Analisis *Big Data* Berbasis *Stream Processing* Menggunakan Apache Spark

Herry Sitepu[#1], Claudia Zefanya Tumbel[#2], Maclaurin Hutagalung[#3]

[#]*Departemen Teknologi Informasi, Institut Teknologi Harapan Bangsa*
Jl. Dipatiukur No. 80-84, Bandung, Indonesia

[1]herry@ithb.ac.id
[2]clauzefanya6@gmail.com
[3]maclaurin@ithb.ac.id

***Abstract*— Big data technology has three key attributes, i.e., volume, speed, and complexity of the data. Big data processing can be performed in real time by using stream processing. Some of the methods used to process big data, i.e., Tuple, Micro Batching, and Windowed Real-Time Stream Processing. The method used in this study is Windowed Real-Time Stream Processing. This study utilizes Apache Spark to implement the processing method. It begins with stream source integration so that Apache Spark is able to obtain the data. Next, the data is analyzed by Apache Spark in real-time. The key result of this data is big data processing system in real-time.**

***Keywords*— Big Data, real-time, stream processing, Apache Spark, open-source software**

***Abstrak*— Teknologi *Big Data* memiliki 3 ciri utama yaitu volume, kecepatan, dan kompleksitas data. Pengolahan *Big Data* dapat dilakukan secara *real time* dengan menggunakan *stream processing.* Beberapa metode yang digunakan untuk mengolah *Big Data*, yaitu *Tupel, Micro Batching,* dan *Windowed Real-Time Stream Processing*. Metode yang digunakan dalam penelitian ini adalah *Windowed Real-Time Stream Processing*. Penelitian ini memanfaatkan Apache Spark untuk menerapkan metode pengolahan tersebut. Penerapan Apache Spark untuk *Big Data* dimulai dengan tahap mengintegrasikan *stream source* sehingga Apache Spark dapat memperoleh *stream* yang akan dianalisis. Hasil akhir dari penerapan metode ini berupa suatu sistem pengolah *Big Data* secara *real-time*.**

Kata Kunci*— Big data, real-time, stream processing, Apache Spark, perangkat lunak *open-source

I. PENDAHULUAN

Peningkatan jumlah akses layanan melalui berbagai perangkat mengakibatkan volume, kecepatan, dan keragaman data menjadi meningkat secara eksponensial. Layanan seperti Facebook, Twitter, Gmail, dan Youtube menyimpan data dari pengguna dalam jumlah yang sangat besar. Untuk mendapatkan informasi yang berguna dari data tersebut, dibutuhkan sistem pengolah *real-time stream processing.*

Stream processing adalah metode untuk menganalisis dan mengolah aliran data dengan kecepatan tinggi secara *real-time*. Contoh *real-time streaming* data, yaitu Twitter. Untuk mendapatkan *trending topic*, *Twitter* harus memproses setiap *tweet* dari semua pengguna secara *real-time*. Sistem pengolah *real-time* tersebut harus efisien dalam penggunaan sumber daya komputasi.

Apache Spark merupakan salah satu *open-source software* yang dapat digunakan untuk menganalisis dan mengolah *streaming* data. Penelitian ini menggunakan data layanan Uber berbentuk CSV (*comma separated value*) dengan jumlah data 14.270.480. Penelitian ini menggunakan data tersebut untuk diimplementasikan dengan metode *windowed real-time stream processing*.

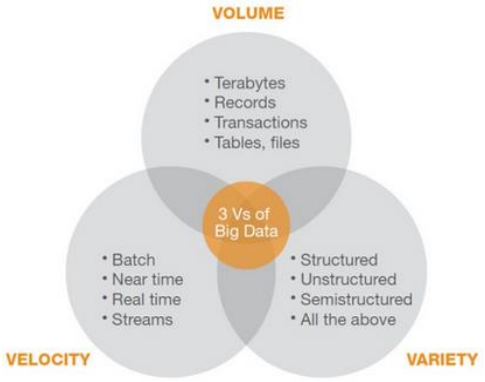
II. KAJIAN PUSTAKA

A. Big Data

Big Data adalah salah satu istilah yang populer saat ini. Karakteristik *Big Data* dapat ditinjau dari aspek volume, *velocity*, dan *variety*. Volume merupakan kriteria ukuran jumlah data. *Variety* merupakan kriteria seberapa banyak jenis data, sumber, dan format data. *Velocity* mengacu pada kecepatan data [6]. Tiga karakteristik *Big Data* dapat dilihat pada Gambar 1.

