PENINGKATAN KINERJA SISTEM MULTI AGEN DENGAN OPTIMALISASI ALOKASI BEBAN

Muhammad Rizka
Teknik Informatika, Jurusan Teknologi Informasi dan Komputer, Politeknik Negeri Lhokseumawe
Jalan Banda Aceh-Medan Km. 280,3 Buketrata, Lhokseumawe, Indonesia
E-mail: muhammad.rizka910@gmail.com

ABSTRAK

Sistem multi agen merupakan sekumpulan agen yang saling berinteraksi dan berkomunikasi untuk melaksanakan suatu tujuan tertentu. Setiap agen akan melakukan kolaborasi dan berkoordinasi untuk meyelesaikan suatu job yang diberikan. Dalam menangani job yang didistribusikan oleh sistem ke setiap agen dapat saja terjadi ketidakseimbangan workload diantara agen-agen. Load balancing merupakan sebuah solusi ketika ketidakseimbangan workload terjadi. Sistem multi agen yang diusulkan merupakan metode load balancing yang dapat mengalokasikan workload ke setiap agen secara dinamis. Sistem multi agen 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 agent worker dan penggunaan daya komputasi komputer dimana agent worker berada. Studi kasus yang diterapkan pada penelitian ini adalah enkripsi data dengan menggunakan algoritma AES (Advanced Encryption Standard). Hasil pengujian menunjukkan bahwa metode Distribusi Dinamis Berbasis Alokasi (DDBAB) Beban dapat meningkatkan kinerja sistem multi agen mencapai 37.85% dibandingkan dengan Distribusi Uniform (DU).

Kata Kunci: *Sistem Multi Agen, Alokasi Beban, Komunikasi Agen, Enkripsi Data, AES.*

ABSTRACT

Multi-agent system is a set of agents that interact and communicate to accomplish a specific purpose. Each agency will collaborate and coordinate to settle a given job. In dealing with jobs that are distributed by the system to each agent may be an imbalance of workload among agents. Load balancing is a solution when the workload imbalance occurs. The proposed multi-agent system is a load balancing method to allocate the workload to each agent dynamically. Multi-agent systems consist of a worker agent in charge of doing the job execution and monitor agent who is responsible for overseeing state worker agent and allocate jobs to each agent worker. Load balancing system is done with consideration of the three parameters, namely the condition of load agent worker, job queue worker agent and the use of computing power the computer where the agent is located worker. The case studies were applied in this research is data encryption algorithms using AES (Advanced Encryption Standard). The test results showed that the method of Dynamic Distribution Based Allocation (DDBAB) Expenses can improve system performance multi agency reached 37.85% compared to the Uniform Distribution (DU).

Keywords: Multi-Agent Systems, Allocation Load, Communications Agent, Data Encryption, AES

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[8]. Sekumpulan agen dalam sistem multi agen melakukan kolaborasi dan berkoordinasi untuk

meyelesaikan suatu *job* yang diberikan.

Dalam pendistribusian *job* dapat saja terjadi ketidakseimbangan *workload* diantara agen. Metode *load balancing* merupakan sebuah solusi ketidakseimbangan sistem dalam jaringan sistem multi agen. Dalam menangani masalah ketidak seimbangan sistem dalam jaringan sistem multi agen ada beberapa penelitian yang membahas mengenai metode *load balancing* pada sistem multi agen yaitu seperti yang dilakukan oleh [15]. Metode *load balancing* yang diusulkan mengalokasikan *resource* komputasi hanya kepada agen aktif dalam suatu komputer.

Setiap agen mendapatkan jumlah alokasi *resource* komputasi yang sama didalam sebuah komputer. Jumlah agen dalam suatu komputer mengindikasikan sebuah nilai *weight* yang digunakan untuk proses *load balancing*. *Load balancing* dilakukan pada saat agen mengalami *overload* sehingga agen tersebut harus dipindahkan ke komputer lain untuk mengurangi *workload*.

