TESIS – KI142502

Pengembangan Mekanisme Konstruksi *Behaviour Mobile Agent* Secara *Runtime* Pada Lingkungan Heterogen Dengan Efisiensi Penggunaan Dan Pemilihan Sumber Daya

M Misbachul Huda
NRP. 5113201049

DOSEN PEMBIMBING
Waskitho Wibisono, S.Kom., M.Eng., Ph.D

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI BERBASIS JARINGAN
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

TESIS – KI142502

DEVELOPMENT CONSTRUCTION OF MECHANISM RUNTIME BEHAVIOUR MOBILE AGENT ON HETEROGENEOUS ENVIRONMENT WITH RESOURCES EFFICIENCY AND SELECTION

M Misbachul Huda
NRP. 5113201049

SUPERVISOR
Waskitho Wibisono, S.Kom., M.Eng., Ph.D

MAGISTER PROGRAMME
INFORMATICS ENGINEERING DEPARTMENT
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Bismillahirrohmanirohim.

Alhamdulillahirabil'alamin, segala puji bagi Allah SubhanahuWata'alla, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tesis yang berjudul "**Pengembangan Mekanisme Konstruksi Behaviour Mobile Agent secara Runtime pada Lingkungan Heterogen dengan Efisiensi Penggunaan dan Pemilihan Sumber Daya**" dengan baik.

Dalam pelaksanaan dan pembuatan Tesis ini tentunya sangat banyak bantuan-bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tesis ini dengan baik.
2. Kedua orang tua penulis, saudara, dan keluarga besar penulis.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku dosen pembimbing yang telah memberikan kepercayaan, dukungan, bimbingan, nasehat, perhatian, serta semua yang telah diberikan kepada penulis.
4. Teman-teman seperjuangan bidang minat NCC terima kasih atas kebersamaannya selama ini.
5. Seluruh teman S2 Teknik Informatika ITS angkatan 2013, dua tahun bersama kalian sadar atau tidak telah membentuk karakter dan kepribadian penulis. Terima kasih atas rasa kekeluargaan yang telah kalian berikan kepada penulis. Semoga kita semua sukses di dunia dan akhirat.
6. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu terselesaikannya Tesis ini.

"Tiada Gading yang Tak Retak", begitu pula dengan Tesis ini. Oleh karena itu, penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Januari 2016

Penulis

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
M. Misbachul Huda
Nrp. 5113201049

Dengan judul :
Pengembangan Mekanisme Runtime Komposi Mobile Agent Pada Lingkungan Heterogen Dengan Efisiensi Penggunaan&Ketersediaan Sumber Daya

Tanggal Ujian : 19-1-2016
Periode Wisuda : 2015 Gasal

Disetujui oleh:

Waskitho Wibisono, S.Kom, M.Eng, Ph.D
NIP. 197410222000031001 (Pembimbing 1)

Prof.Ir.Supeno Djanali, M.Sc, Ph.D
NIP. 194806191973011001 (Penguji 1)

Tohari Ahmad, S.Kom, MIT, Ph.D
NIP. 197505252003121002 (Penguji 2)

Baskoro Adi Pratomo, S.Kom, M.Kom
NIP. 198702182014041001 (Penguji 3)

Direktur Program Pasca Sarjana,

Prof.Ir.Djauhar Manfaat, M.Sc., Ph.D.
NIP. 196012021987011001

Pengembangan Mekanisme Konstruksi *Behaviour Mobile Agent* secara *Runtime* pada Lingkungan Heterogen dengan Efisiensi Penggunaan dan Pemilihan Sumber Daya

Nama Mahasiswa : M Misbachul Huda
NRP : 5113 201 049
Pembimbing : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Abstrak

Sejak *Mobile Agent* (MA) dikembangkan pada lingkungan yang heterogen, banyak penelitian yang berkontribusi pada efisiensi penggunaan sumber daya. Salah satunya adalah proyek VERSAG yang dikembangkan oleh Gunasekera. VERSAG membuat mekanisme konstruksi *behavior* yang terpisah dari MA sehingga dapat di *load,* eksekusi*,* dan *unload* sesuai dengan kebutuhan ketika *running*. Kemampuan ini dinamakan *runtime compositional* MA. Namun kemampuan ini akan relevan jika beroperasi pada lingkungan yang memiliki sumber daya yang selalu cukup untuk menjalankan suatu *behavior.* Permasalahan akan muncul jika sumber daya yang dimiliki tidak cukup untuk menjalankan suatu *behavior.*

Dalam proposal ini diusulkan sebuah mekanisme konstruksi *behaviour* MA secara *runtime* pada lingkungan heterogen dengan efisiensi penggunaan dan ketersediaan sumber daya. MA yang dapat memperhitungkan keterbatasan sumber daya pada lingkungan heterogen. MA dikembangkan menjadi aplikasi cerdas yang dapat berinteraksi dengan *context* dan beradaptasi dengan lingkungan operasi. Adaptasi dapat terjadi ketika MA *running* pada *node* dengan sumber daya terbatas.

Hasil pengujian menunjukkan bahwa metode yang diusulkan mampu menangani permasalahan pemilihan sumber daya terbaik untuk melakukan menjalankan *behaviour*. Dari hasil yang didapat, bahwa metode yang diusulkan mampu mendapatkan 100% node dengan waktu respon terbaik.

Kata kunci: *mobile agent*, ketersediaan sumber daya*,* konstruksi *runtime*.

Development Construction of Mechanism Runtime Behaviour Mobile Agent on Heterogeneous Environment with Resources Efficiency and Selection

Student Name : M Misbachul Huda

NRP : 5113 201 049

Supervisor : Waskitho Wibisono, S.Kom, M.Eng., Ph.D.

Abstract

Since the Mobile Agent (MA) developed in a heterogeneous environment, a lot of research that contribute to resources efficiency. One is a project developed by Gunasekera VERSAG. VERSAG makes separated behavioral mechanisms from the MA so that it can be loaded, executed and unloaded when it is running according to the needs. This ability is named as runtimes composite MA. However, this ability will be an advantage when it is used to run a behavior in the sufficient resources environment. Some issues will arise when a node does not have sufficient resource to run a behavior. Problem will arise if the resources is not enough to run a behavior.

In this proposal proposed a runtime mechanism composite MA in heterogeneous environments with resources efficiency. It is MA that can consider to resource limitations in heterogeneous environments. MA is developed as an intelligent application that can interact with the context and it adapts with the operation environment. The adaptation can be reached when MA runs on the limited resources.

The test results show that the proposed method is able to handle the problems selecting the best resources to execute behavior. From the results obtained, that the proposed method is able to get 100% node with the best response time.

Keywords: *mobile agent, resources availability, runtime construction*

DAFTAR ISI

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Pada Bab ini akan dijelaskan mengenai beberapa hal dasar dalam pembuatan penelitian yang meliputi latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, dan batasan masalah.

1.1. Latar Belakang

Mobile Agent (MA) telah dikembangkan bertahun-tahun karena bersifat *autonomous*, terdistribusi, dan dapat berpindah. Telah banyak wilayah penelitian yang melibatkan MA sebagai solusi teknologi seperti yang dilakukan Pavle Skocir pada area *Game* (Skocir, et al., 2012), Paul Anderson pada area *Cloud Computing* (Anderson, et al., 2012), Ko Sibata pada area Robotik (Sibata, et al., 2014), Damir Šoštarić pada area *Wireless Sensor Network* (WSN) (Šoštarić, et al., 2012), dan pada area-area lain seperti *distributed computing* dan kecerdasan buatan. Dengan wilayah penggunaan yang luas, MA menjadi topik yang menarik untuk diteliti.

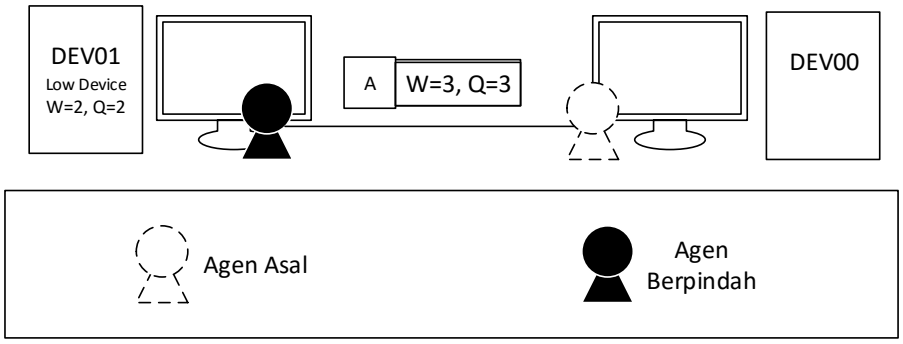
Sejak MA dikembangkan pada *pervasive computing,* telah banyak metodologi yang dikembangkan. Beberapa metodologi menghasilkan *framework* sebagai kerangka penelitian yang dapat dikembangkan. Salah satu *framework* penelitan pada MA adalah *VERsatile Self-adaptive AGents* (VERSAG) (Gunasekera, et al., 2008). VERSAG adalah proyek disertasi yang dikerjakan oleh Gunasekera untuk menempuh pendidikan doktoral di Universitas Monash di Australia.

Umumnya, MA diwujudkan sebagai sebuah *thread* untuk menjalankan satu atau sekumpulan tugas. *Behaviour* adalah sebuah objek yang merepresentasikan tugas yang dilakukan oleh MA. Gunasekera menekankan VERSAG pada optimasi MA menjadi ringan dengan membuat *behaviour* MA yang dapat dibangun pada saat *runtime. Behaviour* MA yang dapat di *load*, eksekusi, dan *unload* oleh MA saat *runtime*. Dengan menggunakan metode konstruksi *behavior* pada saat *runtime*, MA pada VERSAG dapat bermigrasi dari suatu *node* ke *node* lain tanpa membawa beban *behaviour* yang tidak perlu. Metode yang seperti itu

menjadi keunggulan dibandingkan yang dilakukan MA konvensional sebab dapat mengurangi *network traffic* saat migrasi.

Gunasekera melanjutkan penelitian berbasis VERSAG dengan membuat *service provider* pada tahun 2010 (Gunasekera, et al., 2010). Secara umum sebuah *service* akan menyediakan atau mengolah suatu data, namun Gunasekera mengajukan *service provider* sebagai penyedia *behaviour* bagi MA. Gunasekera beranggapan (Gunasekera, et al., 2010) bahwa sebuah *service* akan lebih optimal jika yang di-*request* oleh *client* berupa *behaviour* untuk menjalankan kepentingan *client.* Umumnya ukuran *behaviour* lebih kecil dibandingkan dengan ukuran data. Hal ini dirasa lebih optimal dari pada mengirimkan data untuk diolah pada *service* konvensional.

Anggapan Gunasekera tersebut akan terjadi jika *node* memiliki sumber daya yang cukup untuk menjalankan *behaviour* dari MA. Namun kerugian dapat terjadi ketika sebuah *behaviour* dari MA yang mempunyai kompleksitas yang tinggi dieksekusi pada *node* yang memiliki sumber daya terbatas. Hal ini dapat terjadi karena MA mampu berjalan pada sumber daya yang heterogen, mulai dari yang mempunyai sumber daya yang melimpah, sampai sumber daya yang terbatas.



Gambar 1.1 Contoh Kasus MA yang Tidak Memperhitungkan Sumber Daya

Pada Gambar 1.1 diilustrasikan sebuah kasus MA berpindah dari **DEV00** ke **DEV01** untuk menjalankan *behaviour* **A** yang memiliki kompleksitas (W) 3 satuan, dan kebutuhan ruang penyimpanan (Q) sebesar 3 satuan. **DEV01** merupakan perangkat sumber daya rendah yang hanya mampu mengolah W=2 dan Q=2. *Behaviour* A tidak dapat dieksekusi oleh MA pada **DEV01** karena sumber

daya yang tidak cukup. Dalam masalah ini, kemampuan MA untuk mempertimbangkan ukuran sumber daya menjadi penting.

Permasalahan yang sejenis juga pernah dibahas pada penelitian Han (Han, et al., 2008). Han mengajukan sebuah *software partisioning* yang *responsive* terhadap ukuran sumber daya. *Software partisioning* adalah pemecahan *software* yang digunakan sebagai solusi atas sumber daya yang terbatas. *Responsive* adalah sifat adaptif terhadap ukuran sumber daya. Han berangkat dari domain *software* yang dapat dipecah menjadi komponen-komponen yang didistribusikan ke beberapa *node*, identik dengan MA pada VERSAG yang dipartisi menjadi MA dan *behaviour-behaviour*-nya. Hanya saja dalam uraian disertasi Gunasekera (Gunasekera, 2011), *behaviour-behaviour* MA dipartisi menjadi komponen yang tidak dapat dibagi lagi. Kondisi ini identik dengan yang dilakukan Han (Han, et al., 2008). Namun *software partisioning* yang diusulkan Han tidak bisa mengatasi sebuah *atomic software* yang memiliki bobot kompleksitas tinggi.

Dalam Tesis ini diajukan mekanisme efisiensi penggunaan dan pemilihan sumber daya pada konstruksi behaviour MA secara runtime di lingkungan heterogen. Dengan pertimbangan penggunaan dan pemilihan sumber daya tampaknya dapat mengurangi kegagalan dalam mengeksekusi behaviour yang disebabkan keterbatasan sumber daya pada lingkungan yang heterogen. MA dikembangkan menjadi aplikasi cerdas yang dapat berinteraksi dengan sumber daya dan beradaptasi dengan lingkungan operasi. Adaptasi dapat terjadi ketika MA berjalan pada node dengan sumber daya terbatas. Pemaksaan eksekusi behaviour pada node yang tidak memiliki sumber daya yang cukup dapat menjadi kerugian. Terlebih jika terjadi pada node dengan sumber daya yang sangat terbatas seperti WSN.

Hasil yang diharapkan dari pemilihan sumber daya untuk mengkonstruksi behaviour MA secara *runtime* adalah memberikan alternatif solusi atas ketidakcukupan sumber daya suatu *node* untuk mengeksekusi *behaviour.* Bentuk dari alternatif tersebut adalah pemilihan suatu *node* lain yang dianggap terbaik sebagai alternatif dari *node* asal yang tidak mampu mengeksekusi suatu *behaviour.*

Dengan memilih *node* terbaik diantara beberapa kandidat solusi akan memberikan keuntungan kepada MA dalam mengurangi waktu komputasi dari *behaviour* yang dimiliki.

1.2. Perumusan Masalah

Berdasarkan latar belakang yang telah disebutkan, maka rumusan masalah dalam penelitian ini dapat dijabarkan menjadi beberapa poin berikut ini:

1. Bagaimana mengatasi permasalahan kegagalan mengeksekusi *behaviour* MA yang dikonstruksi secara *runtime* pada *node* yang tidak memiliki sumber daya yang cukup.
2. Bagaimana menentukan *node* alternative terbaik sebagai lokasi eksekusi *behaviour* untuk mengurangi kegagalan eksekusi *behaviour.*
3. Bagaimana menentukan kriteria *node* alternative terbaik.
4. Bagaimana mengukur bobot dari *behaviour* dan sumber daya yang tersedia.

1.3. Tujuan

Tujuan dari tesis ini adalah mengembangkan mekanisme pemilihan sumber daya untuk konstruksi *runtime* MA yang dapat digunakan untuk mengurangi kegagalan dalam mengeksekusi suatu *behaviour* sebab kekurangan sumber pada lingkungan yang heterogen.

1.4. Manfaat

Dari penelitian ini diharapkan dapat meningkatkan kualitas konstruksi MA secara *runtime* dalam eksekusi *behaviour* pada lingkungan heterogen. Peningkatan kualitas ini dapat diindikasikan dengan berkurangnya kegagalan dalam mengeksekusi *behaviour* pada lingkungan heterogen.

1.5. Kontribusi Penelitian

Kontribusi dalam penelitian ini adalah mengembangkan mekanisme konstruksi *behaviour* MA secara *runtime* pada lingkungan heterogen dengan efisiensi penggunaan dan ketersediaan sumber daya.

