Peningkatan Kinerja Sistem Multi Agen dengan Optimalisasi Alokasi Beban (Studi Kasus Enkripsi Data dengan Algoritma AES)

Muhammad Rizka[1], Waskitho Wibisono[2], Tohari Ahmad[3]
Fakultas Teknologi Informasi,
Jurusan Teknik Informatika,
Institute Teknologi Sepuluh Nopember
e-mail: muhammad.rizka910@gmail.com

Abstrak

Sistem multi agen merupakan sekumpulan agen yang saling berinteraksi dan berkomunikasi untuk melaksanakan suatu tujuan tertentu. Dalam mewujudkan sistem multi agen yang skalabel dan efisien maka agen-agen dalam sistem multi agen harus dapat bersifat autonomous, proactive dan flexible terhadap lingkungan dalam keadaan tertentu. Dalam menangani job yang didistribusikan oleh sistem ke setiap agen dapat saja terjadi ketidakseimbangan workload diantara agen-agen. Load balancing merupakan salah satu solusi ketika ketidakseimbangan workload terjadi dalam sistem multi agen. Sistem multi agen yang penulis usulkan yaitu menerapkan load balancing dalam pengalokasikan workload ke setiap agen secara dinamis. Sistem multi agen yang dibangun terdiri dari agent worker yang bertugas dalam melakukan eksekusi job dan agent monitor yang bertanggung jawab dalam mengawasi kondisi agent worker dan mengalokasikan job kesetiap agent worker. Load balancing system dilakukan dengan pertimbangan tiga parameter yaitu kondisi load agent worker, antrian job dan resource komputasi komputer dimana agen-agen tersebut berada. Studi kasus yang diterapkan pada penelitian ini adalah enkripsi data dengan algoritma AES (Advanced Encryption Standard). Hasil pengujian menunjukkan bahwa metode usulan dapat meningkatkan kinerja sistem multi agen dalam melakukan proses enkripsi job hingga mencapai 30,99 % dibandingkan dengan Distribusi Uniform (DU).

Kata kunci: *Sistem Multi Agen, Alokasi Beban, Komunikasi Agen, Enkripsi Data, AES (Advanced Encryption Standard)*

Abstract

Multi-agent system is a set of agents that interact and communicate with each other to accomplish a particular purpose. In a multi-agent system mewujutkan scalable and efficient the agents in multi-agent systems must be able to be autonomous, proactive and flexible to the environment in certain circumstances. In dealing with jobs that are distributed by the system to each agent may be an imbalance of workload among agents. Load balancing is one of the solutions when the workload imbalance occurs in multi-agent systems. Multi-agent system that the authors propose that implement load balancing in the workload allocation to each agent dynamically. Multi-agent system that is built consisting of worker agent in charge of doing the job execution and monitoring agent are responsible for monitoring the condition of workers and allocate job agent kesetiap worker agent. Load balancing system is done with consideration of three parameters, namely the condition of load agent worker, job queuing and computer computing resource where agents are located. The case studies were applied in this research is the data encryption algorithm AES (Advanced Encryption Standard). The results show that the proposed method can improve the performance of multi-agent system in the process of encryption jobs up to 30.99% compared with the Uniform Distribution (DU).

Keywords: *Multi Agent System, Weight Allocation, Agent of Communication, Data Encrypt, AES (Advanced Encryption Standard)*

1. Pendahuluan

Perkembangan teknologi sistem komputasi *ubiquitous* yang terus meningkat pesat membutuhkan sebuah kolaborasi efisiensi yang adaptif terhadap suatu aplikasi yang berjalan pada jaringan *homogenous* maupun *heterogenous*. Sebuah sistem efisiensi yang dapat menyediakan kostumisasi terhadap suatu perubahan lingkungan. Sistem multi agen merupakan sebuah teknologi yang didesain untuk memenuhi kebutuhan tersebut. Sistem multi agen merupakan sebuah paradigma dalam hal membangun suatu sistem dengan kompleksitas tinggiyang berbasis *distributed*, *knowledge*, *computing* dan *adaptif*. Sistem multi agen terdiri dari sekumpulan *intelligent agent* dan *resource* yang saling berinteraksi untuk mencapai suatu tujuan tertentu. Agen merupakan entitas *autonomous* yang dapat bertindak *proactive* dan *flexible* terhadap suatu lingkungan dalam keadaan tertentu [1].

Dalam pendistribusian job dapat saja terjadi ketidakseimbangan *workload* diantara agen. Dalam menangani masalah ketidak seimbangan sistem ada dua penelitian yang terkait yaitu yang dilakukan oleh Shin [2]. Metode *load balancing* yang diusulkan pada saat agen mengalami *overload* sehingga agen tersebut harus dipindahkan ke komputer lain untuk mengurangi *workload*. Dengan hanya melakukan migrasi agen dari suatu komputer ke komputer lain masih memungkinkan terjadinya *overload* karena adanya penambahan agen dan *task* pada komputer tujuan sehingga dapat mengakibatkan terjadi proses *reload balancing* pada sistem yang pada akhirnya akan membuat sistem mengalami *overload* dan menjadi lambat. Penelitian lain mengenai *load balancing* yaitu yang dilakukan oleh Lee [3] yaitu mengalokasikan *resource* komputasi berdasarkan kebutuhan komputasi agen. Metode *load balancing* yang diusulkan membutuhkan waktu *presprocessing* yang lama karena harus mengestimasi terlebih dahulu waktu penyelesaian job.

Metode yang penulis usulkan yaitu sebuah skema *load balancing* dimana sistem melakukan alokasi *job* keseluruh *agent worker* pada setiap komputer secara dinamis. Sistem multi agen terdiri dari *agent monitor* dan *agent worker*. *Agent worker* bertugas sebagai pekerja yang melakukan proses eksekusi *job*. *Agent monitor* bertanggung jawab dalam mengawasi kondisi *load agent worker*. *Load balancing* pengalokasian *job* ditentukan berdasarkan pertimbangan kondisi *load agent worker*, antrian *job* dan *resource* komputasi komputer dimana *agent worker* berada. Dalam penelitian ini sistem multi agen diaplikasikan pada studi kasus enkripsi data dengan menggunakan algoritma AES (*Advanced Encryption Standard*).