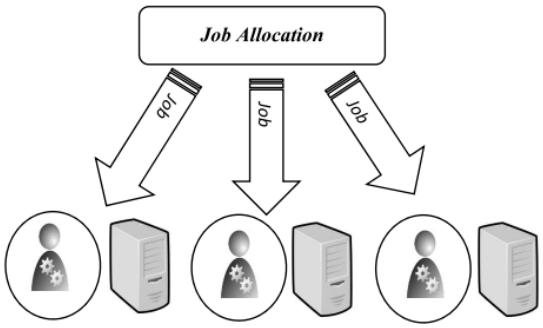
Dengan hanya melakukan migrasi agen dari suatu komputer ke komputer lain masih memungkinkan terjadinya *overload* karena adanya penambahan agen dan *task* pada komputer tujuan sehingga dapat mengakibatkan terjadi proses *reload balancing* pada sistem yang pada akhirnya akan membuat sistem mengalami *overload* dan menjadi lambat.

Pada penelitian ini metode yang diusulkan yaitu 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*. Mekanisme *load balancing* untuk pengalokasian *job* pada penelitian ini ditentukan berdasarkan pertimbangan kondisi *load agent worker*, antrian *job* dan penggunaan daya 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 METODE PENELITIAN

Pada penelitian ini diusulkan sebuah mekanisme *load balancing* dengan mempertimbangkan kondisi *load* agen dan penggunaan 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 memeriksa kondisi *load agent worker* pada setiap komputer dan selanjutnya mengalokasikan sejumlah *job* (blok data) ke *agent worker*.

Agent monitor akan mengalokasikan *job* ke setiap *agent worker* sesuai dengan kondisi masing-masing *agent worker*. Informasi tentang kondisi *agent worker* akan mempengaruhi *load control* dari *agent monitor* untuk menentukan jumlah *job* ke setiap *agent worker*. Pada Gambar 3.2 ditunjukkan bagan alokasi *job* dari *agent monitor* ke seluruh *agent worker* dalam sistem.

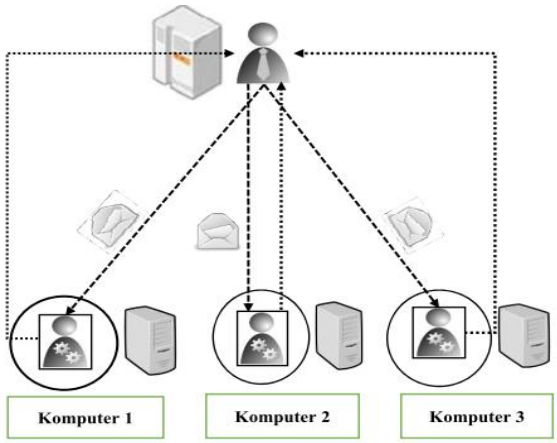


Gambar 1. Alokasi *Job*.

Pada Gambar 1 tersebut dapat dilihat bahwa *agent monitor* mengalokasikan *job* ke seluruh *agent worker* dalam sistem. *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*. Mekanisme *load balancing* data dilakukan dengan tiga pertimbangan yaitu kondisi *load agent worker*, antrian *job* dan penggunaan daya komputasi komputer dimana *agent worker* berada.

a. 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:



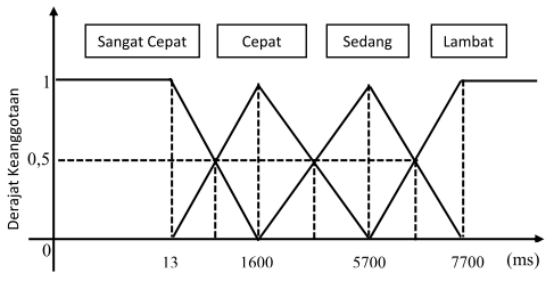
RTT=*receiving time* R(t) – *forwarding time* F(t)……..(3.1)

Gambar 2. Proses Deteksi Kondisi *Load Agent Worker*.