1.6. Batasan Masalah

Penelitian ini menekankan pada mekanisme pemberian alternatif *computing node* kepada konstruksi *behaviour* MA secara *runtime* dan tidak membahas sekuritas dan otoritas penggunaan sumber daya.

1.7. Sistematika Penulisan

Pada Bagian ini dipaparkan secara singkat sistematika penulisan bab-bab yang terdapat pada laporan Tesis ini untuk memudahkan pembacaan. Sistematika tersebut disusun sebagai berikut:

1. Bab I Pendahuluan. Bab ini memaparkan pendahuluan yang menjelaskan latar belakang permasalahan, perumusan masalah, tujuan, manfaat, kontribusi penulisan, batasan masalah, dan sistematika penulisan.
2. Bab II Kajian Pustaka dan Dasar Teori. Bab ini memaparkan uraian yang meliputi teori-teori dasar dari *agent* dan MA berserta komponen dan fitur-fiturnya, konstruksi *behaviour* secara *runtime* yang telah diteliti sebelumnya, dan dasar teori yang terait dengan Tesis.
3. Bab III Metodologi Penelitian. Bab ini memaparkan tentang langkah-langkah penelitian yang telah dilakukan meliputi perumusan masalah, studi literature, desain kerangka kerja untuk solusi yang ditawarkan, mekanisme kerja dari kerangka kerja yang disusun, dan perancangan skenario pengujian MA pada kerangka kerja yang diusulkan.
4. Bab IV Hasil Penelitian dan Pembahasan. Bab ini memaparkan tentang tahapan pengujian dan pembahasan hasil yang dilakukan. Tahapan pengujian meliputi skenario pengujian, variabel pengujian, dan lingkungan pengujian. Pembahasan hasil pengujian meliputi analisis hasil yang didapat dari pengujian secara nyata.
5. Bab V Kesimpulan dan Saran. Bab ini memuat kesimpulan yang dapat diambil dari penelitian yang dilakukan, serta disampaikan juga hal-hal yang perlu dikembangkan dari penelitian yang telah dilakukan untuk penelitian lebih lanjut.

(Halaman ini sengaja dikosongkan)

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Pada Bab ini dijelaskan dasar teori dan kajian pustaka yang digunakan sebagai landasan ilmiah penelitian. Kajian pustaka yang dilakukan mencakup Agent dan MA pengukuran sumber daya, dan *multiple criteria decision making,* serta analisa regresi.

2.1. *Agent* dan *Mobile Agent*

Sebuah Agent didefinisikan (Siddiqui, et al., 2012) sebagai sebuah software yang independen yang dapat menjalankan kepentingan pengguna. Agent memiliki karakteristik kecerdasan dan kemampuan untuk belajar. Sebuah Agent dapat memanfaatkan sumber daya dari lingkungan tempat dia berjalan untuk menjalankan tugas yang diberikan. Sebuah MA memiliki tambahan kemampuan untuk bermigrasi ke banyak tempat dalam jaringan dengan menyimpan state dan membangkitkannya pada node yang baru. Karena dapat bermigrasi, maka sebuah MA dapat bekerja sesuai dengan kepentingan pengguna, seperti mengumpulkan informasi atau mengirimkan permintaan.

Sebuah MA adalah instansiasi dari sebuah mobile code dimana sebuah entitas software bermigrasi dari suatu node ke node lain dalam suatu jaringan dan melanjutkan eksekusi berdasarkan kehendaknya (Gunasekera, 2011). Kemampuan bermigrasi ini dapat memperbaiki produktifitas setiap node dan menciptakan lingkungan computing yang powerful. Infrastruktur MA mencakup protokol, aturan untuk perpindahan yang aman, dan direktori untuk menyimpan segala informasi tentang node.

2.1.1. Komponen MA

Komponen dari MA dijelaskan (Siddiqui, et al., 2012) sebagai berikut:

a. *Agent Code* merupakan algoritma untuk program MA. Sebuah MA setidaknya memiliki satu *agent code* yang menjadi struktur hidup dari MA. *Agent code*

dapat berupa kernel dari MA itu sendiri, maupun *agent code* dari *behaviour* MA.

b. *Agent execution thread:* kumpulan eksekusi yang harus dikerjakan oleh MA. *Agent execution thread*
c. *Agent data* merupakan nilai dari variabel global MA. Nilai ini akan dilanjutkan jika MA berpindah dari suatu *node* ke *node* lain.

2.1.2. Fitur MA

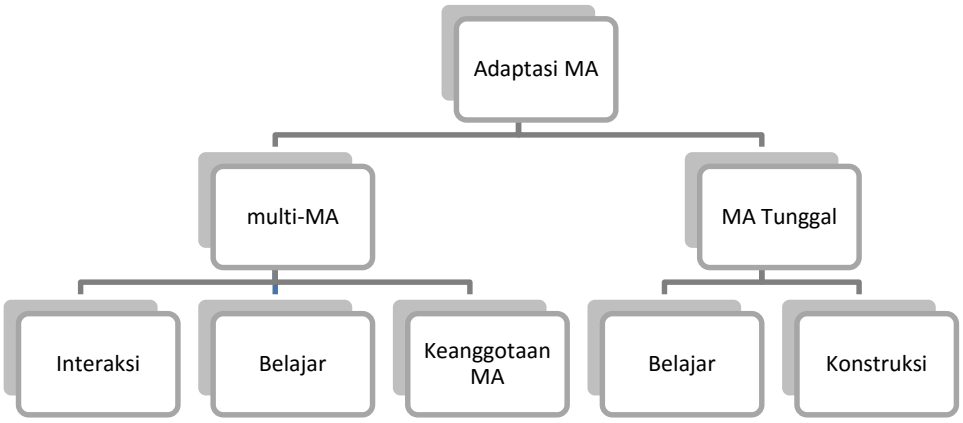
Siddiqui dalam publikasinya (Siddiqui, et al., 2012) menuliskan bahwa MA memiliki sifat khusus yang membedakan dari program konvensional. Beberapa sifat unik dari MA adalah sebagai berikut:

a. *Automonous* adalah kemampuan untuk bertindak tanpa arahan dari manusia. Sebagai contoh, pengguna hanya memerlukan spesifikasi dari sebuah tujuan yang umum dan *software agent* harus mempu membangun rencana dan memutuskan dalam rangka memenuhi tujuan yang diberikan.
b. *Temporal continuity* adalah fitur yang dapat membuat MA dapat menjadi proses yang berjalan secara kontinyu (meskipun dalam fase pasif atau pun aktif dalam *foreground*).
c. *Goal oriented* adalah sifat dari MA yang mampu untuk mengatasi atau menjalankan tugas untuk mencapat tujuan yang diinginkan.
d. *Mobility* adalah kemampuan untuk menjelajah dalam jaringan komputer.
e. *Communicative* adalah kemampuan untuk berkomunikasi dengan MA lain maupun manusia untuk bertukar informasi atau pun tindakan lain untuk memenuhi tujuannya.
f. *Collaborative* adalah kemampuan untuk sifat kooperasi sebuah MA seharusnya dapat menjalankan tugas yang membutuhkan kooperasi dengan MA yang lain untuk memenuhi suatu tujuan.
g. *Learning* adalah kemampuan dari MA yang mampu untuk mempelajari faktor lingkungan, pilihan pengguna, dll, dan mengembangkan sebuah *degree of reasoning* untuk membuat keputusan cerdas yang mampu meningkatkan efisiensi dalam sistem.

MA adalah program yang berbeda dari program konvensional. Sifat-sifat unik dari MA membuatnya memiliki kelebihan dan keterbatasan dibandingkan dengan program konvensional. MA dapat beroperasi pada lingkungan heterogen dan mampu berinteraksi dengan entitas yang memiliki sumber daya yang beragam. Kondisi tersebut menyebabkan sebuah MA harus memiliki kemampuan beradaptasi.

2.2. Konstruksi Behaviour Mobile Agent

Bedasarkan disertasi Gunasekera (Gunasekera, 2011), adaptasi dari suatu MA dapat dibagi seperti yang digambarkan pada Gambar 2.1. Adaptasi MA dapat dikelompokkan menjadi dua berdasarkan jumah MA yaitu MA tunggal dan multi-MA. MA tunggal memiliki adaptasi belajar dan konstruksi. Belajar adalah istilah yang digunakan untuk menunjukkan bahwa MA mampu memilih tindakan yang sesuai dengan kondisi lingkungan. Sedangkan konstruksi adalah kemampuan MA untuk mengubah susunan *behaviour* yang dimiliki untuk mendapatkan *behaviour* atau komponen yang baru. Sedangkan multi-MA memiliki adaptasi berupa interaksi antar MA, kemampuan belajar, dan mengubah keanggotaan dengan menambah atau menghapus MA.



Gambar 2.1 Hirarki Adaptasi MA

Biasanya sebuah MA didisain memiliki sekumpulan *behaviour* yang telah ditetapkan. *Behaviour* merupakan kemampuan dari suatu MA untuk melakukan tugas tertentu. Sehingga sebuah MA terbatasi melakukan tugas yang dingin dicapainya. Dalam prespektif yang lebih komplek, lingkungan yang dinamis,

seperti lingkungan heterogen pada lingkungan *pervasive*, sebuah MA dapat menjumpai situasi yang membutuhkan sebuah *behaviour* yang belum didefinisikan. Pada kasus seperti itu, sebuah MA akan diganti dengan MA lain yang memiliki fungsi yang lebih relevan. Itulah sebabnya adaptasi MA melalui konstruski penting untuk diterapkan.

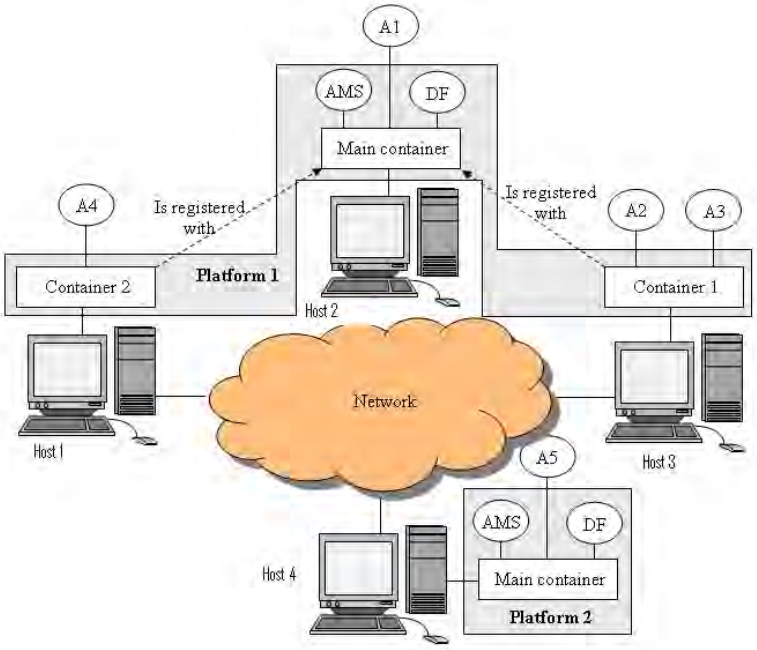
Metode adaptif konstruksi *behaviour* MA telah banyak dikembangkan, beberapa diantaranya adalah sebagai berikut:

a. Java Agent DEvelepment Framework

Java Agent DEvelopment Framework (JADE) (Bellifemine, et al., 2003) adalah sebuah *toolkit* MA yang dikembangkan oleh Telecom Italia. JADE merupakan sebuah *framework* yang dibangun untuk mengembangkan paradigm *agent.* Sebagai sebuah *middleware* yang memberikan kemudahan dalam membangun *agent,* JADE menyediakan beberapa beberapa fasilitas sebagai berikut:

- *Runtime environment* yang menjadi tempat di mana *agent* dapat berjalan dan aktif.
- Pustaka berupa kelas-kelas yang dapat digunakan untuk mengembangkan *agent.*

Seperti pada Gambar 2.2*, Runtime environment* dalam JADE dikenal dengan istilah *container*. Satu *host* dapat menjalankan lebih dari satu *container* dan setiap *container* bisa menangani beberapa *agent*. Sekumpulan *container* yang aktif disebut sebagai *platform*. Sebuah *platform* dapat memiliki *container* yang berasal dari *host* yang berbeda-beda. Satu *platform* harus memiliki satu *container* yang memiliki atribut sebagai *main container* yang aktif. Semua *container* yang aktif dan ingin bergabung dalam sebuah *platform* harus bergabung dengan mendaftarkan diri pada *main container* dan tidak boleh beratribut sebagai *main container* atau disebut juga normal container.



Gambar 2.2 Arsitektur JADE (jade.tilab.com)

Selain menerima pendaftaran dari normal *container* sebuah *main container* selalu memiliki dua buah *agent* yang aktif secara otomatis ketika main *container* dijalankan. Kedua agent itu adalah:

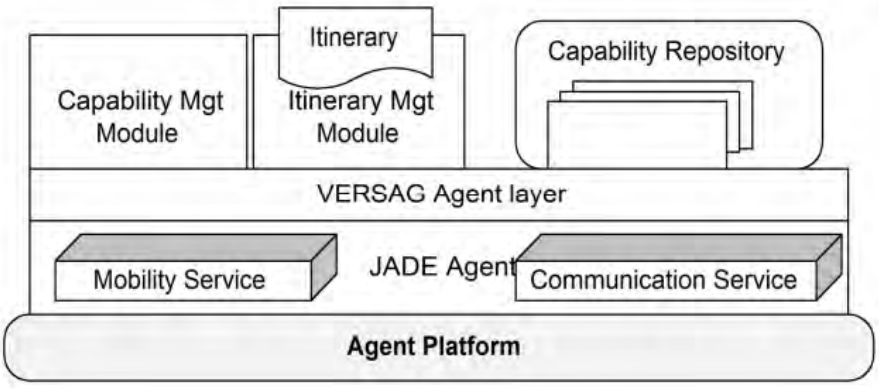
a) AMS (*Agent Management Sistem*) yang menyediakan *naming service* yang memastikan setiap *agent* dalam *platform* memiliki identitas yang unik. Selain itu AMS dapat merepresentasikan otoritas dalam *platform* di mana melalui AMS dapat menjalankan atau menghentikan *agent* dalam *container* yang terdaftar.
b) DF (*Directory Facilitator*) adalah *agent* yang berfungsi sebagai "*yellow pages*" bagi *platform*. Melalui DF sebuah *agent* dapat mencari *agent* yang aktif dan layanan yang diberikan *agent* tersebut.

Didalam JADE, sebuah fungsi MA direpresentasikan dengan sebuah *behaviour* yang mengembangkan kelas **Behaviour**. Dukungan terhadap konstruksi behaviour MA secara runtime disediakan melalui kelas **LoaderBehaviour**. Dengan menggunakan kelas tersebut, antar MA dapat bertukar pesan untuk me-load sebuah behaviour secara dinamis. *Behaviour* direpresentasikan sebagai *bytecode* yang dipertukarkan.

b. VERsatile Self-adaptive AGents

Gunasekera mengusulkan sebuah konsep (Gunasekera, 2011) yang dapat digunakan untuk mengatasi permasalahan. Dengan menggunakan konstruksi MA secara *runtime*. Sebuah MA di bangun tanpa memiliki *behaviour* untuk menjalankan tugasnya, namun pada saat beroperasi pada suatu lingkungan heterogen, MA tersebut dapat me-*load,* dan meng-*upload behaviour* sesuai dengan kebutuhan tujuan yang ingin dicapai. Untuk itu Gunasekera mengembangkan proyek VERSAG.

