SINKRONISASI DATA DENGAN PEMROSESAN PARALEL MENGGUNAKAN MODEL PEMROGRAMAN *MAPREDUCE*

Murti Retnowo
Jurusan Manajemen Informatika, UTY, Yogyakarta
e-mail: nowo.yogya@gmail.com

ABSTRAK

Penelitian dalam pemrosesan data menunjukan bahwa semakin besar data semakun membutuhkan waktu yang lebih lama. Pemrosesan data berukuran besar pada komputer tunggal memiliki keterbatasan yang dapat diatasi dengan memproses data secara paralel. Penelitian ini memanfaatkan model pemrograman MapReduce pada sinkronisasi data dengan melakukan duplikasi data dari database client ke database server. MapReduce adalah model pemrograman yang dikembangkan untuk mempercepat pemrosesan data berukuran besar. Penerapan model MapReduce pada proses pelatihan dilakukan pada pembagian data yang disesuaikan dengan jumlah sub-proses (thread) dan pemasukan data ke database server serta menampilkan data hasil sinkronisasi data. Percobaan dilakukan dengan menggunakan data sebesar 1.000, 10.000, 100.000 dan 1.000.000 data, serta menggunakan thread sebanyak 1, 5, 10, 15, 20 dan 25 thread. Hasil penelitian menunjukan bahwa penggunaan model pemrograman MapReduce dapat menghasilkan waktu yang lebih cepat, namun waktu untuk membuat thread ketika jumlah thread yang banyak memerlukan waktu yang lebih lama. Hasil penggunaan model pemrograman MapReduce dapat memberikan efisiensi waktu dalam melakukan sinkronisasi data baik pada database tunggal maupun database terdistribusi.

Kata Kunci: *mapreduce, sinkronisasi data, multithread, message passing, Static Tasks Assigment*

ABSTRACT

Research in the processing of the data shows that the larger data increasingly requires a longer time. Processing huge amounts of data on a single computer has limitations that can be overcome by parallel processing. This study utilized the MapReduce programming model data synchronization by duplicating the data from database client to database server. MapReduce is a programming model that was developed to speed up the processing of large data. MapReduce model application on the training process performed on data sharing that is adapted to number of sub-process (thread) and data entry to database server and displays data from data synchronization. The experiments were performed using data of 1,000, 10,000, 100,000 and 1,000,000 of data, and use the thread as much as 1, 5, 10, 15, 20 and 25 threads. The results showed that the use of MapReduce programming model can result in a faster time, but time to create many thread that many require a longer time. The results of the use of MapReduce programming model can provide time efficiency in synchronizing data both on a single database or a distributed database

Keywords: *data synchronization, mapreduce, message passing interface multithread, static tasks assigment*

I. PENDAHULUAN

Kebutuhan akan kemampuan komputasi yang besar pada saat ini untuk melakukan pemrosesan data dalam skala besar dan waktu yang singkat menjadi hal yang sangat mendasar guna memaksimalkan kerja dan mencapai tujuan yang akan dicapai dalam perusahaan. Beberapa bidang yang membutuhkan komputasi tingkat tinggi adalah simulasi, perhitungan dan pemrosesan data. Masalah-masalah tersebut sering kali membutuhkan kalkulasi repetitif pada sejumlah data yang besar untuk memperoleh hasil yang valid. Selain itu sistem komputasi harus dapat menyelesaikan masalah dalam waktu yang masuk akal.

Komputasi tradisional yang memiliki prosesor tunggal untuk melaksanakan tugas-tugas dari suatu program merupakan salah satu cara untuk meningkatkan kinerja dalam melakukan pemrosesan data. Akan tetapi dengan prosesor tunggal masih memerlukan waktu yang lama, hal ini disebabkan karena komputer akan mengambil data satu per satu dari masing-masing *server* data yang berada pada lokasi yang berbeda-beda. Pada kenyataannya perusahaan sering kali menuntut agar simulasi, perhitungan dan pemrosesan data dapat diselesaikan dalam hitungan menit bahkan detik. Simulasi yang diselesaikan dalam hitungan hari biasanya tidak dapat diterima. Waktu yang diperlukan untuk melakukan simulasi, perhitungan dan pemrosesan data haruslah sesingkat mungkin agar informasi dapat diterima dalam waktu yang singkat dan akurat. Ada beberapa masalah yang memiliki tenggang waktu lama dalam pemrosesan komputasinya. Sebagai contohnya adalah pemrosesan data dalam jumlah yang besar dimana *server* data terletak pada beberapa komputer yang berbeda sehingga program memerlukan waktu yang lama untuk melakukan proses sinkronisasi atau penggabungan data dari masing-masing komputer [1].

