Analisa Multithreading Pada Sistem Rekomendasi Menggunakan Metode *Collaborative Filtering* Dengan *Apache* Mahout

Multithreading Analysis On Recommended System Using Collaborative Filtering Method With Apache Mahout

Aminudin[1], Muhammad Alwi[2]
[1,2]Jurusan Teknik Informatika, Universitas Muhammadiyah Malang, Malang
E-mail : [1]aminudin2008@umm.ac.id, [2]alwi.muhammad007@gmail.com

Abstrak

Apache mahout saat ini melakukan pengembangan sistem rekomendasi yang didalamnya menggunakan metode Collaborative Filtering, namun dalam implementasinya masih memiliki kekurangan didalam waktu pemrosesan yang masih memakan waktu cukup lama untuk memproses data yang berukuran besar. Penelitian ini akan memanfaatkan multithreading untuk mempercepat waktu pemrosesan data menggunakan library Apache Mahout. Dalam penelitian ini, diketahui bahwa adanya multithreading mampu meningkat kinerja dalam pemrosesan waktu eksekusi data pada Apache Mahout. Pengujian yang telah dilakukan dengan jumlah 20 juta data, didapatkan hasil pengujian pada single thread dengan lama waktu pemrosesan datanya selama 2218 detik. Dan pada pengujian untuk 4 thread didapatkan hasil 764 detik, dan kemudian untuk 8 thread didapatkan hasil 691 detik dan pada pengujian untuk 16 thread didapatkan hasil 1097 detik. Dari berapa pengujian yang telah dilakukan telah membuktikan bahwa multithreading mampu meningkatkan kinerja apache mahout dalam sistem rekomendasi asalkan jumlah thread tidak melebihi kapasitas ukuran thread yang ada di processor.

Kata kunci—Apache Mahout, Collaborative Filtering, Multithreading

Abstract

Apache mahout is currently developing a recommendation system that uses the Collaborative Filtering method, but in its implementation still lacks in processing time that still takes a long time to process large data. This research will utilize multithreading to speed up data processing time on apache mahout. In this research, it is known that the existence of multithreading able to increase its performance in processing data execution time on apache mahout. From the test that has been done with the amount of 20 million data, obtained the test results on the singlethread with the duration of data processing for 2218 seconds. And on the test for 4 threads obtained 764 seconds, and then for 8 threads obtained 691 seconds results and on the test for 16 threads obtained results 1097 seconds. From how many tests have been done has proven that multithreading able to improve apache mahout performance in the recommendation system as long as the number of threads does not exceed the capacity of thread size in the processor.

Keywords—*Apache Mahout, Collaborative Filtering, Multithreading*

1. PENDAHULUAN

Perkembangan teknologi informasi dan telekomunikasi pada saat ini mengalami pertumbuhan yang sangat pesat, hal ini membuktikan bahwa kebutuhan akan teknologi informasi dan telekomunikasi sangat dibutuhkan oleh manusia pada era sekarang. Kemudahan akses data dan kecepatan akses data pada teknologi informasi dan telekomunikasi merupakan fokus utama dalam perkembangan teknologi tersebut. Multithreading progamming sering digunakan sebagai salah satu teknik yang dapat digunakan untuk mengatasi hal kecepatan akses data pada sebuah teknologi telekomunikasi dan informasi

Multithreading menerapkan suatu proses eksekusi sebuah kerja program komputer yang dikerjakan secara bersamaan sehingga mampu mempercepat proses pada suatu program pada komputer [1]. Multithreading memanfaatkan multiple processor yang ada pada komputer, untuk mengoptimalkan proses pada komputer tersebut, sehingga memungkinkan komputer untuk mengerjakan suatu proses bisa lebih cepat.

Machine learning merupakan salah satu teknologi yang belakangan ini sering digunakan dan dikembangkan. *Machine learning* sendiri mempunyai jenis – jenis yang berbeda, ada klasifikasi, klastering, sistem rekomendasi dan lain sebagainya. Dalam implementasi *machine learning* juga membutuhkan kinerja processor yang sangat tinggi dikarenakan didalamnya terdapat perhitungan dari beberapa algoritma untuk mengeksekusi perintah – perintah didalamnya. Banyak aplikasi yang pada saat ini sedang fokus dalam pengembangan *machine learning* salah satunya ialah Apache Mahout. Apache Mahout menerapkan algoritma-algoritma di dalam *machine learning* di dalam lingkungan yang memerlukan skalabilitas yang besar [2].

Apache Mahout pada saat ini melakukan pengembangan tentang aplikasinya yang mengimplementasikan *machine learning* dalam bentuk sistem rekomendasi. Sistem rekomendasi sering digunakan sebagai teknonolgi *machine learning* pada saat ini [3] hal ini dikarenakan banyaknya sistem *e-commerce* yang bermunculan. Dalam pembuatan sistem rekomendasi Apache Mahout menggunakan metode Collaborative Filtering yang merupakan metode paling umum yang digunakan pada sistem rekomendasi pada saat ini [4] Dalam pengembanganya Apache Mahout diperuntuhkan pada data yang berskala besar, hal itu memungkinkan penggunaan *resource*-nya juga besar dan memakan waktu yang lama.

