Penggunaan Paralelisme dan Terdistribusi untuk Pemrosesan *Phylogenetic Tree* pada *Sequence* Biologi dengan Menggunakan *MapReduce*

Renaning Karutami Susilo[1], Setyorini[2], Siti Amatullah Karimah[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]renaaks@students.telkomuniversity.ac.id, [2]setyorini@telkomuniversity.ac.id,
[3]karimahsiti@telkomuniversity.ac.id

Abstrak

***Multiple Sequence Alignment* (MSA) merupakan proses penting dalam analisis sequence biologi dengan melalukan perbandingan pada sejumlah *sequence* biologi. Pada beberapa algoritma MSA (seperti pada CLUSTALW, misalnya), pembentukan *phylogenetic tree* sebagai *guidence* dalam proses *alignment* memiliki peran penting dalam menentukan akurasi hasil akhir *alignment*. Dari keseluruhan proses MSA, pembentukan *phylogenetic tree* memerlukan waktu komputasi yang meningkat seiring dengan peningkatan jumlah *sequence*. Komputasi score similaritas untuk semua kombinasi pasangan *sequence* yang dilaksanakan secara berurutan menjadi masalah pada waktu komputasi. Tugas Akhir ini meneliti potensi efisiensi komputasi *phylogenetic tree* secara paralel dan terdistribusi pada lingkungan Hadoop menggunakan *MapReduce*. Hasil penelitian menunjukkan *phylogenetic tree* dapat dibangun dengan menggunakan komputasi *MapReduce*.**

Kata kunci : Biology sequence, Multipe Sequence Alignment, Phylogenetic Tree, Hadoop, *MapReduce*.

Abstract

Multiple Sequence Alignment (MSA) is an important process in the analysis of biological sequences by making comparisons of a number of biological sequences. In some MSA algorithms (such as CLUSTALW, for example), the formation of phylogenetic trees as a guideline in the alignment process has an important role in determining the accuracy of the final alignment results. From the whole MSA process, the formation of phylogenetic tree computational time increased as the number of sequences increases. Computational score similarity for all combinations of sequence pairs carried out sequentially becomes a problem at computational time. This Final Project examines the potential efficiency of computational phylogenetic trees in parallel and is distributed to the Hadoop environment using *MapReduce*. The results showed that phylogenetic tree can be generated using *MapReduce* computation.

Keywords: Biology sequence, Multipe Sequence Alignment, Phylogenetic Tree, Hadoop, *MapReduce*.

1. Pendahuluan

Latar Belakang

Asam dioksibirinukleat atau yang secara umum dikenal dengan DNA adalah molekul biologis yang menyimpan berbagai informasi genetika sebuah organisme. DNA merupakan asam nukleat, salah satu molekul yang hakiki bagi makhluk hidup yang ada. DNA umumnya memiliki dua untai polinukleotida yang masing-masing terdiri dari nukleotida dengan salah satu jenis basa nitrogen (guanina, adenine, timina atau sitosina) yang saling terikat menjadi satu rantai [1]. Dari rantai ini didapatkan kombinasi *string* yang berbeda-beda. Kombinasi *string* ini menyimpan banyak informasi mengenai organisme tersebut, contohnya seperti fungsi, keturunan, struktur dan lainnya [1]. Dalam bioinformatika, analisis kemiripan antar sequence dilakukan dengan metode *sequence alignment.* Informasi kemiripan pola sequence membawa kemiripan pada fungsi, keturunan, struktur dan informasi genetik [1]. Algoritma untuk melakukan s*equence alignment* diantaranya yaitu algoritma Needleman-Wunsch yang menghasilkan *global alignment* dan algoritma Smith-Waterman yang menghasilkan *local alignment*. *Global alignment* mencari nilai kecocokan tertinggi dari ujung awal DNA hingga ujung akhir DNA, sehingga hasil alignment akan sama panjang, sementara *local alignment* hanya mencari hasil kecocokan tertinggi berdasarkan bagian DNA tersebut [2].

Untuk kebutuhan membandingkan sequence dalam lebih dari 2 menggunakan metode *multiple sequence alignment* (MSA). Salah satu contohnya Clustal yang menggunakan algoritma Needleman-Wunch pada proses di dalamnya [3], ada juga T-COFFEE (*Tree Based Of Consistency Objective Function For Evaluation Alignment*) proses didalamnya menggunakan gabungan dari algoritma Smith-Waterman dan juga Needleman-Wunsch. Hasil T-COFFEE lebih akurat dibandingkan dengan Clustal karena menggunakan *local* dan *global alignment* [4]. Pada MSA dibangun sebuah *phylogenetic tree* (pohon evolusi) yang menunjukkan kedekatan tiap sekuens DNA yang ada dan merepresentasikan hubungan evolusi dalam organisme tersebut. Namun, algoritma yang digunakan untuk MSA membutuhkan banyak waktu komputasi yang lama [2]. Untuk mengatasi masalah ini, eksekusi algoritma

MSA dapat dilakukan secara paralel terdistribusi. Hadoop merupakan sebuah *framework* yang membantu dalam melakukan proses secara terdistribusi dan paralel [5]. Di dalam *framework* Hadoop terdapat sebuah metode bernama *MapReduce* dimana metode ini dapat melakukan proses paralel dari data-data berukuran besar [5]. Algoritma MSA dapat memanfaatkan *framework* Hadoop dan juga metode *MapReduce* untuk mempercepat waktu komputasi walaupun data yang diproses berukuran besar [6]. Dalam melakukan melakukan proses terhadap DNA dapat digunakan *MapReduce* yang bertugas untuk melakukan *sequence alignment* pada tiap baris DNA dari hasil tersebut kemudian dapat digunakan untuk melihat kedekatan hubungan antar baris pada DNA.