2. Metodologi Penelitian

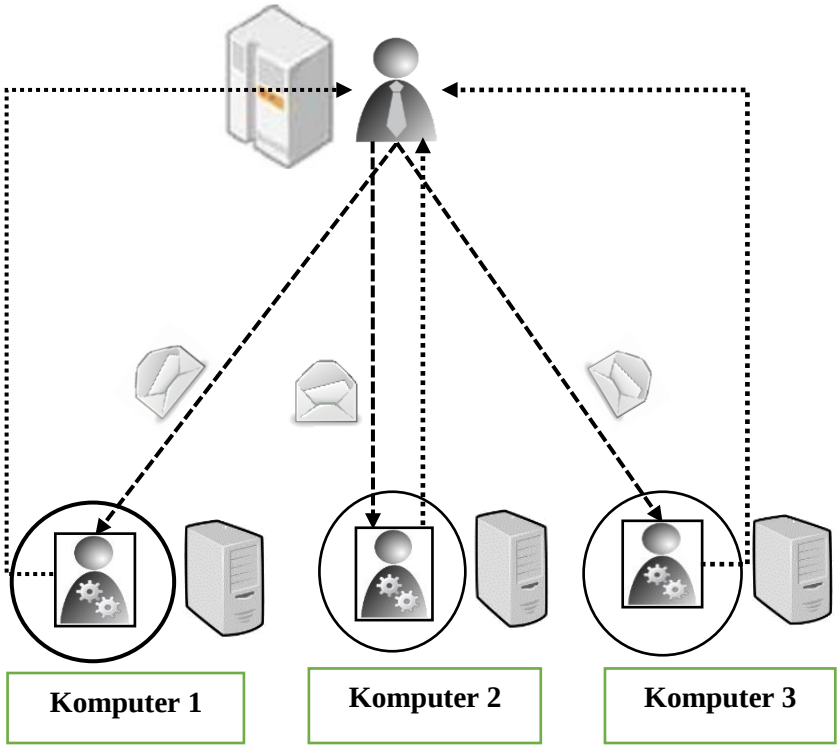
Pada penelitian ini diusulkan sebuah mekanisme *load balancing* dengan mempertimbangkan kondisi *load* agen, antrian *job* dan daya komputasi komputer. Dalam penerapannya ada dua jenis agen yang diimplementasikan yaitu *agent monitor* dan *agent worker*. *Agent worker* yang bertugas dalam melakukan proses enkripsi data sedangkan *agent monitor* bertanggung jawab dalam memonitoring *agent worker* pada setiap komputer dan selanjutnya mengalokasikan sejumlah *job* ke *agent worker*. *Agent monitor* akan melakukan pengecekan secara periodik mengenai kondisi *agent worker* yang sedang melakukan eksekusi *job* selanjutnya kondisi tersebut akan dipertimbankan pada proses *load control* untuk menentukan alokasi *job* untuk setiap *agent worker*.

2.1 Kondisi *Agent Worker*

Dalam menperkirakan kondisi *agent worker*, *agent monitor* akan mengirimkan sebuah pesan ACL secara periodik ke setiap *agent worker* yang berada pada setiap komputer. *Agent monitor* akan segera mencatat waktu *forwarding* F(t) pesan dari *agent monitor* ke *agent worker* dan waktu *receiving* R(t) yaitu pesan balasan dari *agent worker*. *Agent monitor* akan mengkalkulasi nilai *round trip time* (RTT) pesan berdasarkan persamaan 1 berikut ini:

$$RTT = \textit{receiving time}\ R(t) - \textit{forwarding time}\ F(t) \quad (1)$$

Nilai *Round Trip Time* (RTT) dari setiap *agent worker* digunakan untuk menentukan kondisi *agent worker*. Nilai RTT yang didapatkan oleh *agent monitor* akan mendeskripsikan kondisi *load agent worker* yang juga merupakan kondisi penggunaaan daya komputasi oleh *agent worker* saat melakukan proses enkripsi *job*. Proses Deteksi Kondisi *Load Agent Worker* dapat dilihat pada Gambar 1.



Gambar 1. Proses Deteksi Kondisi *Load Agent Worker*

2.2 Antrian *Job*

Dalam menperkirakan antrian *job* pada *agent worker*, *Agent monitor* akan mengirimkan sebuah pesan ACL (*Agent Communication Language*) secara periodik ke setiap *agent worker* yang berada pada setiap komputer. Pesan ACL akan dikirimkan ke seluruh *agent worker* dalam rentang waktu lima detik. *Agent worker* akan mencatat secara *real time* antrian *job* yang sedang terjadi.

Pada saat *agent worker* menerima pesan dari *agent monitor* maka jumlah antrian *job* yang sedang terjadi pada saat itu akan dimasukkan kedalam pesan ACL dan selanjutnya pesan tersebut dikirimkan *agent monitor*. *Agent monitor* akan menerima pesan ACL dari setiap *agent worker* yang berisi jumlah antrian *job* yang sedang terjadi pada masing-masing *agent worker*. Dalam penelitian ini antrian *job* yang terjadi pada *agent worker* diklasifikasikan kedalam tiga kelompok yaitu: rendah, sedang, dan padat.

2.3 Daya Komputasi Komputer

Dalam sebuah sistem multi agen dapat terdiri dari beberapa *agent worker* yang melakukan proses eksekusi *job*. Mekanisme alokasi jumlah *job* untuk setiap *agent worker* yaitu berdasarkan daya komputasi pada suatu komputer dimana *agent worker* berada.