B. Stream Processing

Stream processing adalah metode yang dirancang untuk menganalisis dan mengolah *real-time streaming* data. Permintaan data *stream processing* terus meningkat. Oleh karena itu, data harus diproses dengan cepat dari sumber yang berbeda. Contoh aplikasi *streaming* berupa perdagangan *on line*, jaringan sosial, *internet of things*, monitoring sistem dan



Gambar 1 Karakteristik *big data* [8]

lain-lain [2]. Penerapan *stream processing* terbagi menjadi 3 proses, yaitu *Tupel*, *Micro Batching*, *Windowed Real-Time Stream Processing* (Gambar 2). Tupel berfungsi untuk memproses data yang masuk ketika data diterima oleh penerima. *Micro Batching* mempunyai fungsi yang berbeda dari Tupel. Tupel mengirimkan data satu per satu. Sebaliknya, *Micro* mengumpulkan semua data dari pengirim lalu dikirimkan ke penerima apabila datanya sudah lengkap. Contoh *Windowed Real-Time Stream Processing* adalah *Datatorrent*. *Datatorrent* menggunakan konsep *Beachon*. *Beachon* memiliki *latency* yang sangat rendah. *Beachon* berfungsi memproses dua sistem, yaitu *Tupel* dan *Micro*. Tujuan dari menggabungkan *Tupel* dan *Micro*, yaitu untuk mengurangi *latency*, mendeteksi kesalahan, dan memperbaiki kesalahan sebelum data yang dikirim gagal. Setiap segmen pada *Beachon* disebut *Window*.

C. *Apache Spark*

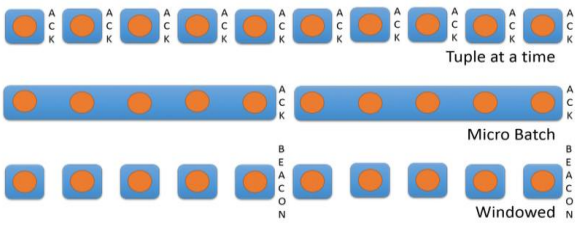
Apache Spark awalnya dikembangkan pada tahun 2009 di UC Berkeley AMPLab. Selanjutnya, dikembangkan menjadi project *open-source* Apache pada tahun 2010. Apache Spark adalah sebuah *open-source software* yang digunakan untuk menganalisis data secara *real-time*. Apache Spark menerapkan API pada mesin pengolahan data. Apache Spark terdiri dari Spark *core* dan satu set *library*.

Core adalah mesin eksekusi yang terdistribusi dengan Java API, Scala, dan Python sebagai *platform* dalam pengembangan aplikasi. Apache Spark merupakan bagian dari Hadoop [1]. Apache Spark memungkinkan penggunaan sumber daya yang lebih efisien dalam satu *cluster*. Spark dapat mengelola data yang berukuran besar. Jenis data yang diolah berupa teks, grafik, dan lain-lain, sedangkan dari sisi sumber data berupa *batch* dan *real-time streaming* data. Aplikasi Spark memproses data pada memori 100 kali lebih cepat dan memproses data pada *disk* 10 kali lebih cepat di *cluster* Hadoop. Spark memudahkan pengguna menulis program di Java, Scala, dan Phyton. *Built-in* yang disediakan oleh Apache Spark sebanyak 80 operator yang digunakan pada *query* data dalam *shell* [5].

III. Analisis dan Perancangan

A. *Cara Kerja Spark Streaming*

Sebelum mulai mengolah data yang masih dalam bentuk *file* CSV terlebih dahulu *user* harus mengetahui cara kerja dari Spark *streaming*.



