Analisis Performansi NoSQL file system pada Hadoop Distribute File System dan Cassandra File System

Juan Yudha[1], Sidik Prabowo[2], Siti Amatullah Karimah[3]

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

Abstrak

Pada saat ini teknologi berkembang dengan pesat. Berdasarkan penelitian yang dilakukan Big Data akan menguasai pasar teknologi. Tingkat pertumbuhan Big Data meningkat setiap tahunnya, dan file system (FS) yang dibutuhkan semakin besar. Hadoop merupakan OS open source yang sangat populer dikalangan Big Data. Namun dengan meningkatnya perkembangan dan penggunaan Big Data, spesifikasi pada file system juga mulai meningkat. Sistem yang ada hanya mampu untuk mentukan berdasarkan dari kategori bottleneck pada Big Data. Namun sistem ini dinilai masih kurang, karena tidak ada hasil keluaran dalam menentukan performansi dari suatu file system. Sehingga dibutuhkan sebuah sistem yang mampu untuk menentukan tigkat performansi dari suatu file system.File System saat ini menggunakan data yang tersusun secara terstruktur salah satu contohnya yaitu SQL. Untuk menangani data yang tidak terstruktur diperlukan suatu mekanisme File System yang menggunakan NoSQL.Untuk itu pada penelitian ini sudah di lakukan pengujian pada file system berdasarkan load-write dan load-read dengan menggunakan Hadoop Distribute File System (HDFS) dan Cassandra File System (CFS). Klasifikasi hasil dibagi menjadi dua yakni, bagus atau tidak nya dari sebuah file system. Dari penelitian yang sudah dilakukan di dapat kan hasil berupa bahwa cassandra memiliki performansi throughput yang lebih bagus dibandingkan dengan HDFS selisihnya yakni sebesar 1818.75 detik untuk setiap operasi yang dijalankan, dan untuk kecepatan dalam memproses data didapatkan hasil berupa bahwa cassandra lebih bagus dibandingkan dengan HDFS selisihnya yakni sebesar 6028 detik.

Kata kunci : Big Data, performance, HDFS, CFS, load-read, load-write merupakan kata-kata kunci yang menjelaskan isi tulisan, biasanya bisa diambil dari judul dan abstrak. Maksimal enam buah dan ditulis dengan huruf kecil, kecuali singkatan

Abstract

At this time technology is growing rapidly. Based on research carried out by Big Data will dominate the technology market. The growth rate of Big Data increases every year, and the required file system (FS). Hadoop is an OS open source that is very popular among Big Data. But with the development and use of Big Data, the specifications on the file system also began to increase. A system that only provides to be made from the bottleneck category on Big Data. But this system starts still lacking, because there is no output in determining the performance of the file system. A system is needed to determine the performance of a file system. File System currently uses structured data based on one example, namely SQL. To return unstructured data, it is necessary to do it through File System which uses NoSQL. For this reason, we must test file system based on load-write and load-read using Hadoop Distribute File System (HDFS) and Cassandra File System (CFS). The results classification is divided into two, good or not from file system. From the research that has been done, the results can consist of cassandra having a performance of throughput which is better than the difference between HDFS which is equal to 1818.75 seconds for each operation carried out, and for the speed in the process of data obtained results from Cassandra better than HDFS the difference is 6028 seconds.

Keywords: Big Data, performance, HDFS, CFS, load-read, load-write.

1. Pendahuluan

Latar Belakang

Pada zaman ini teknologi berkembang dengan sangat pesat. Sebagai pengembang aplikasi yang berurusan dengan data diperlukan kemampuan Big Data Analytics. Big Data adalah sebuah teknik untuk menangkap, menganalisa, memproses serta memvisualisasikan dataset yang berpotensi sangat besar(1). Big Data mempunyai beberapa karakteristik yang sering dijumpai yaitu Volume, Varieties, dan Velocity. Di dalam karakteristik Varieties tipe data yang biasa digunakan yaitu terstruktur, semi-terstruktur, dan tidak terstruktur(2). File System saat ini menggunakan data yang tersusun secara terstruktur salah satu contohnya yaitu SQL. Untuk menangani data yang tidak terstruktur diperlukan suatu mekanisme File System yang menggunakan NoSQL.

