INTISARI

IMPLEMENTASI BULKLOAD UNTUK DATABASE HBASE PADA LINGKUNGAN HDFS

Danny Steven
12/334695/PA/14928

Saat ini, dengan perkembangan teknologi seperti *Mobile Data* yang semakin pesat dan data yang dihasilkan semakin banyak jumlah dan ragamnya, akibatnya *Relational Database Management System* (RDBMS) tidak mampu mengkontrol data yang terkumpul. Salah satu solusi alternatif adalah menggunakan *Not Only SQL* (NoSQL) *database* seperti HBase. Salah satu fitur spesial yang dimiliki HBase adalah *BulkLoad*. *BulkLoad* adalah fitur HBase untuk menyimpan data dalam jumlah besar sekaligus dan tidak harus memiliki bentuk tabel yang terstruktur dan kaku seperti *Relational Database Management System* (RDBMS).

Penelitian ini ditujukan untuk mengeksplorasi penggunaan *BulkLoad* pada HBase dari sisi durasi waktu yang diperlukan untuk menyimpan data. Untuk menjalankan program *BulkLoad*, yang pertama dilakukan adalah mengumpulkan sampel data. Sampel data yang digunakan adalah dokumen XML. Setiap dokumen XML akan di *preprocessing* menjadi *input splits* sebanyak 150 dan akan disimpan kedalam *Hadoop Distributed File System* (HDFS) sebagai dataset dengan nama *batch*. *Batch* sebanyak 14 akan terdiri dari *input splits* dengan jumlah yang berbeda setiap *batch*. Selanjutnya, program *BulkLoad* akan dijalankan untuk setiap *batch* dan masing-masing *batch* akan dijalankan sebanyak 3 kali pengulangan dengan menggunakan *core* yang berbeda setiap kali dijalankan. *Core* yang digunakan pada penelitian ini sebanyak 2,3 dan 4.

Program *BulkLoad* memberikan hasil yang berbeda setiap kali program dijalankan. Hasil paling jelas terlihat pada *batch 14* yang memiliki *input splits* paling banyak, dimana durasi waktu pada *2-core* sebesar 13 menit, jauh lebih lama dibandingkan *3-core* dan *4-core* dengan durasi waktu 9 menit. Hasil penelitian juga menunjukkan *3-core* dan *4-core* tidak berbeda jauh dan *4-core* memberikan hasil yang lebih buruk dibandingkan *3-core*, seperti pada batch 4 dan batch 11 dimana durasi waktu *3-core* lebih cepat 1 menit dibandingkan *4-core*. Hal ini terjadi karena durasi waktu dipengaruhi oleh *error* berupa *killed map task* yang menunjukkan berapa kali program *BulkLoad* gagal membaca *input splits*. *Killed map task* pada *2-core* sebesar 18, *3-core* sebesar 13 dan *4-core* sebesar 15.

Kata Kunci: HBase, *BulkLoad, core, memory, killed map task*

ABSTRACT

BULKLOAD IMPLEMENTATION FOR HBASE DATABASE ON HDFS ENVIRONMENT

Danny Steven
12/334695/PA/14928

The rapid development of technology, such as Mobile Data technology resulting in increase of volume and variety of data. This increase will likely continue and limit the usage of Relational Database Management System (RDBMS). Therefore, it is a necessity to find an alternative database such as Not Only SQL (NoSQL) database like HBase. One of the special feature of HBase is BulkLoad. BulkLoad is a method prepared by HBase to move large amount data to HBase database at the same time without the need to create the table in a rigid form like on Relational Database Management System (RDBMS).

This research intend to observe the usage of BulkLoad in HBase using time measurement in minutes to see the speed of BulkLoad. Firstly, data sample is collected in the form XML document. After that, every XML document enter preprocessing phase to create input splits as much as 150. All input splits then moved to Hadoop Distributed File System (HDFS) as batch. Each Batch have different amount of input splits and batch is created up to 14 batch. Finally, BulkLoad run on each batch for 3 times, each times using different core. This research used 3 different numbers of core, which is 2-core, 3-core and 4-core.

The result showed different elapsed time for every BulkLoad that is ran on each batch. The most significant result showed that in batch 14, which has the largest amount of input splits, BulkLoad need 13 minutes to finish its job at 2-core, slower than 3-core and 4-core which need only 9 minutes. Next, experiment also showed that 3-core and 4-core did not have much difference in elapsed time and 4-core shows result worse than 3-core. On batch 11, BulkLoad in 3-core showed elapsed time 1 minute faster than 4-core. The elapsed time difference in BulkLoad execution is influenced by the number of errors in killed map task. Killed map task recorded the numbers of how much BulkLoad failed to read the input splits. Killed map task in 2-core is 18 errors, 3-core is 13 errors and 4-core is 15 errors.

Keywords: HBase, BulkLoad, core, memory, killed map task