Gambar 2 Penerapan *stream processing* [4]

Cara kerja dari Spark *streaming* terdiri atas beberapa langkah, yaitu:

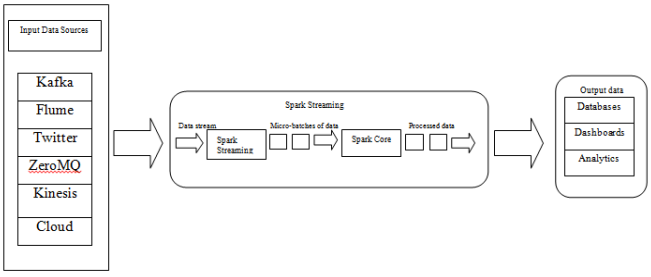
1. Persiapan data yang akan diolah dan jenis data. Data yang diambil untuk penelitian ini berasal dari *awesome public datasets*. *Awesome public datasets* merupakan salah satu penyimpanan data lewat *cloud*. Selain itu, *awesome public datasets* menyimpan data dalam jumlah besar dan bervariasi, dengan kata lain *Big Data*.
2. Data yang dalam bentuk *file* tersebut harus diubah menjadi bentuk *stream* yang dikirimkan melalui jaringan.
3. Spark *streaming* langsung memproses data yang telah diterima. Spark *streaming* akan terus menerus melakukan perhitungan ketika ada data yang masuk dan juga akan menampilkan total waktu *delay* pada GUI Spark.
4. Data yang diproses oleh Spark *streaming* menggunakan *micro batches*. *Micro batches* mengumpulkan semua data dari pengirim dan langsung diproses apabila data tersebut telah lengkap, kemudian langsung dikirim ke penerima.
5. Setelah selesai diproses oleh Spark *streaming* menggunakan teknik *micro batches*, kemudian akan lanjut diproses oleh Spark *core*.
6. Lalu ketika selesai diproses oleh Spark *core* maka hasilnya akan berupa informasi dalam bentuk *interface* yang bermacam-macam seperti *database*, *dashboard*, dan *analytics*. Untuk lebih jelasnya dapat dilihat pada Gambar 3.

B. *Perhitungan Dispatching Terbanyak*

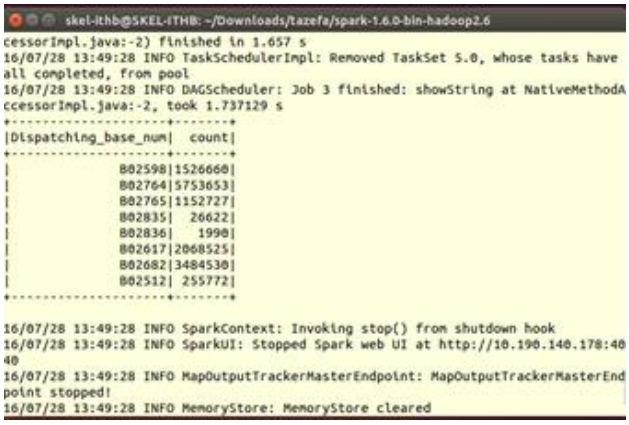
Aplikasi Spark yang sangat kaya dapat melakukan banyak perhitungan, sehingga dari perhitungan yang ada dapat menghasilkan beberapa informasi. Salah satunya menghitung *dispatching* terbanyak dan tersedikit. Tujuan dari informasi yang didapat adalah agar *user* mengetahui kode *dispatching* mana yang paling aktif, sehingga dapat meningkatkan performansi perusahaan. Perusahaan juga bisa memberikan apresiasi kepada orang tersebut. Hasil perhitungan yang dilakukan dapat dilihat pada Gambar 4.

C. *Perhitungan Affiliated*

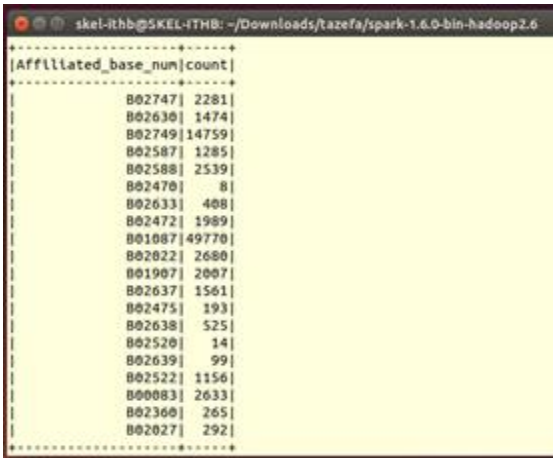
Data Uber memiliki 4 kolom yang masing-masing kolom berisi 14.270.479 data. Oleh karena itu, untuk menghitung kolom afiliasi *user* meggunakan PySpark. Hasil dari perhitungan ditunjukan pada Gambar 5.