Proyek VERSAG dikembangkan untuk menjadi landasan konstruksi *behaviour* MA pada lingkungan heterogen. VERSAG merupakan arsitektur MA dimana MA didesain sebagai pembawa komponen *software.* Arsitektur VERSAG dapat dilihat pada Gambar 2.3. *Behavior* MA diimplemenatasikan menjadi komponen *software* yang portabel yang dapat berubah ketika *running.*



Gambar 2.3 Struktur VERSAG (Gunasekera, et al., 2008)

Kedua *framework* tersebut telah menjadi banyak landasan pengembangan MA. Namun tidak mempunyai kemampuan untuk mengenali dan memperhitungkan sumber daya yang ada sehingga dapat menyebabkan kegagalan dalam mengeksekusi *behaviour*. Pengenalan dan peritungan sumber daya ini menjadi penting seperti yang diuraikan pada latar belakang Bab 1.

2.3. Pengukuran Sumber Daya

Sumber daya merupakan segala sesuatu yang bersifat terbatas. Baik yang bersifat virtual maupun yang non-virtual. Sumber daya dalam penelitian ini

merupakan sumber daya yang berhubungan dengan konstruksi dan eksekusi *behaviour.* Pengukuran sumber daya telah menjadi bagian dari *high-performance computing* (HPC). Pengukuran ini perlu dilakukan untuk mengetahui kemampuan dari sumber daya sehingga pembagian tugas dapat berjalan sesuai dengan kapasitas sumber daya. Pengukuran ini biasanya digunakan untuk mendapatkan estimasi waktu komputasi, jumlah energi yang digunakan, dsb.

Sebuah penelitian telah dikemukakan oleh Choi (Choi, et al., 2014) menghitung estimasi waktu komputasi dan energi yang dibutuhkan berdasakan kondisi sumber daya yang ada dan bobot dari kompleksitas aritmatika yang dilakukan. Untuk mengukur estimasi waktu dan energi yang digunakan, Jee (Choi, et al., 2014) merumuskan beberapa atribut yang diperlukan untuk mengukur estimasi waktu dan energi yang digunakan pada Tabel 2.1.

Tabel 2.1 Variabel Pengukuran Ketersediaan Sumber Daya

variable	deskripsi
W	kompleksitas dari behaviour yang akan dieksekusi, (flops)
Q	transfer memori yang dilakukan, ("mops")
I	Intensitas ($\frac{W}{Q}$), (*flops per byte*)
τ_{flop}	waktu yang perlukan setiap statement, (detik per *flops*)
τ_{mem}	waktu per "mop"
$B\tau$	keseimbangan dalam waktu, $\frac{\tau_{flop}}{\tau_{mem}}$,
ϵ_{flop}	energi yang perlukan setiap statement, (joule per *flops*)
ϵ_{mem}	energi per "mop"
$B\epsilon$	keseimbangan dalam energi, $\frac{\epsilon_{flop}}{\epsilon_{mem}}$,
π_1	power yang digunakan oleh mesin, (watt)
$\Delta\pi$	selisih power yang digunakan oleh mesin ketika *idle* dan sedang bekerja (watt)
T	total waktu
E	total energi

Dari informasi tersebut disusun sebuah formula untuk pemodelan estimasi waktu komputasi dan energi yang dibutuhkan sebagai berikut:

Energi yang dibutuhkan:

$$E = E(W, I) = W\epsilon_{flop}\left(1 + \frac{B_\epsilon}{I}\right) + \pi_1 T\left(W, \frac{W}{I}\right) \qquad \text{(Persamaan 1)}$$

Waktu yang dibutuhkan:

$$T = T(W, Q) \equiv \max(W\tau_{flop}, Q\tau_{mem}, \frac{W\epsilon_{flop} + Q\epsilon_{mem}}{\Delta\pi}) \qquad \text{(Persamaan 2)}$$

2.4. Multiple Criteria Decision Making

Tabucanon (Tabucanon, 1988) mendefinisikan *Multiple Criteria Decision Making* (MCDM) sebagai metode pengambilan keputusan yang didasarkan pada teori, proses, dan metode analisis dengan mempertimbangkan beberbagai komponen ketidakpastian dari kriteria pengambilan keputusan. Proses pengambilan keputusan dilakukan dengan mempertimangkan alternatif-alternatif yang mungkin terjadi sehingga mampu menghasilkan keputusan terbaik dari himpunan alternatif berdasarkan berbagai kriteria. Kriteria dapat didenifisikan sebagai himpunan aturan yang digunakan sebagai dasar pokok pengambilan keputusan.

Dalam MCDM terdapat beberapa komponen yang terdiri dari, atribut, obyektif, dan tujuan.

a. Atribut menidentifikasi ciri kepada suatu entitas, misalnya besar memori, kecepatan I/O suatu media penyimpanan.
b. Obyektif menerangkan arah kecenderungan terhadap suatu atribut, misalnya meminimalkan memori, mempercepat penulisan, dsb.
c. Tujuan merupakan alternatif terbaik yang ingin dicapai, misalnya suatu proses mempunyai obyektif meminimalkan memori, maka tujuan yang ingin dicapai misalnya penggunaan memori dibawah 3Kb.

Kriteria merupakan standar, aturan, ataupun ukuran yang dijadikan pijakan suatu pengambilan keputusan. Pengambilan keputusan dilakukan dengan memformulasikan atribut, obyektif, maupun tujuan yang berbeda-beda. Dengan demikian, maka atribut, obyektif, maupun tujuan dianggap sebagai kriteria. Kriteria dibangun berdasakan kebutuhan, dan keinginan yang ingin dicapai.

Salah satu teknik MCDM adalah yang banyak digunakan adalah *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS) yang pertama kali diperkenalkan oleh Yoon dan Hwang pada tahun 1981. TOPSIS

didasarkan pada pemilihan alternatif yang dilihat dari jarak terpendek dari solusi ideal, namun memiliki jarak terpanjang dari solusi ideal negative dari sudut pandang geometris dengan menggunakan jarak *Euclidian* untuk menentukan kedekatan relatif dari suatu alternatif dengan solusi optimal. Solusi ideal positif didefinisikan sebagai jumlah dari seluruh nilai terbaik yang dapat dicapai untuk setiap atribut; sedangkan solusi negatif-ideal terdiri dari seluruh nilai terburuk yang dicapai untuk setiap atribut.

TOPSIS mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil jarak terhadap solusi ideal negative dengan mengambil kedekatan relative terhadap solusi ideal positif. Berdasarkan perbandingan terhadap jarak relatifnya, susunan popularitas alternatif bisa dicapai. Metode ini banyak digunakan pada beberapa model MCDM untuk menyelesaikan masalah dalam mengambil keputusan secara praktis.

Beberapa langkah dalam metode TOPSIS adalah sebagai berikut:

1. *Membuat matriks keputusan yang ternormalisasi.*
   Contoh dalam suatu pemilihan *node* komputasi tersedia 5 alternatif pilihan, N1, N2, N3, N4, dan N5. Pemilihan didasarkan pada *node* yang memiliki waktu komputasi C1 (keuntungan), energi C2 (keuntungan), ruang penyimpanan C3 (biaya), dan biaya jaringan C4 (biaya). Pemberian jenis kriteria berupa biaya bermaksud semakin tinggi nilai yang diberikan semakin baik, sedangkan keuntungan berarti sebaliknya. Contoh pembuatan matriks keputusan dapat dilihat pada Tabel 2.2.

Tabel 2.2 Contoh Matriks Keputusan

	C1	***C2***	***C3***	***C4***
N1				
N2				
N3				
N4				
N5				

2. *Membuat matriks keputusan yang ternormalisasi terbobot.*

Pada tahap ini matriks pada Table 2.2 diisi dengan menggunakan bobot yang telah ditetapkan. Contoh pemberian bobot matriks dapat dilihat pada Tabel 2.3.

- Sangat Baik = 5
- Baik = 4
- Cukup = 3
- Kurang = 2
- Buruk = 1

Tabel 2.3 Matriks Nilai Keputusan

	C1	***C2***	***C3***	***C4***
N1	3	1	4	3
N2	2	2	2	4
N3	4	5	4	1
N4	5	3	1	3
N5	3	3	3	3

Sedangkan untuk membuat matriks keputusan yang ternormalisasi berbobot dapat dihitung dengan menggunakan rumor berikut:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \qquad \text{(Persamaan 3)}$$

Contoh perhitungannya adalah sebagai berikut:

$x_1 = \sqrt{3^2 + 2^2 + 4^2 + 5^2 + 3^2} = 7.937$

$r_{11} = \frac{3}{7.937}$

sehingga didapatkan matriks baru seperti yang ada pada Tabel 2.3.

Tabel 2.4 Matriks Bobot

	C1	***C2***	***C3***	***C4***
N1	0.377964	0.144338	0.589768	0.452267
N2	0.251976	0.288675	0.294884	0.603023
N3	0.503953	0.721688	0.589768	0.150756
N4	0.629941	0.433013	0.147442	0.452267
N5	0.377964	0.433013	0.442326	0.452267

Setelah nilai **r** didapatkan langkah selanjutnya adalah dengan mengalikan nilai r yang ada pada Table 2.4 dan nilai bobot yang ada pada Table 2.3. Sehingga menghasilkan matriks keputusan ternormalisasi berbobot seperti yang ada pada Tabel 2.5.

Tabel 2.5 Matriks Keputusan Ternormalisasi Berbobot

	C1	***C2***	***C3***	***C4***
N1	1.133893	0.144338	2.359071	1.356801
N2	0.503953	0.57735	0.589768	2.412091
N3	2.015811	3.608439	2.359071	0.150756
N4	3.149704	1.299038	0.147442	1.356801
N5	1.133893	1.299038	1.326978	1.356801

3. *Menentukan matriks solusi ideal positif & matriks solusi ideal negatif.*

   Proses ini adalah menentukan matriks solusi ideal positif dan ideal negative dengan menggunakan persamaan berikut:

$$D_i^+ = \sqrt{\sum_{j=1}^{n}\left(y_i^+ - y_{ij}\right)^2} \qquad \text{(Persamaan 4)}$$

$$D_i^- = \sqrt{\sum_{j=1}^{n}\left(y_{ij} - y_i^-\right)^2} \qquad \text{(Persamaan 5)}$$

4. *Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan matriks solusi ideal negatif.*

   Menentukan jarak pada tahap ini dapat menggunakan pengukuran jarak Euclidian.

5. *Menentukan nilai preferensi untuk setiap alternatif.*

   Untuk menentukan nilai preferensi, data diukur dengan menggunakan persamaan 6 berikut:

$$V_i = \frac{D_i}{D_i^+ D_i^-} \qquad \text{(Persamaan 6)}$$

2.5. Analisis Regresi

Regresi adalah sebuah alat statistic yang memberikan penjelasan tentang pola hubungan (model) antara dua variabel atau lebih. Variabel dalam analisa regresi dibedakan menjadi dua jenis, yaitu:

a. Variabel Respon

Variabel Respon disebut juga variable dependen. Sebuah variable yang keberadaannya dipengaruhi oleh variable lainnya dan dinotasikan dengan variable **Y.**

b. Variabel Prediktor

Variabel Prediktor disebut juga dengan variable independen yaitu variabel yang bebas, tidak dipengaruhi oleh variabel lainnya, dan dinotasikan dengan **X**.

Regresi linear dibagi menjadi dua untuk membedakan jumlah variabel prediktornya. Berikut adalah pembagiannya.

a. Analisis Regresi Sederhana

Regresi linier sederhana digunakan untuk mendapatkan hubungan matematis dalam bentuk suatu persamaan antara **X** dengan variabel **Y**. Bentuk persamaan umum dari persamaan regresi linier untuk sebuah populasi ditunjukkan pada Persamaan 7. Untuk n data, nilai Y dihasilkan dari penjumlahan parameter *intercept* (a) dan perkalian antara koefisien regresi variabel bebas (b) dengan variabel bebasnya (X).

$$Y = a + bX \qquad \text{(Persamaan 7)}$$

Nilai-nilai (a) dan (b) dapat dihitung dengan menggunakan Persamaan 8 dan 9.

$$a = \frac{(\sum Y)(\sum X^2) - (\sum X)(\sum XY)}{n(\sum X^2) - (\sum X)^2} \qquad (Persamaan\ 8)$$

$$b = \frac{n(\sum XY) - (\sum X)(\sum Y)}{n(\sum X^2) - (\sum X)^2} \qquad (Persamaan\ 9)$$

b. Analisis Regresi Berganda

Analisis regresi berganda untuk mengukur pengaruh antara lebih dari satu variabel prediktor terhadap variabel bebas. Variabel prediktor disimbolkan dengan X dan variabel respon disimolkan dengan Y, sedangkan koefisien regresi dan

variabel *intercept* masing-masing disimbolkan dengan (a) dan (b). Persamaan yang digunakan untuk menghitung nilai respon ditunjukkan pada Persamaan 10.

$$Y = a + b_1X_1 + b_2X_2 + b_3X_3 + \cdots + b_nX_n \qquad (Persamaan\ 10)$$

Untuk mensimulasikan proses analisa regresi berganda, maka diangkat sebuah contoh untuk data kasus seperti yang ditunjukkan pada Tabel 2.6.

Tabel 2.6 Simulasi Data Kasus Analisa Regresi Linier Berganda

No	*X1*	*X2*	*Y*
1	10	7	23
2	2	3	7
3	4	2	15
4	6	4	17
5	8	6	23
6	7	5	22
7	4	3	10
8	6	3	14
9	7	4	20
10	6	3	19
Jumlah	**60**	**40**	**170**

Proses dalam analisis regresi linear terjadi dalam beberapa tahap sebagai berikut:

a. Menentukan Hipotesis

Hipotesis digunakan untuk memberi dugaan terhadap pengaruh variabel prediktor dan variabel respon. Hipotesis dalam analisa regresi berganda terdiri dari dua macam sebagai berikut:

H_o: $\beta_1 = \beta_2 = 0$, Variabel X_1, X_2 tidak berpengaruh terhadap variabel Y.

H_a: $\beta_1 \neq \beta_2 \neq 0$, Variabel X_1, X_2 berpengaruh signifikan terhadap variabel Y.

b. Memenuhi Persamaan 11-13

Persamaan umum yang digunakan untuk analisis regresi linier dipaparkan pada Persamaan 11-13. Untuk mempermudah menyelesaikan ketiga persamaan tersebut, maka disusunlan sebuah tabel bantu yang ditunjukkan pada Tabel 2.7.

$$\sum Y = an + b_1 \sum X_1 + b_2 \sum X_2 \quad \text{(Persamaan 11)}$$

$$\sum X_1 Y = a \sum X_1 + b_1 \sum X_1^2 + b_2 \sum X_1 X_2 \quad \text{(Persamaan 12)}$$

$$\sum X_2 Y = a \sum X_2 + b_2 \sum X_2^2 + b_1 \sum X_1 X_2 \quad \text{(Persamaan 13)}$$

Tabel 2.7 Tabel Bantu Analisis Regresi Linier Berganda

No	X_1	X_2	Y	X_1Y	X_2Y	X_1X_2	X_1^2	X_2^2
1	10	7	23	230	161	70	100	49
2	2	3	7	14	21	6	4	9
3	4	2	15	60	30	8	16	4
4	6	4	17	102	68	24	36	16
5	8	6	23	184	138	48	64	36
6	7	5	22	154	110	35	49	25
7	4	3	10	40	30	12	16	9
8	6	3	14	84	42	18	36	9
9	7	4	20	140	80	28	49	16
10	6	3	19	114	57	18	36	9
Jumlah	**60**	**40**	**170**	**1122**	**737**	**267**	**406**	**182**

Dari Persamaan 11-13, didapatkan hasil sebagai berikut:

170 = 10 a + 60 b_1 + 40 b_2……. (1)

1122 = 60 a + 406 b_1 + 267 b_2……. (2)

737 = 40 a +267 b_1 + 182 b_2……..…. (3)

Dengan menggunakan pemecahaan persamaan linier, maka didapat bahwa a=3.9186, sedangkan nilai dari b_1= 2.4909, dan nilai dari b_2 = -0.466. Sehingga didapat persamaan sebagai berikut:

$$Y = 3.9186 + 2.4909\, X_1 - 0.466\, X_2$$

BAB III
METODOLOGI PENELITIAN

Pada penelitian ini disusun sebuah metode untuk memilih *node* terbaik sebagai *node* untuk melakukan komputasi pada mekanisme konstruksi *behaviour* MA secara *runtime.* Untuk memfokuskan tujuan penelitian ini, maka disusun beberapa langkah penelitian sebagai berikut:

1. Perumusan Masalah
2. Studi Literatur
3. Pemetaan Skenario dan Konsep Kerja
4. Desain Framework dan Implementasi
5. Pengujian dan Evaluasi
6. Penyusunan Buku Tesis

3.1. Perumusan Masalah

Masalah yang dihadapi dalam penelitian ini adalah eksekusi *behaviour* yang atomik dari suatu konstruksi MA secara *runtime* ketika berada pada suatu *node* dengan sumber daya yang terbatas pada lingkungan yang heterogen. Berangkat dari kasus ini penulis mengemukakan sebuah ide untuk mengatasi permasalahan tersebut dengan cara memilih *node* lain yang memiliki sumber daya yang cukup untuk mengeksekusi *behaviour* tersebut.

