CPU @ 2.30 GHz (4CPUs)
RAM	12 GB
Storage	931 GB

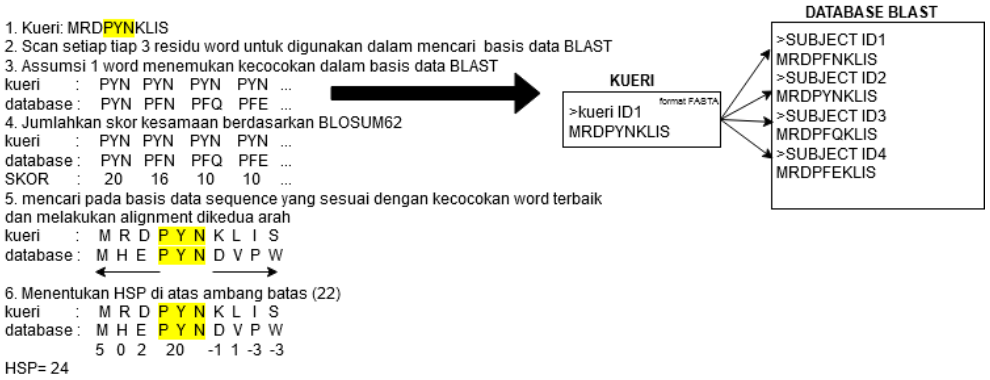
Spesifikasi *software* yang digunakan pada virtual machine sebagai berikut:

Tabel 3 Spesifikasi software

Software	Versi
Operating System	CentOS 7, 64-bit
Apache Hadoop	Hadoop 2.10.1
BLAST	Blast 2.2.23
Java	JDK 1.8
Virtualisasi	Virtual Box 6.0

3.2. Proses Independen pada BLAST

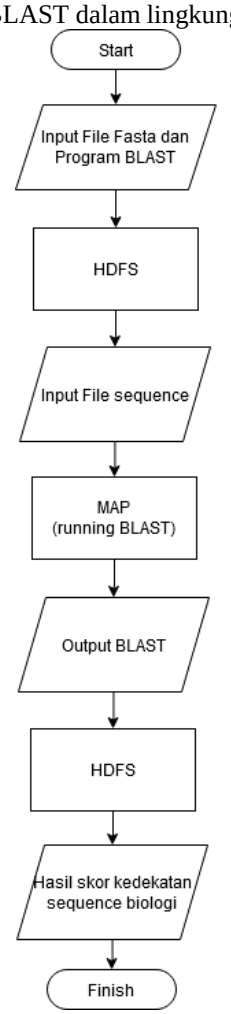
Proses independen adalah unit proses yang tidak saling bergantung satu dengan yang lainnya. Identifikasi proses independen pada komputasi BLAST diperlukan untuk kebutuhan perbandingan ketika dijalankan secara parallel dan terdistribusi. Komputasi skor dilakukan untuk semua kombinasi pasangan sequence, semakin banyak kueri dan semakin besar basis data perbandingan maka semakin lama prosesnya.



Gambar 8. Contoh kerja program BLAST
(jumlah residu kueri sequence × jumlah residu basis data sequence)

3.3 Rancangan Algoritma BLAST Hadoop

Tahapan proses pembangunan komputasi BLAST dalam lingkungan hadoop digambarkan sebagai berikut:



Gambar 9 Flowchart Sistem

Berdasarkan flowchart system dilakukan beberapa langkah untuk membangun system, yaitu:

1. Input data berisi 8 file *sequence* dengan format Fasta yang di upload dan di simpan di HDFS sebelum di distribusikan ke node-node. Program dan basis data BLAST juga di upload ke HDFS nantinya menggunakan distributed cache untuk meng-cache program ke setiap Mapper.
2. *Framework* Hadoop membaca application records dari HDFS dengan Interface InputFormat dan menghasilkan InputStream pasangan key value. Input Format di *custom* menggunakan DataFileInputFormat.java untuk menghasilkan key sebagai nama file dan value sebagai path pada HDFS untuk memasukan seluruh file ke map task.

3. Mapper bertujuan mengatur cache terdistribusi (distributed cache) dan membuat proses java untuk untuk memanggil dan eksekusi program BLAST.
4. Output file dari BLAST berisi skor kedekatan sequence yang di upload kembali ke HDFS.
5. Alternatif untuk menggabungkan beberapa file hasil dari proses mapper dapat menggunakan command Hadoop -getmerge

4. Evaluasi

4.1 Hasil Pengujian

Hasil *running* file fasta pada program BLAST dengan format output tabular terdiri dari *query id, subject id, % identity, alignment length, mismatches, gap openings, query start, query end, subject start, subject end, e-value,* dan *bit score.* Dengan keterangan sebagai berikut:

- Query id : ID sequence kueri
- Subject id : ID sequence subject (sequence yang berada dalam basis data BLAST)
- % Identity : persen sequence yang cocok antara satu sama lain dalam basis data
- Alignment length : Panjang *alignment*
- Mismatches : jumlah ketidakcocokan dalam alignment (dianggap sebagai proses mutasi)
- Gap openings : jumlah celah.
  Diasosiasikan dengan proses insersi (bertambahnya kumpulan asam amino yang baru dalam sequence) atau proses delesi (hilangnya sejumlah asam amino dalam suatu sequence)
- Query start : Mulai alignment dalam kueri
- Query end : Selesai alignment dalam kueri
- Subject start : Mulai alignment pada subject
- Subject end : Selesai alignment pada subject
- E-value : *expect value* atau nilai kemungkinan kecocokan sequence dari hasil peluang acak
- Bit score : skor kemiripan sequence.
  Skor ini tidak bergantung pada panjang kueri sequence dan ukuran basis data

```
BG3:2_30MNAAAXX:7:1:773:180/1   gi|296434297|ref|NP_001171814.1|        100.00   12      0       0       36      1       212     223     2.6     29.3

BG3:2_30MNAAAXX:7:1:773:180/1   gi|296317309|ref|NP_001171743.1|        100.00   12      0       0       36      1       212     223     2.6     29.3

BG3:2_30MNAAAXX:7:1:773:180/1   gi|298497488|gb|ADI82826.1|     91.67   12      1       0       36      1       18      29      7.6     27.7
BG3:2_30MNAAAXX:7:1:577:259/1   gi|298723306|gb|EFI64066.1|     100.00  12      0       0       37      2       265     276     9.9     27.3
BG3:2_30MNAAAXX:7:1:577:259/1   gi|296504059|ref|YP_003665759.1|        100.00   12      0       0       37      2       188     199     9.9     27.3

BG3:2_30MNAAAXX:7:1:577:259/1   gi|296503366|ref|YP_003665066.1|        100.00   12      0       0       37      2       265     276     9.9     27.3
```

Gambar 10 Hasil keluaran BLAST

Pengujian dengan menjalankan Stand-Alone BLAST dan Hadoop cluster dengan menambahkan node satu persatu untuk melihat pengaruh waktu pada proses Mapping. Kueri file yang digunakan berisi 2500, 13000 dan 25000 sequence nukleotida. Test dilakukan tiga kali dan waktu yang digunakan adalah hasil rata-rata dari *running* program.

Tabel 4 Rata-rata waktu komputasi

	Stand-Alone BLAST	**Hadoop Single-Node**	**Hadoop Cluster A 2 nodes**	**Hadoop Cluster B 3 nodes**
2500 sequence	96,319 (s)	96,443 (s)	65,188 (s)	72,630 (s)
13000 sequence	10572,413 (s)	421,736 (s)	401,773 (s)	389,589 (s)
25000 sequence	16508,768 (s)	600,74 (s)	587,162 (s)	493,212 (s)

4.2 Analisis Hasil Pengujian

Program BLAST pada *framework* Hadoop mendapatkan hasil waktu komputasi berbeda disebabkan perbedaan lingkungan yang dibangun.

Pada saat membaca 8 file yang masing-masing berisi 2500 kueri *sequence*, Hadoop single-node menghabiskan waktu yang lebih lama yaitu 0.124 detik untuk menyelesaikan pekerjaan dibandingkan dengan stand-alone BLAST. Hadoop cluster A melakukan pekerjaan 1,4x lebih cepat dibanding single-node Hadoop. Akan tetapi terjadi kenaikan waktu 7.442 detik pada Hadoop cluster B, meskipun masih lebih cepat dibandingkan penggunaan BLAST secara independen. Hal ini dikarenakan pada saat semakin sedikit kueri sequence dan node yang digunakan semakin banyak, potensi kinerja Hadoop menurun karena Hadoop harus terlebih dahulu melakukan metadata.

Sedangkan pengaplikasian BLAST pada Hadoop lebih efisien digunakan saat kueri data lebih banyak. Proses komputasi BLAST mengalami percepatan. Dilihat pada penggunaan Hadoop dengan 13000 kueri sequence,

Hadoop single-node 25x lebih cepat dibanding tanpa menggunakan Hadoop. Begitu pula pada Hadoop cluster A dan B mengalami speed up secara berturut-turut.