Berdasarkan latar belakang masalah yang telah diuraikan diatas maka rumusan masalah pada penelitian ini adalah: melakukan proses sinkronisasi data guna memberikan informasi yang cepat dan akurat dengan memanfaatkan komputer yang sudah ada pada masing-masing *server* data dengan menggunakan model pemrograman *mapreduce*.

Permasalahan yang akan dibahas pada penelitian ini memiliki ruang lingkup yang cukup luas, sehingga permasalahannya dibatasi sebagai berikut:

1) Proses *MapReduce* dilakukan pada jaringan lokal dengan spesifikasi komputer yang berbeda-beda (*grid*).
2) Komputer-komputer yang digunakan untuk membentuk *grid* atau sebagai *node* atau *worker* adalah beberapa komputer yang terhubung dalam suatu jaringan dan berada dalam satu *switch* (jaringan lokal).
3) Metode yang digunakan *Message Passing Interface* dan *Static Task Assigment* digabungkan dengan model pemrograman *MapReduce*.

Iqbal [2], melakukan penelitian tentang bagaimana menganalisis suatu Identitas unik yang dimiliki satu oleh setiap penduduk pada KTP agar tidak terjadi duplikasi KTP pada saat pembuatan, *database* yang disimpan berukuran *Very Large Database* maka membutuhkan sebuah *Framework MapReduce* dan suatu *data warehouse* agar dapat menganalisis Identitas penduduk yang disimpan pada *database*. Data di-*export* ke dalam format Text kemudian dilakukan proses pemetaan [3]. Kurniawan [4] melakukan penelitian untuk mencari anggota milis linux di Indonesia dengan *MapRecude*, dimana *MapReduce* difungsikan untuk melakukan pencarian terhadap data anggota, penambahan, penghapusan, melihat topic milis dan menjalankan *data node* pada sistem operasi berbasis linux[4].

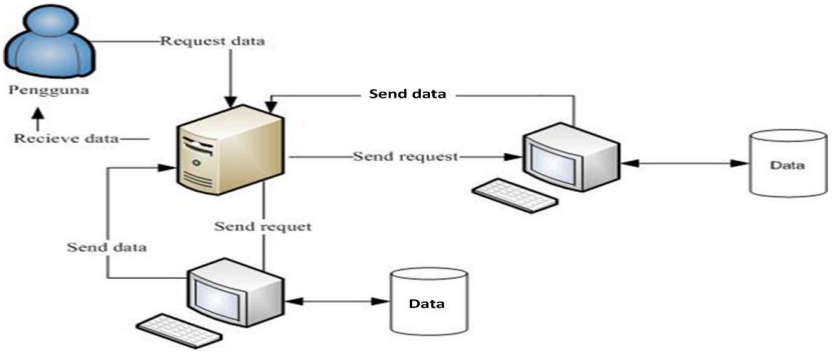
II. METODE

Kebutuhan akan informasi yang cepat dan akurat menjadi hal yang penting pada saat ini. Informasi dibutuhkan oleh pihak manajemen untuk mengambil keputusan dalam proses produksi dan pembelian bahan baku periode berikutnya. Untuk mendapatkan informasi yang cepat dan akurat tentunya diperlukan komputer dengan spesifikasi *hardware* yang mendukung untuk pemrosesan data yang cepat dan tingkat kinerja yang tinggi dan program berbasis *client-server* dengan *database* yang terdistribusi dan dapat diakses dari semua komputer dalam sebuah jaringan lokal maupun jaringan publik merupakan salah satu cara meningkatkan efisiensi waktu dalam pengaksesan data.

Pergantian *hardware* komputer tentunya sangat membantu dalam menambah kecepatan pemrosesan data tetapi dengan pergantian *hardware* komputer memerlukan biaya yang banyak dan program yang selama ini sudah berjalan dengan baik belum tentu di *support* pada *hardware* komputer yang baru. Selain itu juga diperlukan perubahan *setting* dari komputer yang baru agar program dapat berjalan dengan baik.

Merubah program yang ada pada saat ini tentunya memerlukan waktu yang lama karena *source code* dari program yang berjalan tidak diketahui mana yang paling akhir dan bagaimana alur dari program yang berjalan, hal ini disebabkan karena *programmer* yang membuat program sudah pindah ditempat kerja yang baru. Membuat program yang baru tentunya memerlukan waktu yang lebih lama lagi karena *programmer* yang baru harus dapat memahami alur kerja dari program yang berjalan pada saat ini dan mengimplementasikannya pada program yang baru, memerlukan waktu lama untuk pembelajaran program baru bagi pengguna dan *testing* program untuk mengetahui kesalahan-kesalahan yang ada serta akan mengganggu proses kerja yang sudah berjalan dengan baik.