Penelitian yang dilakukan oleh [5] mengatakan bahwa Apache Mahout memerlukan waktu yang sangat lama untuk memproses dataset yang besar dalam klastering menggunakan algoritma K-Means sehingga dipergunakanya sistem parallel atau multithreading untuk mempercepat waktu eksekusi datanya. Dalam penelitian [6] mengatakan bahwa besarnya data yang ada pada internet untuk sistem rekomendasi mengakibatkan penurunan performa sistem rekomendasi tersebut sehingga menggunakan hadoop multinode cluster untuk mengembalikan performa sistem rekomendasi dalam Apache Mahout. Dan juga dalam penelitian lain [7] juga meneliti tentang penggunaan multithreading untuk mendeteksi ikan dibawah laut dengan menggunakan metode Fuzzy C-Means.

Beberapa penelitian telah banyak menjelaskan bahwa Apache Mahout masih mempunyai kekurangan dalam waktu pemrosesan data pada data yang berskala besar. Hal inilah yang menjadi alasan kenapa penelitian ini dilakukan. Lama waktu pemrosesan data juga sangat penting dalam sebuah sistem rekomendasi, untuk mengetahui hal itu maka penelitian ini akan mencoba menggunakan solusi dari [5] di dalam mengimplementasikan sistem rekomendasi menggunakan metode item based Collaborative Filtering, dengan harapan agar dapat meningkatkan kecepatan waktu pemrosesan datanya.

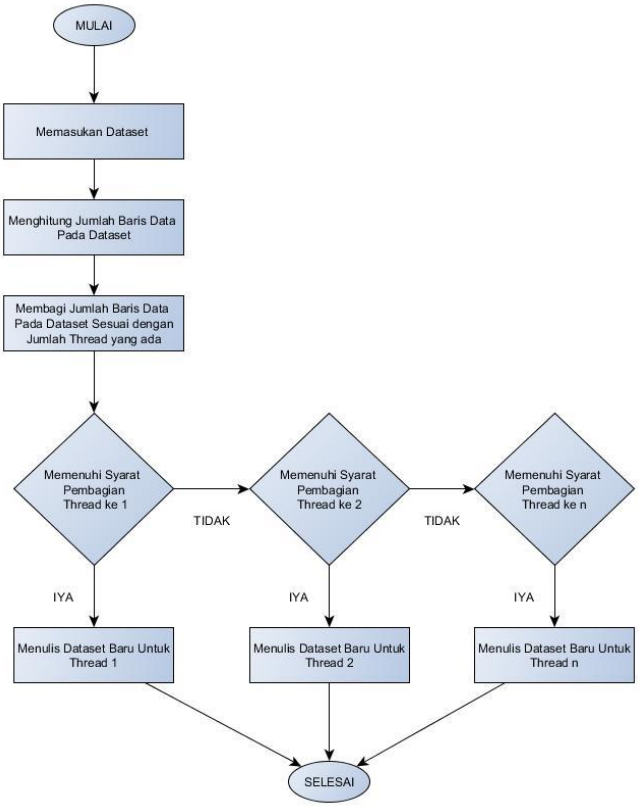
2. METODE PENELITIAN

Metode dalam penelitian ini menggunakan tahapan – tahapan penelitian pada umumnya, yang pertama yaitu mencari studi pustaka, yaitu dengan menganalisis dari berbagai

sumber seperti paper dan juga artikel dari internet yang terkait [8] dan kemudian melakukan pemgumpulan data, setelah itu melakukan langkah pembuatan desain sistem dan skenario pengujian. Desain sistem ini akan mempunyai beberapa tahapan diantaranya yaitu pemecahan data, pemecahan data pada sistem ini bertujuan agar data yang digunakan bisa dibagi rata pada beberapa thread yang akan digunakan untuk memproses data. Setelah setiap thread menerima data masing – masing thread akan melakukan pemrosesan data tersebut dengan menggunakan metode *item based Collaborative filtering* menggunakan *library* Apache Mahout untuk membuat sistem rekomendasi.

2.1 Rancangan Pemecahan Dataset

Tahapan awal pada sistem yang terpenting ialah pemecahan data, hal ini dikarenakan sistem yang dibangun merupakan jenis sistem yang didalamnya terdapat pemrosesan data yang dikerjakan dalam satu waktu atau bisa disebut juga dengan parallel programming. Pemecahan data bertujuan untuk membagi rata dataset yang akan dibagikan untuk beberapa thread yang akan memproses dataset tersebut untuk dijadikan sebuah rekomendasi. Gambar 1 menunjukan alur pemecahan dataset dilakukan.