Permasalahan yang sering dijumpai dalam Big Data yaitu dalam hal storage dan processing. Dalam processing kemampuan prosesing akan menjadi terhambat ketika pembacaan pada storage masih lambat,oleh karena itu dibutuhkan file system yang memiliki kemampuan pengaksesan data yang cepat.

Hadoop Distribute File System (HDFS) suatu File System yang mampu menyimpan data yang besar, menghubungkan blok data yang besar untuk aplikasi dalam sistem terdistribusi(3).

Cassandra File System (CFS) merupakan open-source, distribusi, penyimpanan luas kolom, dan sistem manajemen database NoSQL yang dirancang untuk menangani sejumlah besar data di banyak server komoditas, menyediakan ketersediaan tinggi tanpa titik kegagalan tunggal(4). untuk menangani sejumlah besar data di banyak server komoditas, menyediakan ketersediaan tinggi tanpa titik kegagalan tunggal, dengan replikasi tanpa master asinkron yang memungkinkan operasi latensi rendah untuk semua klien. CFS menawarkan pos relasional sebagai solusi basis data yang berarti jika melalui pos ini CFS tidak menawarkan semua fitur dari database tradisional tetapi tetap mengikuti keluarga kolom keyspace untuk penyimpanan basis data dan memperkenalkan indeks primer dan sekunder untuk data-data yang besar. CFS itu sendiri memiliki skema data yang tidak rasional dalam mendapatkan informasi data yang tidak terstruktur yaitu NoSQL.

Pada penelitian terkait (5), peneliti melakukan analisis performansi file system menggunakan HDFS dan CFS. Namun, dalam penelitian tersebut, peneliti menggunakan load-read dan load-wirte. Sehingga hasil througthput yang dibutuhkan belum maksimal. Atas latar belakang tersebut, penulis membahas mengenai analisis HDFS dan CFS pada performansi penyimpanan pada file system menggunakan NoSQL.

Topik dan Batasannya

Adapun batasan yang terdapat pada tugas akhir ini adalah sebagai berikut :

- Analisis menggunakan Stressing tools Yahoo! Cloud Serving Benchmarking
- Konfigurasi file system seluruhnya default, kecuali record count dalam menguji performansi difile system tersebut.
- Parameter pengujian hanya menggunakan run time dan throughtput.

Berdasarkan latar belakang yang sudah dipaparkan sebelumnya permasalahan yang ditarik adalah:

- Untuk membaca data yang tidak terstruktur dibutuhkan suatu mekanisme yang disebut NoSQL dan stressing tools yahoo! cloud sering benchmarking

Tujuan

1. Mengimplementasikan NoSQL file system menggunakan HDFS,HBase dan CFS untuk menentukan performansi file system tersebut.
2. Analisis performansi HDFS dan CFS dengan menggunakan stressing tools untuk menguji kemampuan file sistem berdasarkan load-read,load-write di kedua metode tersebut.

Organisasi Tulisan

Urutan penulisan laporan ini adalah sebagai berikut : Bagian 2 menunjukkan penelitian-penelitian terkait dengan tugas akhir ini. Sistem yang diajukan untuk analisis performansi NoSQL pada HDFS dan CFS akan dijelaskan di bagian 3. Pada bagian 4 akan didiskusikan mengenai hasil pengujian dan evaluasi sistem. Akhirnya, kesimpulan akan dipaparkan pada bagian 5.

2. Studi Terkait

Pada penelitian sebelumnya(4), software hadoop dianalisis menggunakan HDFS untuk menentukan apakah performansi penyimpanan tersebut baik atau buruk dalam membaca data yang tidak terstruktur. Acuan yang digunakan adalah HBase yang di deklarasi pada software hadoop tersebut. Pada penelitian ini dikatakan bahwa HBase bisa dijadikan sebagai acuan untuk menentukan performansi dalam membaca data yang tidak terstruktur dari penyimpanan di hadoop, namun tidak ada penjelasan detail dalam keamanan data berdasarakan node yang didapat. Sehingga metode ini masih belum bisa dijadikan sebagai acuan dalam menentukan performansi dari suatu software penyimpanan hadoop.

Pada penelitian berikutnya(2), diterapkan CFS yang sama konsepnya dengan HDFS tetapi ada hal yang membedakan dari file system tersebut yakni hanya metode yang digunakan pada software hadoop. Sehingga dengan data yang dihasilkan bisa dijadikan sebagai suatu acuan untuk menganalisis software hadoop lebih lanjut.