Pada Gambar 2 merupakan ilustrasi dari proses pengecekan kondisi *agent worker*. Nilai *round trip time* (RTT) dari setiap *agent worker* digunakan untuk menentukan kondisi *agent worker*.

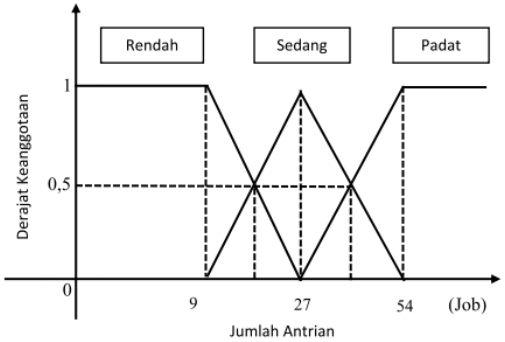
Nilai RTT yang didapatkan oleh *agent monitor* akan mendeskripsikan kondisi *load agent worker* yang juga merupakan kondisi penggunaaan daya komputasi oleh *agent worker* saat melakukan proses enkripsi *job*. Dalam penelitian ini nilai RTT *agent worker* diklasifikasikan kedalam beberapa kelompok yaitu: sangat cepat, cepat, sedang dan lambat. Pengelompokan nilai RTT tersebut berdasarkan waktu pemprosesan pesan yang dilakukan oleh *agent worker* sehingga semakin lama waktu pemprosesannya maka akan berdampak pada semakin lama *agent worker* membalas pesan dari *agent monitor*. Dalam hal ini kondisi trafik pada jaringan diabaikan. Pada Gambar 3 ditunjukkan grafik *membership* nilai RTT dari *agent worker* dalam sistem multi agen.

Gambar 3. Grafik *Membership* Nilai RTT.



b. 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. Pada Gambar 3.5 ditunjukkan grafik *membership* antrian *job* pada *agent worker*.



Gambar 4. Grafik *Membership* Antrian *Job*.

c. Penggunaan 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 penggunaan daya komputasi pada suatu komputer dimana *agent worker* berada.

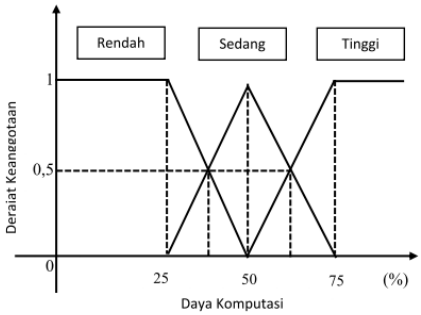
Parameter pertimbangan pengalokasian *job* berdasarkan penggunaan daya komputasi komputer dengan cara menentukan persentase tingkat penggunaan komputasi komputer dimana *agent worker* berada. Persentase tingkat penggunaan komputasi komputer dimana *agent worker* berada dapat ditunjukkan pada Tabel 1:

Tabel 1. Penggunaan Daya Komputasi Komputer

Tingkat Komputasi	Rendah	Sedang	Tinggi
Persentase (%)	25	50	75

Pada Tabel 1 ditunjukkan tingkat penggunaan daya komputasi komputer yang digunakan untuk menjadi parameter penggunaan daya komputasi. Dalam mendapatkan informasi

terkait penggunaan daya komputasi pada setiap komputer digunakan *library* sigar versi 1.6.4 yang mendukung pemrograman java. Ketiga tingkatan tersebut akan mempengaruhi jumlah pengalokasi *job* untuk setiap *agent worker*. Pada Gambar 5 ditunjukkan grafik *membership* penggunaan daya komputasi komputer dalam jaringan *prototype* sistem multi agen yang dibangun.



