Jurnal Informatika, Vol. 13, No. 2, November 2015, 51-55 DOI: 10.9744/informatika.13.2.51-55
ISSN 1411-0105

LOAD BALANCING DINAMIS MENGGUNAKAN ALGORITMA *DYNAMIC DISTRIBUTION UPON DEMAND* PADA SISTEM ENKRIPSI/DEKRIPSI DATA TERDISTRIBUSI

Lauw Reduardy I. Laulyta[1], Mohammad Fajar[2], dan Syamsul Bahri[3]
[1,2,3] Program Studi Informatika, STMIK KHARISMA Makassar
Email: lawreduardy@kharisma.ac.id; fajar@kharisma.ac.id; syamsulbahri@kharisma.ac.id

Abstrak: Beragamnya kemampuan komputer pekerja pada sistem terdistribusi berdampak pada alokasi sumber daya dan waktu penyelesaian setiap pekerjaan di sistem. Studi ini bertujuan mengimplementasikan *load balancing* dinamis menggunakan algoritma *Dynamic Distribution upon Demand* (DDD) untuk optimalisasi alokasi sumber daya pada sistem enkripsi/dekripsi data terdistribusi. Evaluasi dilakukan dengan membandingkan sistem yang diusulkan dengan sistem tanpa *load balancing* dan sistem yang menggunakan *load balancing* statis. Hasil pengujian menunjukkan bahwa alokasi sumber daya pada sistem yang menggunakan algoritma DDD lebih optimal. Selain itu, pada proses dekripsi 3 dan 4 karakter menggunakan nilai inisialisasi beban yang sesuai memperlihatkan waktu penyelesaian pekerjaan yang lebih cepat dibandingkan dua sistem lainnya.

Kata kunci: Komputasi Paralel/Terdistribusi, *Enkripsi, Dekripsi, Dynamic Distribution upon Demand, Load Balancing* Dinamis, Alokasi Sumber Daya

Abstract: *The diversity of capability of computers worker in distributed system influences resource allocation and completion time for each jobs on the system. This study aims to implement a dynamic load balancing technique using dynamic distribution upon demand (DDD) algorithm to optimize the resource allocation in distributed data encryption/decryption system. Evaluating was performed by comparing the proposed system with two other models that is a system without load balancing and with static load balancing method. Evaluation result showed that the resource allocation in the proposed system using DDD algorithm more optimized and at the decryption process of three and four characters with proper workload initialization values presented that the job completion time much faster than the two other systems..*

Keywords*: Parallel/Distributed Computing, Encryption, Decryption, Dynamic Distribution upon Demand, Dynamic Load Balancing, Resource Allocation*

PENDAHULUAN

Sistem komputer paralel dan terdistribusi merupakan salah satu cara yang banyak digunakan untuk meningkatkan kekuatan pemrosesan pada sistem komputer. Pemakaian algoritma komputasi yang kompleks dan melibatkan jumlah data yang besar membuat proses komputasi membutuhkan sumber daya yang intensif. Penelitian Jhongsong Hoya dan Mohammad Fajar [4] tentang enkripsi/deksripsi data terdistribusi memperlihatkan bahwa untuk menyelesaikan proses dekripsi sebuah *file* dengan ukuran 8,29 *MB* menggunakan algoritma RSA 2048 bit dengan sebuah komputer pemroses membutuhkan waktu 58 menit, 46 detik, sedangkan untuk menyelesaikan proses enkripsi-nya membutuhkan waktu yang lebih cepat yaitu 1 menit 58 detik. Sebaliknya, jika pekerjaan yang sama tersebut dikerjakan menggunakan lima buah komputer pemroses dalam komputasi paralel dan terdistribusi membutuhkan waktu 9 menit 43 detik untuk proses dekripsi dan hanya 20 detik untuk proses enkripsinya. Penelitian tersebut menunjukkan keberhasilan pemanfaatan komputasi terdistribusi untuk meningkatkan kekuatan dan kecepatan pemrosesan data pada sistem enkripsi/dekripsi data. Akan tetapi, studi tersebut tidak mempertimbangkan kemampuan komputer pemroses ketika melakukan pembagian beban kerja dan alokasi sumber daya sistem. Setiap komputer pemroses atau pekerja diasumsikan memiliki kemampuan yang sama sehingga diberikan porsi beban kerja yang sama. Hal ini berdampak pada penggunaan sumber daya yang tidak optimal. Terlebih lagi pada sistem terdistribusi yang melibatkan komputer-komputer pemroses di jaringan yang lebih luas seperti internet, memiliki kemampuan dan spesifikasi yang beragam. Oleh karena itu, diperlukan sebuah mekanisme yang disebut dengan *load balancing* untuk mengatur pembagian beban kerja dan alokasi sumber daya, sehingga pemakaian sumber daya sistem dapat lebih optimal. Studi ini bertujuan untuk mengimplementasikan load balancing dinamis menggunakan algoritma

Dynamic Distribution upon Demand untuk mengatur pembagian beban kerja dan alokasi sumber daya pada sistem enkripsi/dekripsi data terdistribusi. Setiap komputer pemroses akan diberikan pekerjaan menurut kemampuannya masing-masing. Selain itu, proses distribusi beban pekerjaan dilakukan secara bertahap. Jika sebuah komputer pemroses telah menyelesaikan pekerjaannya maka komputer tersebut akan meminta pekerjaan berikutnya ke manajer atau server untuk dikerjakan, hingga semua pekerjaan selesai atau hasil pemrosesan (enkripsi/dekripsi) telah didapat.

TINJAUAN PUSTAKA

Komputasi Paralel/Terdistribusi

Komputasi paralel adalah penggunaan dua atau lebih prosesor (inti, komputer) dalam kombinasi untuk memecahkan satu masalah [6]. Beberapa keuntungan menggunakan komputasi paralel, yaitu [2]:

a. Lebih efisien waktu dan uang, dengan menggunakan komputasi paralel kita dapat melakukan sebuah pekerjaan dengan lebih cepat karena banyak prosesor yang ikut membantu menyelesaikan pekerjaan. Selain itu, komputasi paralel juga dapat dibangun dengan menggunakan komponen-komponen yang murah dan umum.
b. Menyelesaikan masalah besar yang tidak mungkin diselesaikan oleh satu komputer saja.
c. Dapat menyelesaikan lebih dari satu proses akibat penggunaan banyak prosesor.

Penggunaan komputasi paralel/terdistribusi dijadikan solusi untuk mempersingkat waktu yang dibutuhkan untuk penyelesaian pekerjaan, akan tetapi dalam eksekusi program mempunyai beberapa hambatan yaitu:

a. Hukum Amdahl, yaitu hukum mengenai komputasi paralel yang menyatakan bahwa peningkatan kecepatan yang mungkin dapat dikerjakan secara paralel dari suatu permasalahan yang diberikan dibatasi oleh proses yang hanya bisa dikerjakan secara serial sehingga komputasi paralel tidak akan pernah mencapai kesempurnaan karena performa akhir ditentukan oleh waktu yang diperlukan untuk menyelesaikan proses secara serial [3].
b. Hambatan yang disebabkan oleh karena beban jaringan dalam penggunaan prosesor yang terdapat pada mesin yang berbeda.
c. Hambatan yang terkait dengan beban waktu untuk inisialisasi pekerjaan, terminasi pekerjaan, dan sikronisasi.

Terdapat perbedaan antara komputasi paralel dan komputasi terdistribusi [5], yaitu:

a. Komputasi paralel menggunakan banyak prosesor sedangkan komputasi terdistribusi menggunakan banyak komputer yang terhubung dalam sebuah jaringan.
b. Secara geografis, komputasi paralel tidak bisa dilakukan pada jarak yang jauh sedangkan komputasi terdistribusi dapat dilakukan meski komputer berada pada jarak yang sangat jauh dan terhubung ke dalam jaringan.
c. Komputasi paralel menggunakan memori sistem yang digunakan bersama (*shared memory*) sedangkan komputasi terdistribusi menggunakan memori sistem yang berbeda tetapi terdistribusi.
d. Komputasi paralel membagi pekerjaan menjadi beberapa bagian pekerjaan yang memiliki komputasi yang sama sedangkan komputasi terdistribusi membagi pekerjaan menjadi beberapa bagian pekerjaan yang mungkin memiliki komputasi yang berbeda yang kemudian dikumpulkan hasilnya untuk menyelesaikan pekerjaan.

Load Balancing

Load Balancing adalah sebuah teknik yang diterapkan pada sistem paralel yang digunakan untuk mencapai kondisis sistem optimal, dimana pekerjaan dibagi ke banyak komputer secara merata, dan sebagai dampaknya akan mengurangi waktu eksekusi program. Load Balancing adalah membagi jumlah pekerjaan yang akan dilakukan dua atau lebih komputer sehingga lebih banyak pekerjaan yang dilakukan dalam jumlah waktu yang sama sehingga pengguna bisa dilayani lebih cepat. Load Balancing dapat diterapkan pada perangkat keras, perangkat lunak, atau kombinasi antara keduanya [7]. Tujuan dari load balancing yaitu untuk mengoptimalkan penggunaan sumber daya, memaksimalkan throughput, meminimalisir waktu respon, dan menghindari overload pada satu komputer pekerja. Salah satu pendekatan yang digunakan pada load balancing yaitu *Process Farm*. *Process Farm* merupakan pendekatan yang sangat umum dijumpai, dimana sebuah master akan melakukan beberapa bagian dari langkah-langkah paralel program dan membuat sejumlah prosesor pekerja untuk melakukan bagian program paralel. Ketika pekerja telah selesai menyelesaikan pekerjaannya, pekerja tersebut akan memberitahu kepada master kemudian diberikan beban pekerjaan baru [8].

Dynamic Distribution upon Demand

Dynamic Distribution upon Demand dimana *Li* merupakan persentase dari total pekerjaan *Wt* akan dibagikan ke arsitektur prosesor *B* pada saat aplikasi dijalankan, sesuai dengan prediksi fungsi *F(Pi ,Wi)*

dan kemudian setiap prosesor akan meminta pekerjaan lebih ketika pekerjaannya telah selesai. Adapun langkah-langkah dari DDD adalah [1]:

1. Pekerjaan dibagikan oleh prosesor utama (dalam penelitian ini merupakan *server*):
   - *Server* akan menghitung jumlah pekerjaan (C_i) yang akan dibagikan pada mulanya
     $$C_i = \frac{W_t \times L_i}{100}$$
   - *Server* akan menghitung jumlah pekerjaan awal (cc_i) berdasarkan kemampuan prosesor P_i
     $$PowTotal = \sum_{i=1}^{B} P_i$$
     $$cc_i = \frac{C_i \times P_i}{PowTotal}$$
   - *Server* akan membagi pekerjaan awal (cc_i) kepada komputer pekerja m_i, dimana $i = 1..B$.
2. Pekerjaan yang tersisa (C_r) akan dibagikan oleh *server*, dimana $C_r = C - Ci$.
   - Komputer pekerja (m_i) akan meminta pekerjaan untuk dikerjakan ketika pekerjaan telah diselesaikan untuk setiap $i = 1..B$.
   - Server akan mengirimkan pekerjaan baru kepada pekerja yang meminta pekerjaan.

HASIL PENGUJIAN SISTEM

Evaluasi dalam penelitian ini dilakukan terhadap tiga tipe sistem enkripsi/dekripsi data terdistribusi, yaitu: 1) sistem yang menggunakan algoritma DDD untuk load balancing dinamis, 2) sistem dengan load balancing statis, dan 3) sistem tanpa mekanisme load balancing. Untuk algoritma deskripsinya, penulis menggunakan *Brute Force*. Sistem dikembangkan menggunakan bahasa pemrograman Java. Melibatkan 3 komputer pemroses (worker) dengan spesifikasi kemampuan yang beragam.

Untuk proses dekripsi, pengujian dilakukan menggunakan nilai inisialisasi beban 10%, 25%, 50%, 75%, dan 100%. Untuk pemrosesan beban kerja yang kecil, yaitu proses dekripsi 1 dan 2 karakter dapat diselesaikan dalam waktu yang cepat (Detik). Pada proses pencarian 2 karakter, pemakaian *load balancing* dinamis dengan nilai inisialisasi 10% dan 25% membutuhkan waktu 0:02 Detik dibanding skenario lainnya (Lihat Tabel 1). Hal ini disebabkan waktu yang diperlukan untuk komunikasi antara *server* dan komputer pekerja (*worker)*.

Sedangkan untuk pemrosesan (dekripsi) 3 karakter, memperlihatkan bahwa penggunaan algoritma load balancing memperlambat waktu penyelesaian proses dibandingkan dengan menggunakan load balancing statis dan yang tidak menggunakan mekanisme load balancing. Penyelesaian proses tercepat yaitu dengan menggunakan load balancing statis yaitu 11 detik (Lihat Tabel 2). Waktu penyelesaian proses menggunakan load balancing dinamis lebih lambat disebabkan adanya waktu yang diperlukan untuk komunikasi antara server dan worker, sementara bila tidak menggunakan mekanisme load balancing atau dengan load balancing statis, worker dan server hanya berkomunikasi ketika server memberikan pekerjaan kepada worker dan ketika worker telah mendapatkan hasil atau menyelesaikan pekerjaannya.

Tabel 1. Hasil Pengujian 1 dan 2 Karakter

Inisialisasi	Waktu Penyelesaian (Menit:Detik)	
	Pencarian 1 Karakter	Pencarian 2 Karakter
No LB	0:01	0:01
10%	0:01	0:02
25%	0:01	0:02
50%	0:01	0:01
75%	0:01	0:01
90%	0:01	0:01
LB Statis	0:01	0:01

Tabel 2. Hasil Pengujian 3 Karakter

Inisialisasi	Pekerjaan Awal		Pekerjaan Tambahan (Permintaan dari *Worker*)		Pekerjaan yang Tidak Dibagi	Waktu yang Dibutuhkan Hingga Hasil Ditemukan (Menit: Detik)
	Worker 1	*Worker* 2	*Worker* 1	*Worker* 2		
Tanpa LB	33.33%	33.33%	0%	0%	0.00%	0:12
10%	5.14%	4.86%	41%	46%	2.73%	0:14
25%	12.84%	12.16%	32%	41%	2.28%	0:14
50%	25.68%	24.32%	16%	30%	3.12%	0:13
75%	38.51%	36.49%	7%	15%	3.21%	0:14
90%	46.22%	43.78%	0%	7%	2.64%	0:13
LB Statis	51.35%	48.65%	0%	0%	0.00%	0:11

Akan tetapi hal ini berbeda dari aspek alokasi sumber daya sistem. Seperti yang ditunjukkan pada gambar 2 bahwa dalam pemakaian *load balancing*, beban kerja lebih besar diberikan ke *worker* 1 dibandingkan *worker* 2. Hal ini disebabkan spesifikasi (kemampuan prosesor) *worker* 1 diidentifikasi oleh sistem lebih tinggi dibanding kemampuan prosesor *worker* 2. Selain itu, manajer tidak perlu mendistribusikan semua pekerjaan ke komputer *worker* untuk diselesaikan. Dari evaluasi menggunakan *load balancing* dinamis dengan nilai inisialisasi 75%, memperlihatkan bahwa 3,21% dari total pekerjaan yang dimuat tidak didistribusikan lagi ke *worker* untuk dikerjakan karena hasil dekripsi sudah didapatkan.

Untuk pemrosesan 4 karakter, evaluasi menunjukkan bahwa penggunaaan algoritma *load balancing* dinamis dengan nilai inisialisasi 75%, membuat proses lebih cepat diselesaikan dibandingkan meng-

gunakan skenario inisialisasi lainnya, atau menggunakan *load balancing* statis, dan tanpa menggunakan *load balancing* dengan waktu 6 menit 56 detik (Lihat Tabel 3), sedangkan penyelesaian proses terlama yaitu tanpa menggunakan *load balancing* (No LB) membutuhkan waktu 15 menit 1 detik. Selain itu, penulis mendapati bahwa menggunakan *load balancing* statis, proses diselesaikan dalam waktu 7 menit 39 detik. Dalam hal optimalisasi dan alokasi sumber daya, seperti terlihat pada gambar 3, bahwa dengan menggunakan *load balancing* dinamis, *worker* 1 yang memiliki kemampuan pemroses (prosesor) yang lebih tinggi mendapatkan pekerjaan yang lebih besar dibandingkan *worker* lainnya. Selain penyelesaian proses yang lebih cepat menggunakan *load balancing* dinamis dengan nilai inisialisasi 75% pada pencarian 4 karakter, penggunaan sumber daya pun lebih optimal yaitu 13,33% dari total pekerjaan yang tidak perlu didistribusikan ke *worker* untuk dikerjakan, karena hasil dekripsi juga telah diperoleh.

Gambar 1. Grafik Hasil Pengujian 3 Karakter

Tabel 3. Hasil Pengujian 4 Karakter.

Inisialisasi	Pekerjaan Awal			Pekerjaan Tambahan (Permintaan dari *Worker*)			Pekerjaan yang Tidak Dibagi	Waktu yang Dibutuhkan Hingga Hasil Ditemukan
	Worker 1	*Worker* 2	*Worker* 3	*Worker* 1	*Worker* 2	*Worker* 3		
No LB	33.33%	33.33%	33.33%	0%	0%	0%	0.00%	15:01
10%	5.04%	2.66%	2.31%	43%	31%	10%	7.08%	8:15
25%	12.59%	6.64%	5.77%	36%	26%	6%	7.67%	7:41
50%	25.18%	13.28%	11.54%	18%	22%	0%	9.83%	7:46
75%	37.77%	19.92%	17.31%	7%	5%	0%	13.33%	6:56
90%	45.32%	23.91%	20.77%	0%	7%	0%	2.82%	13:33
LB Statis	50.36%	26.57%	23.08%	0%	0%	0%	0.00%	7:39

Gambar 2. Grafik Hasil Pengujian 4 Karakter

KESIMPULAN DAN SARAN

Berdasarkan studi yang telah dilakukan, dapat disimpulkan bahwa:

1. Untuk optimalisasi dan alokasi sumber daya pada sistem enkripsi/dekripsi data terdistribusi menggunakan *Dynamic Distribution upon Demand* (DDD) sebagai algoritma load balancing dinamis. Inisialisasi awal pembagian beban berperan penting dalam proses pembagian pekerjaan dan mempengaruhi penggunaan waktu untuk mendapatkan hasil yang ingin dicapai. Beberapa nilai inisialisasi yang digunakan dalam pekerjaan ini yaitu 10%, 25%, 50%, 75%, dan 90% untuk dibandingkan dengan pembagian pekerjaan tanpa menggunakan *load balancing* dan sistem yang menggunakan *load balancing* statis.
2. Hasil evaluasi menunjukkan bahwa untuk penyelesaian proses menggunakan algoritma *Dynamic Distribution upon Demand* pada *load balancing* dinamis untuk kasus yang kecil (dekripsi 1 dan 2 karakter) memerlukan waktu yang lebih lama dibandingkan dengan menggunakan *load balanceing* statis dan tanpa menggunakan *load balancing*, meski terdapat pekerjaan yang tidak dibagikan kepada *worker* dan tetap mendapatkan hasil yang sama seperti pada pencarian 3 karakter, menggunakan *load balancing* statis hanya membutuhkan waktu 11 detik dibandingkan tanpa menggunakan *load balancing* yang membutuhkan waktu 12 detik dan menggunakan *load balancing* dinamis yang menggunakan waktu 13 hingga 14 detik meski pada inisialisasi 75% terdapat 3,21% dari total pekerjaan yang tidak didistribusikan. Hal ini disebabkan adanya waktu komunikasi antara *server* dan *worker*. Akan tetapi untuk pengujian lain, *load balancing* akan mengurangi waktu pemrosesan dan alokasi sumber daya lebih optimal, seperti pada pencarian 4 karakter menggunakan *load balancing* statis membutuhkan waktu 7 menit 39 detik, tanpa menggunakan *load balanceing* menggunakan waktu 15 menit 1 detik dibandingkan dengan menggunakan algoritma *load balancing* dinamis DDD dengan nilai inisialisasi 75% yang hanya membutuhkan waktu 6 menit 56 detik dan terdapat 13,33% dari total pekerjaan tidak didistribusikan ke *worker* untuk menyelesaikan proses yang sama.

Untuk pekerjaan selanjutnya, perlu dipertimbangkan penggunaan pendekatan selain *process farm* dalam implementasi load balancing, seperti *Gradient Model* dan *Phase Parallel*. Hal ini diprediksi dapat mempengaruhi teknik pembagian pekerjaan, lama waktu penyelesaian pekerjaan dan alokasi penggunaan sumber daya sistem.

DAFTAR PUSTAKA

[1] Armando E. De Giusti, Marcelo R. Naiouf, Laura C. De Giusti & Franco Chichizola, *Dynamic Load Balancing in Parallel Processing on Non-Homogeneous Clusters*. Instituto de Investigación en Informática LIDI (III-LIDI), Facultad de Informática–UNLP, JCS&T, 5(4), 2005.

[2] Blaise Barney, *Introduction to Parallel Computing*, Lawrence Livermore National Laboratory. Available on https://computing.llnl.gov/tutorials/parallel_comp/

[3] Gene M. Amdahl, *Validity of the single processsor approach to achieving large scale computing capabilities*, AFIPS spring joint computer conference, IBM Sunnyvale, California, 1967.

[4] Jhongsong Hoya, Mohammad Fajar, Reduksi Waktu Enkripsi/Dekripsi pada Algoritma RSA menggunakan Komputasi Terdistribusi, Prosiding Konferensi Nasional Sistem Informasi (KNSI 2010), STMIK MDP Palembang, 2010.

[5] N. Taing, *Parallel and Distributed Computing*. Retrieved May 14, 2014. from http://lycog.com/distributed-systems/parallel-and-distributed-computing/

[6] Quentin F. Stout, *What is Parallel Computing? A Not Too Serious Explanation*, Computer Science and Engineering. University of Michigan. Ann Arbor, MI 48109-2121. Retrieved May 11, 2014. from http://web.eecs.umich.edu/~qstout/parallel.html

[7] Sherihan Abu Elenin & Masato Kitakami, *Performance Analysis of Static Load Balancing in Grid*, International Journal of Electrical & Computer Sciences IJECS-IJENS, 11(3), 2011.

[8] Zubair Khan, Ravendra Singh, Jahangir Alam and Shailesh Saxen, *Classification of Load Balancing Conditions for parallel and distributed systems*. IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No. 1, September 2011.