Dari sini permasalahan dipecah menjadi beberapa pembagian yaitu pengukuran bobot *behaviour* dari MA, pengukuran sumber daya suatu *node*, dan mekanisme pemilihan *node* alternatif sebagai tempat eksekusi *behaviour.*

3.2. Studi Literatur

Tahapan ini merupakan tahapan pencarian referensi pembelajaran yang dapat dipertanggungjawabkan seperti referensi yang bersumber dari buku, jurnal. Pencarian referensi berguna untuk membangun pemikiran awal dan kerangka

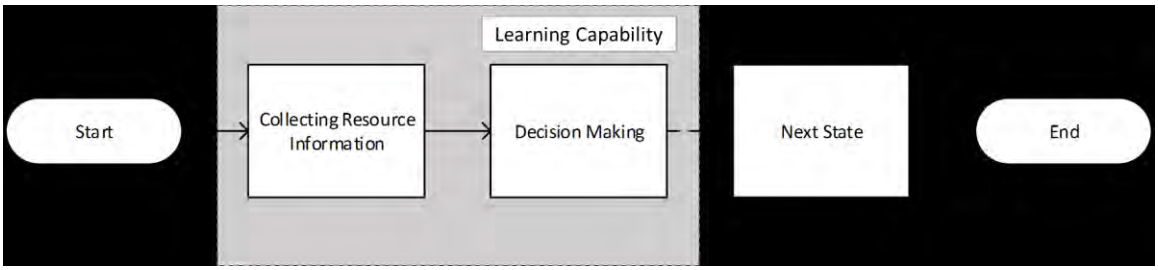
pengetahuan dari penelitian terdahulu yang terkait. Dalam tahap studi litelatur dikaji berbagai referensi yang berkaitan dengan MA dan perkembangannya, pengembangan sumber daya yang telah diteliti pada *pervasive computing,* mekanisme pengukuran bobot sumber daya terhadap *behaviour*, dan pengambilan keputusan berdasarkan hasil pengukuran sumber daya.

3.3. Pemetaan Skenario dan Konsep Kerja

Dalam penelitian ini disusun sebuah tahapan Pemetaan Skenario dan Konsep Kerja dalam bentuk transformasi permasalahan kedalam bentuk konsep solusi. Permasalahan-permasalahan yang mungkin muncul di transformasikan sebagai bahan masukan untuk menentukan konsep solusi. Dengan melakukan analisa dan pemetaan permasalahan menjadi skenario dan konsep serja diharapkan mampu mengurangi kasus-kasus yang mungkin luput pada tahap desain solusi dan implementasi.

Pada permasalahan yang diangkat, MA dapat mengalami kegagalan eksekusi disebabkan sumber daya yang terbatas. Hal tersebut diakibatkan oleh MA tidak memiliki pengetahuan tentang sumber daya yang dijadikan tujuan. Penelitian ini mengusulkan sebuah kemampuan belajar MA untuk dapat mengenali dan menghitung kemampuan sumber daya yang menjadi tujuannya. Dalam konsep *pervasive computing*, MA akan memiliki beberapa alternatife tujuan dan dapat memilih tujuan yang paling baik untuk melakukan tugasnya.

Kemampuan belajar itu dibagi menjadi menjadi dua fungsi yaitu mendapatkan informasi dari sumber daya dan membuat keputusan seperti yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Konsep Kontribusi

Fungsi untuk mendapatkan informasi dari sumber daya menghasilkan data yang diolah oleh fungsi pembuat keputusan untuk memilih *node* terbaik. Data-data yang dibutuhkan oleh fungsi pembuat keputusan didapatkan oleh fungsi ini. Data-data atau atribut yang diperlukan dirancang sebagai berikut:

1. Kesibukan node yang diindikasikan dengan derajat kesibukan CPU dan penggunaan RAM.
2. Waktu yang diperlukan oleh setiap node untuk mengeksekusi satu statemen.
3. Waktu yang diperlukan oleh setiap node untuk memindahkan memori dari memori lambat ke memori cepat.
4. Kompleksitas dari behavior yang akan dieksekusi.
5. Kecepatan pengiriman data dalam jaringan.

Sedangkan fungsi yang digunakan untuk mengambil keputusan adalah fungsi yang mampu menghasilkan perhitungan perkiraan waktu terkecil untuk dipilih sebagai node terbaik. Fungsi pengambil keputusan ini yang mampu mengarahkan MA untuk berpindah dari suatu *node* ke *node* yang lain.

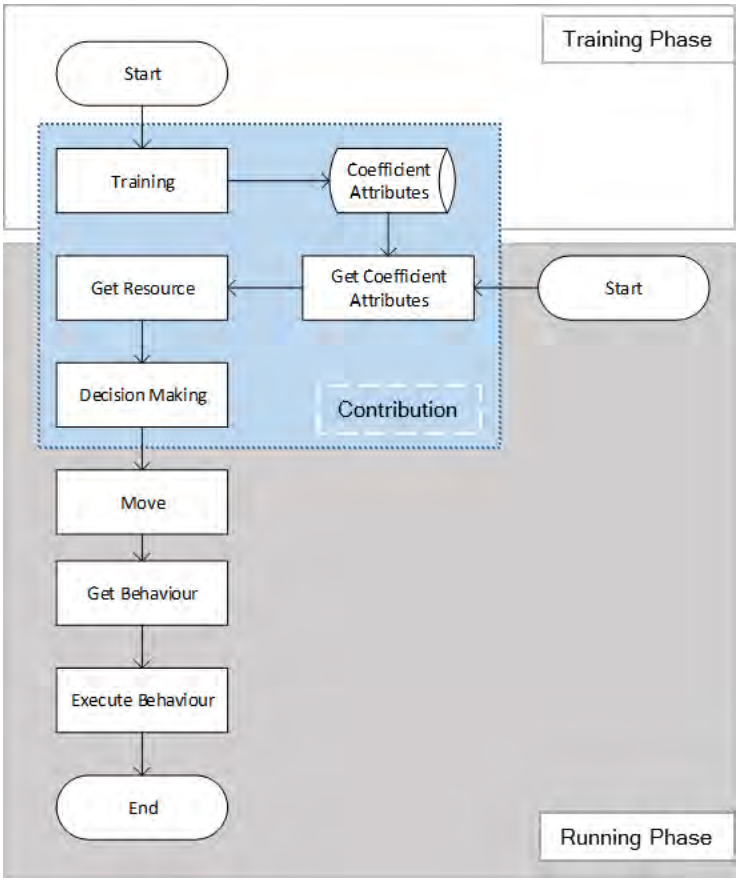
Untuk memperjelas peran dari konsep kedua fungsi diatas dipaparkan dalam Gambar 3.1 siklus hidup MA yang telah ditambah dengan usulan yang dibuat dalam penelitian ini. Tahap pertama dimulai dengan tahap *Training* MA dan tahap kedua adalah pengambilan keputusan oleh MA.

Pada gambar 3.2, MA yang dibangun oleh Gunasekera melalui VERSAG, memiliki siklus sebagai berikut:

1. **Start**

2. **Move**
3. **Get Behaviour**
4. **Execute Behaviour**
5. **End**

Start merupakahan tahapan *instance* dari MA. MA kemudian menentukan perpindahan berdasarkan tujuan yang direncanakan menggunakan fungsi **Move**. Dari *node* perpindahan tersebut, MA kemudian melakukan **Get Behaviour** untuk mendapatkan *behavior* dari tempat yang paling rendah biayanya. *Behaviour* yang didapatkan kemudian di eksekusi dengan menggunakan fungsi **Execute Behaviour**. Setelah selesai mengeksekusi *behavior*, maka berakhir siklus kerja dari MA.



Gambar 3.2 Siklus Hidup MA

Pada solusi yang diusulkan dalam Tesis ini, ditambahkan beberapa tahapan sebelum *running* dan saat *running*. Seperti yang ditunjukkan oleh Gambar 3.2, fase

sebelum *running* atau *training* dilakukan sekali sebelum fase *running*. Fase *training* terdiri dari beberapa tahap sebagai berikut:

1. Start
2. Training
3. Menghasilkan Koefisien Atribut

Fase *running* adalah fase dimana MA beroperasi pada lingkungan kerja. Pada tahapan ini, juga mengalami penambahan beberapa tahapan yaitu:

1. Mendapatkan Koefisien Regresi
2. Mendapatkan Nilai Atribut pada Sumber daya
3. Melakukan Decision Making

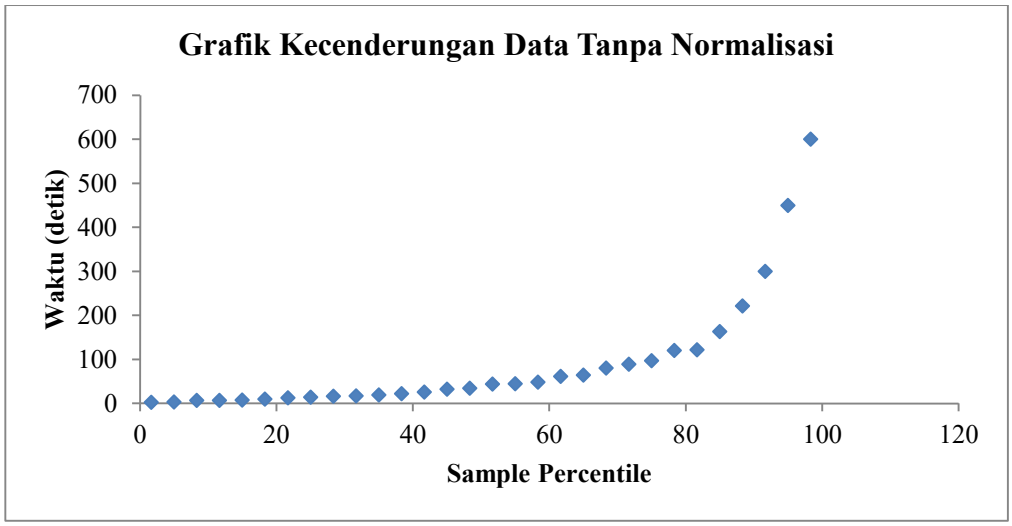
3.3.1. Fase Traning

Pada fase ini, dilakukan beberapa percobaan sederhana untuk mendapatkan sekumpulan data. Digunakan tiga buah jenis CPU (CPU01, CPU02, dan CPU03) dengan nilai atribut seperti yang dipaparkan pada Tabel 3.1. Pada pengambilan data pada tahap *Training* menggunakan sebanyak 30 kombinasi dari semua kasus yang mungkin terjadi untuk membuat pembelajaran yang lebih lengkap.

Dari data pada Tabel 3.1, kemudian dilakukan regresi untuk mendapatkan nilai seberapa besar atribut yang disusun berpengaruh pada waktu eksekusi MA. Hasil yang didapat dari fungsi regresi terlihat pada Gambar 3.3. Pada Gambar 3.3 terlihat bahwa hubungan data-data memiliki hubungan eksponensial dengan standar eror yang tinggi (52.2) seperti yang ditunjukkan oleh Tabel 3.2. Standar eror yang tinggi menunjukkan bahwa model yang dibangun memiliki simpangan eror baku yang tinggi, oleh karenanya model yang dibangun akan memiliki akurasi yang kurang baik. Parameter *Multiple R* adalah parameter yang menunjukkan kuatnya hubungan antara atribut yang diusulkan dengan waktu komputasi. Semakin besar nilai *Multiple R* maka model yang dibangun semakin konvergen.

Tabel 3.1 Tabel data training

NO	CPU	Tflop	Tmem	Data (Satuan)	Kesibukan (%)	Speed	Waktu (detik)
1				5	0.00	0.8	25.9
2				10	3.18	0.4	44.5
3				20	1.14	0.5	88.7
4				50	1.91	0.4	221.4
5	CPU01	8.36E-06	9.51E-06	100	0.78	3.9	449.9
6				5	100.00	3.1	43.7
7				10	99.28	3.2	64.3
8				20	98.14	5.4	120
9				50	82.13	8.3	299.6
10				100	83.83	18.9	599.9
11				5	6.91	0.7	6.9
12				10	6.67	0.1	9.6
13				20	6.79	0.1	19.2
14				50	6.36	0.1	48.2
15	CPU02	8.05E-07	7.86E-07	100	24.98	0	96.8
16				5	100.00	1.6	14.3
17				10	100.00	0	16.3
18				20	100.00	0.1	32.2
19				50	100.00	0.2	80.5
20				100	100.00	0.5	163.1
21				5	0.00	0.4	2.4
22				10	7.25	0	3.5
23				20	2.61	0	7.1
24				50	1.90	0	17.2
25	CPU03	2.89E-07	2.89E-07	100	0.59	0	34.4
26				5	100.00	1	7.9
27				10	100.00	0.3	12.4
28				20	100.00	0.1	22.4
29				50	100.00	0.6	61.6
30				100	100.00	1.8	121.6



Gambar 3.3 Kecenderungan atribut terhadap waktu eksekusi sebelum normalisasi

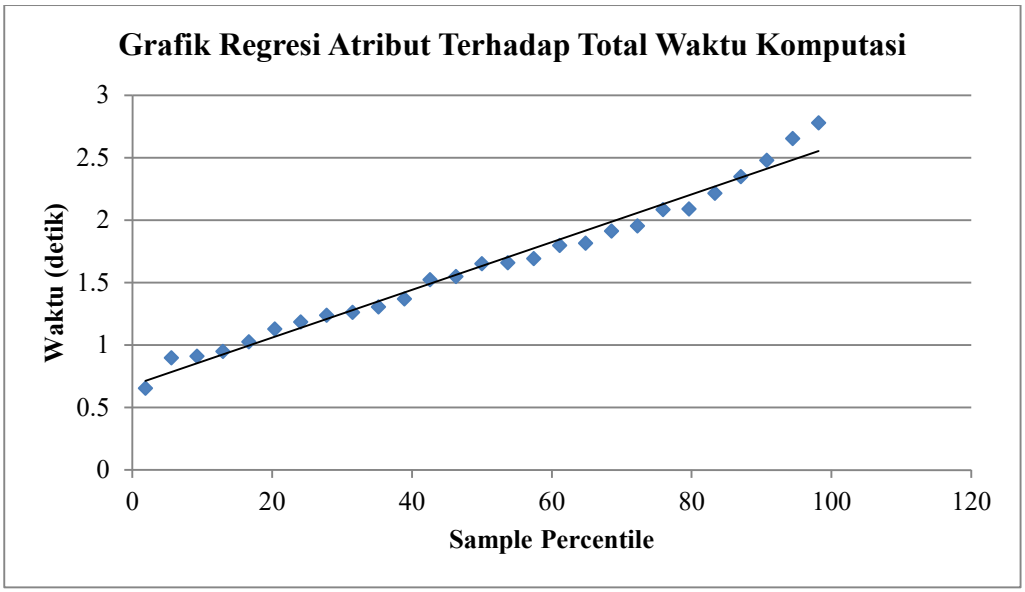
Data tersebut menunjukkan bahwa diperlukan sebuah normalisasi data eskponeksial untuk ditransformasikan menjadi bentuk linear dengan menggunakan fungsi logaritma.