Topik dan Batasannya

Berdasarkan hal-hal yang sebelumnya telah dibahas di latar belakang, berikut ini merupakan rumusan-rumusan masalah yang akan dibahas dalam penelitian ini:

1. Bagaimana membangun *phylogenetic tree* menggunakan model komputasi *MapReduce*?
2. Proses mana yang dapat diparalelkan dalam pembangunan *phylogenetic tree*?

Adapun batasan-batasan yang didetapkan dalam menjawab rumusan masalah diatas, yaitu:

1. Menggunakan *framework* Hadoop dengan *MapReduce* dan Hadoop File System (HDFS) .
2. Metode yang digunakan untuk membangun *phylogenetic tree* adalah NJ (Neighbor Joining).

Tujuan

Tujuan dibuatnya tugas akhir ini adalah untuk membangun *phylogenetic tree* dengan menggunakan metode NJ (*Neighbor Joining*) dengan Hadoop *MapReduce* sebagai model komputasi.

Organisasi Tulisan

Pada jurnal ini, terdiri dari lima bab utama, yang pertama merupakan Pendahuluan. Bab kedua merupakan Studi Terkait. Bab ini berisi teori-teori penunjang penilitian yang dilakukan, yang dibahas disini meliputi Bioinformatika, Algoritma-algoritma yang digunakan dalam pemrosesan sekuens biologi, Penjelasan mengenai *framework* Hadoop dan juga model komputasi *MapReduce*. Selanjutnya bab ketiga yaitu Sistem yang Dibangun, pada bab ini dijelaskan mengenai rancangan sistem yang akan dihasilkan. Bab keempat yaitu Evaluasi. Pada bab ini terdiri dari hasil rancangan sistem yang telah dibangun yang dilakukan sesuai rancangan sistem pada bab tiga. Terakhir yaitu bab kelima, berisi tentang kesimpulan dari hasil pengujian sistem yang dibangun dan juga saran untuk penelitian yang akan dilakukan selanjutnya.

2. Studi Terkait

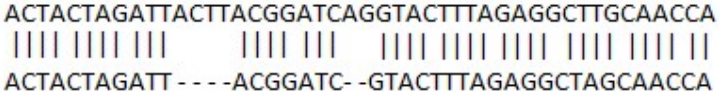
Sequence Alignment

Bioinformatika adalah ilmu yang mempelajari mengenai penerapan teknik komputasional dalam informasi di bidang biologi. [7] Pada bidang ini digunakan penerapan metode matematika, statistika dan informatika dalam memecahkan berbagai permasalahan biologi. Salah satu masalah populer pada bidang bioinformatika adalah mengenai deretan DNA atau protein lainnya. [7]

Multiple Sequence Alignment atau yang biasa disebut MSA adalah cara menyusun urutan baris DNA untuk mengidentifikasi suatu kesamaan sifat, fungsi dan informasi genetik lainnya dengan melibatkan lebih dari dua buah baris DNA, sehingga disebut *multiple.* MSA ini memiliki berbagai macam algoritma, contohnya adalah algoritma Needleman-Wunsch dan algoritma Smith- Waterman. [2]

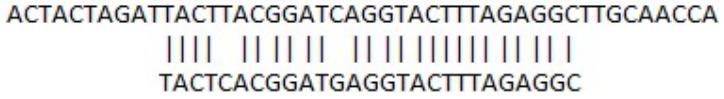
1. Algoritma Needleman-Wunsch

   Algoritma ini digunakan untuk melakukan *global alignment* dimana panjang sekuens seluruhnya terlibat dalam perhitungan yang dilakukan untuk menemukan *alignment* yang optimal dari sepasang DNA ini. [2]



Gambar 1 Contoh Global Alignment

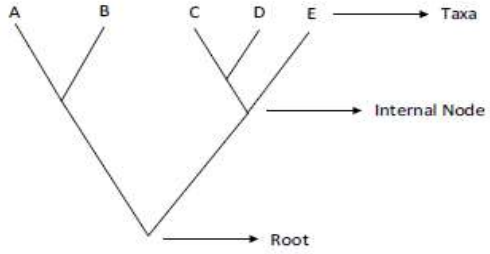
2. Algoritma Smith-Waterman
   Algoritma ini digunakan untuk melakukan *local alignment* dimana hanya sebagian dari panjang sekuens yang digunakan dalam perhitungan untuk menemukan *alignment* yang optimal. [2]



Gambar 2 Contoh Local Alignment

Phylogenetic Tree

Hasil dari algoritma diatas merupakan skor yang merepresentasikan kedekatan dari dua *sequence* DNA dan juga bentuk sequence alignment yang baru, skor ini dapat digunakan untuk membangun sebuah pohon kedekatan atau yang biasa disebut *phylogenetic tree*. *Phylogenetic Tree* atau Pohon Filogenetika adalah diagram percabangan (*tree*) yang menunjukkan hubungan evolus atau kedekatan antar berbagai spesies makhluk hidup. [8] Pohon Filogenetika ini dapat digunakan untuk membuat sistematika biologi, contohnya adalah pohon kehidupan. Selain itu, pohon ini dapat berperan untuk mencari fungsi/sifat suatu gen atau protein.