Selain masalah *hardware* komputer dan program yang berjalan adalah transaksi atau pemrosesan data yang dilakukan dibanyak lokasi dan *database* yang berbeda hal tersebut disebabkan program yang masih berjalan secara *stand alone*. Transaksi yang terjadi dibeberapa tempat yang berbeda dengan program yang masih berjalan secara *stand alone* tentunya akan mengakibatkan masalah tersendiri dalam melakukan proses sinkronisasi data walau sudah menggunakan *database* yang terdistribusi dalam hal ini menggunakan *database MySQL*, tetapi *MySQL* masih di-*install* pada masing-masing komputer dengan menggunakan *setting user* yang hanya dapat mengakses dari satu tempat saja (lokal komputer).



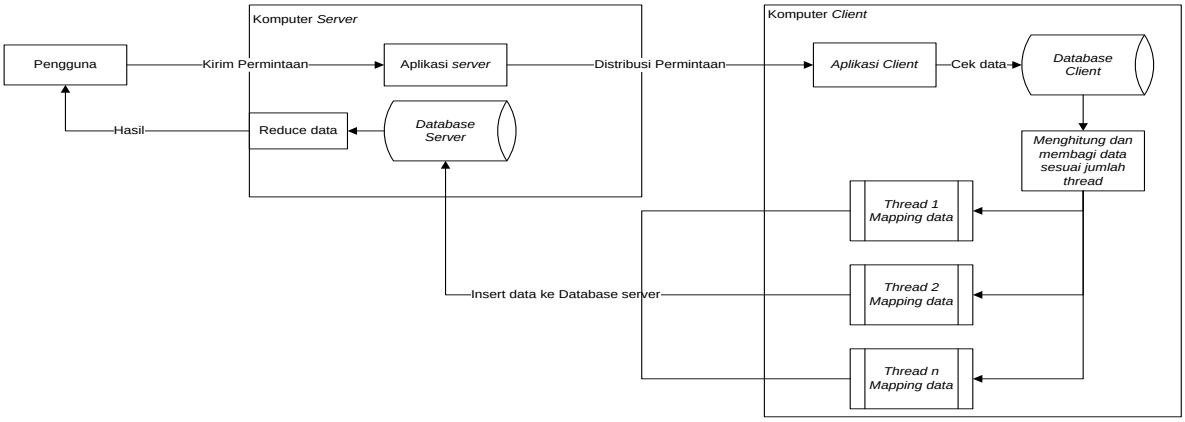
Gambar 1. Deskripsi sistem.

Salah satu tren yang sekarang ini digunakan pada perusahaan IT yang besar dalam menangani jumlah data yang besar dan terdistribusi adalah dengan model pemrograman *MapReduce*, dimana terdapat dua proses yaitu proses *Map* dimana data akan dibagi menjadi beberapa bagian sama besar dan proses *Reduce* dimana data akan digabungkan kembali dan dihilangkan jika ada data-data yang sama dari proses *mapping* sebelumnya. Proses

MapReduce akan dijalankan dalam sebuah *cluster* atau *grid* yang terdiri atas beberapa komputer *independent* dengan melakukan proses secara simultan atau paralel (*Multithread*) yang digunakan untuk melakukan sikronisasi data, *import* data atau pencarian data pada *database* yang terdistribusi.

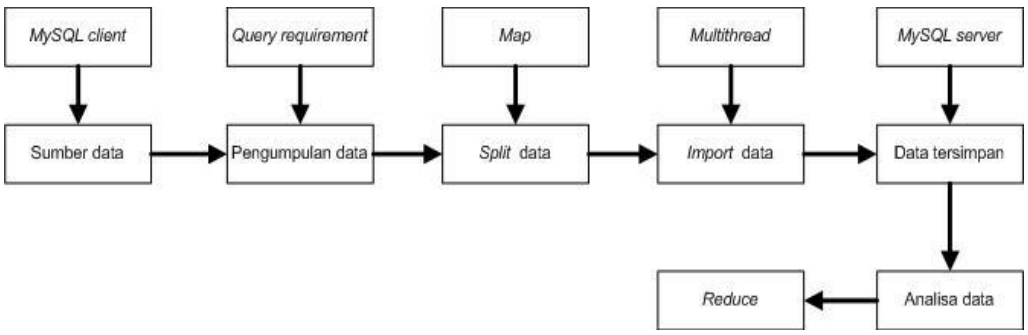
Model pemrograman *MapReduce* yang dipadukan dengan metode *Message-Passing Interface* (MPI) dan *Static Tasks Assignment* (STA) dan digunakan dalam bahasa pemrogramman *Delphi* selain diharapkan mampu meningkatkan kemampuan kinerja sistem dalam menangani data dalam ukuran yang besar dan terdistribusi juga mampu meningkatkan efisiensi dan efektivitas waktu kerja dalam melakukan proses sinkronisasi data dari banyak *database client* ke *database server*.