Tabel 3.2 Statistika Regresi untuk Data sebelelum Normalisasi

Regression Statistics

Multiple R	0.938673
R Square	0.881106
Adjusted R Square	0.856337
Standard Error	**52.20935**
Observations	30

Dengan menggunakan normalisasi data training menggunakan fungsi logaritma didapatkan sebuah grafik linear seperti yang ditunjukkan pada Gambar 3.4 dengan standar eror yang cukup rendah, yakni sebesar 0.1 seperti yang ditunjukkan pada Tabel 3.3.



Gambar 3.4 Kecenderungan Data Ternormalisasi

Antara atribut atau variabel prediktor dan waktu komputasi atau variabel respon memiliki hubungan yang sangat signifikan. Hal tersebut didasari oleh tingginya nilai *Multiple R* (0.9984). Dengan tambahan dukungan dari standar eror yang cukup rendah, maka model yang dibangun sudah sesuai dengan kebutuhan yang diinginkan.

Tabel 3.3 Statistik Regresi Ternormalisasi

Regression Statistics	
Multiple R	**0.998420532**
R Square	0.996843558
Adjusted R Square	0.948472976
Standard Error	**0.109958533**
Observations	30

Tujuan dari analisa regresi berganda ini adalah untuk mengetahui koefisien regresi atau koefisien kontribusi untuk setiap atribut dalam menentukan besaran waktu komputasi yang diperlukan. Analisa regresi dengan standar eror yang rendah dan *Multiple R* yang tinggi mengindikasikan analisa regresi yang dilakukan sudah mampu menghasilkan nilai koefisien yang berakurasi tinggi.

Tabel 3.4 Koefisien Regresi

	Coefficients
Intercept	0
Tflop	**5130684.63**
Tmem	**-4305456.739**
Data (Satuan)	**0.946531331**
Kesibukan (%)	**0.242429482**
Memory	**-0.140105654**
Speed	**0.036401847**

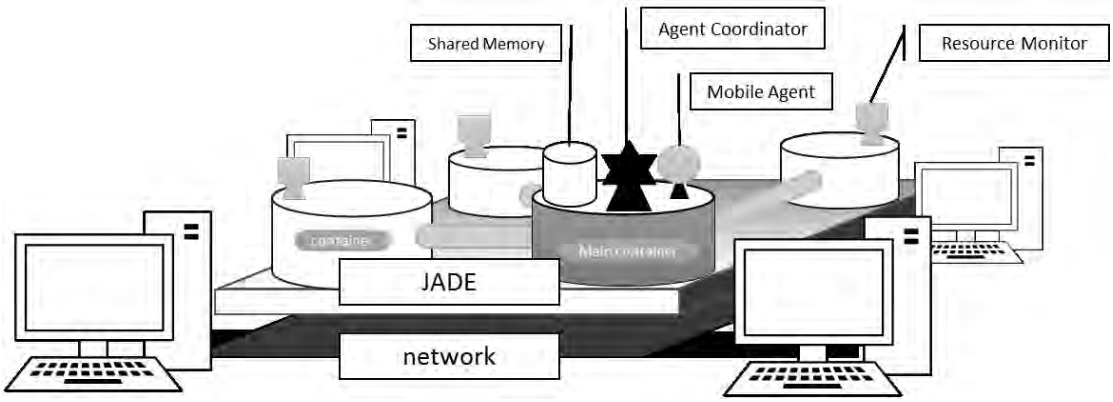
Pada data yang ditunjukkan oleh Tabel 3.4, masing-masing nilai koefisien dari masing-masing atribut adalah, Tflop=5130684,63, Tmem=-4305456.739, Ukuran data=0.946531331, Kesibukan=0.242429482, Memory = -0.140105654, Kecepatan transfer data=0.036401847. Dari data tersebut dapat disusun sebuah formula untuk menentukan besaran waktu komputasi yang diperlukan seperti yang ditunjukkan pada Persamaan 14.

$$Waktu = 5130684.63\ \text{Tflop} - 4305456.739\ \text{Tmem} + 0.946531331\ \text{Data} + 0.242429482\ \text{Kesibukan} - 0.140105654\ \text{Memory} + 0.036401847\ \text{Speed} \quad \text{(Persamaan 14)}$$

3.4. Desain Framework dan Implementasi

Kegagalan eksekusi *behaviour* karena keterbatasan sumber daya pada lingkungan heterogen menjadi permasalahan yang ingin diselesaikan dalam Tesis ini. Pada subbab ini akan dibahas perancangan dari framework yang diusulkan. Seperti yang diilustrasikan pada Gambar 3.5. Komponen utama yang disediakan adalah sebagai berikut:

1. Resource Monitor
2. Shared Memory
3. Mobile Agent



Gambar 3.5 Struktur dari Solusi yang Diusulkan

3.4.1. Resource Monitor

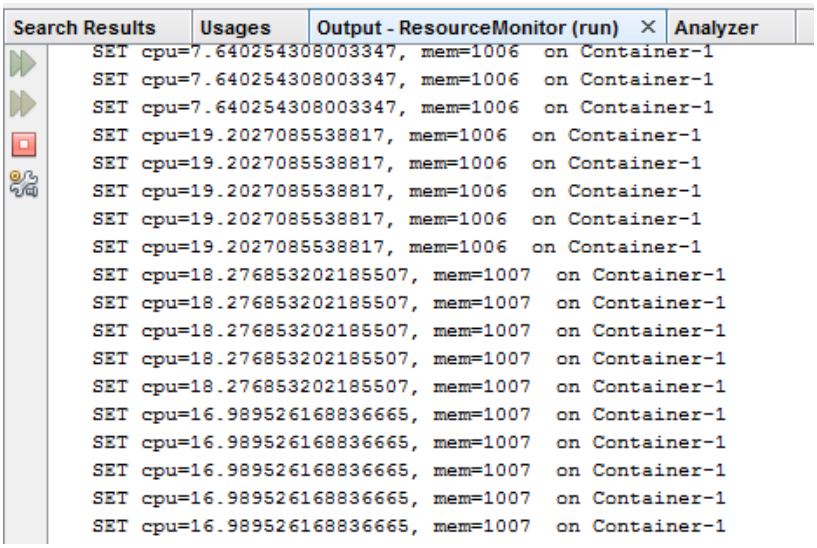
Resource Monitor adalah sebuah program yang digunakan untuk melakukan update informasi atau kondisi dari sumber daya yang ada. Setiap sumber daya memiliki sebuah Resource Monitor untuk selalu memberikan update informasi setiap 300ms.

```
JavaSysMon context = new JavaSysMon();
CpuTimes cpu_usa = context.cpuTimes();
String update = "";
OperatingSystemMXBean bean = (com.sun.management.OperatingSystemMXBean) ManagementFactory
                             .getOperatingSystemMXBean();
cpu = bean.getSystemCpuLoad()*100;
Memory memContext = new GlobalMemory();
memory = memContext.getAvailable() / 1000000 ;
```

Gambar 3.6 Potongan kode Resource Monitor

Seperti yang dicontohkan pada Gambar 3.7, Resource Monitor pada gambar tersebut memantau dua parameter yaitu CPU dan Memori pada Container-1. Hasil dari pemantauan ini kemudian dikirimkan kepada Shared Memory yang menjadi pengumpul informasi kondisi setiap sumber daya yang ada.

Diaplikasikan dengan menggunakan pustaka JavaSysMon (Java System Monitor), potongan kode dari program ini ditunjukkan pada Gambar 3.6.



Gambar 3.7 Reporting Resource oleh ResourceMonitor

3.4.2. Shared Memory

Shared Memory adalah sebuah modul yang menangani permasalahan penyimpanan data yang digunakan untuk penyimpanan *behavior* dan data-data penting yang diperlukan oleh MA. Fungsi dari *Shared Memory* ini adalah untuk memberikan informasi kepada MA segala informasi yang dibutuhkan oleh MA untuk menentukan keputusan. Pada Gambar 3.6 dicontohkan bagaimana *Shared Memory* yang dibangun menerima protocol update kondisi untuk sumber daya bernama **Container-1**.

id	con	cpu	mem
2016-01-07 10:38:25	Container-3	0	3,373
2016-01-14 14:31:57	Container-1	10	576
2016-01-14 14:32:26	Container-2	2.9	1,009

Gambar 3.8 Contoh Tampilan Shared Memory

Pada Gambar 3.8 ditunjukkan sebuah contoh data yang tersimpan pada Shared Memory. Terdapat asal sumber daya, id yang merupakan waktu pengambilan data, dan beberapa data yang menunjukkan kondisi sumber daya.

3.4.3. Mobile Agent

Mobile Agent atau MA memiliki kecerdasan untuk menentukan *node* yang paling baik untuk mengeksekusi behavior yang dimilikinya. Dengan kecerdasan berupa MCDM, koefisien yang didapat pada fase training, dan data kondisi sumber daya yang ada, MA dapat memberikan perangkingan berdasarkan urutan waktu eksekusi paling rendah.

3.5. Pengujian dan Evaluasi

Pengujian dan Evaluasi digunakan untuk mendapatkan data empirik untuk menjawab rumusan masalah dan membahas berbagai informasi yang berguna bagi penelitian ini maupun penelitian pada masa depan.

3.5.1. Skenario Pengujian

Pengujian yang dilakukan untuk membuktikan atau memvalidasi dua hal. Validasi yang pertama adalah validasi internal. Validasi internal adalah validasi yang dilakukan untuk menjawab hipotesan dan rumusan masalah yang telah dikemukakan. Validasi yang kedua adalah validasi eksternal. Validasi eksternal dilakukan untuk menunjukkan keumuman konsep yang diajukan untuk memecahkan permasalahan lain yang sejenis. Skenario pengujian disusun untuk mevalidasi kedua hal tersbut.

Skenario pengujian pada Tesis ini dibagi menjadi beberapa skenario sebagai berikut:

1. Waktu Eksekusi

   Skenario pengujian waktu eksekusi adalah pengujian yang digunakan untuk mengetahui durasi eksekusi *behavior* pada *node* yang terbaik menurut MA. Tujuan dari pengujian ini adalah MA memilih *node* yang mampu menyelesaikan *behavior* dengan waktu paling kecil.
2. Flowtime

   Menurut Xhaf dan Abraham (Xhafa, et. al 2010), *flowtime* mengacu pada waktu respon eksekusi bersamaan yang terkait dengan penjadualan. Dalam kaitannya

dengan metode yang diusulkan, *flowtime* merupakan nilai varian maksimum dari eksekusi *behavior* dari sejumlah MA. Sehingga dengan rekayasa mengurangi nilai *flowtime* akan mengurangi rata-rata waktu total eksekusi.

3. Makespan

   Makespan adalah waktu maksimal yang dihasilkan oleh MA dari sejumlah MA yang dijalankan untuk menyelesaikan *behavior* MA. Makespan menunjukkan lama waktu yang dibutuhkan untuk menyelesaikan seluruh *behavior* dari semua MA.

3.5.2. Evaluasi Pengujian

Evaluasi pengujian adalah tahapan untuk mencari fakta-fakta dari pengujian untuk menjawab rumusan masalah yang dikemukakan pada Bab I dan untuk menggali informasi yang berguna untuk penelitian pada masa depan dengan topik yang berhubungan.

3.6. Penyusunan Buku Tesis

Penyusunan buku Tesis dilakukan sebagai dokumentasi terhadap serangkaian penelitian yang dikerjakan agar dapat dijadikan sebagai bahan pembelajaran, referensi maupun sebuah perbaikan penelitian pada masa depan yang berhubungan efisiensi penggunaan dan pemilihan sumber daya untuk mekenisme konstruksi *behavior* MA maupun Topik yang sejenis atau berkaitan.

[Halaman ini sengaja dikosongkan]

BAB IV
UJI COBA DAN PEMBAHASAN

Pengujian dilakukan untuk membuktikan kebenaran hipotesa dan untuk menjawab pertanyaan penelitian yang diuraikan pada Bab II.

4.1. Langkah-langkah Ujicoba

Tahapan uji coba pada penelitian ini meliputi penentuan skenario pengujian, variable pengujian, lingkungan pengujian, dan perbandingan dengan metode konvensional. Tahap pertama menentukan kasus ujicoba yang dilakukan dan parameter yang ditentukan. Tahap kedua adalah menentukan lingkungan pengujian yang membahas topilogi, dan spesifikasi dari sumber daya yang digunakan. Tahap terakhir adalah pengujian dan perbandingan dengan metoda yang telah ada.

4.1.1. Lingkungan Pengujian

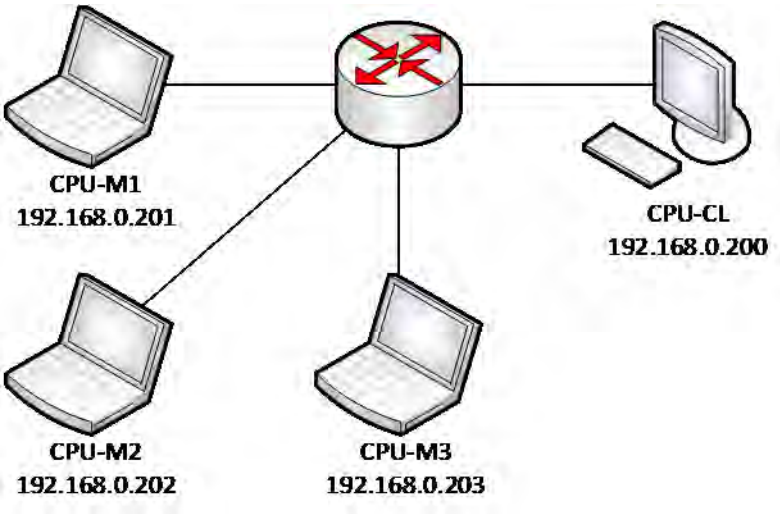
Pengujian dilakukan pada lingkungan yang nyata dengan menggunakan beberapa perangkat yang terdiri dari empat buah komputer dan sebuah *router* untuk menghubungkan keempat komputer dalam satu jaringan.

1. Tiga buah komputer (CPU-M1, CPU-M2, dan CPU-M3) yang memiliki spesifikasi yang berbeda untuk mewakili *node* yang memiliki sumber daya rendah, sedang, dan tinggi, serta sebuah komputer (CPU-C1) yang mewakili asal dari MA. Spesifikasi dari keempat komputer dapat dilihat pada Tabel 4.1.

Table 4.1 Spesifikasi Komputer Pengujian

No	*Kode*	*CPU*	*RAM*
1	CPU-M1	Intel Atom 1.5 GHz	1GB
2	CPU-M2	Intel Core i3 1.4 GHz	4GB
3	CPU-M3	Intel Core i7 2.8 Ghz	6GB
4	CPU-C1	Intel Core i7 2.4 Ghz	8GB

2. Sebuah *router* untuk menghubungkan keempat komputer yang digunakan. *Router* ini memiliki akses *wifi* dan LAN.
3. Semua komputer tersebut memiliki sistem operasi yang sama yaitu Windows yang telah terpasang *framework* Java, dan JADE.
4. Topologi Jaringan yang digunakan dapat dilihat pada Gambar 4.1.



Gambar 4.1 Topologi jaringan pengujian.

4.1.2. Variabel Uji Coba

Variabel yang digunakan dalam pengujian adalah sebagai berikut:

1. Ukuran sumber daya

   Ukuran sumber daya menjelaskan kemampuan sumber daya dalam melakukan operasi. Ukuran sumber daya meliputi waktu yang diperlukan oleh sumber daya untuk mengekeskusi sebuah operasi.