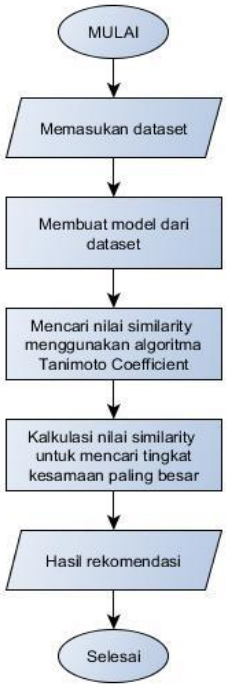


Gambar 1 Alur Pemecahan Dataset

2.2 Rancangan Item Based Collaborative Filtering

Penerapan metode Item Based Collaborative Filtering di Apache Mahout dilakukan dengan cara membuat program dengan ekstensi .java yang kemudian program tersebut ditempatkan dalam satu package dengan *library* Apache Mahout [5]. Gambar 2 menjelaskan alur pembuatan sistem rekomendasi menggunakan apache mahout yang didalamnya menggunakan pendekatan Item Based Collaborative Filtering yang menggunakan algoritma Tanimoto Coefficient Similarity untuk mencari tingkat kesamaan item didalamnya. Pada tahapan awal yaitu dengan memasukan dataset kemudian dari dataset tersebut akan dibuat modelnya dan kemudian dari model dataset tersebut akan dicari tingkat kesamaan itemnya dengan item yang lain menggunkan algoritma Tanimoto Coefficient Similarity , dan setelah

mendapatkan hasil dari nilai kesamaan setiap item tersebut, nilai - nilai tersebut akan dikalkulasi untuk mencari tingkat kesamaan yang paling besar untuk dijadikan sistem rekomendasi.



Gambar 2 Rancangan Metode Collaborative Filtering di Apache Mahout

2.3 Tanimoto Coefficient Similarity

Tanimoto coefficient merupakan salah satu algoritma item similarity pada Apache Mahout, algoritma ini biasa disebut juga dengan Jaccard similarity coefficient, dalam persamaan ini nilai dari rating diabaikan, tingkat kesamaan hanya diukur dari prefensi user. Persamaan ini menggunakan rasio perpotongan item dengan beberapa kumpulan user sebagai ukuran kesamaanya [9].

$$T(a,b) = \frac{\mathrm{Nc}}{\mathrm{N}a + Nb\text{-}Nc} \quad (1)$$

Dimana : Na merupakan user yang merating item a; Nb merupakan user yang merating item b dan Nc merupakan user yang merating keduanya

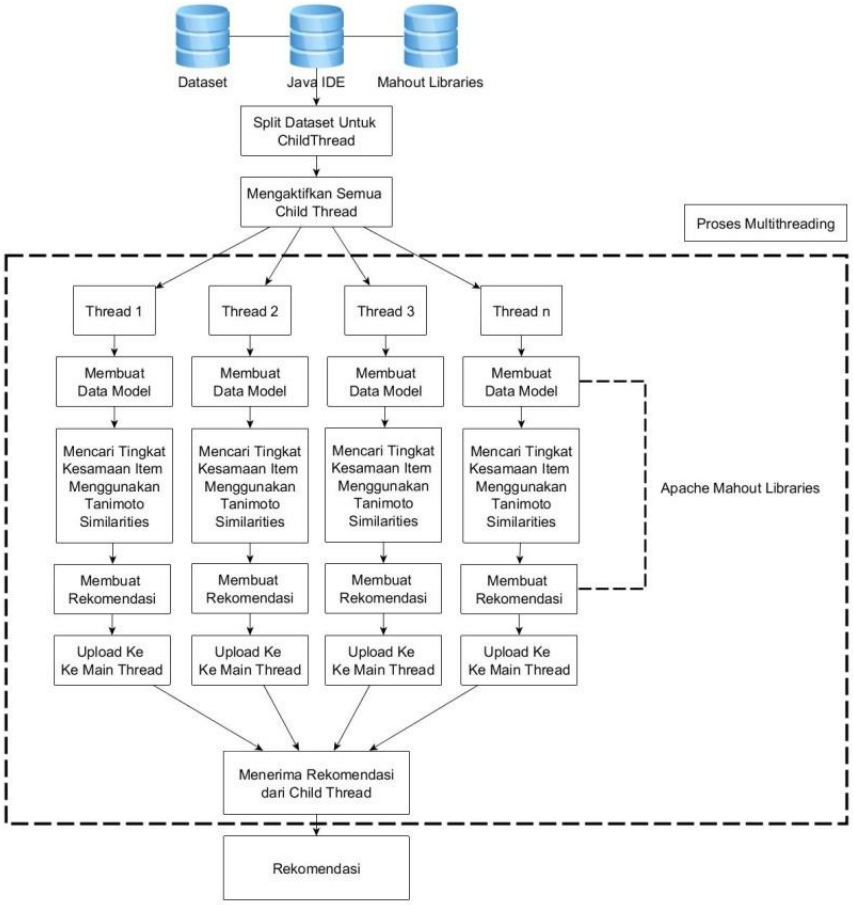
2.4 Rancangan Multithreading pada Apache Mahout

Pada tahapan kali adalah tahapan inti dari penelitian ini, disini proses penggabungan antara Apache Mahout dan multithreading akan dilakukan. Dalam tahapan ini dataset yang telah dipecah akan diproses untuk membuat sebuah sistem rekomendasi yang menggunakan metode Collaborative Filtering yang akan dijalankan dengan *library* Apache Mahout yang pada pemrosesan datanya akan menggunakan multithreading.

Desain alur sistem yang digunakan untuk penelitian seperti yang ada pada Gambar 3 yang menjalaskan alur sistem ini akan berjalan. Ada beberapa tahapan dalam sistem ini berikut merupakan tahapan – tahapanya. Tahapan awal ialah memasukan libraries apache mahout dan dataset yang akan digunakan sebagai data rekomendasi kedalam Java IDE. Tahapan selanjutnya ialah upload data kedalam main thread, setelah main thread menerima dataset-nya kemudian dataset tersebut akan dipecah menjadi beberapa bagian untuk dibagikan kepada beberapa thread lainya yang dalam penelitian ini disebut *child thread*. Setelah *child thread* menerima dataset

masing – masing dari *main thread*, setiap *child thread* akan memproses dataset tersebut untuk dijadikan sebuah rekomendasi.

Tahapan dalam pembuatan rekomendasi ini akan mengunakan *library* Mahout sebagai *engine* rekomendasinya. Sistem akan mencari tingkat kesamaan item dengan menggunakan library Mahout. Setelah sistem memperoleh perhitungan kesamaan antara beberapa item, sistem akan mengkalkulasi hasil perhitungan tersebut dan memilih tingkat kesamaan item yang paling tinggi untuk digunakan sebagai rekomendasi. Setelah setiap child thread memiliki rekomendasi masing – masing dari setiap dataset yang dibagikan oleh *main thread* sebelumnya, *child thread* akan mengirimkan hasil rekomendasi tersebut kepada *main thread*. Kemudian *main thread* menggabung semua rekomendasi yang telah dikirim oleh *child thread* untuk dijadikan rekomendasi.



Gambar 3 Rancangan Multithreading pada Apache Mahout

3. HASIL DAN PEMBAHASAN

Pengujian pada sistem ini akan dilakukan beberapa kali sesuai dataset dan juga thread yang digunakan. Pengujian dilakukan untuk menguji waktu pemrosesan data, dan juga penggunaan *resource* pada komputer seperti penggunaan memori dan penggunaan CPU. Setiap setelah melakukan satu kali pengujian komputer akan di restart yang bertujuan untuk refresh *resource* yang sebelumnya telah dipakai untuk pengujian. Kemudian untuk *resource* yang digunakan seperti penggunaan memori dan *processor* akan dicari nilai maksimalnya dalam task manager sebagai acuan pengujian.. Untuk pengujian kecepatan waktu, pencatatan waktu dimulai dari setiap thread di start secara bersamaan dan diakhiri dengan thread yang paling terkahir menyelesaikan pemrosesanya. Jumlah thread yang digunakan dalam pengujian menggunakan variasi thread yaitu dengan menggunakan satu thread sampai enam belas thread.

3.1 Pengujian dan Analisa Kecepatan Pemrosesan Data

Tujuan pengujian pada tahap ini yaitu untuk mengetahui berapa lama Apache Mahout memproses data rekomendasi dengan banyak data 20 juta, 10 juta dan 1 juta dengan membagi 1 thread sampai dengan16 thread. Hasil pengujian kecepatan proses dengan 20 juta data di tampilkan pada Tabel 1.

Tabel 1 Hasil Pengujian Waktu Eksekusi 20 Juta Data

Jumlah Thread	Waktu (Detik)
1 Thread	2218
2 Thread	1213
4 Thread	764
6 Thread	711
8 Thread	691
10 Thread	857
12 Thread	734
14 Thread	692
16 Thread	1097

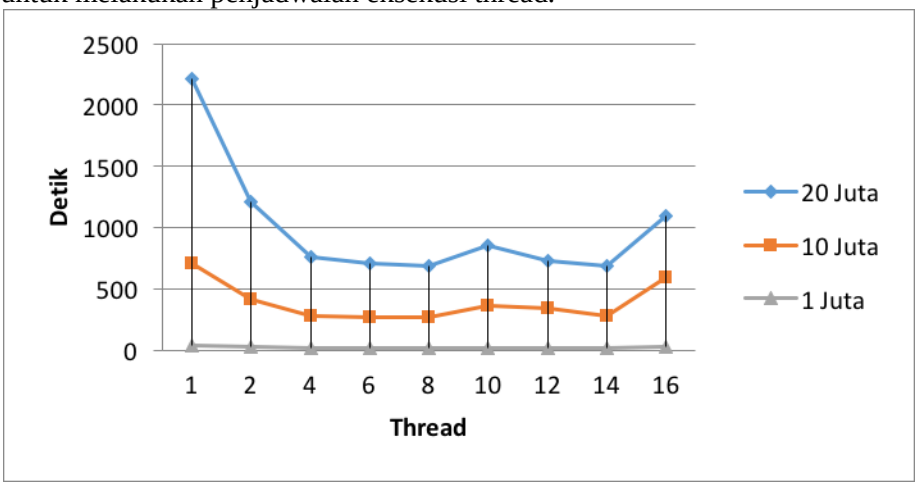
Dari Tabel 1 dapat dilihat hasil dari pengujian waktu eksekusi data yang berjumlah 20 juta data, dalam hasil pengujian tersebut penggunaan *multithreading* mampu meningkatkan kecepatan eksekusi datanya, namun dilihat dari hasil tersebut, penggunaan empat thread sampai dengan empat belas thread tidak memiliki perbedaan yang cukup signifikan, hal itu dikarenakan maksimal thread yang bisa digunakan dalam sistem hanya empat thread sehingga penggunaan thread lebih dari empat thread menjadi kurang maksimal. Dan untuk penggunaan enam belas thread waktu eksekusi datanya semakin naik secara signifikan hal itu dikarenakan thread yang semakin banyak dalam pemrosesan diluar dari kemampuan sistem untuk memproses thread secara bersamaan akan membuat sistem lamban untuk melakukan penjadwalan eksekusi thread.

Tabel 2 Hasil Pengujian Waktu Eksekusi 10 Juta Data

Jumlah Thread	Waktu
1 Thread	713
2 Thread	417
4 Thread	282
6 Thread	273
8 Thread	266
10 Thread	365
12 Thread	340
14 Thread	280
16 Thread	590

Dari Tabel 2 bisa dilihat hasil dari pengujian waktu eksekusi data yang datanya berjumlah 10 juta data, dalam hasil pengujian tersebut hampir sama dengan pengujian untuk 20 juta data, penggunaan Multithreading mampu meningkatkan kecepatan eksekusi datanya, namun dilihat dari hasil tersebut, penggunaan empat thread sampai dengan empat belas thread tidak memiliki perbedaan yang cukup signifikan, hal itu dikarenakan maksimal thread yang bisa digunakan dalam sistem hanya empat thread sehingga penggunaan thread lebih dari empat thread menjadi kurang maksimal. Dan untuk penggunaan enam belas thread waktu eksekusi datanya semakin

naik secara signifikan hal itu dikarenakan thread yang semakin banyak dalam pemrosesan diluar dari kemampuan sistem untuk memproses thread secara bersamaan akan membuat sistem lamban untuk melakukan penjadwalan eksekusi thread.



Gambar 4 Grafik Pengujian Waktu Eksekusi dalam Satuan Detik

Dari Gambar 4 dapat dilihat hasil dari pengujian eksekusi data yang menggunakan beberapa jumlah dataset. Dari hasil pengujian tersebut bisa dilihat bahwa multithreading sangat berperan penting dalam pemrosesan data yang berskala besar, contoh dari grafik diatas bisa dilihat bahwa eksekusi dataset yang bererukuran 10 juta data dan 20 juta data memberikan hasil yang sangat jelas bahwa jumlah thread yang digunakan sangat mempengaruhi hasil pengujian kecepatan eksekusi data, hal itu bisa dilihat dari selisih perbedaan waktu yang dihabiskan untuk mengeksekusi data tersebut dan tentunya semakin banyak thread akan semakin cepat waktu eksekusinya selama jumlah thread yang digunakan sama dengan kamampuan maksimal sistem untuk menjalankan beberapa thread secara bersamaan, yaitu yang bergantung dengan jumlah thread yang ada di processor.

3.2 Pengujian dan Analisa Penggunaan Memori

Tujuan pengujian pada penggunaan memori adalah untuk mengetahui resource memory yang digunakan oleh Apache Mahout untuk memproses data didasarkan atas thread pertama sampai dengan thread ke 16. Hasil pengujian penggunaan memori masing-masing thread dengan 20 juta data di tampilkan pada Tabel 3.

Tabel 3 Hasil penggunaan memori 20 juta data

Jumlah Thread	Memori (MB)
1 Thread	1160
2 Thread	1170
4 Thread	1340
6 Thread	1230
8 Thread	1340
10 Thread	1360
12 Thread	1375
14 Thread	1360
16 Thread	1400

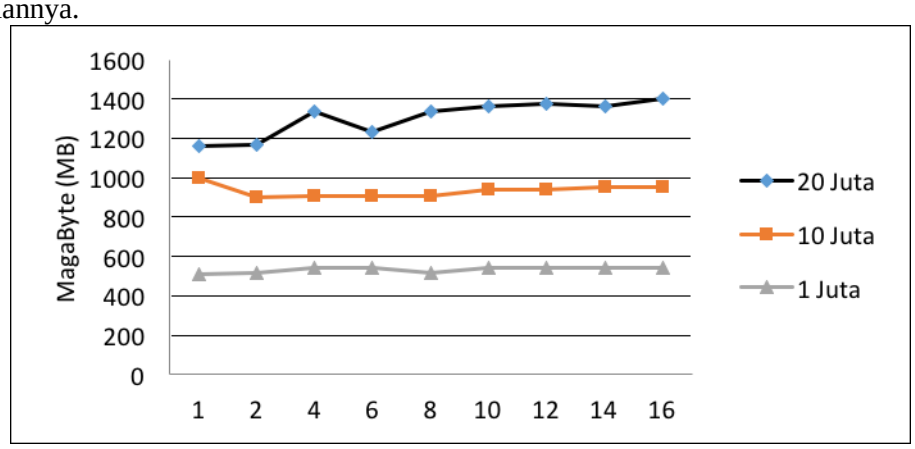
Dari Tabel 3 ditunjukan hasil dari pengujian penggunaan memori pada waktu program tersebut dijalankan, dari hasil pengujian bisa dilihat bahwa penggunaan multithreading tidak terlalu mempengaruhi penggunaan memori dalam sebuah komputer, hal itu dikarenakan hasil

dari penggunaan memori tidak terlalu membuat perubahan secara signifikan dalam beberapa pengujianya.

Tabel 4 Hasil penggunaan memori 10 juta data

Jumlah Thread	Memori (MB)
1 Thread	1000
2 Thread	900
4 Thread	910
6 Thread	910
8 Thread	910
10 Thread	940
12 Thread	940
14 Thread	950
16 Thread	950

Dari Tabel 4 bisa dilihat hasil dari pengujian penggunaan memori pada waktu program tersebut dijalankan, dari hasil pengujian bisa dilihat bahwa penggunaan multithreading tidak terlalu mempengaruhi penggunaan memori dalam sebuah komputer, hal itu dikarenakan hasil dari penggunaan memori tidak terlalu membuat perubahan secara signifikan dalam beberapa pengujiannya.



Gambar 5 Grafik Pengujian Penggunaan Memori

Gambar 5 dapat dilihat hasil dari pengujian penggunaan memori yang menggunakan beberapa jumlah dataset. Dari hasil pengujian tersebut bisa dilihat bahwa multithreading tidak terlalu berperan penting dalam penggunaan memori dalam program tersebut dijalankan, dikarenakan penggunaan memori yang digunakan dalam beberapa thread tidak mengalami perubahan yang terlalu signifikan, namun dalam pengujian yang memiliki perbedaan jumlah dataset yang akan dieksekusi, penggunaan memori mengalami penurunan disaat digunakanya data yang berukuran lebih kecil.

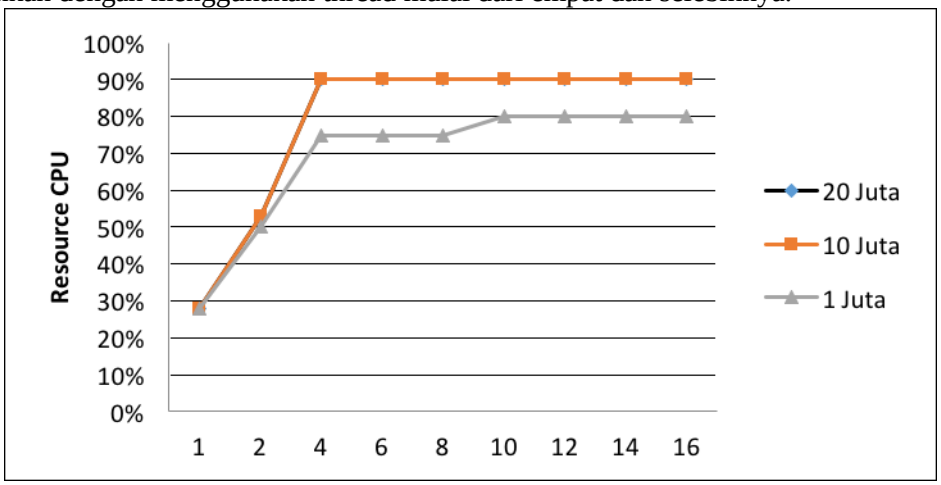
3.3 Pengujian dan Analisa Penggunaan Resource CPU

CPU merupakan salah satu komponen terpenting dari sistem komputer, tanpa CPU komputer tidak akan dapat bekerja dengan semestinya. Maka dari itu, pada penelitian ini akan dilakukan pengujian Apache Mahout untuk memproses data dengan jumlah data yang besar yaitu 20 juta data dengan jumlah thread yang bervariasi. Tabel 5 menunjukan hasil pengujian CPU dari thread ke-1 sampai dengan thread ke-16 dengan satu pengukuran CPU yaitu (%) persen.

Tabel 5 Hasil penggunaan CPU 20 juta data

Jumlah Thread	CPU
1 Thread	28%
2 Thread	53%
4 Thread	90%
6 Thread	90%
8 Thread	90%
10 Thread	90%
12 Thread	90%
14 Thread	90%
16 Thread	90%

Dari Tabel 5 dapat dilihat hasil dari pengujian penggunaan CPU pada waktu program tersebut dijalankan, dari hasil pengujian bisa dilihat bahwa penggunaan multithreading sangat mempengaruhi penggunaan CPU, hal ini dikarenakan multithreading memanfaatkan kemampuan thread yang ada pada CPU, maka dari itu semaking banyak thread yang digunakan maka semakin besar juga kerja yang dibebankan kepada CPU. Bisa dilihat dari data pengujian yang ada pada Tabel 5 terjadi peningkatan penggunaan CPU hingga 90% saat program dijalankan dengan menggunakan thread mulai dari empat dan selebihnya.



Gambar 6 Grafik Pengujian Perbandingan Resource CPU

Dari Gambar 6 dapat dilihat hasil dari pengujian CPU yang menggunakan beberapa jumlah dataset. Dari hasil pengujian tersebut bisa dilihat bahwa multithreading sangat berperan penting dalam penggunaan CPU. hal ini dikarenakan multithreading memanfaatkan kemampuan thread yang ada pada CPU, maka dari itu semakin banyak thread yang digunakan maka semakin besar juga kerja yang dibebankan kepada CPU. Perbadaan penggunaan CPU dalam eksekusi jumlah data 10 juta dan 20 juta relatif sama dikarenakan kedua jumlah data tersebut termasuk memiliki jumlah data yang besar, namun dalam penggunaan jumlah data sebesar 1 juta data penggunaan CPU tidak mengalami peningkatan kegunaan CPU yang sangat besar hal itu dikarenakan bahwa jumlah datanya yang terlalu kecil dan membuat multithreading tidak berjalan secara maksimal.

3.4 Perbandingan dengan Penelitian yang lain

Jika dibandingkan dengan penelitian sebelumnya yang telah dilakukan oleh Hongyi Su et al dengan judul "*Parallel Collaborative Filtering Recommendation Model Based on Expand-Vector*" didalam penelitian tersebut menggunakan dataset MovieLens yang dapatkan dari http://www.movielens.umn.edu dengan data 100 ribu, 1 juta dan 10 juta data[11]. Data tersebut diproses dengan menggunakan pemrograman CUDA dan menggunakan HOD (Hadoop on Demand) dengan membagi beberapa node hasilnya seperti di tunjukan pada Tabel 6.

Tabel 6 Perbandingan dengan Penelitian Sebelumnya[11]

Node /Thread	**RUNTIME**					
	Collaborative Filtering Apache Mahout			**CUDA HOD[11]**		
	1 Juta	**10 Juta**	**20 Juta**	**100 ribu**	**1 Juta**	**10 Juta**
1	44	713	2218	1007	1013	4523
2	27	417	1213	888	883	2879
3	20	282	764	826	836	2180

Tabel 6 menunjukan hasil perbandingan antara penelitian yang sudah dilakukan menggunakan Collaborative Filtering dengan Apache Mahout dengan Collaborative menggunakan Model CUDA dan HOD (*Hadoop on Demand*) didapatkan hasil didalam memproses data dengan satuan waktu (detik). Hasil yang didapatkan dari perbandingan tersebut untuk memproses data dengan 1 juta Apache Mahout membutuhkan waktu paling lama 44 detik untuk 1 thread dan akan selalu menurun berdasarkan banyaknya thread yang digunakan. Kemudian dengan menggunakan CUDA HOD membutuhkan waktu 1017 hampir 10 kali lipat untuk memproses data yang sama. Untuk memproses menggunakan Collborative Filtering menggunakan Apache Mahout dengan memproses data 20 juta hanya membutuhkan waktu 2218 dibandingkan dengan CUDA HOD yang hanya memproses 1 juta data membutuhkan waktu 4523. Hasil perbandingan disimpulkan bahwa pemrosesan multithreading Collaborative Filtering menggunakan Apache Mahout lebih cepat prosesnya daripada menggunakan pemrosesan parallel menggunakan model pemrograman CUDA dan HOD.

4. KESIMPULAN

Multithreading memang sangat membantu untuk mempercepat proses sebuah sistem yang berjalan karena pada multithreading mampu untuk menjalankan beberapa perintah dalam satu waktu. Multithreading juga juga dapat membantu untuk mempercepat proses sebuah sistem rekomendasi seperti halnya yang dilakukan oleh penelitian ini, hal ini terbukti karena dari beberapa percobaan penggunaan multithreading terbukti mampu untuk mempercepat proses datanya, berikut merupakan uraian dari hasil penelitian yang telah dilakukan :

a. Multithreading dapat diimplementasikan pada Apache Mahout yang digunakan untuk membuat sistem rekomendasi, dan dari penelitian ini juga telah membuktikan bahwa adanya multithreading mampu meningkat kinerja dalam pemrosesan waktu eksekusi datanya.
b. Dari pengujian lama waktu pemrosesan data yang telah dilakukan dengan jumlah data sebanyak 20 juta data, didapatkan hasil pengujian pada *single thread* dengan lama waktu pemrosesan datanya selama 2218 detik. Dan pada pengujian untuk 4 thread didapatkan hasil 764 detik, dan kemudian untuk 8 thread didapatkan hasil 691 detik dan pada pengujian untuk 16 thread didapatkan hasil 1097 detik.
c. beberapa pengujian yang telah dilakukan telah membuktikan bahwa multithreading mampu meningkatkan kinerja Apache Mahout dalam sistem rekomendasi, hal ini dibuktikan dengan adanya peningkatan kecepatan pemrosesan eksekusi datanya, dan multithreading akan berjalan secara maksimal jika digunakan sesuai dengan kemampuan

sistem komputer untuk menjalankan thread secara bersamaan, dan jumlah thread yang dibuat tidak melebihi dari kemapuan tersebut.

5. SARAN

Dari penelitian yang sudah dilakukan ini, masih banyak yang bisa dilanjutkan untuk penelitian yang selanjutnya, dalam preprocessing-nya atau bisa disebut dengan proses pemecahan datanya dapat menggunakan metode yang lain, bisa menggunakan metode klastering terlebih dahulu untuk memecah dataset dan kemudian dipecah menjadi beberapa thread, hal itu bertujuan agar setiap thread bisa memproses data yang mempunyai spesifikasi yang hampir sama, dan juga metode sistem rekomendasi gunakan bisa menggunakan pendekatan yang lain-nya dan atau metode lainya. Dan juga bisa menggunakan engine lain untuk memproses sistem rekomendasi ini.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Kepala Laboratorium Teknik Informatika yang telah memberikan ijin kepada penulis untuk melakukan riset tentang multithreading ini.

DAFTAR PUSTAKA

[1] J. Pardede, "Implementasi multithreading untuk Meningkatkan Kinerja Information Retrieval Dengan Metode GVSM," *JSISKOM*, vol. Vol. 4, No, 2014.

[2] E. A. Laksana, "Collaborative Filtering dan Aplikasinya," vol. 1, no. 1, pp. 36–40, 2014.

[3] U. Farooque, "Implementing User Based Collaborative Filtering to Build a Generic Product Recommender Using Apache Mahout," *INDIACom*, pp. 984–987, 2016.

[4] M. Hameed, "Collaborative Filtering Based Recommendation System: A survey.," *... J. Comput. ...*, vol. 4, no. 5, pp. 859–876, 2012.

[5] X. Daoping, Z. Alin, and L. Yubo, "A Parallel Clustering Algorithm Implementation Based on Apache Mahout," *2016 Sixth Int. Conf. Instrum. Meas. Comput. Commun. Control*, pp. 790–795, 2016.

[6] T. S. Kumar and S. Pandey, "Customization of recommendation system using collaborative filtering algorithm on cloud using mahout," *Adv. Intell. Syst. Comput.*, vol. 321, pp. 39–43, 2015.

[7] S. K. Mahapatra, S. K. Mohapatra, S. Mahapatra, and S. K. Tripathy, "A Proposed Multithreading Fuzzy C-Mean Algorithm for Detecting Underwater Fishes," *2016 2nd Int. Conf. Comput. Intell. Networks*, pp. 102–105, 2016.

[8] A. Kurniawan, "Sistem Rekomendasi Produk Sepatu Dengan Menggunakan," vol. 2016, no. Sentika, pp. 18–19, 2016.

[9] A. Jabakji and H. Dag, "Improving item-based recommendation accuracy with user's preferences on Apache Mahout," *Proc. - 2016 IEEE Int. Conf. Big Data, Big Data 2016*, no. 978, pp. 1742–1749, 2016.

[10] H. Mohanty , Soumendra; Jagadeesh, Madhu; Srivatsa, *Big Data Imperatives Enterprise Big Data Warehouse, BI Implementations and Analytics*. Aprees, 2013.

[11] Hongyi Su, Xianfei Lin, Caiqun Wang, Bo Yan, Hong Zheng; *Parallel Collaborative Filtering Recommendation Model Based on Expand-Vector*; International Conference on Intellegent Computing, pp 1-10