Gambar 5. Grafik *Membership* Penggunaan Daya Komputasi Komputer

2.1 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. 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*. 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. Ada dua model *fuzzy* sugeno yaitu:

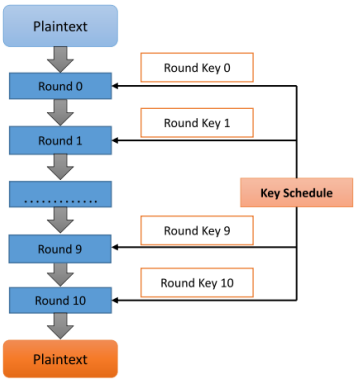
a. Model fuzzy sugeno order nol
   Persamaan model sugeno order nol yaitu:
   IF (x1 is A1) o (x2 is A2) o (x3 is A3) o… o (xN is AN) THEN z=k
   Dengan A_i adalah himpunan fuzzy ke–i sebagai anteseden dan k adalah konstanta tegas sebagai konsekuen.
b. Model fuzzy sugeno orde satu
   IF (x1 is A1) o… o (xN is AN) THEN z = p1*x1+… + pN*xN+q
   Dengan A_i adalah himpunan fuzzy ke–i sebagai anteseden, p_i konstanta tegas ke-i dan q konstanta pada konsekuen.

Dalam penelitian ini ada tiga parameter yang akan dijadikan inputan untuk *fuzzy logic*.

1. Nilai *Round Trip Time* (RTT) pesan. Nilai RTT memiliki empat kategori yaitu, sangat cepat, cepat, sedang dan lambat.
2. Penggunaan daya komputasi komputer memiliki tiga tingkatan yaitu cepat, sedang, lambat.
3. Antrian Job memiliki tiga kategori yaitu rendah, sedang dan padat.

Rule fuzzy logic yang diimplementasikan untuk proses pengalokasian job oleh agent monitor ke *agent worker* ada sebanyak 36 *rule*. *Rule* tersebut merepresentasikan setiap kondisi dari *multi agent system.*

2.2 Enkripsi Blok Data Dengan AES.



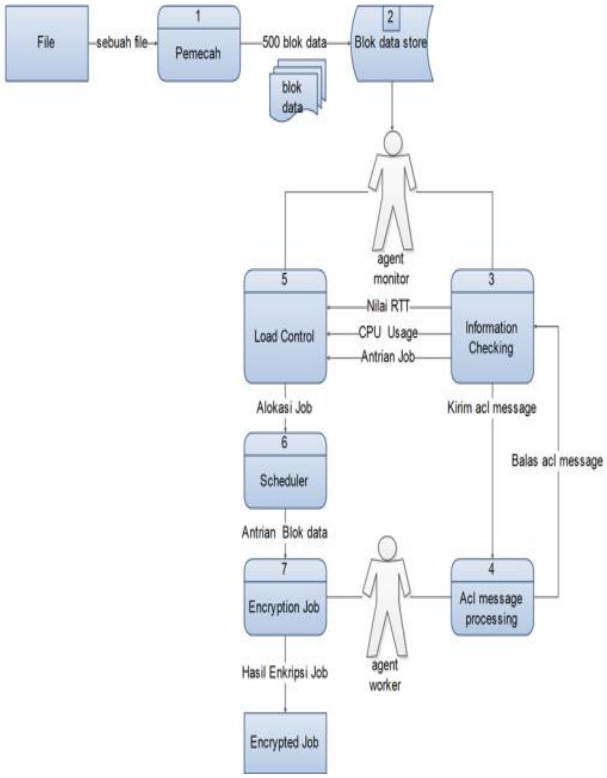
Gambar 6. Enkripsi Blok Data Dengan AES

Dalam penelitian ini sebuah blok data yang berukuran besar akan dipecahkan kedalam beberapa blok data dengan ukuran yang relative kecil. Blok-blok data akan didistribusikan keseluruh agent worker. Agent worker akan melakukan proses eksekusi blok data yang telah didistribusikan oleh agent monitor. Sistem multi agen dalam penelitian ini diaplikasikan pada studi kasus enkripsi data dengan menggunakan algoritma AES (Advanced Encryption Standard). Pada Gambar 6 merupakan skema enkripsi data algoritma AES dengan panjang key 128 bit[1].