Pada penelitian ini HBase di kategorikan menjadi tiga yakni, ext3, ext4, dan XFS. Setiap kategori memiliki karakteristik tersendiri, untuk kategori ext3 hanya memerlukan CPU yang sedikit dan support terhadap Linux , kategori ext4 = memerlukan CPU yang banyak dan hanya support pada OS windows , dan kategori XFS = . Jika karakteristik dari suatu perangkat hanya bisa di linux maka dikategorikan ke dalam ext3, jika karakteristik hanya bisa dijalankan di OS windows maka dikategorikan sebagai ext4 . Sedangkan untuk karakteristik yang memiliki beban yang besar dan suatu perangkat yang tinggi, maka dikategorikan ke dalam XFS.

Penelitian sebelumnya(6), menggunakan NoSQL dalam menentukan performansi di CFS. Tool tersebut dirancang berdasarkan ukuran petabyte yang di distribusikan di beberapa server komoditas. Didalam tabel tidak ada dukungan atomicity, konsistensi, isolasi, dan daya tahan(ACID). seperti data yang tidak rasional, tidak mendukung operasi gabungan, dan tidak ada bahasa SQL(7). Sehingga masih dibutuhkan suatu metode untuk menguji performansi dari kedua file system tersebut. Untuk memperjelas perbandingan untuk setiap penelitian, lihat tabel 1 berikut :

Tabel 1. Perbandingan Penelitian

Judul Paper	Tahun	File System	Metode	Output
Big Data Modeling Methodology for Apache Cassandra	2015	CFS	Mapping dan Chebotko Diagram	Column
Hadoop-HBase for Large-Scale Data	2011	HDFS	MySql , Jobtracker , HQuorum peer	Load-read , Load Write
Benchmarking Cloud Serving Systems with YCSB	2011	HBase, Cassandra, BigTable, PNUTS, Sharded MySQL	MapReduce	Load-Read , Load Write

Berdasarkan tabel 1 diatas, pada penelitian ini akan menggunakan metode yang digunakan pada penelitian sebelumnya(5). Dari penelitian tersebut akan digunakan kategori yang telah ditemukan yakni, load-write dan load-read. Namun untuk metode yang digunakan akan menerapkan YCSB. Metode ini dipilih berdasarkan hasil perbandingan yang telah di paparkan sebelumnya. Lalu untuk parameter yang akan digunakan pada penelitian ini adalah Run Time dengan tambahan parameter throughput. Throughput dijadikan parameter pada penelitian ini, dikarenakan dari paramater ini bisa dianalisis seberapa bagus processing dalam mengolah data, dan untuk Run Time dijadikan parameter dikarenakan dari parameter ini bisa di analisis seberapa cepat processing dalam mengolah data.

2.1 Big Data

Big Data adalah frasa yang digunakan untuk mengartikan volume besar dari data terstruktur dan tidak terstruktur yang begitu besar sehingga sulit untuk diproses menggunakan database tradisional dan teknik perangkat lunak. Dalam kebanyakan skenario perusahaan volume data terlalu besar atau bergerak terlalu cepat atau melebihi kapasitas pemrosesan saat ini(6). Big Data memiliki beberapa karakteristik yaitu volume, variety, dan velocity.

1. Volume

   Volume pada big data memiliki jumlah data yang sangat besar sehingga dalam proses pengolahan data dibutuhkan suatu penyimpanan yang besar dan dibutuhkan analisis yang lebih spesifik.

2. Variety

   Variety big data memiliki bentuk format data yang beragam baik terstruktur ataupun tidak terstruktur dan bergantung pada banyaknya sumber data.

3. Velocity

   Velocity big data memiliki aliran data yang yang cepat dan real time.

2.2 Hadoop

Hadoop adalah framework atau platform open source berbasis Java di bawah lisensi Apache untuk support aplikasi yang jalan pada Big Data(). Terdapat empat layer utama pada hadoop yakni,

1. Hadoop common

   Pada layer ini terdapat library-libray umum yang mendukung library lainnya untuk dapat digunakan. Ini terkait perintah-perintah dasar yang ada pada Hadoop.

2. Hadoop Yarn

   Hadoop YARN merupakan framework yang digunakan untuk menajamen pekerjaan secara terjadwal (schedule) dan manajemen cluster data.