Parameter pertimbangan pengalokasian *job* berdasarkan daya komputasi komputer dengan cara menentukan tingkat komputasi komputer berdasarkan nilai *million instruction per second* (MIPS). Tingkat komputasi setiap komputer dimana *agent worker* berada dapat ditunjukkan pada Tabel 1 berikut:

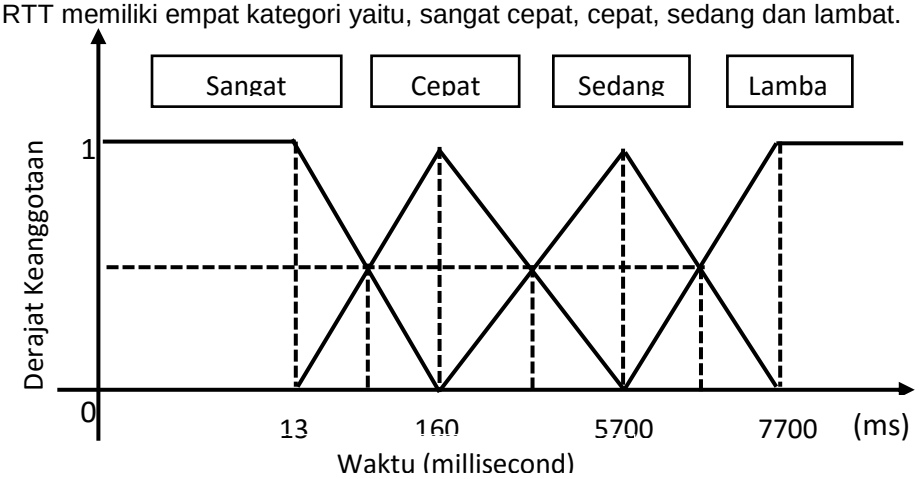
Tabel 1. Tingkat Komputasi Komputer Dimana *Agent Worker* Berada

Agent	Agent 3 (komputer 3)	Agent 2 (komputer 2)	Agent 1 (komputer 1)
MIPS	569	685	2801
Tingkat Komputasi	Lambat	Sedang	Cepat

2.4 Metode *Fuzzy Logic*.

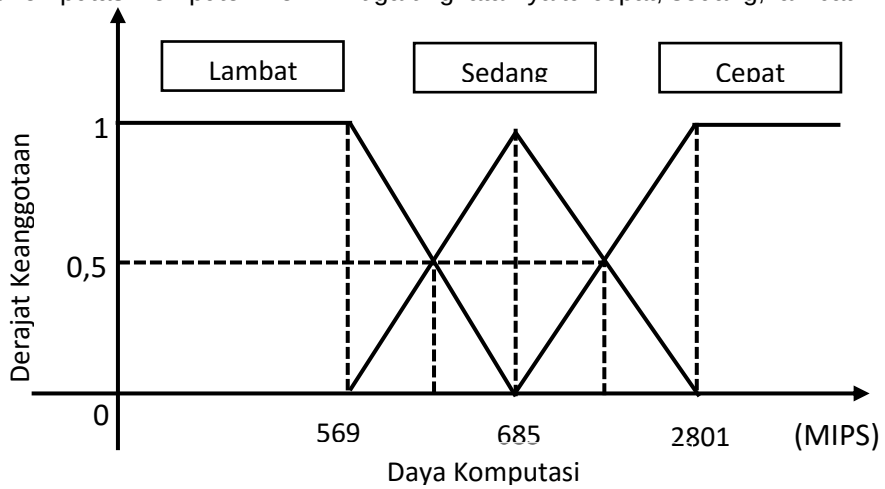
Metode *fuzzy logic* yang digunakan dalam penelitian ini adalah *fuzzy logic* model Sugeno. Dalam penelitian ini ada tiga parameter yang dijadikan inputan *fuzzy logic,* yaitu.

a. Nilai RTT memiliki empat kategori yaitu, sangat cepat, cepat, sedang dan lambat.



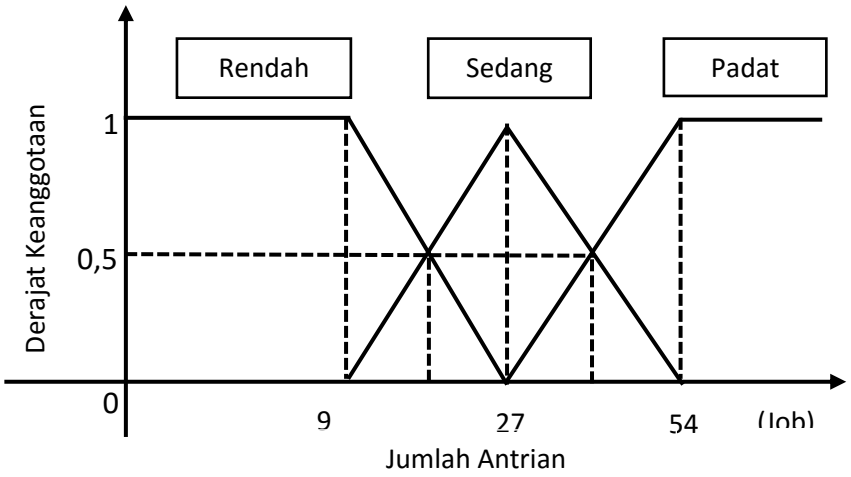
Gambar 2. Grafik *Membership* Nilai RTT

b. Daya komputasi komputer memiliki tiga tingkatan yaitu cepat, sedang, lambat.