Pada percobaan terakhir dengan 25000 sequence, Hadoop single-node membutuhkan waktu 600,74 s sedangkan Hadoop cluster A hanya 587,162 s dan cluster B 33x lebih cepat daripada BLAST stand-alone. Dilihat dari table 4, waktu yang digunakan diambil dari total waktu *distributed cache* dan *map tasks.*

5. Kesimpulan

Dari hasil pengujian disimpulkan bahwa:

1. Model komputasi BLAST dapat dibangun menggunakan framework Hadoop.
2. Perbandingan waktu yang dihasilkan dengan menggunakan Hadoop dipengaruhi oleh jumlah node, jumlah kueri sequence dan ukuran basis data BLAST.
3. Pada saat kueri sequence lebih sedikit, framework Hadoop kurang efisien karena pekerjaan yang dilakukan lebih banyak yaitu distribusi data dan memparalelkannya sedangkan stand-alone BLAST langsung melakukan komputasi. Namun, Hadoop BLAST lebih efisien digunakan pada saat kueri data lebih banyak karena melakukan pembagian pekerjaan dengan semakin banyak node maka waktu komputasi semakin cepat.

Untuk mengembangkan hasil yang sudah didapatkan dari penelitian ini, diharapkan pada penelitian selanjutnya dapat menggabungkan file output dari BLAST dengan menggunakan reducer dan hasilnya berurutan sesuai e-value terkecil dan bitscore (skor kesamaan) terbesar.

Reference

[1] J. Xiong, *Essential bioinformatics*. Cambridge University Press, 2006.
[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990, doi: 10.1016/S0022-2836(05)80360-2.
[3] Z. M. Zhou and Z. W. Chen, "Dynamic programming for protein sequence alignment," *Int. J. Bio-Science Bio-Technology*, vol. 5, no. 2, pp. 141–150, 2013.
[4] C. Kemena and C. Notredame, "Upcoming challenges for multiple sequence alignment methods in the high-throughput era," *Bioinformatics*, vol. 25, no. 19, pp. 2455–2465, 2009, doi: 10.1093/bioinformatics/btp452.
[5] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, 1981, doi: 10.1016/0022-2836(81)90087-5.
[6] K. Arumugam, Y. S. Tan, B. S. Lee, and R. Kanagasabai, "Cloud-enabling sequence alignment with hadoop mapreduce: A performance analysis," 2012.
[7] P. Chou and G. D. Fasman, "Amino acid sequence," *Adv. Enzym. Relat. Areas Molec. Biol*, vol. 47, p. 45, 2009.
[8] P. Stothard, "The Sequence Manipulation Suite: JavaScript Programs for Analyzing and Formatting Protein and DNA Sequences," *Biotechniques*, vol. 28, p. 1102,1104, Jul. 2000, doi: 10.2144/00286ir01.
[9] V. O. Polyanovsky, M. A. Roytberg, and V. G. Tumanyan, "Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences," *Algorithms Mol. Biol.*, vol. 6, no. 1, pp. 1–12, 2011, doi: 10.1186/1748-7188-6-25.
[10] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, Mar. 1970, doi: 10.1016/0022-2836(70)90057-4.
[11] A. O'Driscoll *et al.*, "HBLAST: Parallelised sequence similarity - A Hadoop MapReducable basic local alignment search tool," *J. Biomed. Inform.*, vol. 54, pp. 58–64, 2015, doi: 10.1016/j.jbi.2015.01.008.
[12] S. Leo, F. Santoni, and G. Zanetti, "Biodoop: Bioinformatics on hadoop," *Proc. Int. Conf. Parallel Process. Work.*, no. September, pp. 415–422, 2009, doi: 10.1109/ICPPW.2009.37.
[13] M. Schatz, "BlastReduce: high performance short read mapping with MapReduce," *Univ. Maryland, http//cgis. cs. umd. edu/Grad/ …*, 2008, doi: citeulike-article-id:6496105.
[14] A. Pertsemlidis and J. W. Fondon, "Having a BLAST with bioinformatics (and avoiding BLAST phemy)," *Genome Biol.*, vol. 2, no. 10, 2001, doi: 10.1186/gb-2001-2-10-reviews2002.
[15] Apache Software Foundation, "Hadoop." https://hadoop.apache.org (accessed Dec. 19, 2020).
[16] H. Alshammari, H. Bajwa, and J. Lee, "Hadoop based enhanced cloud architecture for bioinformatic algorithms," *2014 IEEE Long Isl. Syst. Appl. Technol. Conf. LISAT 2014*, no. May, 2014, doi: 10.1109/LISAT.2014.6845204.
[17] D. Borthakur, "HDFS architecture guide. Hadoop Apache Project," pp. 1–14, 2008, [Online]. Available: http://hadoop.apache.org/hdfs/.

[18] J. Zhang, G. Wu, X. Hu, and X. Wu, "A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services," in *2012 ACM/IEEE 13th International Conference on Grid Computing*, 2012, pp. 12–21, doi: 10.1109/Grid.2012.17.
[19] D. L. Eager, J. Zahorjan, and E. D. Lozowska, "Speedup Versus Efficiency in Parallel Systems," *IEEE Trans. Comput.*, vol. 38, no. 3, pp. 408–423, Mar. 1989, doi: 10.1109/12.21127.