2. Ukuran data

   Ukuran data menunjukkan ukuran gambar yang digunakan. Pada Tesis ini digunakan tiga jenis ukuran citra yakni, 100x100, 300x300, dan 500x500 piksel.

3. Jumlah Job

   Jumlah Job menunjukkan seberapa sering algoritma SURF dilakukan.

4. Jumlah MA

   Jumlah MA merupakan jumlah MA atau worker yang dijalankan.

4.1.3. Skenario Pengujian

Kasus yang diangkat dalam pengujian ini adalah *template matching* dengan mengunakan algoritma SURF (*Speeded-Up Robush Feature*) yang diimplementasi dengan menggunakan pustaka BootCV.

Skenario pengujian melibatkan node yang mewakili *low* sumber daya, medium sumber daya, dan *high* sumber daya. Pada Gambar 4.1 diilustrasikan

topologi pengujian yang akan digunakan. Sebuah MA akan dikirimkan dari Agent Repository untuk mengeksekusi beberapa *behaviour* pada *node-node* yang ada.

4.2. Hasil dan Pembahasan

Tujuan uji coba dalam penelitian adalah untuk mengetahui seberapa efektif MA dalam memilih sumber daya terbaik sebagai *node* komputasi. Pemilihan *node* komputasi ini didasarkan pada pengetahuan dari MA dalam fase *training.* Seluruh hasil pengujian didapatkan dengan cara mengolah data yang dihasilkan oleh implementasi *framework* JADE pada lingkungan pengujian yang nyata. Data tersebut kemudian diolah dan disajikan dalam bentuk table atau grafik untuk mempermudah analisis.

4.2.1. Analisis Parameter Waktu Eksekusi

Skenario pengujian waktu eksekusi adalah pengujian yang digunakan untuk mengetahui durasi eksekusi *behavior* pada *node* yang terbaik menurut MA. Tujuan dari pengujian ini adalah MA memilih node yang mampu menyelesaikan *behavior* dengan waktu paling rendah.

Tabel 4.1 Hasil Pengujian Waktu Eksekusi

	Kondisi Node						Node Terbaik	Pengujian Total		
No	CPU-M1		CPU-M2		CPU-M3					
1	**N**	**B**	**N**	**B**	**N**	**B**		CPU-M1	CPU-M2	CPU-M3
2	10	Normal	10	Normal	10	Normal	**CPU-M3**	44.5	6.9	**3.5**
3	10	Tinggi	10	Tinggi	10	Tinggi	**CPU-M3**	64.5	14.3	**7.9**
4	50	Normal	50	Normal	50	Normal	**CPU-M3**	221.4	48.2	**3.5**
5	50	Tinggi	50	Tinggi	50	Tinggi	**CPU-M3**	299.6	80.5	**61.6**
6	50	Normal	50	Normal	50	Tinggi	**CPU-M2**	221.4	**48**	61.6
7	50	Normal	50	Tinggi	50	Tinggi	**CPU-M3**	221.4	80.5	**61.6**

Pengujian dilakukan dengan menjalankan kecerdasan MA untuk memilih *node* komputasi terbaik dari *node* yang ada. Setiap *node* mengirimkan informasi

mengenai kondisi saat ini berupa derajat kesibukan yang dialami dengan menggunakan program **Resource Monitor**. Hasil yang didapat oleh **Resource Monitor** akan dikirimkan ke **Shared Memory.**

MA menggunakan data yang tersimpan pada **Shared Memory** untuk diolah dan diputuskan menggunakan teknik MCDM. *Node* terbaik dari hasil pemilihan keputusan tersebut akan menjadi destinasi dari MA selanjutnya.

Pada pengujian ini disusun dua variable untuk menguji kemampuan dari MA dalam menentukan *node* terbaik. Variabel yang pertama adalah **N** yang merupakan jumlah job yang akan dieksekusi. Jumlah job ini memiliki dua varian yaitu 10 dan 50. Variable yang kedua adalah B yang merupakan derajat kesibukan dari *node.* Variable *d*erajat kesibukan memiliki dua varian, yaitu normal dan tinggi. Untuk membuat *node* mempunyai derajat kesibukan yang tinggi dilakukan *Stress Test* dengan menggunakan aplikasi bernama Prime95.

Dari hasil pengujian yang dipaparkan pada Tabel 4.1, terlihat bahwa MA mampu memilih *node* dengan waktu terbaik untuk mengeksekusi *behavior*. Meskipun CPU-M3 memiliki sumber daya yang paling besar, namun pada percobaan ke 6, terlihat bahwa MA lebih memilih menggunakan *node* dengan kode CPU-M2. Hal ini disebabkan oleh scenario pengujian ke-6, *node* dengan kode CPU-M3 mengalami kesibukan yang tinggi.

4.2.2. Analisis Parameter Flowtime

Flowtime merupakan nilai varian dari eksekusi *behavior* dari sejumlah MA. Nilai *flowtime* ini menunjukkan keragaman setiap MA dalam menjalankan *behavior.* Semakin kecil nilai *flowtime* maka akan semakin baik untuk komputasi yang dilakukan oleh multi MA. Namun jika nilai *flowtime* besar, maka hal tersebut masih perlu ditingkatkan. Sehingga dengan rekayasa mengurangi nilai flowtime akan mengurangi rata-rata waktu total eksekusi. Nilai *flowtime* akan berdampak pada kualitas dari keseimbangan waktu kerja dari multi MA.

Pada pengujian pengukuran parameter *flowtime,* dijalankan multi MA untuk menjalankan satu atau beberapa *behavior* atau job. Pengujian ini menggunakan tiga variable yang berbeda.

a. Job merupaan banyaknya satuan tugas yang harus dilakukan oleh MA pada *behavior.* Satuan tugas ini besarannya adalah satu kali koputasi algoritma SURF oleh pustaka BootCV.
b. Data merupakan ukuran citra yang digunakan. Ukuran citra pada pengujian ini bervariasi dari 100x100 piksel, 300x300 piksel, dan 500x500 piksel.
c. Worker merupakan sebutan untuk MA dalam konteks multi MA.

Pengujian ini menggunakan pembanding dari metode **Round Robin.**

Tabel 4.2 Hasil Pengujian Flowtime

Job	Data	Worker	Usulan Metode	Round Robin
			flowtime	flowtime
1	100	3	0.016	3.283
		6	0.471	4.052
		9	0.085	7.273
		12	0.064	9.45
		15	0.085	52.646
		18	0.167	14.978
	300	3	0.036	2.648
		6	1.319	8.444
		9	2.301	13.603
		12	5.816	20.142
		15	12.918	24.139
		18	11.208	30.489
	500	3	0.512	5.112
		6	5.736	15.358
		9	48.401	66.615
		12	11.167	167.128
		15	58	101.52
		18	56.272	98.615
20	100	3	0.016	42.323
		6	0.031	5.195
		9	0.038	8.143
		12	0.743	52.954
		15	11.851	16.092
		18	15.239	18.788
	300	3	1.332	7.736
		6	2.811	51.94
		9	6.249	18.874

		12	10.661	24.114
		15	10.064	69.799
		18	19.221	36.775
	500	3	2.474	12.869
		6	6.031	60.065
		9	12.398	72.521
		12	58.92	84.356
		15	18.063	95.31
		18	73.012	4712.021

Hasil pengujian ini dipaparkan pada Tabel 4.2. Dari Tabel 4.2, data yang dihasilkan dari pengujian memiliki kecenderungan metode yang diusulkan lebih baik dibandingkan dengan metode Round Robin. Untuk lebih rinci dalam melakukan pengamatan maka pengujian ini dibagi menjadi enam skenario sebagai berikut:

a. Jumlah Job 1 dan ukuran data 100

   Pengujian ini menggunakan jumlah Job sebanyak 1 dan ukuran data sebesar 100x100 piksel.

b. Jumlah Job 1 dan ukuran data 300

   Pengujian ini menggunakan jumlah Job sebanyak 1 dan ukuran data sebesar 300x300 piksel.

c. Jumlah Job 1 dan ukuran data 500

   Pengujian ini menggunakan jumlah Job sebanyak 1 dan ukuran data sebesar 500x500 piksel.

d. Jumlah Job 20 dan ukuran data 100

   Pengujian ini menggunakan jumlah Job sebanyak 20 dan ukuran data sebesar 100x100 piksel.

e. Jumlah Job 20 dan ukuran data 300

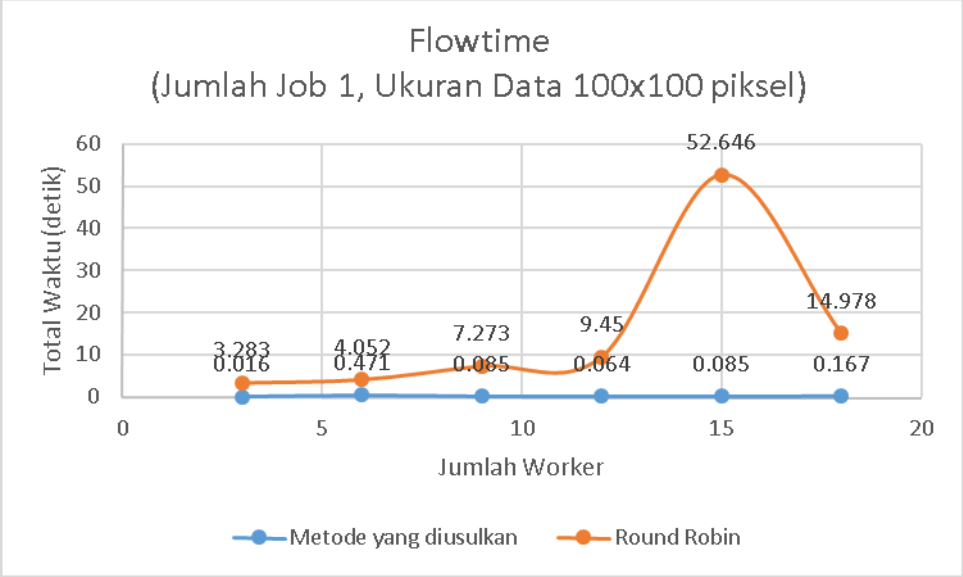
   Pengujian ini menggunakan jumlah Job sebanyak 20 dan ukuran data sebesar 300x300 piksel.

f. Jumlah Job 20 dan ukuran data 500

Pengujian ini menggunakan jumlah Job sebanyak 20 dan ukuran data sebesar 500x500 piksel.

4.2.2.1. Skenario Job=1 dan Data=100

Pada skenario pengujian dengan menggunakan 1 Job dan ukuran data 100x100 piksel didapatkan data seperti yang tergambar pada Gambar 4.2. Pada Gambar tersebut, metode yang diusulkan memiliki nilai flowtime yang cenderung lebih stabil dan terhadap perubahan jumlah *worker*. Hal ini berbeda dengan apa yang diperlihatkan oleh hasil pengujian dengan menggunakan Round Robin. Pengujian dengan menggunakan metode pemilihan *node* menggunakan Round Robin, memiliki nilai yang paling tinggi pada jumlah worker 15.



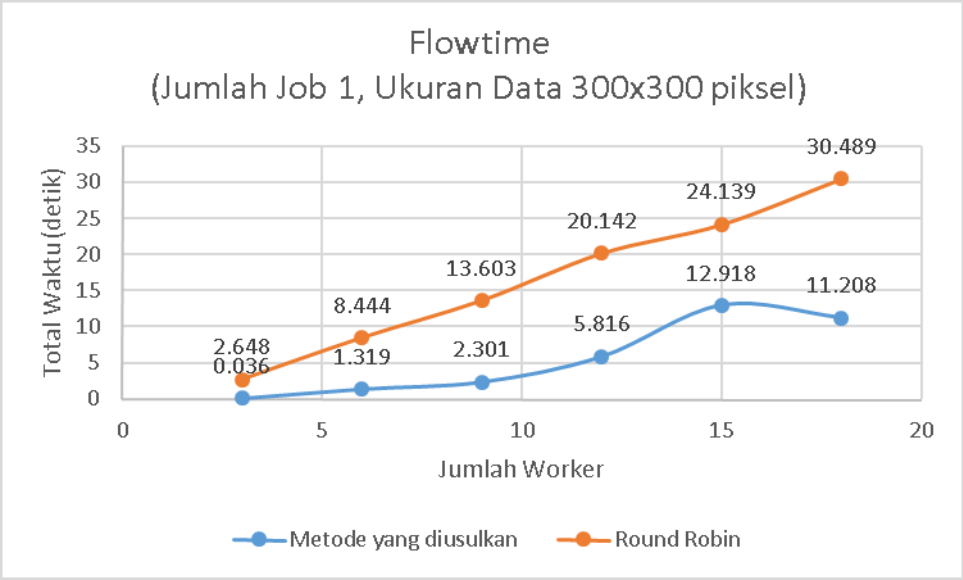
Gambar 4.2 Grafik Flowtime Job=1 dan Data=100x100piksel

Pada skenario pengujian Job=1,d an Data=100, metode yang diusulkan memiliki waktu *flowtime* yang lebih baik dibandingkan dengan metode Round Robin. Nilai yang didapat oleh metode yang diusulkan memiliki kecenderungan yang stabil, namun hal ini berbeda dengan hasil yang ditunjukkan oleh metode Round Robin.

4.2.2.2.Skenario Job=1 dan Data=300

Skenario kedua dari pengujian *Flowtime* adalah dengan menggunakan jumlah Job 1 dan ukuran Data 300x300 piksel. Hasil yang didapat dari pengujian

dengan menggunakan skenario ini ditunjukkan pada Gambar 4.3. Pada Gambar tersebut, metode yang diusulkan masih memiliki nilai yang lebih baik dibandingkan dengan hasil yang diperoleh dari metode Round Robin. Kedua hasil memili kecenderungan memiliki nilai *flowtime* yang meningkat searah dengan pertambahan *worker* yang bekerja. Selain data pada *worker* ke 18, pada metode yang diusulkan, seluruh data bergerak naik searah dengan pertambahan jumlah *worker*.



Gambar 4.3 Grafik Flowtime Job=1 dan Data=300x300piksel

Dari pengujian pada skenario ini, ditunjukakan bahwa jumlah worker mempengaruhi nilai dari *flowtime*. Namun terjadi ketidakstabilan pada data worker ke 18 pada metode yang diusulkan.

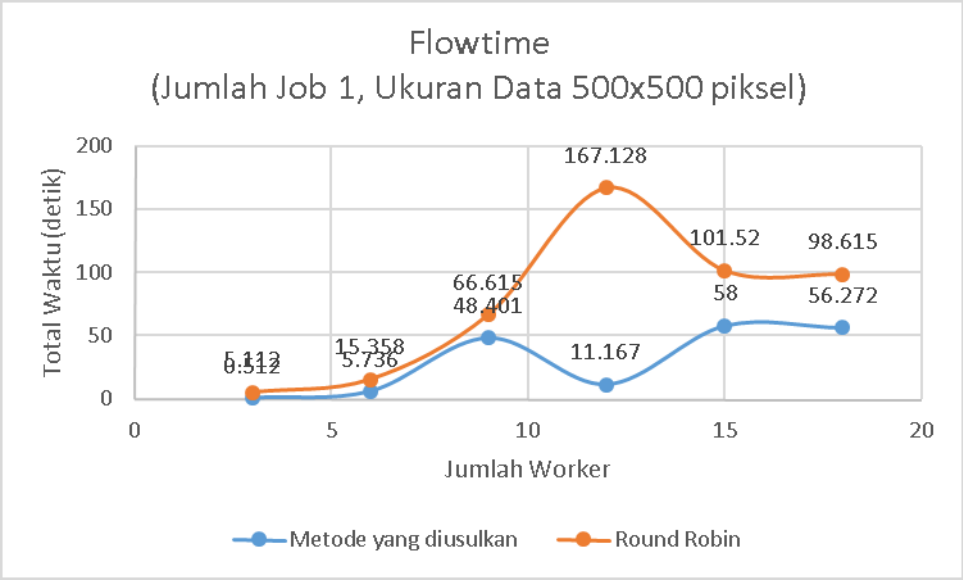
4.2.2.3.Skenario Job=1 dan Data=500

Skenario selanjutnya adalah Job 1 dan Data 500 yang menunjukkan jumlah Job yang dieksekusi berjumlah 1 dan menggunakan Data sebesar 500x500 piksel. Hasil Pengujian yang dilakukan pada skenario ketiga menunjukkan bahwa metode yang diusulkan memberikan data yang lebih baik dibandingkan dengan hasil yang diperlihatkan oleh metode Round Robin.