Gambar 3. Grafik *Membership* Daya Komputasi

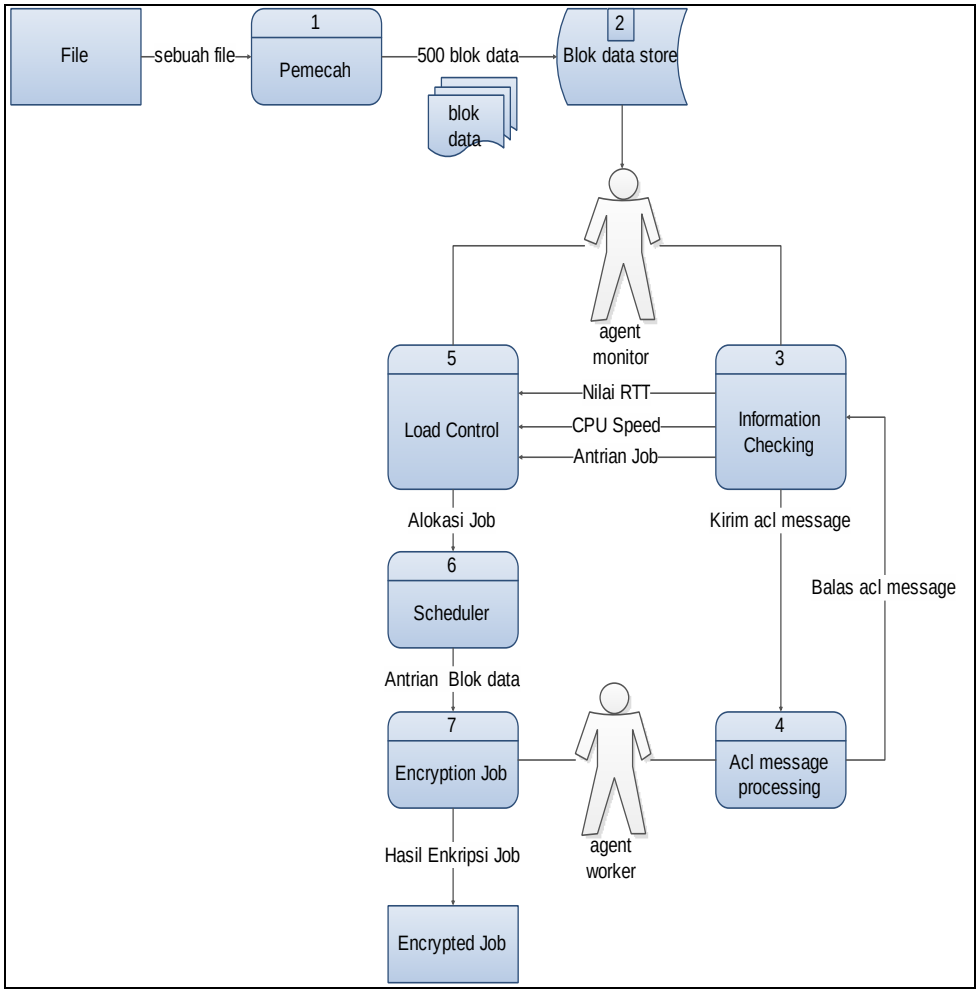
c. Antrian *Job* memiliki tiga kategori yaitu rendah, sedang dan padat.



Gambar 4. Grafik *Membership* Antrian *Job*

2.6 Skema Usulan

Dalam membangun sistem multi agen yang seimbang (*balance*) peneliti mengusulkan *load balancing* terhadap *workload agent worker* dalam sistem multi agen. Skema usulan sistem multi agen dapat dilihat pada Gambar 5 berikut ini.



Gambar 5. Alur Sistem

Sistem *load balancing* yang diusulkan terdiri dari tujuh proses utama yaitu proses pemecahan sebuah *file* menjadi sekumpulan blok data, proses blok data *store*, proses *information checking*, proses acl *message*, proses *load control*, proses *scheduler*, dan proses enkripsi blok data.

3. Sistem Multi Agen

Sistem multi agen merupakan teknologi yang telah banyak diterapkan diberbagai bidang diantaranya komputasi *ubiquitous*, *distributed simulation*, sistem komunikasi *mobile*, *game* dan masih banyak lagi. Pada umumnya bidang tersebut memiliki karakteristik yang kompleks dan memiliki *behaviour* yang tidak dapat diprediksi sehingga agen-agen dalam sistem didesain untuk saling berinteraksi dan juga berkolaborasi dalam menyelesaikan suatu *task* yang diberikan.

Distributed simulation yang dibangun berdasarkan sistem multi agen dapat menghasilkan sebuah sistem yang sangat efektif karena sistem multi agen dapat memenuhi kebutuhan terkait skalabilitas dan otonom. Sejumlah peneliti telah membuktikan bahwa sistem multi agen dapat menyediakan suatu model yang dinamis terhadap penelitian dibidang biologi, sosial, sistem ekonomi, logistik militer dan lain-lain. Penelitian tersebut membutuhkan desain model yang spesifik dan sering kali tidak dapat ditangani dengan metode simulasi tradisional.