Gambar 3 *Phylogenetic Tree*

Berikut bagian-bagian yang ada pada *phylogenetic tree:*
1. Root: Merupakan akar awal terjadinya percabangan pada sebuah tree.
2. Taksa: Bagian ujung pada sebuah cabang, merepresentasikan spesies atau sekuens
3. Internal Node: Titik yang menghubungkan dua cabang, node merepresentasikan keturunan. [9]

Dalam membangun *phylogenetic tree* dapat menggunakan bebagai metode. Metode yang banyak digunakan adalah UPGMA (Unweighted Pair Group Method using Arithmetic Average) dan NJ (Neighbor Joining)

1. UPGMA (Unweighted Pair Group Method using Arithmetic Average)
   Menggunakan sekuensial clustering method. Menggabungkan 2 taksa dengan nilai *pairwise distance* terkecil pada *matrix distance*. Kemudian dihitung lagi dengan menempatkan node ditengah antara dua taksa itu, kemudian dihitung lagi nilainya sehingga terbentuk matriks yang terlah terreduksi. [9]

   Misalkan ada 4 buah sekuens biologi yaitu A, B, C dan D, dengan menggunakan rumus kombinasi:

$$C(4,2) = \frac{4!}{(2!\,(4-2)!)} = \frac{24}{4} = 6$$

   maka terdapat 6 pasangan sekuens. Berikut adalah himpunan pasangan sekuens dengan masing-masing skornya {(AB,3),(AC,3),(AD,6),(BC,7),(BD,8),(CD,9)}.

   a. Membangun matriks
      Ukuran matriks yang dibangun sama dengan jumlah sekuens yang dimiliki, contohnya disini ada 4 sekuens maka matriks yang dibangun berukuran 4x4. Matriks ini kemudian diisi dengan skor tiap pasangan sekuens berdasarkan himpunan skor.

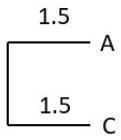
	A	B	C	D
A	0	5	3	6
B	5	0	7	8
C	3	7	0	9
D	6	8	9	0

Gambar 4 Matriks skor kedekatan

b. Mencari *alignment* terdekat dalam matriks
Dari matriks yang telah dibangun, cari dua *alignment* yang terdekat, dilihat dari skornya yg terkecil, kemudian dua *alignment* dibentuk menjadi satu node karena cabang *equidistant* maka skor AC dibagi dua.

$$\frac{AC}{2} = \frac{3}{2} = 1.5$$

	A	B	C	D
A	0	5	3	6
B	5	0	7	8
C	3	7	0	9
D	6	8	9	0



Gambar 5 Nilai terkecil pada matriks bobot dan node yang dibentuk

c. Reduksi matriks dan hitung skor terbaru
Setelah mendapatkan skor terkecil maka matriks direduksi menjadi seperti pada gambar 7. Kemudian nilai jarak untuk B dan D menuju AC dihitung kembali dengan rumus:

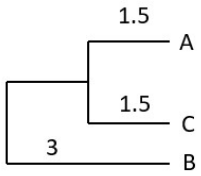
$$ACB = \frac{AB + BC}{2} = \frac{5 + 7}{2} = \frac{12}{2} = 6$$

$$ACD = \frac{AD + CD}{2} = \frac{6 + 9}{2} = \frac{15}{2} = 7.5$$

	AC	B	D
AC	0	6	7.5
B	6	0	8
D	7.5	8	0

Gambar 6 Matriks yang telah direduksi dan skor terbaru

Setelah didapatkan matriks terbaru seperti pada gambar 8, lakukan hal yang sama seperti pada langkah b. Skor terkecil adalah 6 yaitu dari AC ke B maka dibentuk cabang baru dari AC ke B seperti pada gambar 9. Karena *equidistant* maka skor 6 dibagi 2, $\frac{6}{2} = 3$.

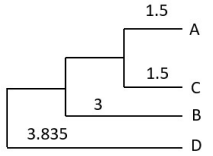


Gambar 7 Cabang B yang telah ditambahkn ke node AC

Ulang langkah b dan c hingga seluruh *alignment* terbentuk cabangnya. Disini tersisa jarak ACB ke D maka dihitung dengan rumus

$$\frac{AD + CD + BD}{3} = \frac{8 + 6 + 9}{3} = 7.67$$

karena *equidistant* maka 7.67 dibagi 2, $\frac{7.76}{2} = 3.835$.