Kedua data mengalami ketidaklurusan pada worker ke 12. Metode Round Robin menghasilkan nilai yang paling tinggi pada titik ke worker 12. Sedangakn

metode yang diusulkan memiliki nilai yang malah menurun pada titik worker yang sama.

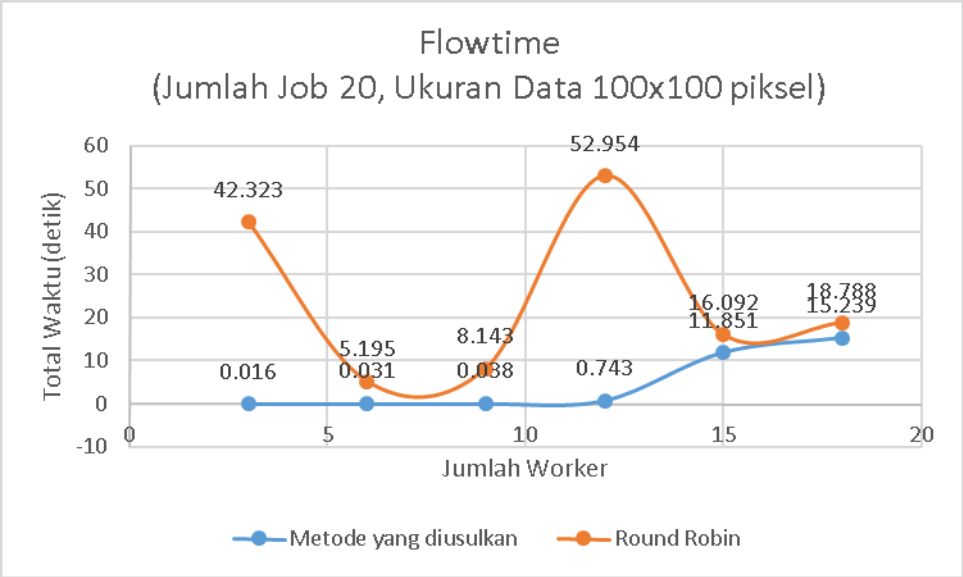
Selain data pada titik worker ke 12, seluruh data mengalami peningkatan seiring dengan peningkatan jumlah worker seperti yang ditunjukkan pada Gambar 4.4



Gambar 4.4 Grafik Flowtime Job=1 dan Data=500x500piksel

4.2.2.4.Skenario Job=20 dan Data=100

Skenario keempat pada pengujian *flowtime* mengalami peningkatan pada jumlah Job. Jumlah job pada skenario ini meningkat dari 1 menjadi 20 Job. Hal ini menunjukkan banyaknya algoritma SURF dengan menggunakan pustaka BootCV dieksekusi. Pada skenario pengujian ini, jumlah data kembali ke 100 piksel persegi.

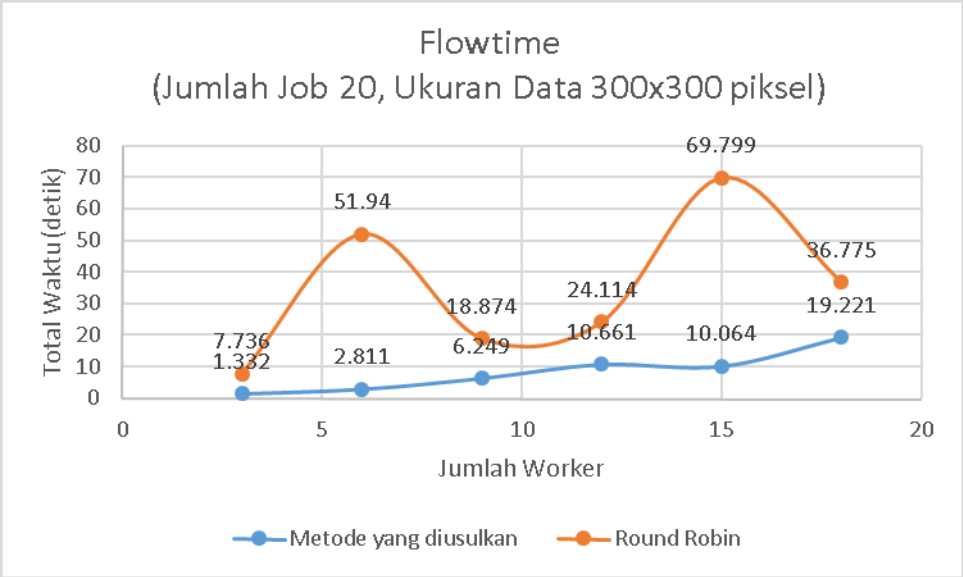


Gambar 4.5 Grafik Flowtime Job=20 dan Data=100x100piksel

Dari hasil yang ditunjukkan pada Gambar 4.5, ditunjukkan nilai yang tidak stabil terjadi pada hasil yang diperoleh dari metode Round Robin. Sedangkan pada metode yang diusulkan mengalami kenaikan yang cukup signifikan pada titik worker ke 15.

4.2.2.5.Skenario Job=20 dan Data=300

Skenario kelima dari pengujian *flowtime* adalah dengan meningkatkan ukuran data menjadi 300 dari skenario pengujian sebelumnya yang bernilai 100. Hasil pengujian pada skenario kelima ini, memiliki ciri yang identik dengan hasil pengujian yang ditunjukkan pada skenario pengujian flowtime keempat.

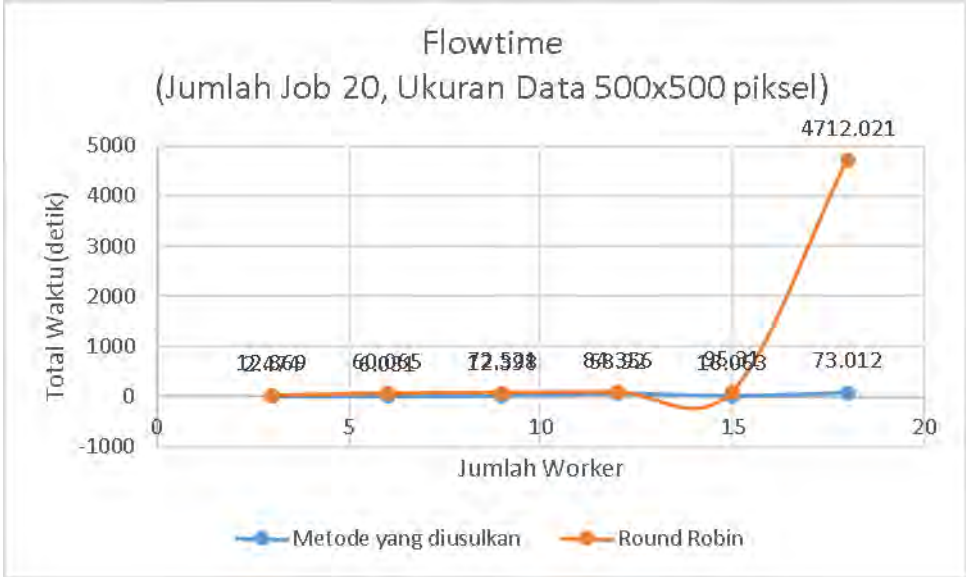


Gambar 4.6 Grafik Flowtime Job=20 dan Data=300x300piksel

Hasil yang diperoleh dari metode Round Robin, masih tidak cukup stabil dibandingkan dengan hasil yang diperoleh dari metode yang diusulkan seperti yang terlihat pada Gambar 4.6.

4.2.2.6.Skenario Job=20 dan Data=500

Skenario pengujian terakhir dari pengujian flowtime adalah dengan menambahkan ukuran data menjadi 500 dari pengujian sebelumnya. Seperti yang ditunjukkan pada Gambar 4.7. Worker yang dialami pada metode Round Robin mengalami kenaikan yang cukup tinggi pada titip 18. Hasil ini dipengaruhi oleh terjebaknya MA atau worker di tempat dengan sumber daya yang rendah. Sehingga membutuhkan waktu eksekusi yang sangat tinggi. Hal ini mengakibatkan perbedaan yang cukup signifikan dengan waktu eksekusi dari *node* yang lain.



Gambar 4.7 Grafik Flowtime Job=20 dan Data=500x500piksel

Pada pengujian *flowtime* ini dapat dianalisa bahwa metode yang diusulkan memiliki waktu flowtime yang lebih baik dibandingkan dengan metode Round Robin.

4.2.3. Analisis Parameter Makespan

Makespan merupakan nilai waktu penyelesaian eksekusi secara menyeluruh dari eksekusi *behavior* dari sejumlah MA. Nilai *makespan* ini menunjukkan lama waktu yang dibutuhkan untuk menyelesaikan seluruh pekerjaan oleh semua worker. Semakin kecil nilai *makespan* maka akan semakin baik untuk komputasi yang dilakukan oleh multi MA. Namun jika nilai *makespan* besar, maka hal tersebut masih perlu ditingkatkan. Sehingga dengan rekayasa mengurangi nilai *makespan* akan membuat rata-rata waktu pengerjaan semakin baik. Nilai *makespan* akan berdampak pada kualitas dari keseimbangan waktu kerja dari multi MA.

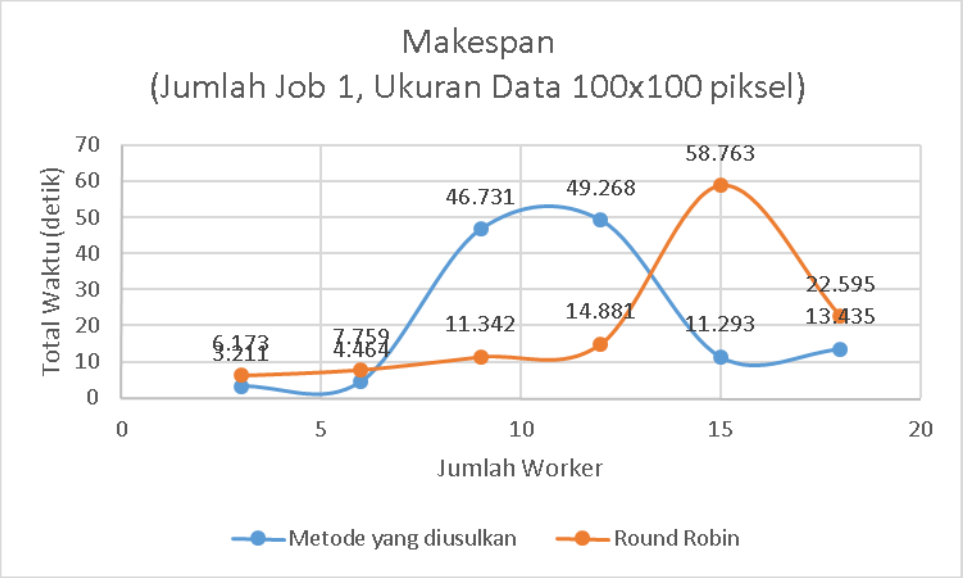
Pada pengujian pengukuran parameter *makespan,* dijalankan multi MA untuk menjalankan satu atau beberapa *behavior* atau job. Pengujian ini menggunakan tiga variable yang berbeda.

d. Job merupaan banyaknya satuan tugas yang harus dilakukan oleh MA pada *behavior.* Satuan tugas ini besarannya adalah satu kali koputasi algoritma SURF oleh pustaka BootCV.

e. Data merupakan ukuran citra yang digunakan. Ukuran citra pada pengujian ini bervariasi dari 100x100 piksel, 300x300 piksel, dan 500x500 piksel.

f. Worker merupakan sebutan untuk MA dalam konteks multi MA.

Hasil pengujian dipaparkan pada Tabel 4.3. Dari Tabel 4.3, data yang dihasilkan memiliki kecenderungan yang tidak stabil dari kedua metode. Semua grafik yang ditunjukkan oleh Gambar 4.8-4.13 menunjukkan bahwa tidak terjadi keseimbangan antara kedua metode yang dibandingkan.

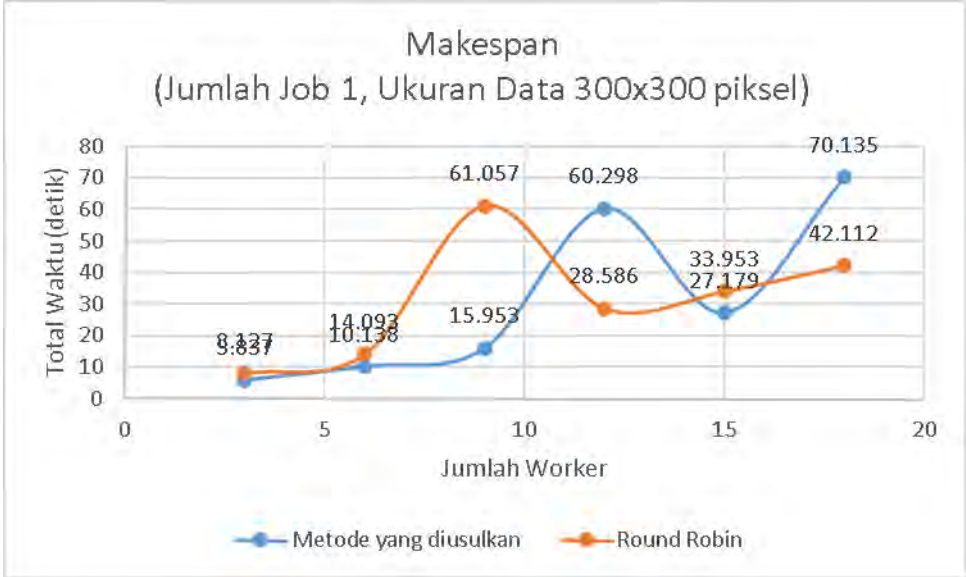


Gambar 4.8 Grafik Makespan Job=1 dan Data=100x100piksel

Pada Gambar 4.8, kedua grafik menunjukkan pola naik turun yang tidak beraturan. Hal tersebut mengindikasikan bahwa jumlah *worker* kurang begitu berpengaruh terhadap waktu *makspan* yang didapat. Pada saat worker berjumlah 3, nilai dari metode yang diusulkan memiliki waktu yang lebih baik dibandingkan dengan metode pembanding, Round Robin. Begitu juga pada jumlah worker enam. Namun ketika jumlah worker menjadi 9 dan 12, metode Round Robin memiliki waktu yang lebih baik. Sedangkan data sisanya menunjukkan fakta sebaliknya.