3. Data Mining

   Bagian ini sebenarnya bisa dikatan adalah API untuk menjalankan Map Reduce. Bagian ini mempermudah membuat dan menjalankan Map Reduce. Dengan demikian akan lebih mudah membuat dan menjalankan query.

4. DataBase NoSQL

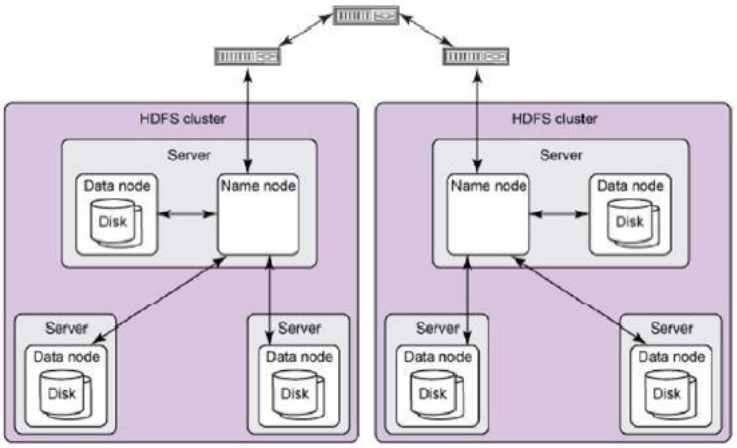
   Bagian ini ada karena proses map reduce biasanya makan waktu lama (karena data yang diproses baisanya besar) dan dilakukan secara periodik dan tidak sewaktu-waktu. Bagian ini memberikan akses data yang lebih cepat dan bisa sewaktu-waktu.

5. Linux Kernel

   Fondasi dasar hadoop adalah linux kernel, seperti ART yang memanfaatkan linux kernel untuk mengurangi penggunaan memori dan fitur threading. Dengan menggunakan linux kernel, sistem bisa memanfaatkan fitur penyimpanan yang ditawarkan linux itu sendiri.

2.3 Hadoop Distribute File System

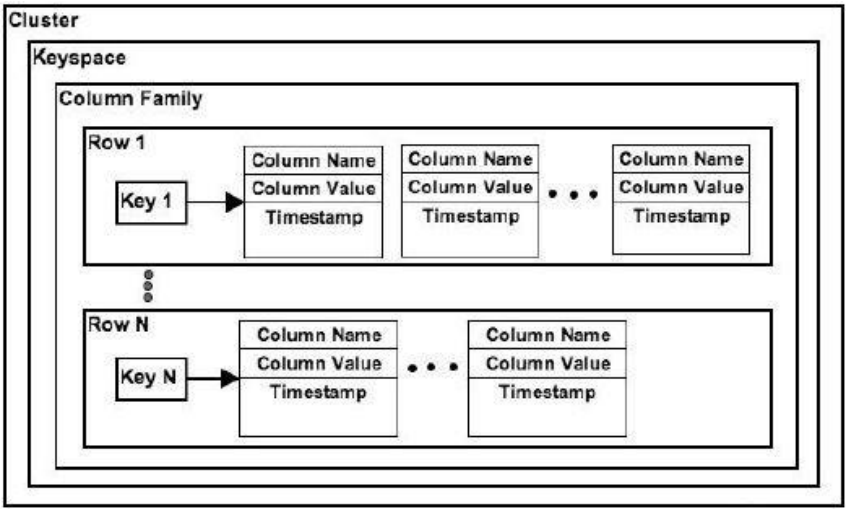
Hadoop Distribute File System adalah suatu konsep file system yang memungkinkan sistem untuk menyimpan suatu data dengan menggunakan suatu algoritma(8). Untuk menyimpan sebuah data HDFS membutuhkan input dan aturan yang jelas sehingga mampu untuk menentukan keluaran yang diinginkan. Pada penelitian sebelumnya, sudah diterapkan beberapa algoritma dan dianalisis akurasinya terhadap permasalahan klasifikasi Hadoop. HDFS menyediakan sebuah akses global ke dalam file di dalam cluster. Untuk portabilitas maksimum, HDFS diimplementasikan sebagai file system tingkat pengguna di java yang mengeksploitasi file system yang asil pada setiap node(9).



Gambar 1. Arsitektur HDFS
(7)

2.4 Cassandra File System

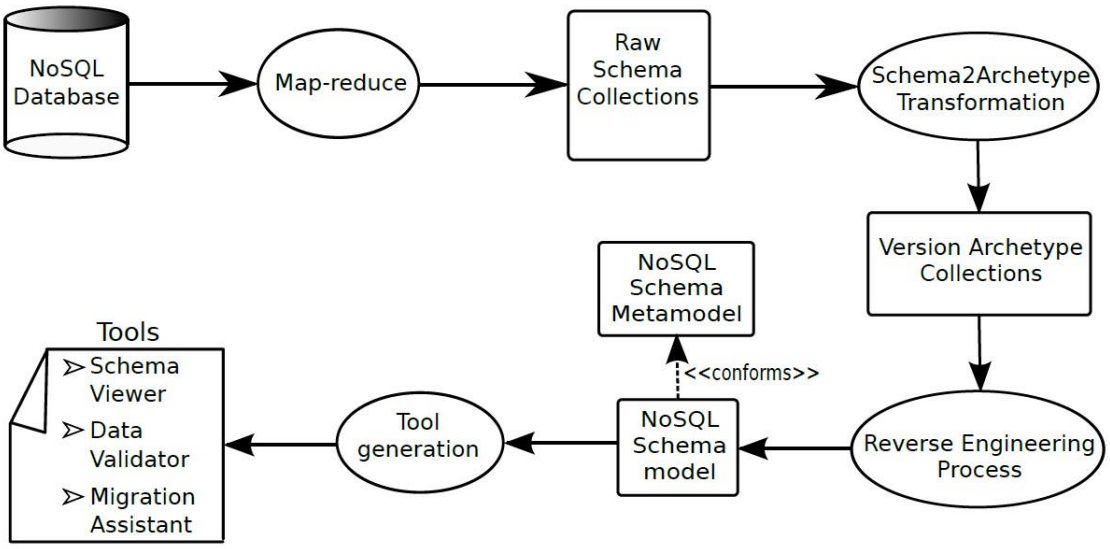
Cassandra File System adalah sebuah open source untuk menajemen database yang konsisten,high available, fault tolerant dan terdistribusi yang sangat scalable (dapat diukur). Cassandra menggunakan metode penglompokan dan distribusi data sekaligus model data Big Table yang juga menerapkan sistem non relasional. File system cassandra memiliki fasilitas replikasi dan terdesentralisasi, sehingga cassandra menawarkan dukungan untuk cluster yang mencakup beberapa pusat data dengan replikasi masterless-asinkron yang memungkinkan operasi latensi rendah disemua client. Meskipun cassandra menggunakan desain sharded, cassandra menggunakan arsitektur ”ring” peer-to-peer yang jauh lebih mudah untuk di koonfigurasi.(4).



Gambar 2. Arsitektur Cassandra
(10)

2.5 NoSQL

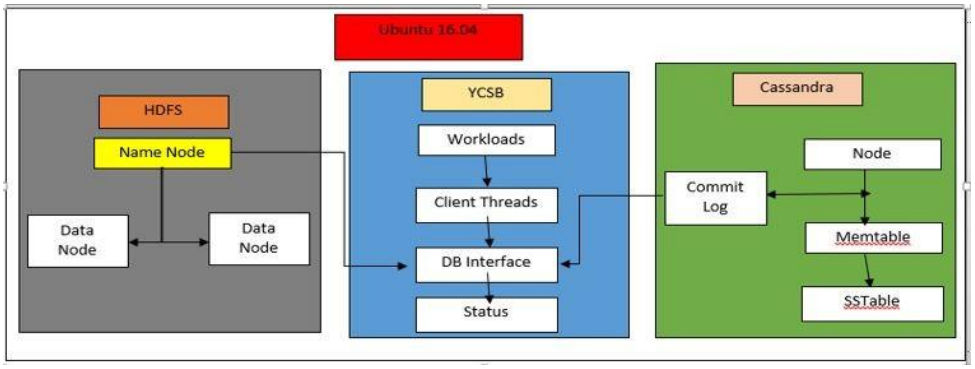
NoSQL merupakan database yang tidak relasional dan tidak terstruktur. NoSQL tidak seperti SQL yang menggunakan database dalam bentuk tabel, tetapi menggabungkan semua database yang tidak membedakan jenis-jenisnya dan tanpa karakteristik umum, dan menyusun sebagian data didalam bagian yang lainnya. Query yang ada di dalam tabel biasanya di nyatakan dalam bentuk Cassandra Query Language atau bisa disingkat CQL yang memiliki konsep seperti SQL. Tidak seperti layaknya SQL, CQL tidak mendukung operasi dalam membaca bilangan biner tetapi memiliki sejumlah aturan dalam memastikan efisiensi dan skalabilitas(11). Berikut gambar dari alur cara kerja NoSQL dan konseptual model data, bisa dilihat pada gambar 3.



Gambar 3. NoSQL Architecture
(7)

3. Sistem yang Dibangun

Penelitian ini memiliki dua arsitektur file system utama, yaitu dengan menggunakan arsitektur hadoop distributed file system dengan cassandra file system. Arsitektur pertama dimulai dengan memasang Hadoop Lalu memasang Hbase di dalam hadoop kemudian memasang tools YCSB untuk di jalan kan. Arsitektur kedua dimulai dengan memasang cassandra di mesin yang berbeda dan memasang tools yang sama seperti arsitektur sebelumnya.



Gambar 4. Sistem yang dibangun

3.1 Skenario Generate Data

3.2 Yahoo Cloud Serving Benchmark

Masalah pada Big Data yang diangkat yaitu adanya hambatan atau Bottleneck yang menghambat pemrosesan pada Big Data, sehingga parameter yang diujikan pada penelitian ini adalah load-read dan load-write yang memp-resentatifkan kecepatan pembacaan data dan penulisan data yang akan dilakukan oleh file system. Berdasarkan paper,Benchmarking Tools yang cocok untuk melakukan pengujian ini adalah Yahoo Cloud Serving Benchmark (YCSB).

Yahoo Cloud Serving Benchmark (YCSB) adalah sebuah framework yang memiliki tujuan perbandingan pada performansi database,file system dan cloud data serving(12). YCSB memiliki lapisan abstraksi untuk beradaptasi dengan API store tertentu, untuk mengumpulkan metrik kinerja yang diakui secara luas dan untuk menghasilkan campuran workloads. Meskipun berguna untuk mengkarakterisasi kinerja dasar dari workloads sederhana, seperti penyisipan baris yang acak atau berurutan, pencarian atau penghapusan, YCSB tidak memiliki dukungan untuk pembandingan fungsi lanjutan yang dianggap penting dan semakin didukung oleh tables store.

3.3 Skenario Data

Pada skenario Data kali ini, ada enam rancangan data dimana rancangan tersebut sudah di sediakan oleh tools YCSB dan keluaran dari tools itu ada operation, record, Disk I/O, insert, dan read dimana tiap langkah-langkah ini akan dijelaskan dibawah:

Workloads	Operations	Record Selection	Application Example
A-Update heavy	Read: 50% Write: 50%	Zipfian	Sesi menyimpan rekaman tindakan terbaru dalam sesi pengguna
B-Read Heavy	Read: 95% Write: 5%	Zipfian	Pemberian tag foto; menambahkan tag adalah pembaruan, tetapi sebagian besar operasi adalah untuk membaca tag
C-Read Only	Read: 100%	Zipfian	Cache profil pengguna, di mana profil dibuat di tempat lain (misal., Hadoop)
D-Read Latest	Read: 95% Write: 5%	Lastest	Pembaruan status pengguna; orang ingin membaca status terbaru
E-Short ranges	Scan: 95% Insert: 5%	Zipfian/Uniform	Percakapan berulir, di mana setiap pemindaian untuk posting di thread yang diberikan (diasumsikan dikelompokkan oleh id thread)
F Read-modify-write	Read: 50% Scan:50%	Zipfian	Menguji sekaligus memasang suatu inputan di dalam database

3.4 Skenario Pengujian

Pada pengujian kali ini dilakukan suatu skenario dalam menjalankan tools yang telah dipasangkan ke dalam kedua file system tersebut. berikut langkah-langkah dalam menjalankan tools tersebut:

1. Siapkan sistem database yang akan diuji
2. Pilih interface layer DB yang sesuai
3. Piliih workload yang sesuai dengan urutannya.
4. Pilih parameter runtime yang sesuai.
5. Memuat Data.
6. Kemudian Eksekusi workload.

Urutan dalam menjalankan workload harus sesuai karena ukuran keluaran data yang sudah dieksekusi bisa optimal. Urutan dalam menjalankan workload yaitu di mulai dari: A-load, A-read, B-read, C-Read, F-Read, D-Read, E-Load, dan E-Read.

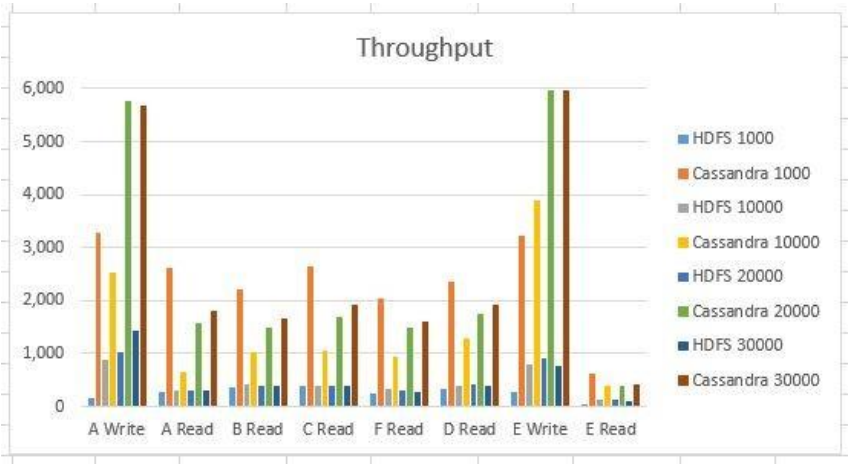
4. Evaluasi

Bagian ini menampilkan hasil analisis penulis. Seperti yang dijelaskan sebelumnya, parameter yang dipilih untuk diukur adalah Run Time dan Throughtput.

4.1 Throughtput

Terlihat dari gambar perbandingan throughtput pada skema HDFS dengan skema Cassandra umum sangat jauh berbeda. Dikarenakan pada arsitektur HDFS didesain untuk dapat mengelola data berukuran super besar dalam suatu sistem terdistribusi dan HDFS memiliki karakteristik ’fault tolerance’ atau mampu menjamin keutuhan data meskipun terjadi kegagalan pada beberapa komputer yang diperkerjakannya. HDFS juga mampu menangani input

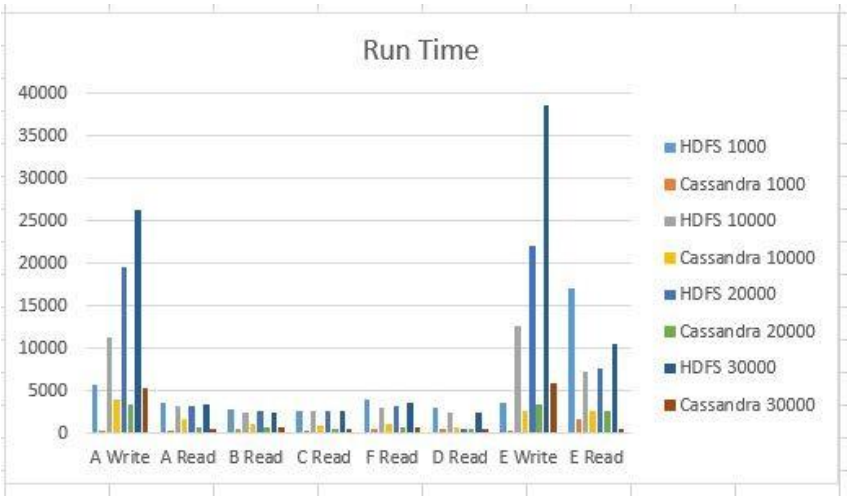
data yang terjadi secara terus-menerus dari ribuan user yang selama menjadi 'bottle neck' pada sistem database sebelumnya, tetapi dikarenakan resource yang terbatas maka HDFS hanya bisa menampung data yang sangat sedikit. HDFS mengelola data secara langsung dalam sistem file. File besar dipecah menjadi blok-blok kecil dan direplikasi pada banyak node data. Sedangkan untuk cassandra sendiri memberikan solusi database post relational. Melalui pos relasional cassandra mengikuti keluarga kolom keyspace untuk menyimpan data dan memperkenalkan indeks primer dan sekunder untuk ketersediaan data yang tinggi. Sehingga di dapatkan hasil berupa bahwa cassandra dalam memproses sebuah data lebih baik dibandingkan HDFS.