Deskripsi sistem secara garis besar dimulai dengan adanya proses permintaan data dari pengguna. Proses permintaan dan pengiriman data seperti ditunjukan pada Gambar 1 dimana ada pengguna melakukan permintaan data, permintaan data akan dikirimkan ke komputer *server*. Komputer *server* yang menerima permintaan data, secara otomatis akan mendistribusikan permintaan data ke masing-masing komputer *client* sampai disini komputer *server* sudah selesai melaksanakan pekerjaannya. Proses selanjutnya terjadi pada komputer *client.* Setelah menerima permintaan data selanjutnya komputer *client* akan mempersiapkan data yang diminta sesuai dengan kriteria yang sudah didapat dari komputer *server*, proses selanjutnya adalah melakukan *mapping* data atau mengumpulkan data yang akan dilakukan proses sinkronisasi ke *database server* secara paralel (*multithread*). Proses sinkronisasi data yang sudah selesai dilakukan akan dapat diakses oleh pengguna yang melakukan permintaan data tersebut sekaligus melakukan proses *reduce* atau menghilangkan data yang sama dengan kriteria tertentu.



Gambar 2. Proses analisis data.

Tahapan analisis data dapat dilihat pada Gambar 2. Proses pengiriman diawali dengan adanya permintaan data yang dibuat oleh pengguna dengan membuat kriteria berupa batasan tanggal dan jumlah *thread* untuk melakukan proses sinkronisasi, kriteria permintaan data tersebut selanjutnya akan dikirim ke komputer *server*, komputer *server* setelah menerima kriteria permintaan data akan mendistribusikan kriteria permintaan data tersebut ke semua komputer *client* yang terdapat dalam satu jaringan. Komputer *client* yang sudah menerima kriteria data akan melakukan pengumpulan data dan membagi data sesuai dengan jumlah *thread* dan melakukan proses sinkronisasi data ke komputer *server*.



Gambar 3. Alur proses pengiriman data.

III. HASIL

Hasil pelatihan digunakan untuk mengetahui perbedaan kecepatan sinkronisasi data yang dilakukan dengan secara konvensional dengan *single* proses yang dibandingkan dengan model pemrograman *MapReduce*. Dengan demikian akan diketahui kecepatan waktu yang diperoleh dari masing-masing program yang selanjutnya dilakukan

perbandingan untuk mengetahui selisih waktu dari masing-masing program. Selain pengecekan waktu pemrosesan data juga dicobakan masalah-masalah yang terjadi selama pembuatan sistem seperti pengecekan koneksi, pengecekan *port* dan pembatasan pengguna (*user*) yang melakukan transaksi.

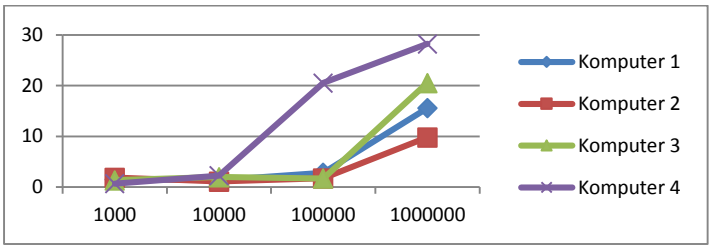
Proses pengujian dimulai dengan melakukan pengujian untuk mengetahui waktu yang digunakan untuk membuat *thread*, melakukan pemrosesan data sebanyak 1000, 10.000, 100.000 dan 1.000.000 data yang akan dilakukan sinkronisasi dengan jumlah pemroses sebanyak 1, 5, 10, 15, 20 dan 25 pada masing-masing jumlah data.

A. *PEMBUATAN THREAD*

Thread atau *leightweight process* (LWP) adalah suatu unit dasar dari *CPU Utilization* yang berisi program *counter*, kumpulan *register*, dan ruang *stack*. *Thread* akan berkerja sama dengan *thread* yang lainnya dalam hal penggunaan bagian kode, bagian data, dan *resource* dari sistem operasi secara kolektif (*task*). *MapReduce* merupakan salah satu model pemrograman yang terdistribusi dan berjalan secara paralel (*multithread*) yang berfungsi untuk mempercepat pemrosesan data dalam jumlah besar. Hasil dari rata-rata waktu yang digunakan untuk membuat *thread* sebanyak 25 *thread* dapat dilihat pada Tabel I. Waktu yang digunakan untuk membuat *thread* dipengaruhi oleh banyaknya data, jumlah *thread* yang sudah dibuat dan *hardware* komputer yang digunakan.