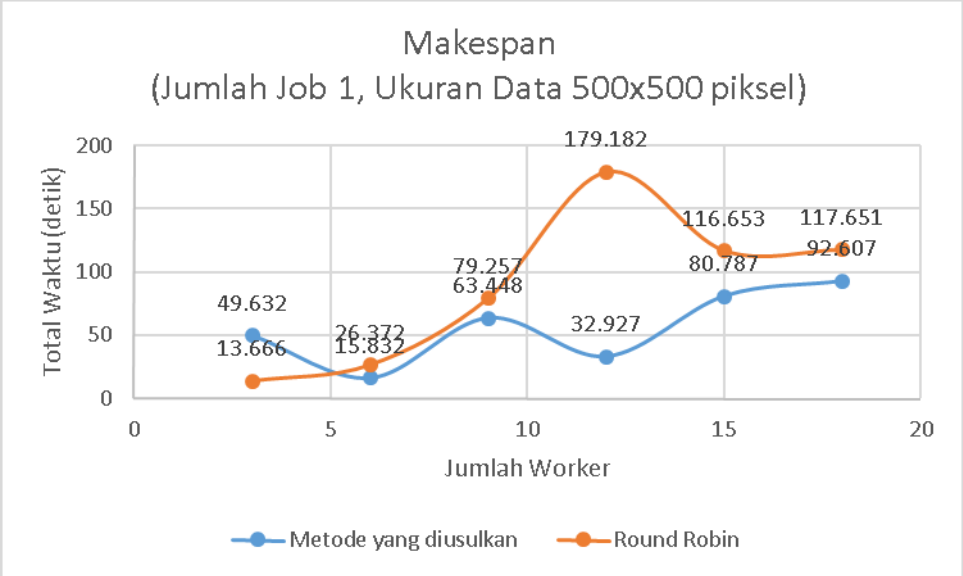
Tabel 4.3 Hasil Pengujian Makespan

job	gambar	worker	Usulan Metode	Round Robin
			makespan	makespan
1	100	3	3.211	6.173
		6	4.464	7.759
		9	46.731	11.342
		12	49.268	14.881
		15	11.293	58.763
		18	13.435	22.595
	300	3	5.837	8.127
		6	10.138	14.093
		9	15.953	61.057
		12	60.298	28.586
		15	27.179	33.953
		18	70.135	42.112
	500	3	49.632	13.666
		6	15.832	26.372
		9	63.448	79.257
		12	32.927	179.182
		15	80.787	116.653
		18	92.607	117.651
20	100	3	4.813	46.349
		6	8.44	10.266
		9	11.625	14.45
		12	16.802	62.097
		15	63.73	64.223
		18	30.338	28.799
	300	3	7.133	13.967
		6	12.672	60.067
		9	18.532	27.603
		12	67.618	76.659
		15	81.646	84.237
		18	91.416	54.39
	500	3	12.037	21.68
		6	21.707	72.256
		9	72.321	86.516
		12	81.819	102.062
		15	97.51	117.357
		18	108.606	4779.15



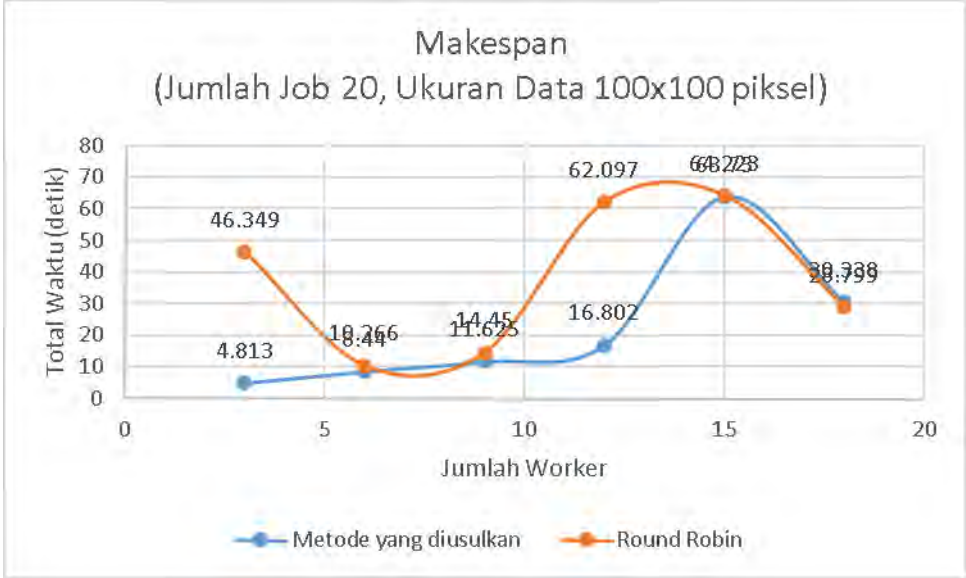
Gambar 4.9 Grafik Makespan Job=1 dan Data=300x300piksel

Pada Gambar 4.9, kedua grafik menunjukkan pola naik turun yang tidak beraturan. Hal tersebut mengindikasikan bahwa jumlah *worker* kurang begitu berpengaruh terhadap waktu *makspan* yang didapat. Pada saat worker berjumlah 3, nilai dari metode yang diusulkan memiliki waktu yang lebih baik dibandingkan dengan metode pembanding, Round Robin. Begitu juga pada jumlah worker enam dan 9. Namun ketika jumlah worker menjadi 12 dan 18, metode Round Robin memiliki waktu yang lebih baik. Sedangkan data sisanya menunjukkan fakta sebaliknya.



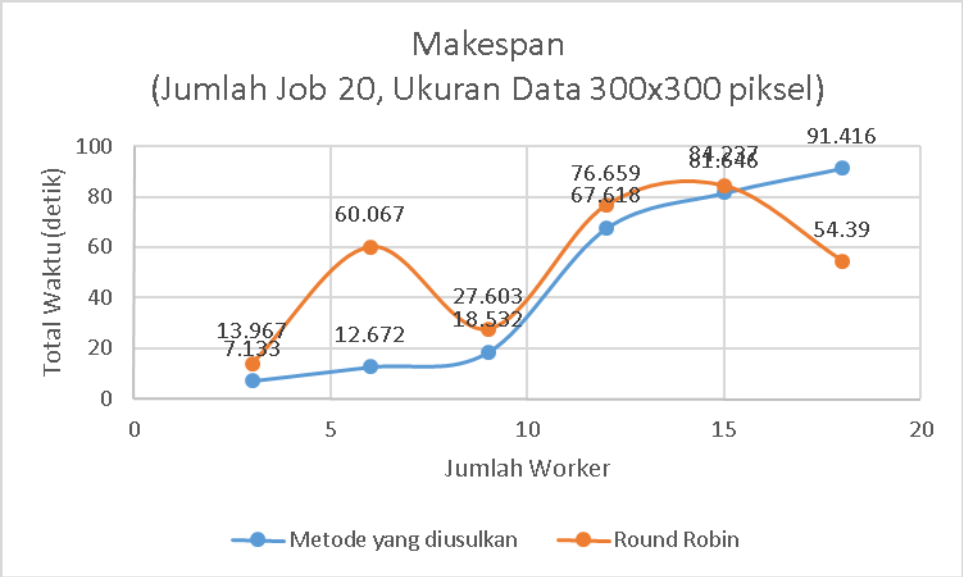
Gambar 4.10 Grafik Makespan Job=1 dan Data=500x500piksel

Pada Gambar 4.10, kedua grafik menunjukkan pola naik turun yang tidak beraturan. Hal tersebut mengindikasikan bahwa jumlah *worker* kurang begitu berpengaruh terhadap waktu *makspan* yang didapat. Pada saat worker berjumlah 3, nilai dari metode yang diusulkan memiliki waktu yang tidak lebih baik dibandingkan dengan metode pembanding, Round Robin. Namun dari data yang tersisa menunjukkan bahwa metode yang diusulkan memiliki waktu yang lebih baik.



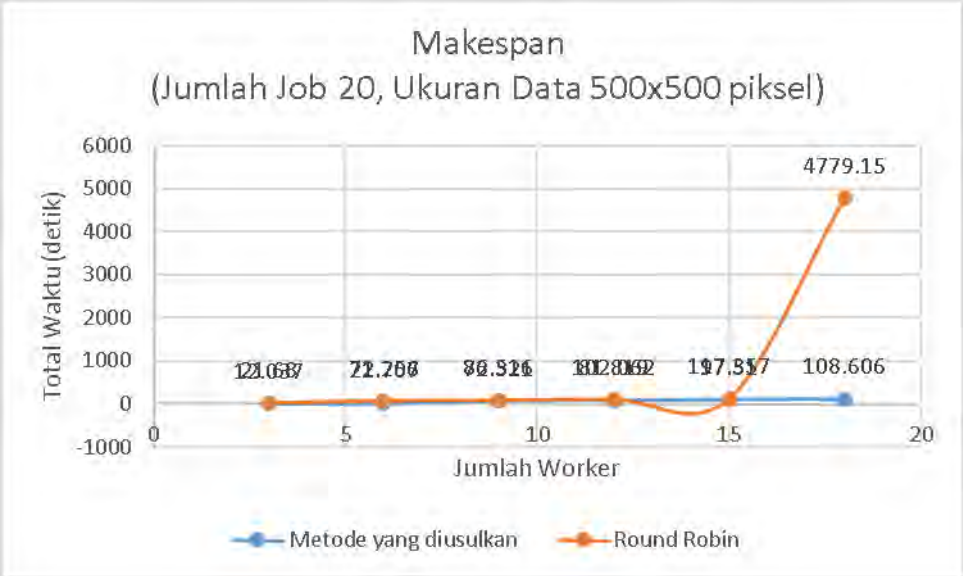
Gambar 4.11 Grafik Makespan Job=20 dan Data=100x100piksel

Pada Gambar 4.11, kedua grafik menunjukkan pola naik turun yang tidak beraturan. Hal tersebut mengindikasikan bahwa jumlah *worker* kurang begitu berpengaruh terhadap waktu *makspan* yang didapat. Namun secara rata-rata metode yang diusulkan memiliki waktu yang lebih baik kecuali pada saat jumlah worker mencapai 18.



Gambar 4.12 Grafik Makespan Job=1 dan Data=300x300piksel

Pada Gambar 4.12, kedua grafik menunjukkan pola naik turun yang tidak beraturan. Hal tersebut mengindikasikan bahwa jumlah *worker* kurang begitu berpengaruh terhadap waktu *makspan* yang didapat. Namun secara rata-rata metode yang diusulkan memiliki waktu yang lebih baik kecuali pada saat jumlah worker mencapai 18.



Gambar 4.13 Grafik Makespan Job=1 dan Data=500x500piksel

Pada Gambar 4.13, kedua grafik menunjukkan pola naik turun yang tidak beraturan. Hal tersebut mengindikasikan bahwa jumlah *worker* kurang begitu

berpengaruh terhadap waktu *makspan* yang didapat. Namun secara rata-rata metode yang diusulkan memiliki waktu yang lebih.

BAB V
KESIMPULAN DAN SARAN

Pada Bab ini dijelaskan kesimpulan akhir yang didapatkan dari penelitian yang telah dilakukan dan juga dipaparkan saran-saran yang bersifat membangun untuk penelitian selanjutnya di masa yang akan datang.

5.1. Kesimpulan

Pengujian dan analisis yang telah dilakukan menghasilkan beberapa kesimpulan penelitian sebagai berikut:

1. Metode yang diusulkan pada penelitian ini menggunakan MA dengan kederdasan MCDM pada atribut sumber daya untuk memililih *node* dengan respon waktu terbaik terbukti mampu mengatasi permasalahan pemilihan *node* eksekusi.
2. Atribute yang diusulkan terbukti mampu menjadi dasar perhitungan pengambilan keputusan pada kasus MA tunggal.
3. Nilai parameter *flowtime* pada semua skenario selalu lebih kecil dibandingkan dengan nilai parameter yang sama pada metode Round Robin.
4. Nilai *makespan* pada metode yang diusulkan dan pada metode Round Robin memiliki pola yang tidak stabil.

5.2. Saran

Saran yang dapat diberikan kepada penelitian selanjutnya adalah sebagai berikut:

1. Penjadwalan MA pada kasus MA untuk kasus komputasi parallel menggunakan multi MA.
2. Optimasi pengaturan beban kerja MA pada kasus komputasi parallel

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

Anderson, P., Bijani, S. & Vincos, A., 2012. *Multy-Agent Negotiation of Virtual Machine Migration Using the Lightweight Coordination Calculus.* Dubrovnik, Springer, pp. 124-133.

Bellifemine, F., Caire, G., Poggi, A. & Rimassa, G., 2003. JADE A White Paper. *TILAB Journal "EXP - in search of innovation" Special issue,* Volume 3, pp. 6-19.

Braun, P. & Rossak, W. R., 2004. *Mobile Agents Basic Concepts, Mobility Models, and the Tracy Toolkit.* 1st ed. s.l.:Morgan Kaufmann Publishers.

Choi, J., Dukhan, M., Liu, X. & Vuduc, R., 2014. *Algorithmic time, energy, and power on candidate HPC compute building blocks.* Phoenix, IEEE, pp. 447 - 457.

Gunasekera, K., 2011. *Compositionally Adaptive Mobile Sotware Agents for Pervasive Environtment,* Melbourne: Monash University.

Gunasekera, K., Zaslavsky, A., Krishnaswamy, S. & Loke, S. W., 2008. *VERSAG: Context-Aware Adaptive Mobile Agents for the Semantic Web.* Turku, IEEE, pp. 521-522.

Gunasekera, K., Zaslavsky, A., Loke, S. W. & Krishnaswamy, S., 2008. *Context Driven Compositional Adaptation of Mobile Agents.* Beijing, s.n.

Gunasekera, K., Zaslavsky, A., Loke, S. W. & Krishnaswamy, S., 2010. *Service Oriented Context-Aware Software Agents for Greater Eficiency.* Berlin, Springer Berlin Heidelberg, pp. 62-71.

Han, S. et al., 2008. A resource aware software partitioning algorithm based on mobility constraints in pervasive grid environments. *Future Generation Computer Systems,* 24(6), p. 512–529.

Sibata, K., Takimoto, M. & Kambayashi, Y., 2014. *Expanding the Control Scope of Cooperative Multiple Robots.* Chania, Springer, pp. 17-26.

Siddiqui, F., Zeadally, S. & Basu, K., 2012. Mobile Agent System. In: *Ubiquitous Multimedia and Mobile Agents Models and Implementations.* s.l.:IGI Global, pp. 31-59.

Skocir, P., Marusic, L., Marusic, M. & Petric, A., 2012. *The MARS - A Multi-Agent Recommendation System for Games on Mobile Phone.* Dubrovnik, Springer, pp. 104-112.

Šoštarić, D., Horvat, G. & Hocenski, Ž., 2012. *Multi-agent Power Management System for ZigBee Based Portable Embedded ECG Wireless Monitoring Device with LabView Application.* Dubrovnik, Springer, pp. 299-308.

Tabucanon, M. T., 1988. *Multiple Criteria Decision Making in Industry.* s.l., Elsavier.

Tu, M. T., Griffel, F., Merz, M. & Lamersdorf, W., 1998. *A Plug-in Architecture Providing Dynamic Negotiation Capabilities for Mobile Agents.* Stuttgart, Springer-Verlag.

Weiss, G., 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* s.l.:The MIT Press.

Yazir, Y., Matthews, C., Farahbod, R. & Neville, S., 2010. *Dynamic Resource Allocation in Computing Clouds using Distributed Multiple Criteria Decision Analysis.* Miami, IEEE.

BIODATA PENULIS

M Misbachul Huda, biasa dipanggil Misbah lahir di kota Tuban pada 24 Juni 1991. Penulis adalah anak pertama dari tiga bersaudara. Penulis menempuh pendidikan SDN Plumpang III (1997-2003), SMPN 1 Plumpang (2003-2006), SMAN 1 Tuban (2006-2009). Penulis kemudian melanjutkan pendidikan di Teknik Informatika ITS pada tahun 2009 lalu melanjutkan pada jenjang Master pada tahun 2013 di perguruan tinggi yang sama. Selama perkuliahan penulis pernah menjadi asisten praktikum untuk matakuliah Sistem Digital, Sistem Operasi, Jaringan Komputer. Penulis juga pernah menjadi asisten dosen untuk matakuliah Sistem Operasi, Jaringan Komputer dan Pemrograman Jaringan.

Selama menempuh perkuliahaan penulis juga aktif dalam melakukan riset. Tiga riset, dua paper untuk seminar internasional dan tiga paper pada jurnal nasional berhasil dikembangkan bersama dosen penulis selama kuliah. Penulis dapat dihubungi di email misbachul.h@gmail.com.