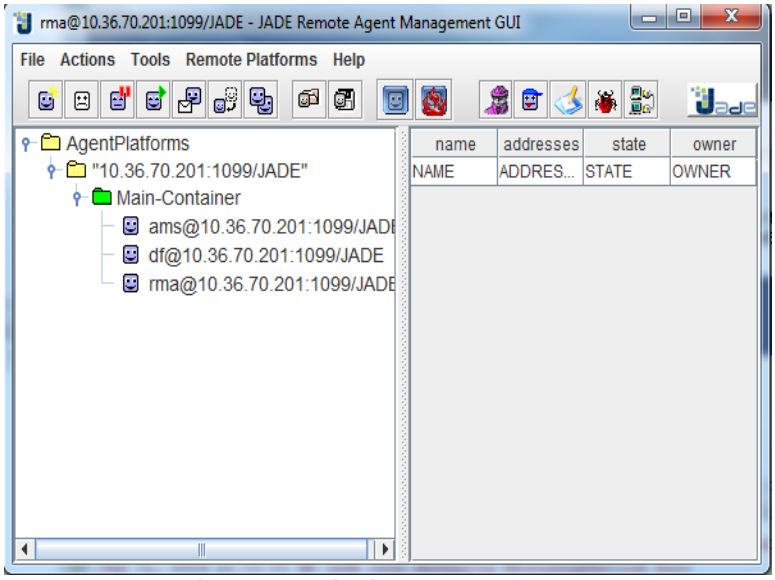
3.1 *Load Balancing*

Load balancing merupakan sebuah mekanisme untuk menyeimbangkan *load* diantara agen. *Load balancing* diperlukan ketika hanya beberapa agen dalam suatu komputer mengalami *overloaded* sedangkan agen lainya dalam keadaan *idle* atau sekumpulan agen hanya terkonsentrasi dalam suatu komputer sehingga komputer tersebut mengalami *overload*. *Load balancing* sangat penting dalam pendistribusian *job* diantara agen dalam sistem. Dalam penerapannya *load balancing* dibedakan kedalam dua kelompok yaitu *static load balancing* dan *dynamic load balancing*. Dalam *static load balancing* keputusan migrasi agen dibuat secara statis atau probabilistik dengan pertimbangan status sistem [4]. *Static load balancing* sangat efektif dan sederhana ketika suatu *load* bersifat statis akan tetapi ketika *load* bersifat fluktuatif maka sistem akan mengalami ketidakefisienan dan *under* utilisasi. *Dynamic load balancing* merupakan sebuah metode dalam menyeimbangkan *load* diantara agen sesegera mungkin ketika kondisi *load* berubah.

3.2 *Agent Platform*

Agent Platform merupakan teknologi arsitektur yang menyediakan *environment* bagi agen dalam melaksanakan operasi untuk menyelesaikan suatu *task* tertentu. Sistem multi agen diterapkan di atas *Agent Platform* untuk mewujudkan sistem yang aman, efisien dan *autonomous* [5]. Dalam membangun sebuah sistem multi agen ada beberapa *Agent Platform* yang tersedia diantaranya adalah JADE.

JADE merupakan suatu *Agent Platform* yang menyediakan fasilitas *middleware* untuk pengembangan sistem multi agen [6]. JADE dibangun berdasarkan pemrograman java. JADE memiliki tampilan grafis yang memudahkan dalam proses administrasi dan monitoring agen-agen seperti yang terlihat pada Gambar 6 berikut ini:



Gambar 6. *Grafical User Interface* JADE

JADE menyediakan fungsi yang siap untuk digunakan dan mudah untuk dikostumisasi. *Agent Platform* JADE menyediakan beberapa desain dasar [7] dalam membangun sistem multi agen, yaitu.

a. JADE mendukung penuh jaringan tersebar. JADE dapat dijalankan pada jaringan komputer.
b. JADE mendukung penuh standarisasi FIPA-*compliant* dalam melakukan interaksi diantara agen.
c. JADE mendukung mobilitas agen
d. Mengimplementasikan *white pages* and *yellow pages* dalam menyediakan layanan agen.
e. Manajemen agen yang *simple* dan efektif serta tampilan berbasis grafis.

3.3 Metode *Fuzzy Logic*

Metode *fuzzy logic* merupakan metode yang mempunyai kemampuan untuk memproses variabel yang bersifat kabur atau yang tidak dapat dideskripsikan secara eksak/pasti seperti misalnya tinggi, lambat, bising, dan lain-lain. Dalam *fuzzy logic* variabel yang bersifat kabur tersebut direpresentasikan sebagai sebuah himpunan yang anggotanya adalah suatu nilai *crisp* dan derajat keanggotaannya (*membership function*) dalam himpunan tersebut [8]. Model *fuzzy logic* yang digunakan dalam penelitian ini adalah *fuzzy logic* dengan model Sugeno.

Fuzzy logic dengan model Sugeno pertama kali dikemukakan Michio Sugeno pada tahun 1985. Model Sugeno termasuk kategori model linguistik karena menggunakan logika matematika dengan *premis* dan *consequent* [9]. Michio Sugeno mengusulkan penggunaan *singleton* sebagai fungsi keanggotaan dari konsekuen. *Singleton* adalah sebuah himpunan *fuzzy* dengan fungsi keanggotaan yang pada titik tertentu mempunyai sebuah nilai dan 0 di luar titik tersebut. Pada model Sugeno hasil keluaran (*consequent*) yang didapatkan dari sistem berupa konstanta atau persamaan linear.

3.4 Enkripsi Blok Data Dengan AES

Algoritma AES adalah blok *chipper text* simetrik yang dapat mengenkripsi (*encipher*) dan dekripsi (*decipher*) informasi data. AES menggunakan sistem permutasi dan substitusi (P-Box dan S-Box). AES memiliki ukuran *block* data yang tetap yaitu 128 bit dan beberapa ukuran blok kunci yaitu mulai dari 128, 192, dan 256 bit [10]. Penggunaan algoritma AES dikelompokkan kedalam tiga tingkatan yaitu berdasarkan panjang kunci yang digunakan untuk mengenkrip dan mendekrip data pada ukuran blok 128 *bits*.

Enkripsi blok data dengan panjang *key* 128 bit memiliki sepuluh *round* untuk setiap proses enkripsi. Setiap masukan 128 bit *plaintext* dimasukkan kedalam *state* yang berbentuk bujur sangkar berukuran 4×4 byte. State ini di-XOR dengan *key* dan selanjutnya diolah sepuluh putaran dengan subtitusi-transformasi *linear-addkey* sehingga menghasilkan *ciphertext*.

Pada penelitian ini digunakan algoritma AES dengan *mode* operasi ECB (*Electronic Code Book*). Pada *mode* operasi ECB setiap blok *plaintext* dienkripsi secara individual dan independen untuk menjadi blok *ciphertext* [11].