Gambar 8 Bentuk akhir tree yang terbentuk

2. NJ (Neighbor Joining)

Mirip dengan UPGMA dalam membangun tree menggunakan *stepwise reduced matrix*. Namun pada NJ nilainya tidak diasumsikan ekuivalen, namun ada *evolutionary rate* pada tiap sekuens dengan menggunakan *conversion step*. Pada *conversion step* digunakan nila r dan juga nilai *transformed* r. [9] rumusnya seperti dibawah ini:

$$d'_{\mathrm{AB}} = d_{\mathrm{AB}} - 1/2 \times (r_{\mathrm{A}} + r_{\mathrm{B}})$$

d'_{AB} adalah *converted distance* antar sekuens A dan B

d_{AB} adalah nilai *actual evolutionary distance* antar sekuens A dan B

$r_{\mathrm{A}} + r_{\mathrm{B}}$ adalah jumlah jarak A atau B pada taksa lainnya, nilai r didapat dari rumus:

$$r_i = \Sigma \mathrm{d} r_{ij}$$

Dimana i dan j adalah dua taksa yang berbeda, nilai transformed r dibutuhkan untuk menentukan jarak dari individual taksa ke node terdekat

$$r'_i = \frac{r_i}{n} - 2$$

Dimana ada n buah taksa, misalnya A dan B berasal dari node U, maka jarak dari A ke u ditentukan dengan rumus:

$$d_{\mathrm{AU}} = \frac{[d_{\mathrm{AB}} + (r'_{\mathrm{A}} - r'_{\mathrm{B}})]}{2}$$

Misalkan ada 3 buah sekuens DNA yaitu A, B dan C, dengan menggunakan rumus kombinasi:

$$C(3,2) = \frac{3!}{(2!\,(3-2)!)} = \frac{6}{2} = 3$$

maka terdapat 3 pasangan sekuens. Berikut adalah himpunan pasangan sekuens dengan masing-masing skornya {(AB,3),(AC,3),(BC,7)}.

a. Membangun Matriks

Ukuran matriks yang dibangun sama dengan jumlah sekuens yang dimiliki, contohnya disini ada 3 sekuens maka matriks yang dibangun berukuran 3x3. Matriks ini kemudian diisi dengan skor tiap pasangan sekuens berdasarkan himpunan skor.

	A	B	C
A	0	5	3
B	5	0	7
C	3	7	0

Gambar 9 Matriks skor Kedekatan

b. Menghitung nilai *r-value* dan *r'-value*

Hitung nilai *r-value* dan *r'-value* dihitung dengan menggunakan rumus dibawah ini:

$$r_A = AB + AC = 5 + 3 = 8$$
$$r'_A = \frac{r_A}{3-2} = \frac{8}{1} = 8$$
$$r_B = BA + BC = 5 + 7 = 12$$
$$r'_B = 13$$
$$r_C = CA + CB = 3 + 7 = 10$$
$$r'_C = 10$$

Nilai *r'-value* sama dengan nilai *r-value* karena pembaginya adalah 1.

c. Menghitung nilai *converted distance*

Hitung nilai *converted distance* dengan menggunakan rumus dibawah:

$$d'_{AB} = d_{AB} - \frac{1}{2} * (r_A + r_B) = 5 - \frac{1}{2} * (8 + 12) = -5$$
$$d'_{AC} = d_{AC} - \frac{1}{2} * (r_A + r_C) = 3 - \frac{1}{2} * (8 + 10) = -6$$
$$d'_{BC} = d_{BC} - \frac{1}{2} * (r_B + r_C) = 7 - \frac{1}{2} * (12 + 10) = -4$$

Ganti nilai matriks menggunakan nilai yang didapatkan pada langkah ini, sehingga matriks menjadi

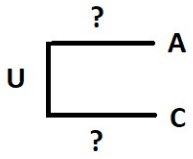
	A	B	C
A	0	-5	-6
B	-5	0	-4
C	-6	-4	0

Gambar 10 Matriks dengan nilai *converted distance*

d. Pilih nilai terkecil

Dari matriks pada gambar, pilih nilai terkecil, didapatkan -6 dari A ke C yang kemudian akan digabungkan menjadi satu *node tree* yang diberi nama U.

	A	B	C
A	0	-5	-6
B	-5	0	-4
C	-6	-4	0



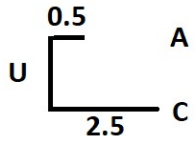
Gambar 11 Nilai terkecil pada matriks bobot dan node yang dibentuk

e. Hitung jarak tiap cabang

Jika pada metode UPGMA jarak cabang dianggap sama, maka pada metode NJ jarak cabang dihitung berbeda, jarak dihitung masing-masing menggunakan rumus

$$d_{AU} = \frac{[d_{AC} + (r'_A - r'_C)]}{2} = \frac{[3 + (8 - 10)]}{2} = 0.5$$
$$d_{CU} = \frac{[d_{AC} + (r'_C - r'_A)]}{2} = \frac{[3 + (10 - 8)]}{2} = 2.5$$

Dari hasil perhitungan didaparkan AU = 0.5 dan CU=2.5, maka cabang digambarkan sebagai berikut



Gambar 12 Tree yang terbentuk

Panjang cabang berbeda sesuai dengan nilai yang didapatkan.

f. Menghitung jarak baru

Karena A dan C sudah menjadi satu node, maka nilainya dengan B harus diperbarui dengan menggunakan rumus

$$d_{BAC} = \frac{[(d_{AB} - d_{UA}) + (d_{BC} - d_{UC})]}{2} = \frac{[(5 - 0.5) + (7 - 2.5)]}{2} = 4.5$$

Hadoop & MapReduce

Pada sistem ini akan digunakan *framework* Hadoop dan juga metode *MapReduce* dalam melakukan pemrosesan sekuens biologi. Hadoop merupakan sebuah *framework* yang mensupport aplikasi atau pengolahan yang melibatkan big data [10], Hadoop melakukan komputasi dengan distribusi ke node. Hadoop dapat dijalankan secara *single-node* dan *multi-node.* Pada *single-node*, seluruh kegiatan dilakukan hanya menggunakan satu perangkat saja sedangkan *multi-node* dilakukan pada banyak perangkat yang ada pada klaster. Klaster ini kumpulan dari perangkat yang saling terhubung dan bekerja dalam Hadoop atau biasanya disebut sebagai Hadoop Cluster. Dalam Hadoop Cluster *multi-node* biasanya ada perangkat yang berperan sebagai kontrol utama dalam proses (*Master*) dan perangkat yang berperan menjalankan proses yang diberikan (*Slave*). Pada Hadoop perhitungan atau proses komputasi itu bisa dibagi ke perangkat lainnya tanpa saling tindih sehingga hasil menjadi cepat semakin banyak perangkat yang terlibat dalam proses maka akan semakin cepat proses tersebut dilakukan.