2.3. Skema Proses.

Penelitian ini memiliki tujuh proses tahapan utama yang saling terkait dan setiap proses memiliki fungsi yang spesifik untuk mewujudkan tujuan dari
sistem multi agen. Dalam membangun sistem multi agen yang seimbang (balance) peneliti

mengusulkan load balancing terhadap workload agent worker dalam sistem multi agen. 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. Penjelasan lebih lanjut mengenai skema sistem multi agen yang diusulkan dapat dilihat pada Gambar 7.



Gambar 7. Alur Usulan Sistem

Proses pertama adalah sebuah file dengan ukuran besar dimasukkan ke dalam proses pemecah. File akan dipecah-pecah ke dalam suatu ukuran blok data dan setiap blok data tersebut memiliki ukuran yang sama. Dalam penelitian ini ada lima skenario pengujian yang setiap pengujian menggunakan 500 blok data. Pada setiap pengujian ukuran blok data yang diuji berbeda-beda mulai dari 100 KB untuk pengujian pertama, 250 KB untuk pengujian kedua, 500 KB untuk pengujian ketiga, 750 KB untuk pengujian keempat dan 1000 KB untuk pengujian kelima. Blok-blok data yang telah dipecah tersebut disimpan kedalam sebuah data store. Setiap blok data yang akan dialokasikan ke agent worker disimpan pada folder yang terpisah. Proses information checking adalah proses memeriksa kondisi agent worker dalam setiap komputer. Pada proses ini acl message akan dikirimkan secara periodik ke agent worker. Acl message akan dikirimkan ke agent worker setiap lima detik. Pesan ACL yang dikirimkan ke agent worker berjumlah tiga buah yang masing-masing pesan berisi waktu pada saat pesan dikirimkan dan sebuah string. Dalam proses information checking akan dicatat waktu pengiriman setiap acl message ke agent worker. Informasi waktu tersebut akan digunakan untuk menentukan seberapa lama agent worker membalas acl message. Proses acl message processing yaitu proses yang berada pada agent worker.

Pada proses tersebut agent worker akan menunggu setiap acl message yang dikirimkan dari proses information checking. Setelah acl message diterima selanjutnya dibuat sebuah acl message yang berisi antrian job yang sedang terjadi dan tingkat komputasi komputer dimana agent worker berada untuk selanjutnya dikirimkan ke proses information checking. Pada proses information checking akan dicatat waktu balasan dari setiap agent worker. Proses information checking akan mengkalkulasi nilai RTT (Round Trip Time) dari acl message yang dikirimkan. Informasi nilai RTT (Round Trip Time) dari setiap agent worker akan dikirimkan ke proses load control untuk dilakkukan pengalokasian job. Proses load control akan melakukan pengalokasian jumlah job ke setiap agent worker. Pada proses load control setiap agent worker yang berada pada setiap komputer akan mendapatkan jumlah job yang berbeda. Pada proses load control akan dialokasikan job untuk agent worker berdasarkan nilai RTT (Round Trip Time), antrian job dan tingkat komputasi komputer. Pengalokasian job dilakukan dengan metode fuzzy logic. Metode fuzzy logic digunakan untuk menetukan jumlah alokasi job yang sesuai untuk setiap agent worker berdasarkan tiga inputan yaitu nilai RTT, antrian job dan tingkat komputasi komputer. Job yang akan dialokasikan berkisar antara 1 sampai 100 job.

Pada proses yaitu scheduler yaitu bertugas dalam mengirimkan sejumlah blok data yang telah dialokasikan ke seluruh agent worker dalam setiap komputer. Pada proses enkripsi blok data, setiap agent worker akan mengenkripsi blok-blok data dengan menggunakan algoritma AES dengan panjang key 128 bit.

3 PENGUJIAN DAN PEMBAHASAN

Skenario pengujian yang dilakukan untuk menguji kinerja sistem multi agen. Setiap agent worker akan dialokasikan sebanyak 500 job. Pengujian dilakukan dengan lima tahapan yaitu:

1. Sistem diuji dengan 500 job dengan ukuran setiap job 100 KB.
2. Sistem diuji dengan 500 job dengan ukuran setiap job 250 KB.