Dalam penerapan *mode* operasi ECB untuk memotong *plaintext* menjadi sejumlah blok dengan ukuran yang telah ditetapkan memungkinkan terjadinya panjang *plaintext* tidak habis dibagi dengan panjang ukuran blok. Hal tersebut mengakibatkan blok terakhir berukuran lebih pendek daripada blok-blok lainnya. Salah-satu cara untuk mengatasinya yaitu dengan *padding*. *Padding* adalah menambahkan blok terakhir dengan pola bit yang teratur agar panjangnnya sama dengan ukuran blok yang ditetapkan [12].

4. Hasil dan Pembahasan

Tujuan uji coba dalam penelitian ini adalah untuk membandingkan dan mengamati kinerja *load balancing* dengan pertimbangan tiga parameter yaitu kondisi *workload agen*, antrian *job* dan daya komputasi komputer dimana agen tersebut berada. Seluruh hasil pengujian didapatkan dengan cara mengolah data yang di hasilkan dari proses percobaan yang dilakukan pada lingkungan sistem multi agen. Data tersebut kemudian di olah menjadi bentuk tabel dan disajikan dalam bentuk grafik untuk memudahkan dalam proses analisis.

4.1 Skenario Pengujian

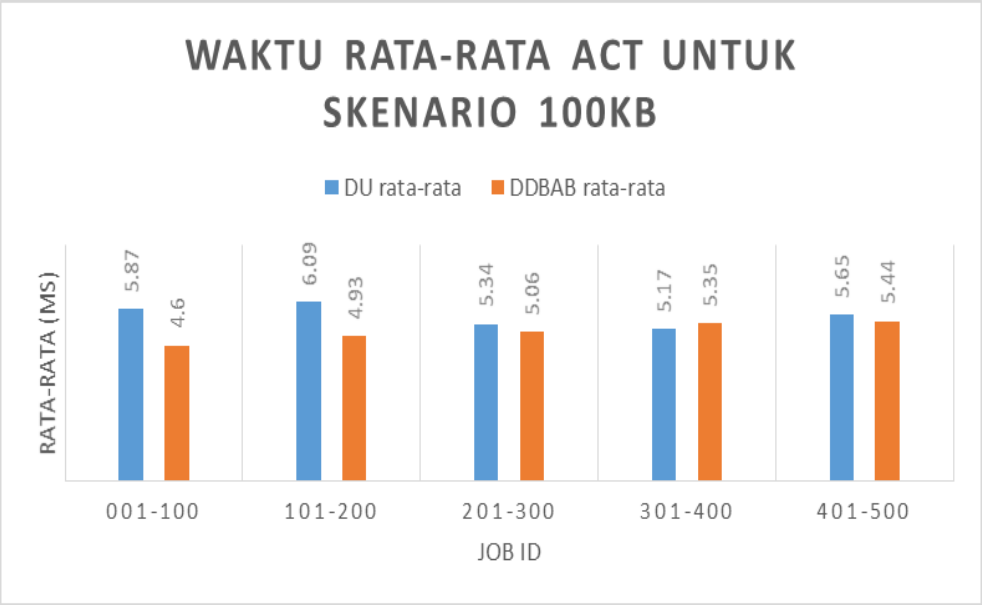
Tujuan dari pengujian sistem adalah untuk mengevaluasi kemampuan dari sistem yang dibangun. Pengujian dilakukan untuk mengetahui kinerja *Load balancing system* saat kondisi agen berubah.

Dalam pengujian ini dilakukan dua skenario pendistribusian *job* yaitu pendistribusian job secara statis dengan Distribusi *Uniform* (DU) dan pendistribusian job secara dinamis dengan Distribusi Dinamis Berbasis Alokasi Beban (DDBAB). Pendistribusian blok data secara dinamis (DDBAB) dengan menggunakan pertimbangan *Round Trip Time* (RTT), antrian *job* dan daya komputasi komputer sesuai dengan metode yang diusulkan. Parameter yang diuji dari kedua skenario tersebut adalah waktu rata-rata (*Actual Completion Time*) ACT.

Skenario yang dilakukan pada penelitian ini adalah setiap komputer pekerja hanya memiliki satu *agent worker*. Setiap *agent worker* akan dialokasikan 500 *job*. Pengujian dilakukan dengan lima tahapan yaitu: tahap pertama sistem diuji dengan 500 *job* dengan ukuran setiap *job* 100 KB, tahap kedua sistem diuji dengan 500 *job* dengan ukuran setiap *job* 250 KB, tahap ketiga sistem diuji dengan 500 *job* dengan ukuran setiap *job* 500 KB, pada tahap keempat sistem diuji dengan 500 *job* dengan ukuran setiap *job* 750 KB, pada tahap kelima sistem diuji dengan 500 *job* dengan ukuran setiap *job* 1000 KB.