Gambar 5. Hasil Throughtput

4.2 Run Time

Terlihat pada gambar, perbandingan Run Time pada skema HDFS dengan skema Cassandra sangat berbeda karena di dalam HDFS dirancang untuk mengambil satu file besar, membaginya menjadi beberapa file yang lebih kecil dan mendistribusikannya di seluruh node. HDFS harus mengumpulkan beberapa file dari node yang berbeda dan memberikan beberapa hasil yang sesuai dengan permintaan client, sedangkan untuk cassandra adalah pilihan yang sempurna untuk menulis dan membaca banyak catatan kecil. Arsitekturnya yang tanpa master dan slave memungkinkan load-write cepat dan load-read dari sembarang node. Sehingga dalam memproses sebuah data cassandra cukup terbilang lebih cepat di banding dengan HDFS.



Gambar 6. Hasil Run Time

5. Kesimpulan

Hasil Analisis dari seluruh pengujian yang dilakukan dalam penelitian tugas akhir ini dapat simpulkan sebagai berikut: Untuk Throughput, Performansi dalam memproses sebuah data didalam sebuah file system, hadoop distri-

bute file system dikatakan lebih buruk dibandingkan dengan cassandra file system yaitu selisihnya sebesar 1818,75 operasi per detik dikarenakan HDFS mengelola data secara langsung dalam sistem file. File besar dipecah menjadi blok-blok kecil dan direplikasi pada banyak node data. Sedangkan untuk cassandra Melalui pos relasional cassandra mengikuti keluarga kolom keyspace untuk menyimpan data dan memperkenalkan indeks primer dan sekunder untuk ketersediaan data yang tinggi.

Dan untuk Run Time, Kecepatan dalam memproses sebuah data didalam sebuah file system, hadoop distribute file system dikatakan lebih lambat dibandingkan dengan cassandra file system yaitu selisihnya sebesar 6028 per detik. Dikarenakan HDFS mengikuti arsitektur Master dan Slave sementara cassandra mengimplementasikan arsitektur peer to peer. Cluster HDFS dapat diskalakan secara horizontal dengan menambahkan lebih banyak node ke cluster atau secara vertikal dengan memutakhirkan konfigurasi perangkat keras dan perangkat lunak. Cassandra menggabungkan google big table yang sangat dinamis dan dapat diskalakan dan Dynamo DB Amazon. Cassandra menggunakan bahasa kueri atau antarmuka baris perintah untuk mengakses data yang disimpan dalam basis datanya.

Daftar Pustaka

[1] Y. Kaneko and T. Ito, "A reliable cloud-based feedback control system." IEEE, 2016.

[2] D. U. R. Pol, "Big data analysis: Comparision of hadoop mapreduce and apache spark." IJESC, 2016.

[3] C.-Y. Lin and Y.-C. Lin, "A load balancing algorithm for hadoop distribute file system." 18th International Conference on Network-Based Information Systems, 2015.

[4] K. Dwivedi and S. K. Dubey, "A taxonomy and comparison of hadoop distributed file system with cassandra file system." Asian Research Publishing Network (ARPN), September 2015.

[5] E. T. R. R. R. S. Brian F. Cooper, Adam Silberstein, "Benchmarking cloud serving systems with ycsb," 2010.

[6] Y. Permana, "Mengenal big data," 2016. [Online]. Available: https://www.codepolitan.com/mengenal-big-data

[7] S. P. A. M. Sugam Sharma, Ritu Shandilya, Leading NoSQL models for handling Big Data: a brief review, 2016.

[8] A. P. Jai Prakash Verma1, "Comparison of mapreduce and spark programming frameworks for big data analytics on hdfs," 2016. [Online]. Available: www.csjournalss.com

[9] S. R. Jeffrey Shafer and A. L. Cox, "The hadoop distributed filesystem: Balancing portability and performance," 2010.

[10] P. K. J. H. M. G. Elif Dede, Bedri Sendir, "Tan evaluation of cassandra for hadoop," 2013.

[11] S. L. Artem Chebotko, Andrey Kashlev, "A big data modeling methodology for apache cassandra." IEEE, 2015.

[12] M. N. Vora, "Hadoop-hbase for large-scale data." International Conference on Computer Science and Network Technology, 2011.