TABEL I
WAKTU PEMBUATAN *THREAD* (25 *THREAD*)

Nama Komputer	Waktu pembuatan (Detik)			
	1.000	10.000	100.000	1.000.000
Komputer 1	1,52	1,36	2,80	15,96
Komputer 2	1,82	1,08	1,72	9,80
Komputer 3	1,32	1,96	1,68	20,53
Komputer 4	0,68	2,24	5,32	28,24



Gambar 4. Grafik pembuatan *thread.*

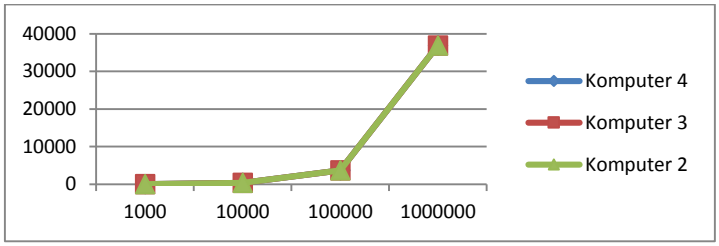
B. *PERCOBAAN DENGAN SINGLE PROSES*

Pengujian proses sinkronisasi data yang dilakukan dengan menggunakan proses tunggal pada masing-masing komputer. Pelatihan sinkronisasi data dengan *single process* (proses tunggal) dilakukan dari 4 buah komputer dengan spesifikasi *hardware* yang berbeda-beda dan menggunakan jumlah data yang sama yang ada pada masing-masing komputer. Pengujian dilakukan guna untuk mengetahui kecepatan pemrosesan data dengan menggunakan *single process*.

TABEL II
HASIL PROSES SINKRONISASI DENGAN *SINGLE PROCESS*

Nama Komputer	Waktu Pemrosesan (Detik)			
	1.000	10.000	100.000	1.000.000
Komputer 1	34	348	3.600	36.001
Komputer 2	36	365	3.694	36.940
Komputer 3	36	365	3.693	36.928
Komputer 4	36	364	3.693	36.932

Pengujian dilakukan pada semua komputer dengan spesifikasi *hardware* yang berbeda. Hasil pengujian proses sinkronisasi data yang dilakukan pada semua komputer dengan menggunakan proses tunggal (*single process*) dengan jumlah data yang sedikit (1.000 data) menunjukan bahwa waktu yang digunakan untuk melakukan proses sinkronisasi data tidak akan lama berkisar antara 34 hingga 36 detik, tetapi ketika dilakukan dengan data yang banyak semisal 1.000.000 data, maka proses sinkronisasi data memerlukan waktu yang lama berkisar antara 36.001 hingga 36.940 detik atau 10 jam 1 detik hingga 10 jam 15 menit 40 detik seperti ditunjukan pada Tabel II. Dari waktu yang diperlukan untuk melakukan sinkronisasi data dengan jumlah data 1.000.000 hingga 10 jam maka diperlukan sebuah program yang diperlukan untuk melakukan pemrosesan data dengan waktu yang relative lebih singkat.



Gambar 5. Grafik dengan *single process.*

C. *PENGUJIAN DENGAN MODEL PEMROGRAMAN MAPREDUCE*

1) *MAPPING DATA*

Pada pengujian kedua dengan menggunakan model pemrograman *MapReduce* yang dipadukan dengan *Message Passing dan Static Task Assignment* mampu menurunkan waktu pemrosesan data 60 % hingga 70 %. Proses diawali dengan adanya pembagian data sesuai dengan jumlah *thread* yang akan melakukan proses sinkronisasi data, selanjutnya dilakukan proses *mapping* data sesuai dengan hasil pembagian data. Proses *mapping* dilakukan bersamaan dengan proses pembuatan *thread*, masing-masing *thread* akan memperoleh data yang sama. Contoh dari pembagian data dapat dilihat pada Gambar 6.

```
1. Jumlah Data          : 1.000.000
2. Jumlah Thread        : 15
3. Sisa hasil bagi      : 1.000.000 mod 15 = 10
4. Data untuk 1 thread  : (1.000.000 - 10)/15 = 66.666
5. Data thread 1 - 14   : 66.666 data
6. Data thread 15       : 66.666 + 10 = 66.676 data
```

Gambar 6. Contoh pembagian data.