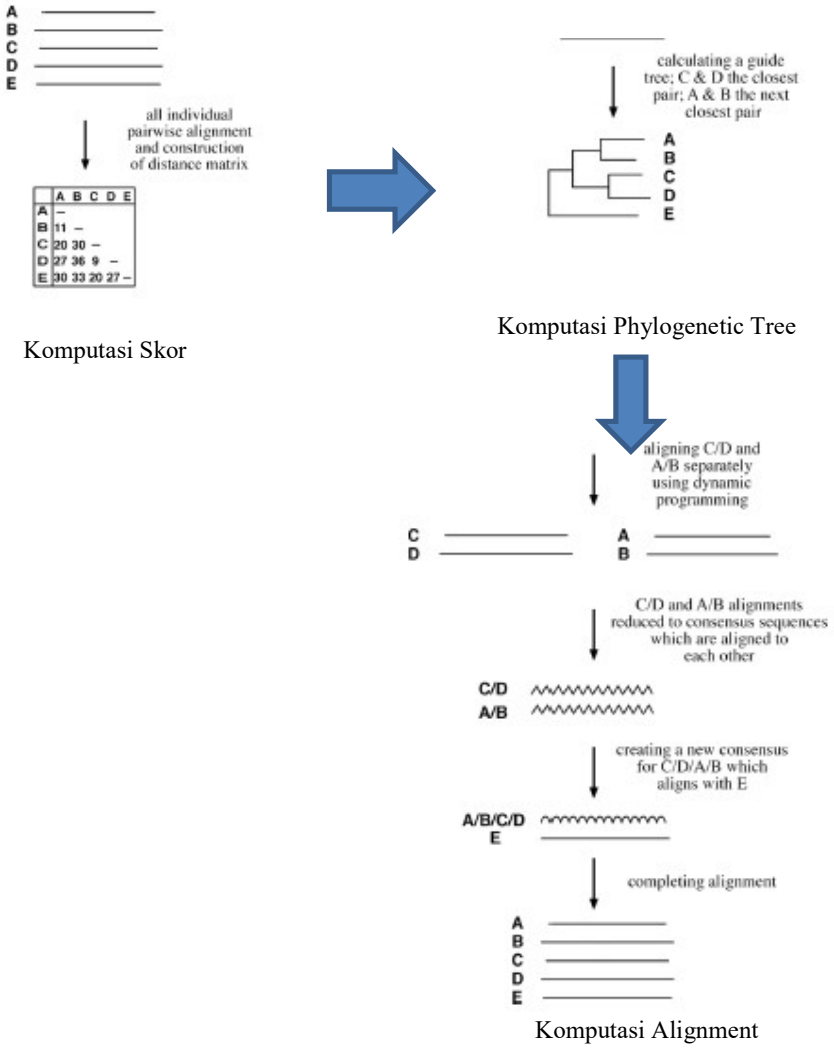
Hadoop memiliki *file system* sendiri yang disebut *Hadoop Distributed File System* (HDFS). HDFS menyimpan data yang digunakan dalam proses yang berjalan di Hadoop Cluster. Dengan adanya HDFS Sistem tersebut dapat membagi beban penyimpanan ke berbagai perangkat, walaupun menggunakan perangkat yang berbeda, namun jika salah satu perangkat mati tidak akan berdampak apapun pada perangkat lain, data akan masih tersimpan dan proses masih bisa terus dijalankan. [10] Ini merupakan salah satu contoh *failure control* yang dimiliki Hadoop. HDFS pada Hadoop *multi-node* juga memiliki kontrol utama seperti *master-slave*, namun dalam HDFS yang berperan sebagai *master* disebut *name node* dan yang berperan sebagai *slave* disebut *data node.*

MapReduce merupakan sebuah metode komputasi yang tersedia dalam framework Hadoop. *MapReduce* ini memiliki *job tracker* dan *task tracker* yang memiliki peran mirip dengan *master*-slave pada Hadoop. *MapReduce* pada sistem terdistribusi memiliki dua proses utama yaitu *map* dan *reduce*. *Map* berfungsi untuk membagi sebuah proses menjadi beberapa bagian yang dapat diproses secara paralel dengan jumlah yang banyak juga dengan ukuran data yang besar, dimana tiap node menerapkan fungsi map ke lokal data dan menuliskan hasil prosesnya ke penyimpanan sementara. [6] *Reducer* berfungsi sebagai penggabung hasil proses yang dilakukan secara terpisah oleh banyak *map.* Fungsi *map* dan *reduce* didefinisikan dengan menggunakan sepasang *key* dan *value*. [7] *Map* mengambil sepasang data (*key, value*) pada satu data domain dan mengembalikan sebagai *list of pair* pada domain yang berbeda. [6] Semua yang dilakukan di dalam Hadoop *MapReduce* terjadi secara otomatis, user hanya perlu menjalankan program java dalam bentuk file jar.