3. Sistem diuji dengan 500 job dengan ukuran setiap job 500 KB.
4. Sistem diuji dengan 500 job dengan ukuran setiap job 750 KB.
5. Sistem diuji dengan 500 job dengan ukuran setiap job 1000 KB

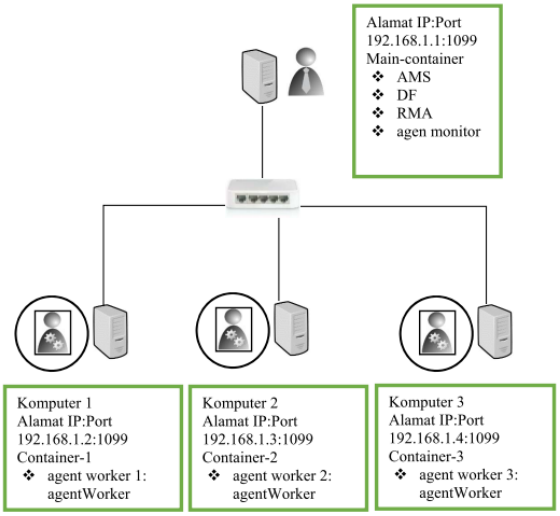
3.1 Implementasi Topologi Jaringan Sistem Multi Agen

JADE (Java Agent DEvelopment) merupakan sebuah framework multi agen yang berbasis java. Topologi jaringan sitem multi agen yang dibangun dengan empat komputer yang memiliki spesifikasi yang berbeda. Tiga komputer sebagai komputer pekerja yang didalamnya akan ditempatkan agent worker dan satu komputer sebagai komputer monitor yang akan menjadi tempat bagi agent monitor. Setiap komputer pekerja terdiri dari satu container dan setidaknya memiliki satu agent worker. Main-container berada pada komputer monitor yang didalamnya terdiri dari AMS (Agent Management System), DF (Directory Facilitator), RMA (Remote Management Agent) dan yang terakhir adalah agent monitor. Setiap komputer pada prototype jaringan sistem multi agen memiliki spesifikasi yang berbeda. Spesifikasi komputer yang bervariasi bertujuan untuk mengukur performa load balancing sistem yang diusulkan pada lingkungan yang heterogen. Pada Tabel 2 ditunjukkan spesifikasi komputer yang digunakan dalam penelitian ini:

Tabel 2. spesifikasi komputer

Nama Komputer	Prosessor	*Clock Speed*	Jumlah *Thread*	*Memory*
Monitor	Core i3 (2120)	3,4 GHz	4	4 GB
1	Core i3 (3240)	3,4 GHz	4	4 GB
2	Core i5 (3317U)	1,7 GHz	4	4 GB
3	Pentium 4	3 GHz	2	1 GB

Agent monitor yang berada didalam main-container yang memiliki IP 192.168.1.1 dengan port 1099. Agen worker-1 yang berada pada container-1 memiliki IP 192.168.1.2 dengan port 1099. Agen worker-2 yang berada pada container-i memiliki IP 192.168.1.3 dengan port 1099. Agen worker-3 yang berada pada container-3 memiliki IP 192.168.1.4 dengan port 1099.



Gambar 8. Testbed Jaringan

3.2 Implementasi Load Balancing

Penelitian ini dibangun berdasarkan pemrograman java agar mendukung framework JADE. Sistem multi agen yang dibangun memiliki dua jenis agen yaitu agent monitor dan agent worker yang saling bekerja sama dalam mewujudkan kondisi workload yang stabil. Agent monitor ditempatkan pada main-container sedangkan agent worker ditempatkan pada setiap container yang tersebar pada jaringan sistem multi agen. Dalam framework sistem multi agen setiap agen memiliki tingkah-laku (behaviour) dalam menangani task yang diberikan. Agent monitor mengimplementasikan dua jenis behaviour dalam perilakunya yaitu, Ticker Behaviour dan Cyclic Behaviour. Ticker Behaviour merupakan behaviour yang melakukan eksekusi program pada suatu periode waktu yang ditentukan. Ticker Behaviour diimplementasikan dalam pengiriman pesan ACL ke agent worker setiap dua detik untuk pengecekan kondisi workload agent worker. Cyclic Behaviour merupakan behaviour yang melakukan eksekusi looping terhadap program didalamnya. Cyclic Behaviour diimplementasikan dalam proses penerimaan pesan balasan dari agent worker selanjutnya agent monitor menghitung waktu round trip time pesan tersebut.