TABEL III
RATA-RATA WAKTU *MAPPING DATA* (DALAM DETIK)

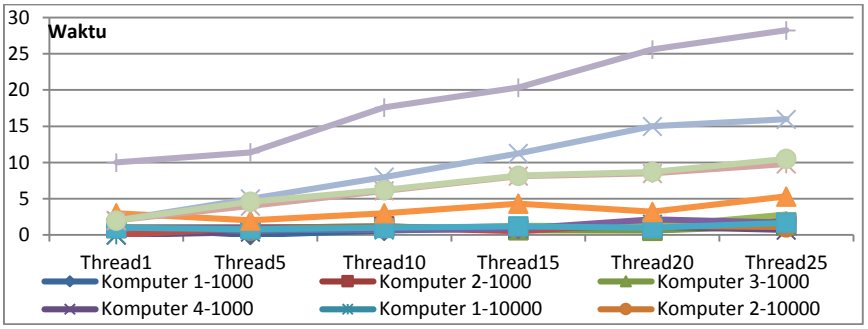
Nama	Data	Jumlah Thread					
		25	20	15	10	5	1
Komputer 1	1.000	1,52	0,85	1,00	0,60	0,0	1
Komputer 2	1.000	1,32	0,70	0,73	1,20	1,0	1
Komputer 3	1.000	1,28	0,60	0,67	1,20	1,0	1
Komputer 4	1.000	0,68	1,15	1,00	0,90	0,4	0
Komputer 1	10.000	1,36	1,50	0,93	0,70	1,0	0
Komputer 2	10.000	1,08	1,25	1,07	0,90	0,6	1
Komputer 3	10.000	1,96	1,20	1,00	0,80	0,6	1
Komputer 4	10.000	2,24	1,60	0,53	1,00	1,0	0
Komputer 1	100.000	2,80	1,20	1,27	0,80	1,0	1
Komputer 2	100.000	1,72	2,15	0,87	0,70	1,0	1
Komputer 3	100.000	1,.68	10,00	1,20	1,00	0,8	1
Komputer 4	100.000	5,32	3,20	4,33	3,00	2,0	3
Komputer 1	1.000.000	15,96	15,00	11,27	8,00	5,0	2
Komputer 2	1.000.000	9,80	8,50	8,13	6,10	4,0	2
Komputer 3	1.000.000	10,52	8,70	8,20	6,20	4,6	2
Komputer 4	1.000.000	28,24	25,60	20,33	17,60	11,4	10

IV. PEMBAHASAN

Hasil rata-rata waktu pengujian yang diperlukan untuk melakukan *mapping* data dan pembuatan *thread* dengan model pemrograman *MapReduce* dapat dilihat pada Tabel III dimana dalam tabel tersebut menunjukan hasil pengujian pembuatan *thread* atau *mapping* data yang dilakukan pada masing-masing komputer dengan jumlah data yang berbeda dan jumlah *thread* yang berbeda. Semakin banyak jumlah data yang akan diproses dan jumlah *thread* yang dibuat semakin banyak maka waktu yang diperlukan untuk melakukan *mapping* data akan semakin lama demikian pula semakin sedikit *thread* yang dibuat akan memerlukan waktu yang sedikit pula.

2) *REDUCE DATA*

Proses *reduce* data digunakan untuk menggabungkan data dan menghilangkan data jika terjadi kesamaan data atau duplikasi data hasil dari proses *mapping*. Proses *reduce* dilakukan oleh masing-masing komputer setelah proses *mapping* dan pembuatan *thread* sudah selesai dibuat. Semua *thread* yang sudah berhasil dibuat akan melakukan proses *reduce* data dengan melakukan duplikasi data dari *database client* ke *database server*.



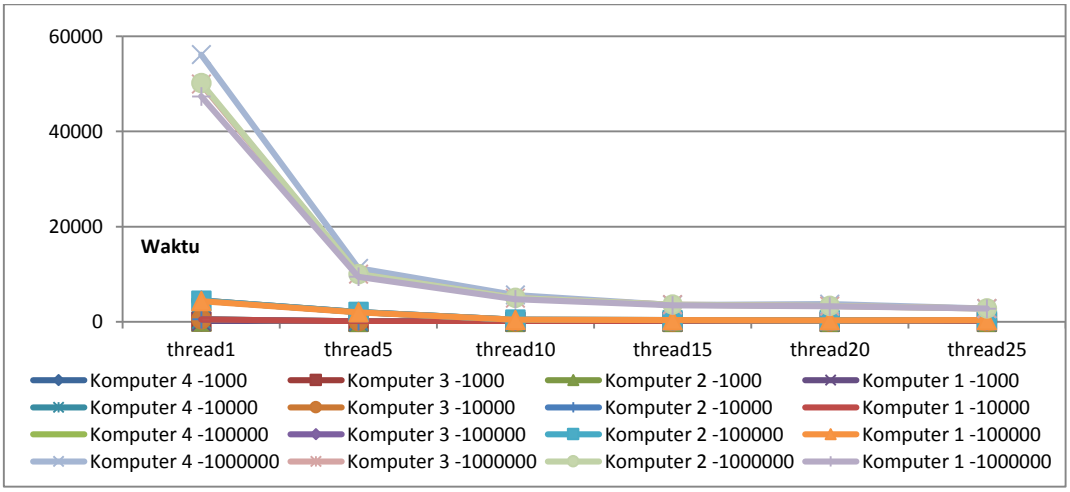
Gambar 7. Rata-rata Mapping data.