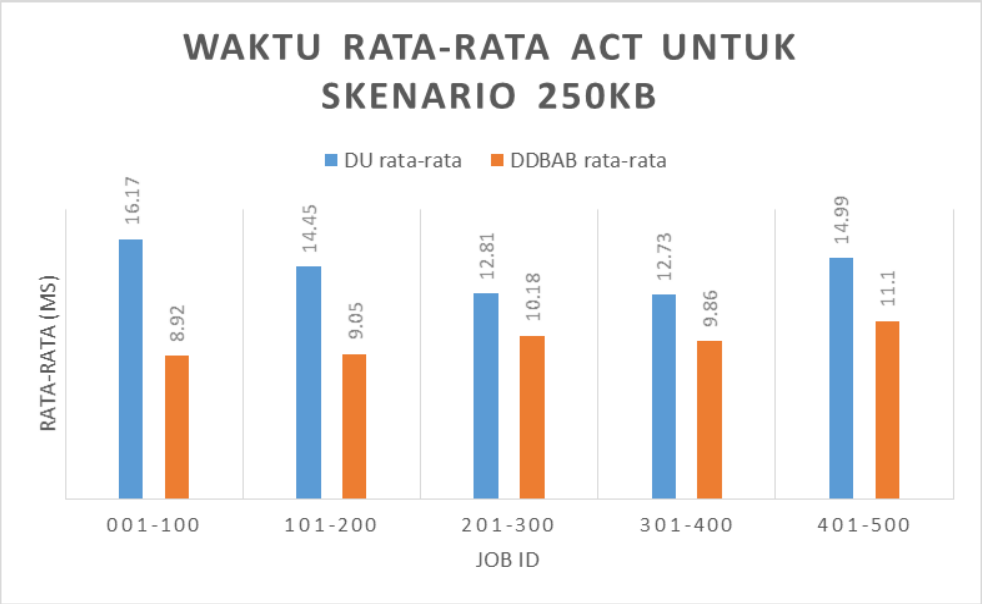
4.2 Analisis Hasil

Pada hasil yang diamati adalah waktu rata-rata penyelesaian suatu *job* oleh setiap *agent worker* dari setiap komputer. Pada bagian ini ditampilkan hasil pengujian pada skenario dengan jumlah *job* 500 dan *job* yang memiliki ukuran mulai dari 100 KB sampai 1000 KB



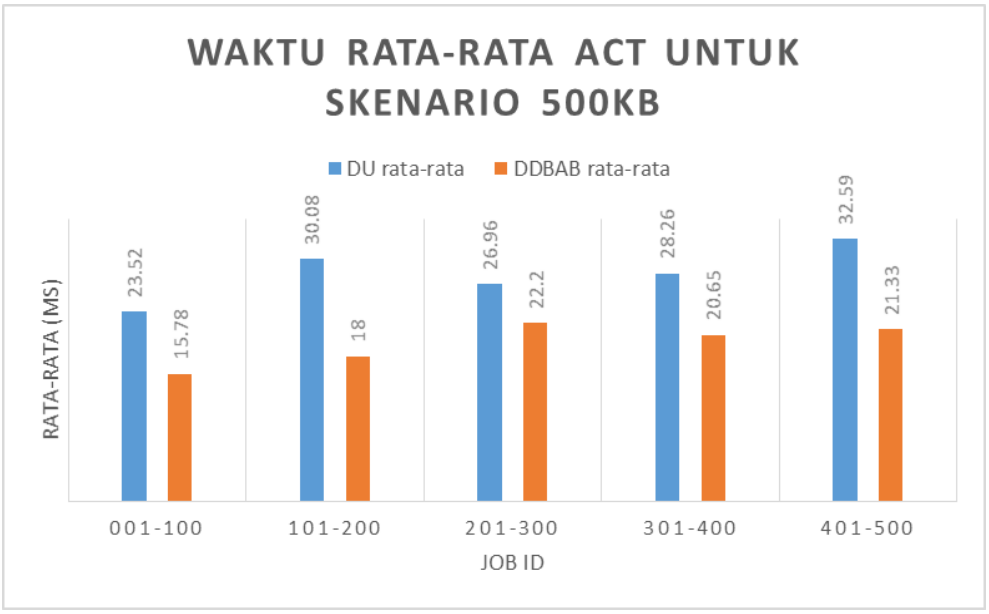
Gambar 7. Grafik Waktu Rata-rata ACT Untuk Skenario Pertama

Pada Gambar 7 dapat dilihat pada skenario dengan ukuran *job* 100 KB pada rentang *job* id 001-100 dan 101-200 terjadi perbedaan nilai ACT yang jelas diantara kedua metode pengujian. Metode DDBAB memiliki nilai rata-rata ACT lebih rendah dari metode DU.



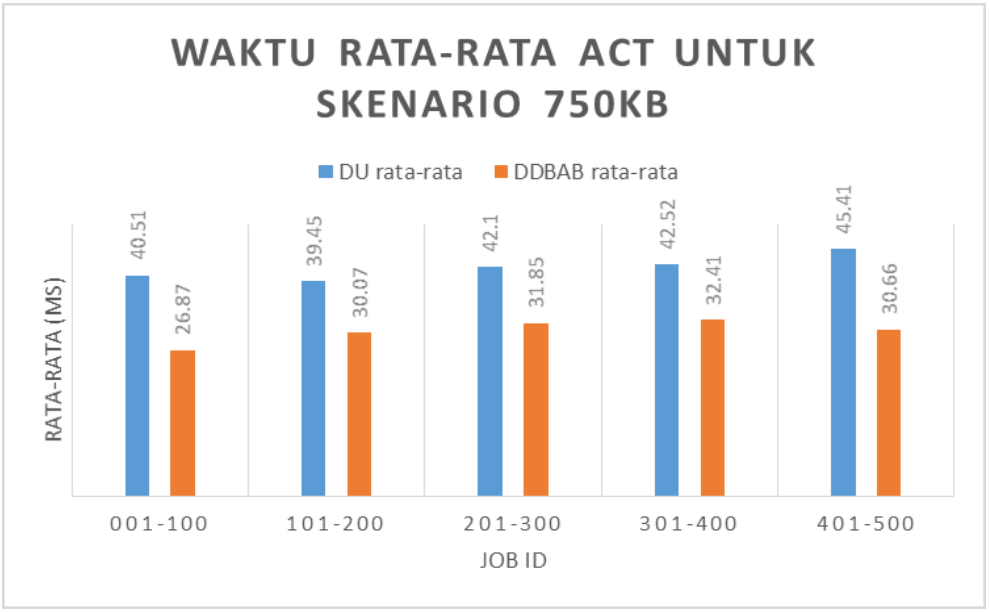
Gambar 8. Grafik Waktu Rata-rata ACT Untuk Skenario Kedua

Pada Gambar 8 dapat dilihat bahwa skenario dengan ukuran *job* 250 KB pada semua rentang *job* id. Metode Distribusi Dinamis Berbasis Alokai Beban (DDBAB) memiliki nilai rata-rata ACT lebih rendah dari metode DU



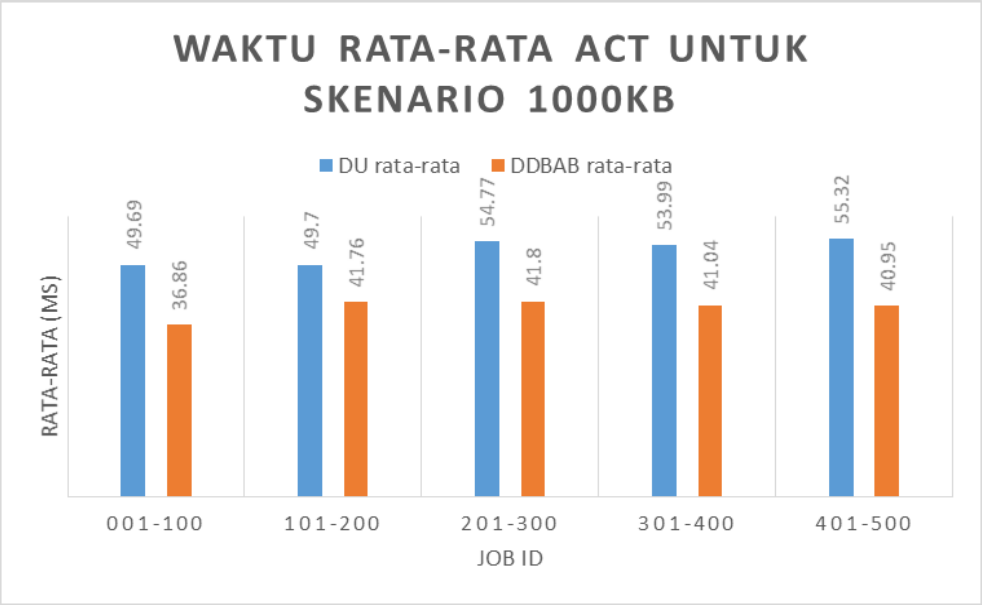
Gambar 9. Grafik Waktu Rata-rata ACT Untuk Skenario Ketiga

Pada Gambar 9 dapat dilihat bahwa skenario dengan ukuran *job* 500 KB pada semua rentang *job* id. Metode Distribusi Dinamis Berbasis Alokai Beban (DDBAB) memiliki nilai rata-rata ACT lebih rendah dari metode DU.



Gambar 10. Grafik Waktu Rata-rata ACT Untuk Skenario Keempat

Pada Gambar 10 dapat dilihat bahwa skenario dengan ukuran *job* 750 KB pada semua rentang *job* id. Metode Distribusi Dinamis Berbasis Alokai Beban (DDBAB) memiliki nilai rata-rata ACT lebih rendah dari metode DU.



Gambar 11. Grafik Waktu Rata-rata ACT Untuk Skenario Kelima

Pada Gambar 10 dapat dilihat bahwa skenario dengan ukuran *job* 1000 KB pada semua rentang *job* id. Metode Distribusi Dinamis Berbasis Alokai Beban (DDBAB) memiliki nilai rata-rata ACT lebih rendah dari metode DU.

5. Kesimpulan

Pendistribusian *job* secara dinamis dapat dilakukan dengan metode Distribusi Dinamis Berbasis Alokasi Beban (DDBAB). DDBAB dapat meningkatkan kinerja sistem multi agen dalam melakukan proses enkripsi *job* hingga mencapai 30,99 % dibandingkan dengan Distribusi *Uniform* (DU). Berdasarkan hasil pengujian menunjukkan bahwa nilai *Actual Completion Time* (ACT) untuk enkripsi *job* dengan panjang kunci 128 bit dipengaruhi oleh ukuran *job* dimana semakin besar ukuran suatu *job* maka akan semakin meningkat nilai ACT yang dihasilkan. Penelitian ini menggunakan lima skenario pengujian yaitu 500 *job* untuk setiap skenario dengan ukuran *job* mulai dari 100 KB, 250 KB, 500 KB, 750 KB dan 1000 KB.

Daftar Pustaka

[1] Drogoul, A., Vanbergue, D. & Meurisse, T., 2003. Multi-Agent Based Simulation: Where are the Agents ?. Lecture Notes in Computer Science, Volume Multi Agent Base simulation II, pp. 43-49.
[2] Shin, S. Y., Lee, H. C., Song, s. K., & Youn, H. Y. (2009). A Load Balancing Scheme for Multi-Agent Systems based on Agent State and Load Condition.
[3] Lee, Y. J., Park, G. Y., Song, H. K., & Youn, H. Y. (2012). A Load Balancing Scheme for Distributed Simulation Based on Multi-Agent System. International Conference on Computer Software and Applications Workshops (pp. 613-618). Sungkyunkwan University.
[4] S, W. & M, L., 1980. Assignment of Tasks and Resources for Distributed Processing. s.l., OMPCON.
[5] Leszczyna, R.,. Evaluation of Agent Platforms. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, Ispra, Italy 30 june. 2004.
[6] F. Bellifemine, G. Caire, and D. Greenwood, "Developing multiagent systems with JADE," Wiley & Sons, Sussex, 2007.
[7] http://jade.tilab.com/, diakses tanggal 12 Oktober 2013.
[8] Lotfi A. Zadeh. Fuzzy Logic Systems: Origin, Concepts, And Trends. Computer Science. Division Department of EECS. UC Berkeley. 10 November, 2004.

[9] Kusumadewi S., Purnomo H., Aplikasi Logika Fuzzy, Untuk pendukung keputusan, Graha Ilmu, 2004.
[10] Daemen, Joan; & Rijmen, Vincent. November 26 2001. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197.
[11] Draft NIST Special Publication 800-17, Modes of Operation Validation System (MOVS): Requirements and Procedures, May 1996.
[12] J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS#1 v2.0. In Advances in Cryptology CRYPTO'01, Santa Barbara, California, U.S.A., Lectures Notes in Computer Sci-ence 2139, pp. 230–238, Springer-Verlag, 2001.