4 HASIL DAN ANALISIS

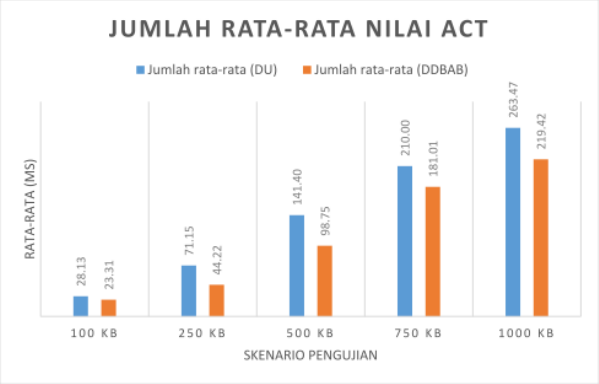
Seluruh hasil pengujian didapatkan dengan cara mengolah data yang di hasilkan dari proses percobaan yang dilakukan pada lingkungan sistem multi agen. Data tersebut kemudian di olah menjadi bentuk tabel dan disajikan dalam bentuk grafik untuk memudahkan dalam proses analisis. Pada Tabel 4.5 ditunjukkan jumlah rata-rata dan jumlah total actual completion time (ACT) dari

setiap skenario mulai dari ukuran job 100 KB, 250 KB, 500 KB, 750 KB dan 1000 KB. Pada tabel tersebut dapat dilihat persentase perbedaan waktu penyelesaian job antara DU dengan DDBAB. Sistem DDBAByang diusulkan dapat bekerja optimal dalam mengurangi waktu penyelesaian job.

Tabel 3. Jumlah rata-rata dan jumlah total ACT Pada Setiap Skenario.

Skenario	Ukuran *Job*	Distribusi *Uniform* (DU) (milliseconds)		Distribusi Dinamis Bebasis Alokasi Beban (DDBAB) (milliseconds)		Persentase (%) Penurunan	
		Jml rata-rata	Jml total	Jml rata-rata	Jml total	Jml rata-rata	Jml total
1	100 KB	28,13	2813,33	23.31	2331.33	17.13	17.13
2	250 KB	71,15	7115,33	44.22	4422.00	37.85	37.85
3	500 KB	141,40	14140,67	98.75	9875.00	30.16	30.17
4	750 KB	210,00	20998,67	181.01	18100.67	13.80	13.80
5	1000 KB	263,47	26346,67	219.42	21942.00	16.72	16.72

Pada Gambar 9 ditunjukkan jumlah nilai rata-rata Actual Completion Time (ACT) dari setiap skenario yang dilakukan. Hasil pengujian menujukkan peningkatan nilai Actual Completion Time (ACT) seiring dengan bertambahnya ukuran job.



Gambar 9. Jumlah Rata-rata Nilai ACT Pada Setiap Skenario.

Berdasarkan Gambar 4.22 dapat dilihat bahwa waktu rata-rata penyelesaikan 500 job mengalami peningkatan pada setiap skenario. Peningkatan tersebut membuktikan bahwa semakin besar ukuran job maka semakin tinggi waktu penyelesaiannya. Pada pengujian dengan menggunakan DDBAB menghasilkan nilai rata-rata ACT yang lebih rendah dari pada nilai ACT DU. Berdasarkan pengujian yang dilakukan didapatkan bahwa waktu rata-rata penyelesaian job yang dihasilkan oleh DDBAB pada setiap skenario mulai dari 100KB sampai dengan 1000KB selalu lebih rendah dari DU.