Pada Tabel IV menunjukan hasil sinkronisasi data yang dilakukan dengan model pemrograman *MapReduce* menunjukan adanya perubahan waktu yang cukup signifikan ketika menggunakan proses tunggal untuk data sebanyak 1.000 menggunakan waktu selama 68 detik, ketika menggunakan model pemrograman *MapReduce* dengan menggunakan 5 pemroses rata-rata waktu yang diperlukan menurun sekitar 6,8 %, ketika menggunakan 25 pemroses turun hingga hampir 90 %, sedangkan menggunakan 20 pemroses turuh hingga 95% perbedaan waktu antara 20 pemroses dan 25 pemroses disebabkan karena banyak sedikitnya *thread* yang dibuat untuk melakukan pemrosesan, semakin banyak *thread* yang dibuat memerlukan waktu yang lebih lama jika dibandingkan dengan *thread* yang lebih sedikit.

TABEL IV
RATA-RATA WAKTU *REDUCE DATA* (DALAM DETIK)

Nama komputer	Data	Thread					
		25	20	15	10	5	1
Komputer 1	1.000	6,00	4,00	4,60	6,00	11,00	68,00
Komputer 2	1.000	4,96	4,00	5,00	5,00	11,00	74,00
Komputer 3	1.000	3,48	4,00	5,00	5,00	11,00	74,00
Komputer 4	1.000	5,32	4,00	4,07	6,00	11,00	75,00
Komputer 1	10.000	30,96	37,05	46,67	60,70	101,40	479,00
Komputer 2	10.000	31,00	38,00	48,00	65,20	108,60	520,00
Komputer 3	10.000	30,68	37,90	48,33	64,80	108,20	520,00
Komputer 4	10.000	30,48	38,00	48,40	63,10	108,40	522,00
Komputer 1	100.000	351,20	297,20	342,93	407,80	1.989,60	4.361,00
Komputer 2	100.000	361,24	311,65	357,93	433,50	2.055,60	4.461,00
Komputer 3	100.000	361,44	309,70	357,13	431,50	2.052,60	4.458,00
Komputer 4	100.000	356,88	307,70	353,60	433,80	2.051,60	4.460,00
Komputer 1	1.000.000	2.754,56	3.244,65	3.449,87	4.731,30	9.464,80	47.360,00
Komputer 2	1.000.000	2.809,80	3.338,00	3.611,07	5.011,50	10.027,60	50.200,00
Komputer 3	1.000.000	2.796,48	3.316,70	3.592,40	4.995,60	9.994,80	49.970,00
Komputer 4	1.000.000	2.765,84	3.694,85	3.552,80	5.613,40	11237.60	56.170,00

Grafik pada Gambar 8 menunjukan rata-rata waktu terbaik terjadi pada komputer dengan spesifikasi *hardware* komputer tertinggi (Komputer 2) dengan selisih waktu yang tidak terlalu banyak. Semakin banyak *thread* dibuat semakin lama proses *mapping* data tetapi semakin cepat proses *reduce* data.



Gambar 8. Rata-rata hasil *reduce* data (sinkronisasi).