3. Sistem yang Dibangun

3.1 Analisis Independen Proses pada Pembentukan *Phylogenetic Tree*

Untuk kebutuhan paralel dan distribusi diperlukan identifikasi proses independen pada komputasi MSA. Proses independen adalah unit proses yang tidak saling bergantung satu dengan yang lainnya. Proses-proses ini yang berpotensi untuk dijalankan secara paralel dan terdistribusi. Pada penelitian ini mengambil contoh algoritma MSA Clustal W. Berikut adalah proses MSA dari CLUSTAL W

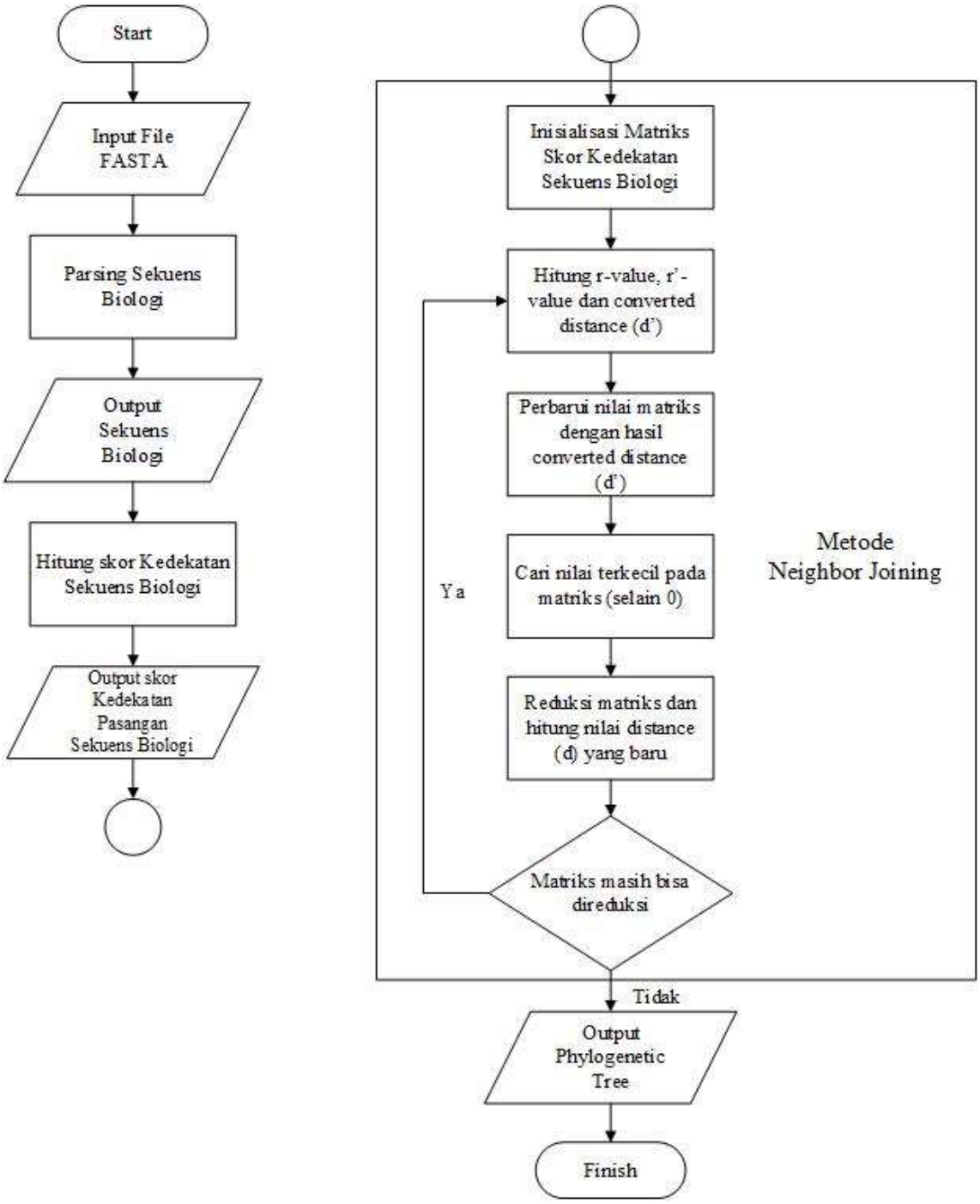


Gambar 13 Alur proses MSA (CLUSTAL W)

Dari gambar 13 komputasi skor memiliki potensi melibatkan banyak proses independen, karena komputasi skor dilakukan untuk semua kombinasi pasangan *sequence* yang terbentuk dari sejumlah input *sequence*. Masing-masing komputasi skor dapat dipandang sebagai unit proses yang independen.