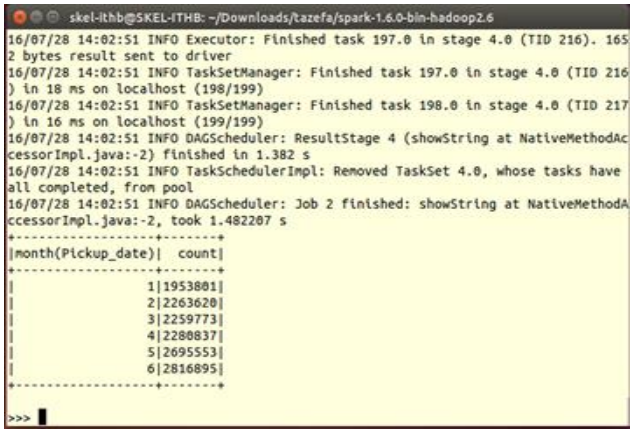
Gambar 3 Cara kerja Spark *streaming*



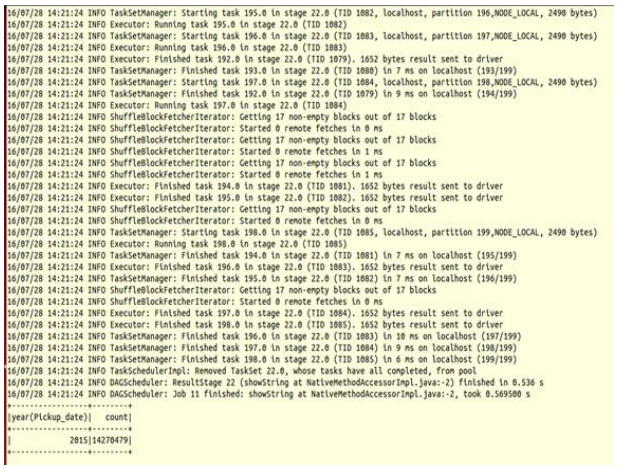
Gambar 4 *Array dispatching*



Gambar 5 *Affiliated array*



Gambar 6 Total *month*



Gambar 7 Total *year*

D. Perhitungan Total Month

Aplikasi Spark yang sangat kaya dapat melakukan banyak perhitungan sehingga dari perhitungan yang ada dapat menghasilkan beberapa informasi. Salah satunya menghitung total *month* dari setiap panggilan selama 6 bulan. Hasil dari perhitungan tersebut ditunjukan pada Gambar 6.

E. Perhitungan Total Year

Apache Spark juga dapat menghitung data total *year* dari sumber data yang sedang diolah. Informasi tersebut dapat dilihat pada Gambar 7.

F. Perhitungan Busy Hour

Ada beberapa informasi penting juga yang perlu diketahui, yaitu *busy hour*. Informasi *busy hour* sangat berguna untuk memperbanyak *driver* pada jam-jam tersebut agar dapat di-*cover* setiap permintaan yang datang. Hasil *busy hour* tersebut dapat dilihat pada Gambar 8.

G. Perhitungan Day of Month

PySpark SQL juga dapat menghitung *day of month* dari kolom *pickup date* yang diolah. Setiap informasi yang dihasilkan merupakan informasi penting untuk bisa mengatasi permintaan terbanyak dan dapat mengirimkan *driver* juga. *Day of month* menunjukan hari-hari terdapat panggilan terbanyak selama 6 bulan. Hasil informasi tersebut ditunjukan pada Gambar 9.

H. Analisis Kebutuhan Perangkat Keras

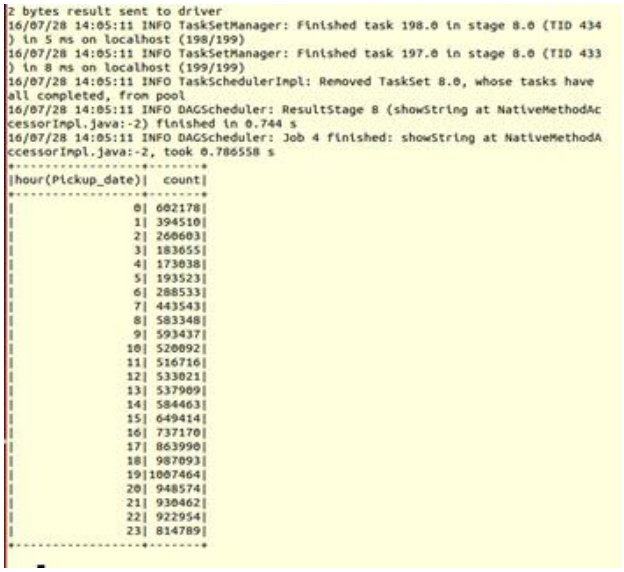
Kebutuhan perangkat keras merupakan kebutuhan karakteristik suatu perangkat yang akan digunakan pada penelitian ini. Perangkat keras yang diharapkan dalam penelitian ini, yaitu:

- 3 buah PC dengan RAM 4 Gb, memori 500 GB, CPU Intel Core i3.
- *Software* Apache Spark terbaru yang lengkap dengan Hadoop.

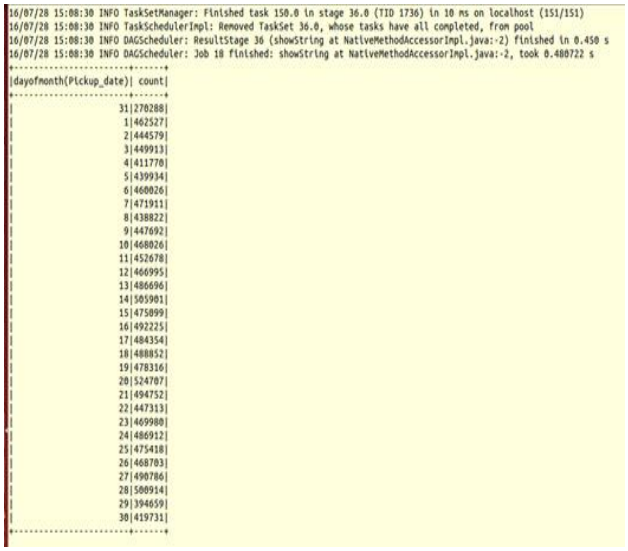
I. Analisis Kebutuhan Fungsional

Kebutuhan fungsional merupakan semua fungsi yang dapat dilakukan oleh aplikasi Apache Spark. Kebutuhan fungsional yang dapat dilakukan oleh aplikasi ini adalah sebagai berikut:

- Apache Spark dapat menampilkan total *delay* hasil pengolahan data di GUI Spark *stream*ing.
- Apache dapat menampilkan hasil perhitungan yang cepat pada *console*.
- Apache Spark dapat menampilkan berbagai macam informasi dengan membuka *port* 4040.



Gambar 8 Total *year*



Gambar 9 *Day of month*

Master Spark dapat menampilkan hasil *cluster* dari beberapa PC lengkap dengan IP *address*, durasi, *driver*, dan lain-lain yang dapat diakses menggunakan *port* 8080.

IV. Implementasi dan Pengujian

A. Implementasi Perancangan

Implementasi rancangan merupakan tahap yang dilakukan setelah analisis kebutuhan sistem. Tujuan dari tahap ini untuk menampilkan data statistik total data dari data yang diolah. Oleh karena itu, data tersebut diproses menggunakan Apache Spark.

Proses implementasi meliputi proses implementasi perangkat keras, implementasi perangkat lunak, implementasi data Uber, dan implementasi program Apache Spark. Proses implementasi perangkat lunak akan dijelaskan tentang deskripsi data yang akan digunakan untuk menampilkan *stream* statistik. Selain itu, dibutuhkan juga 3 PC perangkat keras yang akan digunakan untuk *clustering*.

B. Implementasi Streaming Statistic

Ada beberapa tahap yang dilakukan untuk menghasilkan *stream processing* data Uber, yaitu:

1. Membuat nama aplikasi network_wordcount di SparkContext yang berfungsi untuk memanggil fungsi yang ada di dalam *listing code*.
2. Menjalankan terminal setelah script tersebut disimpan. Terminal pertama berisi perintah `cat uber.csv | nc -lp port 9999` yang berfungsi untuk membaca data lalu di *stream* ke terminal dua. Terminal dua juga menjalankan nama aplikasi networkwordcount.py yang telah dibuat di SparkContext menggunakan perintah: `$./bin/spark-submit NetworkWordCount.py localhost 9999` yang berfungsi untuk melakukan perhitungan data dari terminal pertama.
3. Menuliskan perintah di terminal pertama. Untuk lebih jelasnya, dapat dilihat pada Gambar 10. Penulisan untuk terminal kedua ditunjukan pada Gambar 11.
4. Menjalankan perintah di kedua terminal pada Gambar 10 dan 11. Hasilnya dapat dilihat pada GUI *(graphical user interface)* Spark *streaming* yang ditunjukan pada Gambar 12. Hasil tersebut berupa *statistic* yang menggambarkan total *event* yang diterima dalam waktu 1 *second* dengan data yang direkam sebanyak 14.270.480.

C. Parameter Pengukuran Kinerja

Sebelum menilai kualifikasi hasil pengukuran ada beberapa parameter yang digunakan, yaitu [7]:

1. Kualifikasi kinerja Apache Spark dapat dikatakan baik apabila kecepatan waktu dalam memproses data untuk menghasilkan informasi kurang dari 1 detik.
2. Kualifikasi kinerja Apache Spark dapat dikatakan cukup apabila kecepatan waktu dalam memproses data untuk menghasilkan informasi antara 1 detik sampai dengan 1 menit.
3. Kualifikasi kinerja Apache Spark dapat dikatakan kurang apabila kecepatan dalam memproses data untuk menghasilkan informasi lebih dari 1 menit.

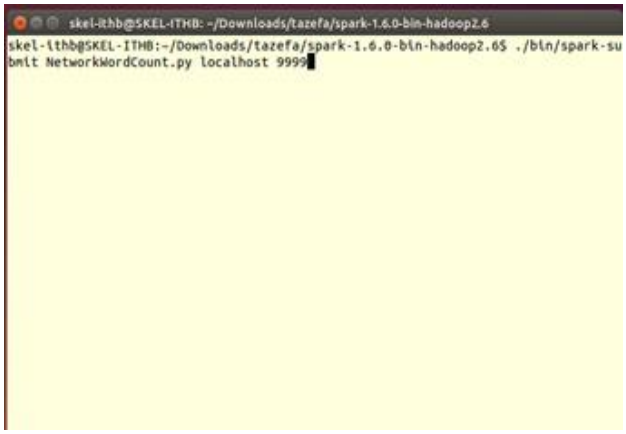
D. Hasil Pengukuran Kinerja Apache Spark

Pengolahan data menggunakan Spark membutuhkan spesifikasi komputer yang bisa mendukung cara kerja Spark. Oleh karena itu, dilakukannya pengukuran kinerja berdasarkan spesifikasi komputer yang digunakan. Hasil pengukuran kinerja Spark dapat dilihat pada Tabel 1.

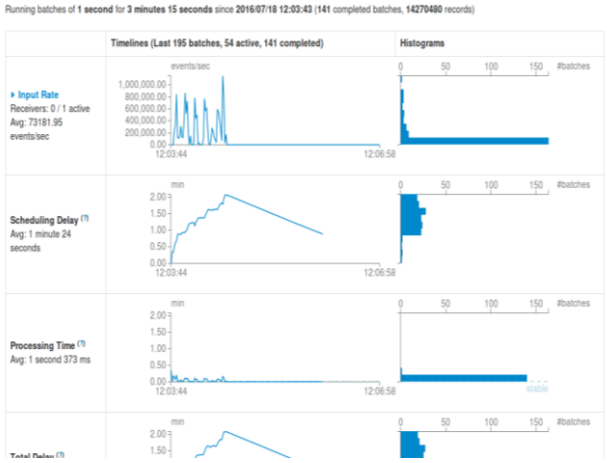
E. Analisis Hasil Pengukuran

Analisis hasil pengukuran dari data yang diolah menggunakan Apache Spark menyatakan kualifikasi kinerja Apache Spark cukup. Hal ini disebabkan karena spesifikasi perangkat keras yang digunakan dalam pengolahan data. Oleh

Gambar 10 *Netcat*



Gambar 11 Spark *Submit*



Gambar 12 Spark *Streaming Statistic*

TABEL I

HASIL PENGUKURAN

ID	Objective	Submitted	Durasi
1	*Count Dispatching*	28/7/2016 14:01:55	55s
2	*Count Affiliated*	28/7/2016 14:04:18	53s
3	*Count Month*	28/7/2016 14:06:22	52s
4	*Count Hour*	28/7/2016 14:18:49	52s
5	*Count Day of Month*	28/7/2016 14:20:34	51s
6	*Count Quarter*	28/7/2016 14:23:05	52s
7	*Count Year*	28/7/2016 14:24:11	54s

karena itu, untuk mendapatkan hasil kualifikasi pegukuran yang baik perlu adanya *upgrade* perangkat kerasnya, mulai dari ukuran RAM, *hardisk*, *intel core* yang mendukung pengolahan data di Spark.

V. KESIMPULAN

Berdasarkan hasil dari analisis, implementasi, dan pengujian, maka dapat diambil kesimpulan sebagai berikut:

1. *Big Data* berhasil diolah dengan metode *stream processing*.
2. Metode *stream processing* yang diterapkan untuk mengolah *Big Data* dapat disinkronisasi dengan Apache Spark sehingga menghasilkan informasi-informasi yang dibutuhkan seperti total *dispatching*, total *month*, total *hour*, total *affiliated*, total *day of month*, dan total *year* dalam waktu yang singkat.
3. Berhasil mengolah data menggunakan Apache Spark sehingga kinerja Apache Spark dapat diukur dari hasil *monitoring* pada *localhost* 9999.

DAFTAR REFERENSI

[1] H. John. ”Definition Real-Time.” Internet: http://whatis.techtarget.com/definition/real-time, Apr. 2006 [Oct. 20, 2015].

[2] K. Wahner. ”Real-Time *Stream* Processing as Game Changer in a *Big Data* World with Hadoop and Data Warehouse.” Internet: http://www.infoq.com/articles/*stream*-processing-hadoop, Sept. 10, 2014 [Oct. 20, 2015].

[3] M. Barlow. (2013, February 25). *Real-Time Big Data Analytics*. (1st edition). [Online]. [Oct 20, 2015].

[4] N. Idoudi, N. Louati, C. Duvallet, at all. (2009, January). "A Framework to Model Real-Time Databases." *International Journal of Computing and Information Sciences*. [Online]. 7(1), pp. 1-8. Available: http://www.ijcis.info/Vol7N1/Vol7P1N1-PP-1-11.pdf [Oct. 20, 2015].

[5] N. Chetan. “Real-Time Event *Stream* Processing.” Internet: https://www.datatorrent.com/real-time-event-*stream*-processing-what-are-your-choices/, March. 9, 2015 [Oct. 20, 2015].

[6] P. Srini. ”*Big Data* Processing with Apache Spark.” Internet: http://www.infoq.com/articles/apache-spark-introduction, Jan. 30, 2015 [Oct. 20, 2015].

[7] T. Das. "Faster Stateful *Stream* Processing in Apache Spark *Stream*ing." Internet: https://databricks.com/blog/2016/02/01/faster-stateful-*stream*-processing-in-apache-spark-*stream*ing.html/, Feb.1, 2016 [Jun. 20, 2016].

[8] W. Kai. "Real-Time *Stream* Processing as Game Charger in a *Big Data* World with Hadoop and Data Warehouse." Internet: http://www.infoq.com/articles/*stream*-processing-hadoop, Sept. 10, 2014 [Oct. 20, 2015].

Herry Imanta Sitepu, menempuh pendidikan S1 di Teknik Elektro ITB dan lulus tahun 1999, dan memperoleh gelar magister dan doktor di jurusan yang sama di ITB. Sejak tahun 2006 aktif sebagai pengajar di Prodi Sistem Komputer ITHB. Minat penelitian: *computer networking*, *programming* dan *distributed system*.

Claudia Zefanya Tumbel, lahir di Manado, 6 April 1994. Pada tahun 2012 menempuh pendidikan S1 di Institut Teknologi Harapan Bangsa (ITHB) di Departemen Teknologi Informasi Jurusan Teknik Elektro. Lulus pada bulan Juli 2016 dengan gelar Sarjana Teknik. Bidang yang diminati adalah *networking*.

Maclaurin Hutagalung, menerima gelar Sarjana Teknik dari Institut Teknologi Bandung jurusan Teknik Elektro, gelar Magister Sains dari University of Twente Belanda di bidang Sinyal, Sistem, dan Kendali, dan gelar Doktor dari Tokyo Institute of Technology di bidang Sistem Kendali Nonlinear. Sejak tahun 2012 aktif sebagai pengajar di Departemen Sistem Komputer ITHB di Bandung. Minat penelitian pada Kendali Sistem Dinamis, Robotika, dan Penerbangan.