5 KESIMPULAN

Pada bagian ini disampaikan kesimpulan berdasarkan penelitian yang telah dilakukan.

1. 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 37.85% dibandingkan dengan Distribusi Uniform (DU).
2. Konsep pendistribusian job secara dinamis dapat diimplementasikan pada sistem multi agen dengan cara memadukan konsep behaviour pada sistem multi agen dengan algoritma fuzzy logic. Perpaduan tersebut berhasil mengontrol kondisi load agent worker pada setiap skenario percobaan

sehingga nilai ACT yang didapatkan relative stabil bila dibandingkan dengan Distribusi Uniform (DU).

3. Proses penentuan agen yang mengalami overload dapat dilakukan dengan menghitung nilai RTT dan antrian job. Penentuan tersebut berhasil menekan laju alokasi job sehinga berdampak pada nilai rata-rata ACT yang dihasilkan oleh agent worker. Penurunan nilai rata-rata ACT yang didapatkan pada setiap skenario dengan DDBAB yaitu berkisar antara 13.80% sampai dengan 37.85% lebih rendah pada DU yang tanpa melibatkan penentuan agen overload dalam pengalokasian job.

6 DAFTAR PUSTAKA

[1] Adiwidya, B. M. D., 2010. Algoritma AES (Advanced Encryption Standard) dan Penggunaannya dalam Penyandian Pengompresian Data. [Online] [Accessed 8 april 2013].

[2] Bellifemine, F., n.d. Java Agent DEvelopment Framework. [Online] Available at: http://jade.tilab.com/ [Accessed 12 October 2013].

[3] Christophe de Canniere, A. B. a. B. P., 2006. An Introduction of Block Cipher Cryptanalysis. s.l., Proceedings of the IEEE.

[4] Drogoul, A., Vanbergue, D. & Meurisse, T., 2003. Multi-Agent Based Simulation:

[5] Where are the Agents ?. Lecture Notes in Computer Science, Volume Multi Agent Base simulation II, pp. 43-49.

[6] Fabio Bellifemine, A. P. R., 2000. JADE – A FIPA-compliant agent framework. [Online] Available at: http://www.researchgate.net [Accessed 9 October 2013].

[7] Fabio Bellifemine, G. C. a. D. G., 2007. Developing multiagent systems with JADE. New York: Wiley & Sons.

[8] Gutierrez, C., Magarino, I. g. & Fernadez, R. f., 2011. Detection of Undesirable Communication Patterns in Multi-agent Systems. Engineering Aplications of Artificial Intelligence, pp. 103-116.

[9] Isizoh A. N., O. S. O. A. O. C., 2012. Temperature Control System Using Fuzzy Logic. International Journal of Advanced Research in Artificial Intelligence, Volume 1, p. 3.

[9] 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. s.l., Sungkyunkwan University, pp. 613-618.

[10] Leszczyna, r., 2004. Evaluation of Agent Platforms, Ispra, Italy: Institute for the Protection and security of the Citizen. 68.

[11] M Azmi, Z. . R. et al., 2011. Performance Comparison of Priority Rule Scheduling Algorithms. International Journal of Grid and Distributed Computing, Volume Vol. 4, No. 3, p. September.

[12] Manger, J., 2001. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS#1 v2.0. In Advances in Cryptology CRYPTO'01. Computer Sci-ence 2139, Issue Springer-Verlag, p. 230–238.

[13] Rojas, R., 1996. Fuzzy sets and fuzzy logic. In: Neural Networks. berlin: Springer-Verlag, p. 289.

[14] Selent, D., 2010. ADVANCED ENCRYPTION STANDARD. RIVIER ACADEMIC JOURNAL, 6(2).

[15] 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.

[16]Wyman, B., 2011. Understanding Encryption. Security Awareness , 8 july.

[17] Zadeh, L. A., 2004. Fuzzy Logic Systems: Origin, Concepts, And Trends, UC Berkeley: Computer Science. Division Department of EECS.