3.2 Rancangan Algoritma Phylogenetic Tree

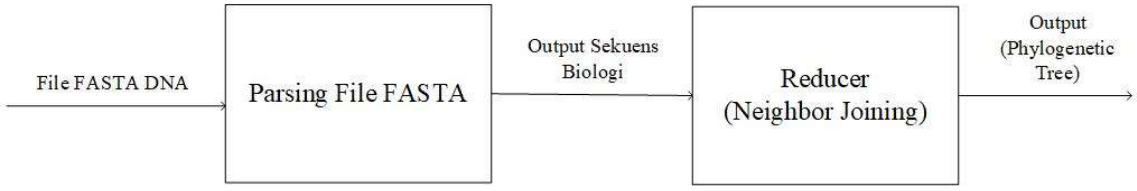
Tahapan proses pembangunan *phylogenetic tree* dapat digambarkan sebagai berikut:



Gambar 14 Alur proses pembentukan *phylogenetic tree*

3.3 Pembangunan Modul *MapReduce*

Pada modul *MapReduce*, yang digunakan hanya *mapper*nya saja, berikut skema yang dibangun pada setiap *mapper*.



Gambar 15 Skema *mapper* yang dibangun

a. Mapper
   - Tujuan: melakukan proses pembangunan tree.
   - Input: sekuens biologi dalam format file FASTA
   - Proses: parsing file FASTA dimana *header* dengan sekuens biologi dipisahkan kemudian dilakukan pembangunan tree dengan menggunakan metode Neighbor Joining.
   - Output: *phylogenetic tree* dalam bentuk newick.

3.4 Skenario Uji

1. Parsing file FASTA dan pembentukan *phylogenetic tree* dengan Java IDE. File FASTA akan diparsing menggunakan bantuan library BioJava [11]. Pembentukan *phylogenetic tree* dilakukan dengan menggunakan hasil parsing yang didapatkan. Dalam pembentukan *phylogenetic tree* ini juga digunakan library BioJava [11] untuk menjalankan metode Neighbor Joining. Parsing ini akan dilakukan dengan menggunakan Java IDE sehingga pada skenario ini belum melibatkan *MapReduce* dan Hadoop
2. Parsing file FASTA dan Pembentukan *phylogenetic tree* dalam bentuk komputasi *MapReduce* pada Hadoop. Program untuk parsing yang telah dibuat dengan menggunakan komputasi *MapReduce* akan dijalankan diatas framewok Hadoop. Program dijalankan dua kali, Hadoop yang digunakan adalah AWS Hadoop dengan jumlah node yang berbeda yaitu dua node dan tiga node yang masing-masing berukuran 8gb dengan 4 core.

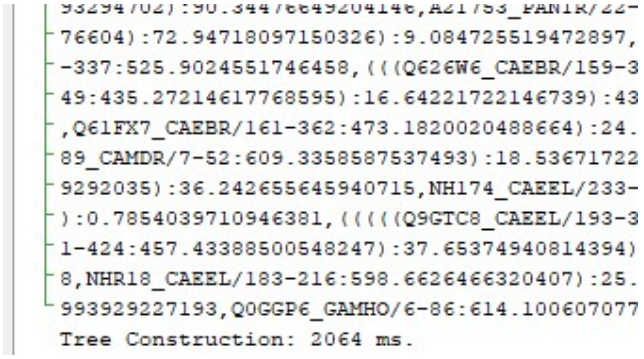
4. Evaluasi

4.1 Hasil Pengujian

1. Hasil Parsing file FASTA dan pembentukan *phylogenetic tree* dengan Java IDE dengan melibatkan 283 sekuens. Berikut adalah *phylogenetic tree* dalam format newick.

Gambar 16 Output tree dari IDE Java

Waktu komputasi yang digunakan dalam membangun *phylogenetic tree,* waktu yang ditunjukkan adalah 2064 ms. Waktu komputasi ini didapatkan dari menghitung selisih waktu saat program ini dijalankan hingga program menghasilkan *phylogenetic tree*.



Gambar 17 Waktu komputasi yang digunakan IDE Java

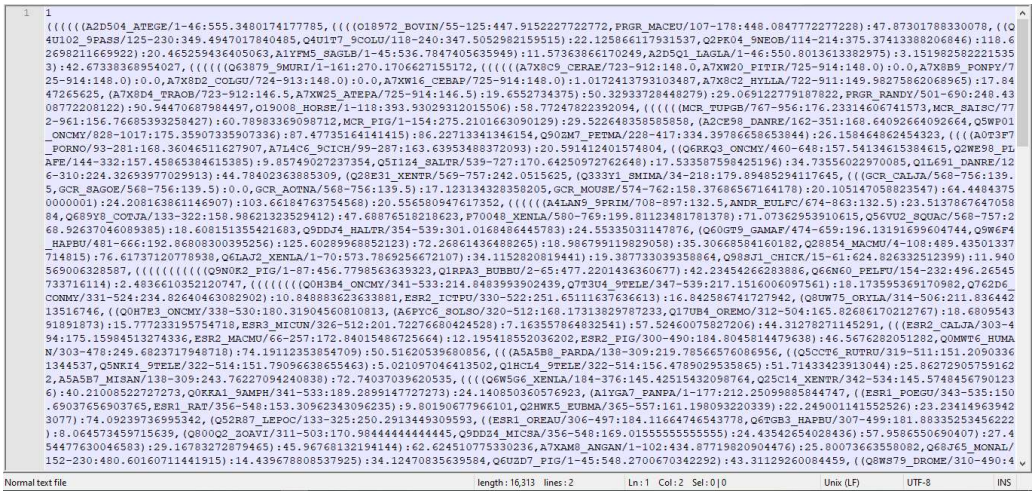
Namun, tiap menjalankan program waktu yang dihasilkan berbeda-beda, berikut waktu yang dihasilkan dalam 4 kali menjalankan program.

Tabel 1 Rata-rata waktu komputasi yang digunakan IDE Java

Run Program ke-	Waktu yang digunkan (ms)
1	2064
2	2649
3	3013
4	2770
Rata-rata	2624

Rata-rata waktu yang digunakan dalam membentuk tree adalah 2624 ms.

2. Hasil Parsing file FASTA dan pembentukan *phylogenetic tree* dengan menggunakan *MapReduce* yang dijalankan pada Hadoop multinode, dengan jumlah node sebanyak dua buah. Berikut *phylogenetic tree* dengan format newick



Gambar 18 Output tree dari komputasi *MapReduce*

Program *MapReduce* dengan menggunakan tiga node menghasilkan output yang sama.

Waktu komputasi yang digunakan mapper dalam membangun *phylogenetic tree* berbeda antara Hadoop yang menggunakan dua node dan tiga node, berikut hasil dari tiap percobaan.

```
Total time spent by all maps in occupied slots (ms)=427104
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=8898
Total vcore-milliseconds taken by all map tasks=8898
Total megabyte-milliseconds taken by all map tasks=13667328
```

Gambar 19 Detail waktu yang digunakan komputasi mapeduce dengan dua buah node

Tabel 2 Waktu komputasi menggunakan program *MapReduce*

Jumlah Node	Waktu Komputasi (ms)
2	8898
3	7468

Waktu komputasi yang digunakan Hadoop *MapReduce* dengan dua node dan tiga node berbeda. Hadoop dengan tiga node membutuhkan waktu lebih sedikit dibandingkan dengan Hadoop yang hanya menggunakan dua node.

4.2 Analisis Hasil

Phylogenetic tree dengan IDE Java dan komputasi *MapReduce* menggunakan waktu komputasi yang berbeda, hal ini disebabkan karena perbedaan lingkungan yang dibangun. Waktu komputasi program menggunakan *MapReduce* dengan dua dan tiga buah node hasilnya berbeda. Untuk *MapReduce* yang menggunakan dua node, waktu yang dibutuhkan 8898 ms sedangkan pada tiga node waktu yang dibutuhkan lebih singkat yaitu 7468 ms. Dilihat dari gambar 19, waktu yang digunakan oleh komputasi *MapReduce* diambil dari total waktu yang digunakan *map tasks*.

5. Kesimpulan

5.1 Kesimpulan

Dari hasil pengujian, dapat disimpulkan bahwa:

1. Dalam membangun *phylogenetic tree* dapat menggunakan komputasi *MapReduce*.
2. Dari waktu yang dihasilkan oleh komputasi *MapReduce* dengan menggunakan dua node dan tiga node dapat disimpulkan bahwa jumlah node mempengaruhi waktu komputasi, semakin banyak node maka waktu komputasi semakin cepat.

5.2 Saran

Untuk mengembangkan hasil yang sudah di dapatkan dari penelitian ini, diharapkan pada penelitian selanjutnya dapat:

1. Mendistribusikan *mapper* dan *reducer* secara tepat sehingga waktu komputasi yang digunakan *MapReduce* dapat dipersingkat.
2. Menjalankan secara paralel dan terdistribusi dalam proses iterasi *aligment* yang dibutuhkan untuk melakukan rangkaian *Multiple Sequence Alignment* (MSA).

Daftar Pustaka

[1] Alberts, Johnson, Lewis, Raff and Roberts, Molecular Biology of the Cell, 6th ed., 2014.

[2] V. O. Polyanovsky, M. A. Roytberg dan V. G. Tumayan, "Comparative Analysis of the Quality of a Global Algorithm and a Local Algorithm for Alignment of Two Sequences," *Algorithms Mol Biol,* 2011.

[3] R. Chenna, H. Sugawara, T. Koike, R. Lopez, T. J. Gibson, D. G. Higgins dan J. D. Thomp, "Multiple Sequence Alignment with the Clustal Series of Programs," *Nucleic Acids Res,* 2013.

[4] C. Notredame, D. G. Higgins dan J. Heringa, "T-coffee: a Novel Method for Fast and Accurate Multiple Sequence Alignment," pp. 205-217, 2000.

[5] J. Alnasir, "The Application of Hadoop in Structural Bioinformatics," *bioRxiv,* 2018.

[6] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski dan C. Kozyrakis, "Evaluating *MapReduce* for Multi-core and Multiprocessor Systems," 2007.

[7] A. M. Lesk, "Bioinformatics," 26 July 2013. [Online]. Available: https://www.britannica.com/science/bioinformatics. [Diakses 1 November 2018].

[8] Q. Zou, "Multiple Sequence Alignment and Reconstructing Phylogenetic Trees with Hadoop," *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM),* 2016.

[9] J. Xiong, Essential Bioinformatics, New York: Cambridge University Press, 2006.

[10] M. Bhandarkar, "*MapReduce* Programming with Apache Hadoop," *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS),* 2010.

[11] A. Lafita, S. Bliven, A. Prlic, D. Guzenko, P. W. Rose, A. Bradley, P. Pavan, D. Myers-Turnbull, Y. Valasatava, M. Heuer, M. Larson, S. K. Burley dan J. M. Duarte, "BioJava 5: A Community Driven Open-source Bioinformatics Library," *PLoS Comput Biol 15(2): e1006791,* 2019.