TABEL V
PERBANDINGAN WAKTU *REDUCE* DATA

Nama Komputer	Jumlah Data	Thread 25			Thread 20			Thread 15			Thread 10			Thread 5			Thread 1		
		avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max
Komputer 1	1.000	6.00	6	6	4.00	4	4	4.60	4	6	6.00	6	6	11.00	11	11	68.00	68	68
Komputer 2	1.000	4.96	4	6	4.00	4	4	5.00	5	5	5.00	5	5	11.00	11	11	74.00	74	74
Komputer 3	1.000	3.48	3	4	4.00	4	4	5.00	5	5	5.00	5	5	11.00	11	11	74.00	74	74
Komputer 4	1.000	5.32	5	6	4.00	4	4	4.07	4	6	6.00	6	6	11.00	11	11	75.00	75	75
Komputer 1	10.000	30.96	30	31	37.05	36	38	46.67	46	62	60.70	59	62	101.40	101	102	479.00	479	479
Komputer 2	10.000	31.00	30	32	38.00	38	38	48.00	47	66	65.20	65	66	108.60	108	109	520.00	520	520
Komputer 3	10.000	30.68	30	31	37.90	37	38	48.33	47	66	64.80	64	66	108.20	108	109	520.00	520	520
Komputer 4	10.000	30.48	30	31	38.00	38	38	48.40	48	64	63.10	62	64	108.40	108	109	522.00	522	522
Komputer 1	100.000	351.20	347	356	297.20	296	300	342.93	337	410	407.80	405	410	1989.60	1984	1993	4361.00	4361	4361
Komputer 2	100.000	361.24	354	365	311.65	309	314	357.93	355	435	433.50	433	435	2055.60	2054	2057	4461.00	4461	4461
Komputer 3	100.000	361.44	357	366	309.70	304	312	357.13	354	433	431.50	430	433	2052.60	2051	2055	4458.00	4458	4458
Komputer 4	100.000	356.88	353	361	307.70	304	312	353.60	351	436	433.80	432	436	2051.60	2050	2055	4460.00	4460	4460
Komputer 1	1.000.000	2754.56	2618	2776	3244.65	3113	3263	3449.87	3434	4739	4731.30	4724	4739	9464.80	9448	9478	47360.00	47360	47360
Komputer 2	1.000.000	2809.80	2692	2825	3338.00	3208	3356	3611.07	3604	5020	5011.50	5007	5020	10027.60	10018	10040	50200.00	50200	50200
Komputer 3	1.000.000	2796.48	2684	2813	3316.70	3183	3348	3592.40	3580	5000	4995.60	4991	5000	9994.80	9990	10000	49970.00	49970	49970
Komputer 4	1.000.000	2765.84	2652	2782	3694.85	3580	3717	3552.80	3546	5620	5613.40	5604	5620	11237.60	11234	11240	56170.00	56170	56170

V. SIMPULAN DAN SARAN

Kesimpulan yang dapat diambil setelah dilakukan penelitan dan percobaan adalah sebagai berikut:

1. Penggunaan model pemrograman *MapReduce* dapat mempercepat waktu untuk melakukan sinkronisasi datadengan cara duplikasi data (parsial sinkronisasi) dari *database client* ke *database server*.
2. Proses sinkronisasi data menggunakan 4 node komputer bila dibandingkan dengan menggunakan 1 node komputer. Hasil terbaik ditunjukan dengan memberikan 25 *thread* untuk melakukan sinkronisasi.
3. Semakin banyak *thread* yang dibuat dan data yang semakin besar mengkibatkan lamanya waktu yang digunakan untuk melakukan proses *mapping* data.
4. Berdasarkan pengujian perbedaan antara jumlah waktu pemrosesan dengan menggunakan *thread* sebanyak 15 dan *thread* sebanyak 25 perbedaan waktu pemrosesan tidak signifikan.
5. Pengujian model pemrograman *MapReduce* dengan menggunakan bahasa pemrograman Delphi 2010. Proses pengujian dengan menggunakan data sebanyak 1.000, 10.000, 100.000 dan 1.000.000 dan *thread* yang digunakan sebanyak 5, 10, 15, 20 dan 25 *thread.*

Saran yang diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Model pemrograman *MapReduce* diujicobakan pada bahasa pemrograman diluar bahasa pemrogrman java dan PHP dengan spesifikasi komputer yang sama maupun spesifikasi yang berbeda.
2. Pengunaan data dan jumlah pemrosesan yang lebih banyak lagi untuk mengetahui kendala-kendala yang timbul ketika model pemrograman *MapReduce* ketika diterapkan pada bahasa pemrograman diluar Java dan PHP.
3. Diujikan pada jumlah komputer yang lebih banyak dengan komputasi awan.

REFERENSI

[1] Wilkinson, B., dan Allen, M., 2005, (Pararrel Programming Techniques and applications Using Networked Workstations and Computers~Second Edition, (diterjemahkan oleh Hidayat, s., Santosa, Y. B. Hery, A. P. dan Himamunanto, R 2010) Penerbit Andi Yogyakarta.

[2] Iqbal M. S., 2012, Implementation MapReduce On Very Large Database System, http://library.gunadarma.ac.id

[3] Khairul, F. A.,dan Suadi, W., 2014, Komputasi Pembobotan Dokumen Berbahasa IndonesiaMenggunakan MapReduce. Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember

[4] Kurniawan, E. K., 2010, Penerapan Model Pemrograman MapReduce Pada Sistem Pencarian Milis Linux Di Indonesia, Jurusan Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia.