Pengembangan Prototipe Sistem *Network Rendering* Alternatif Berbasis JPPF

I Made Putrama[1], I Wayan Adi Sparta[2], I Gede Mahendra Darmawiguna[3], Gede Saindra Santyadiputra[4]

***Abstract*—Rendering is a stage to turn a model into a 3D animated video form. Rendering process takes a long time and requires hardware support with high specifications. Unfortunately, computers for rendering purpose with high specifications are generally very expensive. An alternative solution to this problem is to utilize grid computing technology that can take advantage of some ordinary computers configured in the network. Together, those computers can help complete rendering process more quickly. In this paper, the rendering solution is achieved by combining the grid computing technology-based framework, called JPPF, that can be integrated using Blender 3D. Blender 3D is one of the common software used in making animated films. In this paper, the solution result is also compared with the one from the network-based rendering feature of the Blender 3D, called Network Rendering, as a performance benchmark. The comparison results show that this alternative solution works with relatively better performance and flexibility and is easier to operate than the features provided in 3D Blender.**

***Intisari—Rendering* merupakan suatu tahapan untuk mengubah model menjadi bentuk video animasi 3D. Proses *rendering* memakan waktu cukup lama, dan memerlukan dukungan perangkat keras dengan spesifikasi tinggi. Sayangnya, komputer untuk kepentingan *rendering* dengan spesifikasi tinggi umumnya sangatlah mahal. Oleh karena itu, salah satu alternatif yang dapat ditempuh adalah dengan memanfaatkan teknologi *grid computing* yang dapat memanfaatkan beberapa komputer biasa yang dikonfigurasi dalam jaringan, sehingga secara bersama-sama dapat membantu menyelesaikan proses *rendering* dengan lebih cepat. Pada makalah ini, solusi *rendering* tersebut ditempuh dengan menggabungkan kerangka kerja berbasis teknologi *grid computing*, JPPF, yang dapat diintegrasikan dengan Blender 3D, yang merupakan salah satu perangkat lunak yang umum digunakan dalam pembuatan film animasi. Dalam makalah ini, solusi tersebut juga dibandingkan hasilnya dengan fitur *rendering* berbasis jaringan yang memang telah ada pada Blender 3D, yang disebut Blender 3D *Network Rendering*, sebagai tolok ukur kinerja sistem yang dibangun. Hasil perbandingan menunjukkan bahwa solusi alternatif ini bekerja dengan kinerja dan fleksibilitas yang relatif lebih baik dan juga lebih mudah dioperasikan daripada fitur yang disediakan dalam Blender 3D.**

***Kata Kunci—rendering* jaringan, *grid computing*, Blender 3D, animasi.**

I. PENDAHULUAN

Indonesia sebagai negara yang mempunyai beragam kekayaan, yang salah satunya berhubungan dengan kreativitas sumber daya manusia, kini mulai merambah ke dunia industri animasi kreatif. Beberapa film animasi di Indonesia sudah mulai berkembang pesat yang hasilnya sudah dapat disaksikan di televisi, seperti Si Entong, Keluarga Somat, Adit & Sopo Jarwo, dan yang lainnya, yang beberapa di antaranya dibuat oleh rumah produksi seperti MNC Animation, Dreamtoon Animation Studios dan MD Animation.

Sebagai bagian dari lembaga pendidikan di Indonesia, jurusan Pendidikan Teknik Informatika (PTI) di Fakultas Teknik dan Kejuruan (FTK), Universitas Pendidikan Ganesha ikut menyelenggarakan pendidikan yang berhubungan dengan pengembangan animasi, dimulai dari mata kuliah Teknologi Multimedia yang dilanjutkan di semester selanjutnya menjadi sebuah mata kuliah pilihan bagi mahasiswa yang ingin mendalami pembuatan film animasi dengan mengambil mata kuliah Animasi 3D Lanjutan *(Advance 3D Animation)*. Di PTI, kreativitas mahasiswa dalam pembuatan film animasi sudah mengalami banyak perkembangan, dibuktikan dengan mulai banyaknya mahasiswa mengambil skripsi bertema animasi, yang salah satunya berupa karya sebuah film fiksi dalam bentuk animasi 3D yang menceritakan lahirnya maskot jurusan PTI, *"Tude the Movie"*. Film animasi 3D ini diangkat dalam sebuah skripsi yang kemudian dikembangkan lebih lanjut oleh beberapa mahasiswa yang lain [1].

Antusiasme mahasiswa PTI dalam membuat produk kreatif sangatlah tinggi. Berdasarkan pengalaman proses pembelajaran yang diperoleh mahasiswa, dapat disadari bahwa keterbatasan utama yang ada di jurusan PTI berkaitan dengan ini adalah belum tersedianya fasilitas laboratorium yang memadai, khususnya yang menunjang pembuatan film animasi di *Laboratory of Cultural Informatics* (LCI). LCI merupakan satu-satunya laboratorium PTI yang dapat dimanfaatkan mahasiswa untuk pembuatan film animasi.

Pembuatan film animasi di Lab LCI menggunakan bantuan sebuah perangkat lunak Blender 3D. Untuk membuat sebuah film animasi 3D yang berdurasi hanya sekitar 20 menit, mahasiswa membutuhkan waktu pengerjaan yang sangat lama hingga berbulan-bulan. Selain karena alasan pembuatan produk film animasi yang memerlukan kreativitas dan keterampilan penggunaan perangkat lunak tersebut serta tingkat kompleksitas film animasi yang dibuat, spesifikasi dan kinerja perangkat keras komputer yang digunakan juga menjadi faktor utama penyebab lamanya waktu menyelesaikan sebuah film animasi.

Berkaitan dengan proses pengerjaan pada pembuatan sebuah film animasi, ada beberapa tahapan proses penting yang harus diselesaikan, yaitu *modeling*, *texturing*, *rigging*, *animating*, dan *rendering*. Empat tahap pertama merupakan proses yang umumnya masih banyak berkaitan dengan kreativitas seorang *animator*, sehingga kualitas beserta lama pengerjaan prosesnya sangat bergantung pada tingkat keterampilan pembuatnya. Proses *rendering*, yang merupakan

[1,2,3,4]*Jurusan Pendidikan Teknik Informatika, Fakultas Teknik dan Kejuruan, Universitas Pendidikan Ganesha, Jl. Udayana No. 11, Singaraja, Bali, 81116, INDONESIA (tlp: 0362-27213; e-mail: made.putrama@undiksha.ac.id)*

salah satu tahapan yang tidak kalah penting, lebih bergantung kepada kinerja komputer yang digunakan serta teknik *rendering* yang diterapkan. *Rendering* adalah proses pengubahan data dari objek yang terlihat pada *Camera View* di Blender 3D menjadi *file* gambar atau *file* film animasi [2]. Dalam proses *rendering*, pengaturan format keluaran gambar ataupun film yang diinginkan, baik itu kualitas maupun pengaturan jumlah *frame* per detiknya, memengaruhi lama waktu yang dibutuhkan untuk menyelesaikan proses tersebut secara keseluruhan.

Berdasarkan observasi awal dan penyebaran angket survei terhadap beberapa mahasiswa tingkat akhir di Jurusan PTI yang mengerjakan skripsi bertemakan film animasi 3D, diperoleh informasi bahwa proses *rendering* yang dilakukan mahasiswa rata-rata membutuhkan waktu sangat lama sehingga menghambat dalam penyelesaian tugas akhir. Berdasarkan hasil survei, mahasiswa PTI membutuhkan waktu antara 4--6 bulan hanya untuk mengerjakan *rendering* film animasi. Ada beberapa hal lain yang memperlambat proses *rendering* film animasi, yakni komputer-komputer di Lab LCI masih mengadopsi teknik *rendering* konvensional dengan menggunakan antarmuka Blender 3D yang hanya memanfaatkan kinerja sebuah komputer saja untuk sekali *rendering*. Untuk dapat mengerjakan proses *rendering* secara paralel, saat ini mahasiswa masih harus membagi *file* Blender 3D secara manual menjadi beberapa bagian dan kemudian melakukan proses *rendering* terhadap masing-masing bagian menggunakan komputer yang berbeda-beda secara terpisah. Selain itu, ketika terjadi gangguan karena ketidaksengajaan pengguna lain yang menggunakan komputer yang sedang mengerjakan proses *rendering* dan juga seringnya listrik padam di Lab LCI membuat proses tersebut harus diulang kembali dari awal, sehingga menyebabkan pengerjaan keseluruhan proses menjadi terhambat. Dari pengamatan yang dilakukan terhadap kondisi di Lab LCI, sebuah komputer yang sedang digunakan untuk melakukan *rendering* harus diberikan label yang memberitahukan bahwa komputer tersebut tidak boleh digunakan untuk kepentingan lain karena dapat mengganggu jalannya proses *rendering*.

Untuk mengatasi hal ini, sebuah model penerapan teknologi *rendering* berbasis jaringan yang dikenal dengan istilah *network rendering* atau *rendering farm* dapat digunakan. Model ini dapat memanfaatkan beberapa atau seluruh komputer di Lab LCI untuk dapat melakukan proses *rendering* secara bersama-sama sekaligus dengan tetap mengijinkan komputer-komputer tersebut digunakan untuk kepentingan yang lain. Selain itu, teknologi ini memungkinkan proses *rendering* tetap berlangsung jika salah satu komputer pendukung proses tersebut mengalami gangguan, sehingga tahap *rendering* yang sedang dilakukan tidak harus selalu diulang dari awal seperti apa yang terjadi saat ini.

Sudah ada banyak bentuk penerapan teknologi *rendering farm* yang tersedia saat ini, yakni fasilitas *rendering online* seperti *www.rebusfarm.net*, *www.rendercore.com*, dan lain-lain yang rata-rata berbasiskan *cloud computing*. Namun, teknik ini memiliki kelemahan, karena selain membutuhkan koneksi internet, juga merupakan opsi komersial yang berbayar yang tidak dianjurkan. Dengan mengerjakan *rendering* secara *online*, berarti *file* harus diunggah ke penyedia layanan sehingga memungkinkan terjadinya penyalahgunaan hak cipta produk.

Saat ini Blender 3D sudah didukung fasiltias *rendering farm* yang disebut dengan *Network Render*. Berdasarkan observasi awal, kelebihan fasilitas ini adalah merupakan teknologi bawaan dari Blender 3D yang sudah dimanfaatkan oleh pengguna di seluruh dunia dan tentunya sudah melalui tahap evaluasi sesuai standar internasional. Namun, fasilitas *Network Render* Blender 3D ini mengharuskan setiap komputer terhubung jaringan, baik itu melalui *Local Area Network* (LAN) maupun melalui internet, untuk membuka antarmuka Blender 3D saat *rendering* sedang berlangsung. Akibatnya, komputer tersebut tidak dapat digunakan secara leluasa di saat sedang membantu pengerjaan proses *rendering*. Keterbatasan lain dari fasilitas ini adalah fitur yang ada dibuat sesuai standar yang tidak dapat dikustomisasi dengan mudah karena terintegrasi dengan Blender 3D sehingga untuk melakukannya dibutuhkan pemahaman pengembangan antarmuka Blender 3D yang umumnya membutuhkan penerapan algoritme *low-level* tertentu, yang dapat ditulis dalam bahasa pemrograman Python. Hal ini tentunya membutuhkan waktu evaluasi, penyesuaian, dan penyempurnaan yang tidak singkat untuk dapat digunakan dan diterima sebagai produk yang siap pakai.

II. *RENDERING* DAN *GRID COMPUTING*

Sebuah penelitian membahas tentang distribusi proses *rendering* yang disebut dengan *Distributed System for Automatic Blender Rendering* (DSABR) [3]. Proses ini merupakan solusi yang menerapkan algoritme yang ditulis dengan bantuan *library* Phyton berdasarkan *distributed computing framework*, yakni *Work Queue*. Dikemukakan bahwa hasil penelitian telah memberikan gambaran cara proses *rendering* animasi dapat dikerjakan lebih cepat, dibandingkan dengan metode konvensional. Namun, ada beberapa kendala yang ditemui. Salah satunya adalah ketika salah satu *node* yang membantu pengerjaan proses *rendering* mengalami keadaan stagnan dan memakan waktu sangat lama, keseluruhan distribusi proses menjadi terhambat. Dan penggunaan fitur *Work Queue*, yang disebut dengan istilah *fast-abort*, memerlukan sebuah *multiplier* yang penentuan nilai optimalnya tidak dikemukakan secara jelas.

Grid computing secara sederhana dikatakan sebagai sebuah perubahan evolusioner komputasi terdistribusi ke level yang lebih tinggi. Seperti juga diungkapkan bahwa sebuah sistem *grid* akan menyediakan infrastruktur elektronik yang dapat dimanfaatkan lintas sosial secara global, baik untuk kepentingan bisnis, pemerintahan, penelitian, sains, maupun kepentingan dunia hiburan [4]. Salah satu *open-source grid computing framework* berbasis Java, yang disebut *Java Parallel Processing Framework* (JPPF), menjanjikan sebuah mekanisme penanganan pekerjaan terdistribusi secara paralel yang dapat membagi sebuah aplikasi untuk dieksekusi secara otomatis ke dalam bagian-bagian kecil secara terpisah [5]. Dengan menggunakan JPPF, aplikasi yang membutuhkan kemampuan pemrosesan yang tinggi dapat dijalankan di

banyak komputer yang terhubung untuk mengurangi waktu pemrosesan secara dramatis [5]. Hal ini seperti yang telah dilakukan dalam sebuah penelitian yang menggunakan *framework* JPPF untuk melakukan pengambilan informasi dari basis data MS SQL Server 2000, yang hasilnya menunjukkan bahwa pengambilan data terdistribusi secara paralel mengalami peningkatan kinerja yang sangat signifikan [6].

III. METODOLOGI

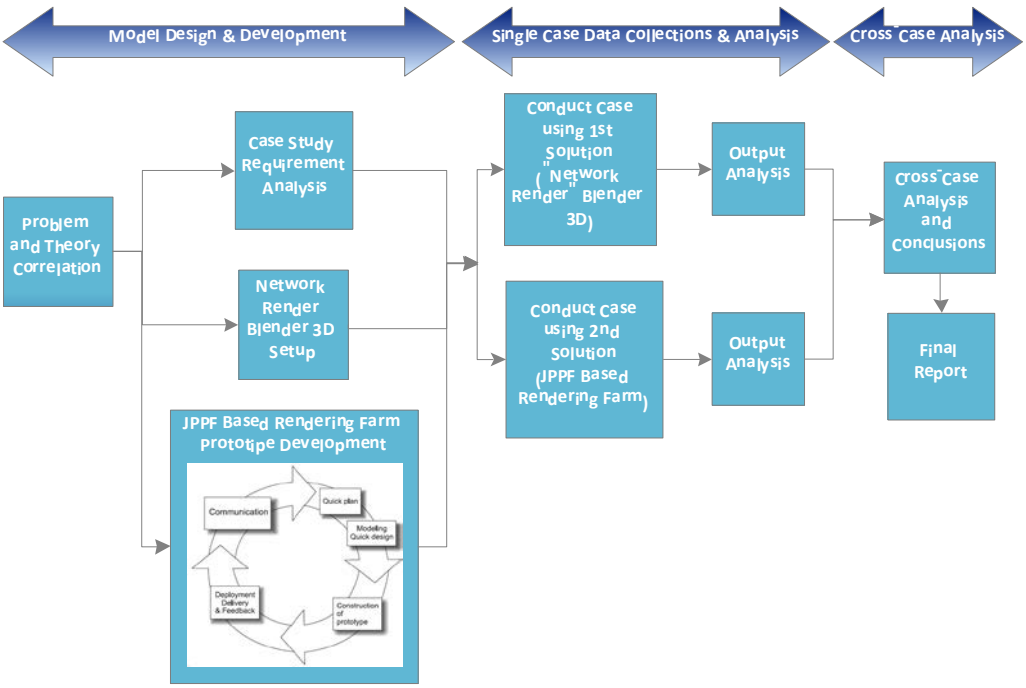
Metode yang digunakan merupakan penggabungan konsep dari tahapan metode penelitian studi kasus penelitian sebelumnya dengan pengembangan prototipe perangkat lunak *Software Development Life Cycle* (SDLC) menggunakan model *prototyping* [7].

Seperti ditunjukkan pada Gbr. 1, ada dua metode yang digunakan, yakni metode dengan cara melakukan *setup* atau konfigurasi *Network Render* yang merupakan bawaan versi Blender 3D yang ada saat ini, dan metode yang menerapkan teknologi *grid computing* menggunakan *framework* Java, JPPF, ke dalam sebuah prototipe perangkat lunak yang diuji dan dibandingkan kinerjanya dengan metode yang pertama. Metode penelitian studi kasus ini berdasarkan metode yang telah dikembangkan dalam sebuah buku yang berjudul *"Case Study Research: Design and methods"* [8].

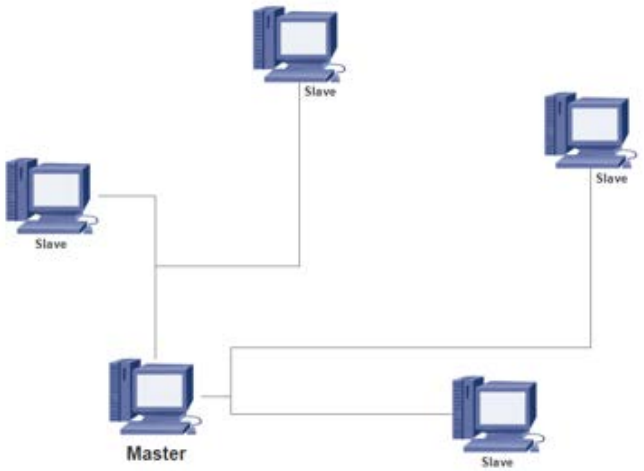
Untuk dapat mengkaji dan membandingkan kedua metode, ada beberapa hal yang perlu dipersiapkan, yaitu: (1) perangkat keras pendukung dengan jumlah dan spesifikasi yang sama (menggunakan *Personal Computer* (PC) yang sama untuk kedua metode); (2) menggunakan *file* masukan yang sama; (3) dilakukan dalam kondisi lingkungan yang sama, yaitu pada jaringan dengan konektivitas yang sama, misalnya *Wired/Wireless Local Area Network*.

Adapun kebutuhan kedua model yang dibuat dapat dirinci sebagai berikut.

1. Kedua model dapat menyimulasikan *rendering farm* dengan topologi: (a) sebuah PC sebagai *server* (pusat *render*) dengan sebuah PC sebagai *slave* (membantu proses *rendering*); (b) sebuah PC sebagai *server* (pusat *render*) dengan dua buah PC sebagai *slave* (membantu proses *rendering*); (c) sebuah PC sebagai *server* (pusat *render*) dengan tiga buah PC sebagai *slave* (membantu proses *rendering*); (d) sebuah PC sebagai *server* (pusat *render*) dengan empat PC sebagai *slave* (membantu proses *rendering*); dan (e) dua buah PC sebagai *server* (pusat *render*) dengan tiga PC sebagai *slave* (membantu proses *rendering*).
2. Sistem dapat melakukan *rendering* sebuah *file* Blender 3D.
3. Sistem dapat melakukan *rendering* beberapa *file* Blender 3D sekaligus.
4. Sistem dapat melanjutkan *rendering* yang bermasalah di pengerjaan sebelumnya secara manual tanpa harus mengulang dari awal.
5. Sistem dapat melanjutkan *rendering* yang bermasalah di pengerjaan sebelumnya secara ototmatis tanpa harus mengulang dari awal.
6. Kedua model dapat menghasilkan keluaran akhir film animasi dalam satu *file* video.
7. Kedua model dapat menghasilkan keluaran akhir film animasi dalam satu *file* video ke lokasi yang dapat dipilih.
8. Kedua model dapat dikonfigurasi untuk dapat mematikan PC *server* dan/atau *slave* yang digunakan secara otomatis.
9. Kedua model dapat dikonfigurasi untuk dapat memberikan notifikasi *email* ketika *server* terhubung ke *internet*.



Gbr. 1 Metode penelitian.



Gbr. 2 Topologi *Network Rendering* Blender 3D.

Pada tahap analisis, sebuah pendekatan pengujian sistem diadopsi, yakni pendekatan *Synthetic Workloads* yang digunakan untuk mengevaluasi kinerja dan perbandingan beban kerja sebuah sistem [9].

Adapun rumus *Synthetic Workloads* yang berkaitan adalah sebagai berikut.

$$WC = \frac{\sum_{j \in \tau \wedge j\ completed} 1}{|\tau|} \quad (1)$$

$$TC = \frac{\sum_{j \in \tau \wedge k \in j \wedge k\ completed} 1}{\sum_{j \in \tau} |j|} \quad (2)$$

$$ETC = \frac{\sum_{j \in \tau \wedge k \in j \wedge k\ completed} 1}{\sum_{j \in \tau \wedge k \in j \wedge k\ enabled} 1} \quad (3)$$

dengan

τ = jumlah *job* dalam sebuah sistem,

j = sebuah job,

k = sebuah *task*.

Selanjutnya, untuk kegagalan yang terjadi saat eksekusi skenario pengujian, digunakan perhitungan *System Failure Factor* (SFF) yang merupakan rasio antara jumlah semua *task* yang gagal dikerjakan dengan jumlah total *task* yang dikerjakan.

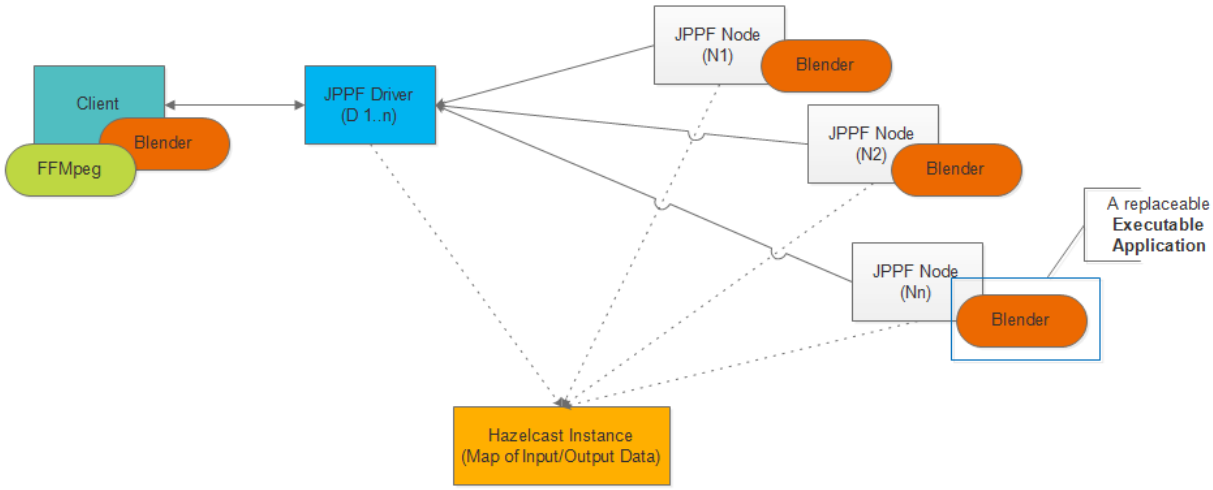
WC = *Workload Completion*, yaitu rasio total *job* yang diselesaikan dengan total *job* yang dikerjakan, seperti pada (1).

TC = *Task Completion* yang merupakan rasio total *task* yang diselesaikan dengan total *task* yang dikerjakan, seperti pada (2).

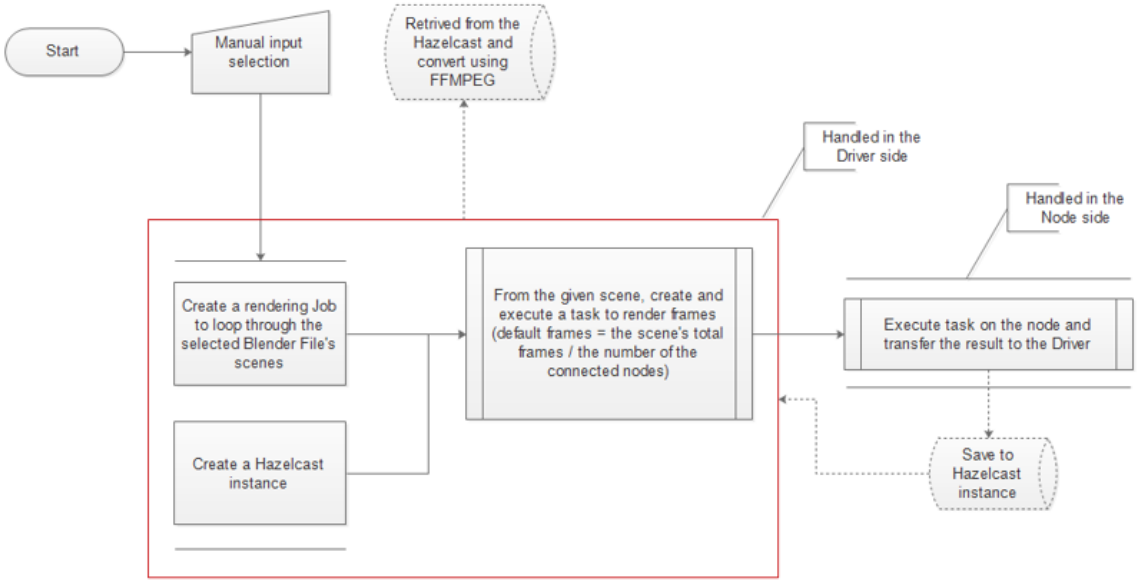
ETC = *Enabled Task Completion*, yakni rasio total *enabled-task* yang diselesaikan dengan total *enabled-task* yang dikerjakan, seperti pada (3). *Enabled-task* adalah *task* yang bisa dikerjakan jika *task* sebelumnya telah selesai dikerjakan.

Untuk kepentingan perbandingan antara sistem *rendering* menggunakan Blender 3D dan sistem prototipe yang dibangun, ada dua konfigurasi yang dibutuhkan. Yang pertama adalah konfigurasi sistem *rendering* dengan Blender 3D dengan topologi jaringan seperti ditunjukkan pada Gbr. 2. Sedangkan konfigurasi yang kedua merupakan sistem *rendering* menggunakan *framework* JPPF yang arsitektur sistemnya ditunjukkan pada Gbr. 3.

Cara kerja sistem *rendering* dengan Blender 3D adalah dengan mengaktifkan fitur *Network Rendering* yang ada pada antarmuka Blender 3D yang berjalan pada masing-masing komputer yang telah terhubung satu sama lain dalam jaringan. Konfigurasi diterapkan berdasarkan panduan tahap demi tahap seperti yang telah dijelaskan dalam sebuah *website* [10]. Diungkapkan bahwa proses *rendering* sebuah *file* film animasi Piero Series dengan total *frame* sebanyak 1.037 awalnya memakan waktu 7,2 hari. Dengan menggunakan *Render Network* dengan empat *nodes*/*slave*, proses *rendering* film tersebut dapat diselesaikan dalam waktu 1,8 hari, yang artinya lebih efisien hingga 75% [10].



Gbr. 3 Arsitektur aplikasi sistem *rendering* berbasis JPPF.



Gbr. 4 Alur algoritme prototipe sistem *rendering* berbasis JPPF.

```
AJNR
|_____config
|       |_____blendfile.py                (-- python script yang dibutuhkan Blender Command dalam membaca
input file)
|       |_____commons-logging.properties
|       |_____jppf.properties
|       |_____log4j.properties
|       |_____logging.properties
|_____lib                                 (--lib folder digunakan untuk menampung semua dependencies)
|_____src
|       |_____main
|               |_____java
|               |       |_____id.ac.undiksha.research.apps.ajnr
|               |               |_____listeners
|               |               |       |_____AJNRNodeListener.java
|               |               |_____services
|               |               |       |_____AJNRDriverStartup.java
|               |               |       |_____AJNRNodeStartup.java
|               |               |_____ui
|               |               |       |_____AJNRBooleanEditor.java
|               |               |       |_____AJNRCheckBoxEditor.java
|               |               |       |_____AJNRCheckBoxRenderer.java
|               |               |       |_____AJNRFileListCellRenderer.java
|               |               |       |_____AJNRFileMouseListener.java
|               |               |       |_____AJNRFilePopup.java
|               |               |       |_____AJNRFileSelectionListener.java
|               |               |       |_____AJNRFileTableModel.java
|               |               |       |_____AJNRTextEditorListener.java
|               |               |       |_____MainUI.form
|               |               |       |_____MainUI.java
|               |               |_____utils
|               |               |       |_____AJNRUtil.java
|               |               |       |_____AJNRZip.java
|               |               |       |_____BlendFileFilter.java
|               |               |       |_____EmailValidator.java
|               |               |       |_____NumberInputVerifier.java
|               |               |_____vo
|               |               |       |_____AJNRData.java
|               |               |       |_____AJNRFile.java
|               |               |       |_____AJNRScene.java
|               |               |       |_____AJNRSetting.java
|               |               |_____AJNRClient.java
|               |               |_____AJNRClientCallable.java
|               |               |_____AJNRJob.java
|               |               |_____AJNRRunner.java
|               |               |_____AJNRTask.java
|               |_____resources
|                       |_____META-INF
|                       |       |_____services
|                       |               |_____org.jppf.node.event.NodeLifeCycleListener
|                       |               |_____org.jppf.startup.JPPFDriverStartupSPI
|                       |               |_____org.jppf.startup.JPPFNodeStartupSPI
|                       |_____AJNR.png
|                       |_____AJNR-icon.png
|                       |_____AJNR-tray.png
|
|_____target                              (--target folder digunakan untuk menampung hasil kompilasi pro-
gram)
```

Gbr. 5 Struktur program dalam prototipe sistem *rendering* berbasis JPPF.

Proses internal dari sistem *rendering* dengan menggunakan Blender 3D tetap merupakan sebuah proses *black-box* yang dalam penelitian ini tidak dikaji lebih dalam, sebab sistem tersebut merupakan produk terbuka siap pakai. Sedangkan pada sistem *rendering* menggunakan JPPF, seperti yang ditunjukkan pada Gbr. 4, algoritme sistem pada dasarnya meliputi pemrosesan masukan dengan memanfaatkan *framework* JPPF dalam melakukan manajemen kerja berbasis *grid*, sehingga sebuah pekerjaan *rendering* dapat dipecah dan dikerjakan oleh beberapa komputer sekaligus secara bersama-sama. Penggunaan *framework* ini diintegrasikan dengan menggunakan komponen lain yang berupa *framework* perangkat lunak multimedia yang disebut dengan FFmpeg [11]. FFmpeg berfungsi dalam melakukan penggabungan dan pengubahan hasil *rendering* masing-masing komputer sehingga menjadi satu keluaran akhir berupa video animasi. Untuk mendistribusikan masukan *file* dari satu komputer yang bertindak sebagai *driver* ke masing-masing komputer *node* dan kemudian mengumpulkan hasil *rendering* dari masing-masing *node* ke dalam sebuah *folder* pada komputer *driver*, digunakan sebuah *framework* perangkat lunak yang disebut dengan Hazelcast [12]. Hazelcast disebut juga dengan istilah *in-memory data grid framework*, yang memungkinkan penyimpanan data dari satu komputer *node* ke komputer *driver* secara langsung tanpa melalui proses pembuatan koneksi yang umumnya diperlukan jika digunakan teknik lain, seperti transfer *file* menggunakan *File Transfer Protocol* (FTP) yang cukup memakan waktu dan dapat mendegradasi kinerja sistem secara keseluruhan.

IV. HASIL

Implementasi rancangan sistem *rendering* dengan JPPF pada Gbr. 3 dan Gbr. 4 adalah berupa struktur program yang dapat diamati pada Gbr. 5. Secara umum, kode program terbagi ke dalam tiga kategori, yakni: kode program yang berkaitan dengan konfigurasi sistem, yang terdapat pada *folder* "config"; *folder* "lib" yang berisi semua *file* yang

terkait yang diperlukan untuk mengintegrasikan *framework* yang digunakan, seperti JPPF, Hazelcast, dan Ffmpeg; dan kode program yang ditulis dalam bahasa Java yang mengintegrasikan semua *framework* yang digunakan untuk menghasilkan sebuah solusi sistem *rendering* yang diinginkan. Detail dari beberapa kode program penting, secara umum dipaparkan pada Tabel I. Antarmuka sistem *rendering* menggunakan JPPF ini ditunjukkan pada Gbr. 6, yang terdiri atas dua bagian, yakni bagian *setud* parameter yang digunakan untuk memasukkan *file* masukan untuk di-*render*, sedangkan pada bagian lain digunakan untuk mengatur notifikasi saat proses *rendering* berlangsung.

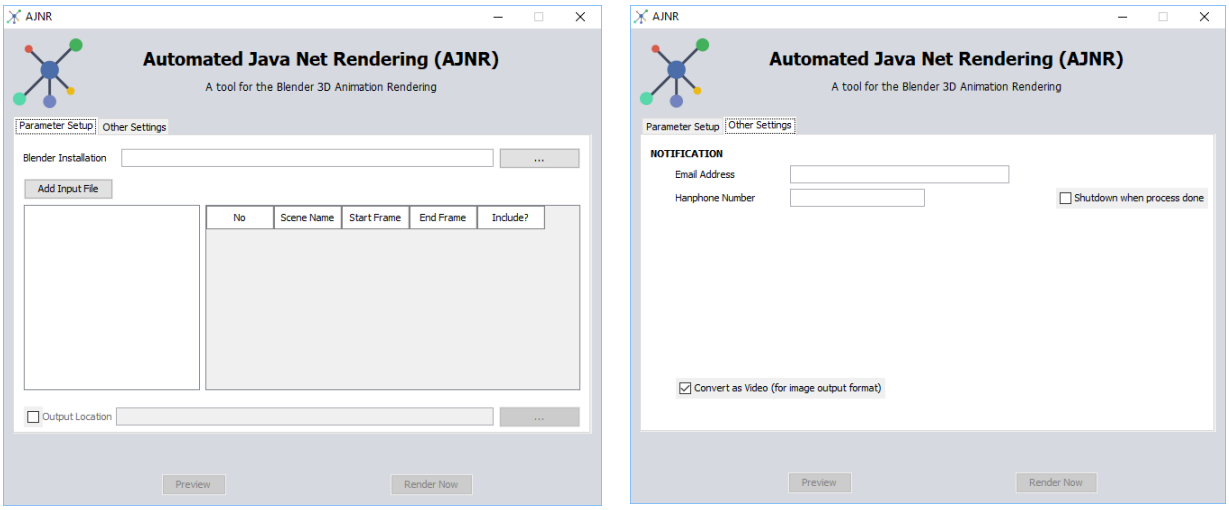
TABEL I
PENJELASAN KODE PROGRAM

Nama Kode Program	Potongan Program	Fungsionalitas

blendfile.py	def info():
	scenes = bpy.data.scenes
	finfo = []
	for s in scenes:
	sinfo = {}
	sinfo = //put scene info into sinfo array
	finfo.append(sinfo)
	return json.dumps({'scenes':finfo})
	print(info())
		File ini merupakan *file* yang berisi kode program yang ditulis dalam bahasa Python yang berfungsi untuk melakukan ekstraksi masukan *file* Blender 3D yang akan di-*render*.

AJNRDriverStartup.java
AJNRNodeStartup.java
AJNRNodeListener.java
	AJNRNodeListener implements MessageListener
	…
	..
	public void onMessage() {
	…
	//handle message callback saat Node selesai melakukan *rendering* dan hasilnya disimpan di dalam Hazelcast
	..
	}
		Ketiga kode program ini mengimplementasikan kode pada *com.hazelcast.core.MessageListener* untuk mendapatkan notifikasi terhadap *event* yang terjadi ketika data ditambahkan/diambil dari Hazelcast.

AJNRClient.java
AJNRRunner.java
AJNRJob.java
AJNRTask.java
	public void run() {
	ListModel<AJNRFile> files = …
	for (int i=0; i<files.getSize(); i++) {
	AJNRFile file = …
	for (AJNRScene scene : file.scenes) {
	AJNRJob job = …
	executor.submit(job);
	}
	}
	}
	executor.shutdown();
	}
		Kode program ini digunakan untuk memulai *rendering* dengan melakukan pembacaan terhadap seluruh *scene* pada *file* masukan dan kemudian membuat satu *job* yang terdiri atas beberapa *task* yang nantinya secara otomatis akan terdistribusi ke masing-masing *node* yang aktif.

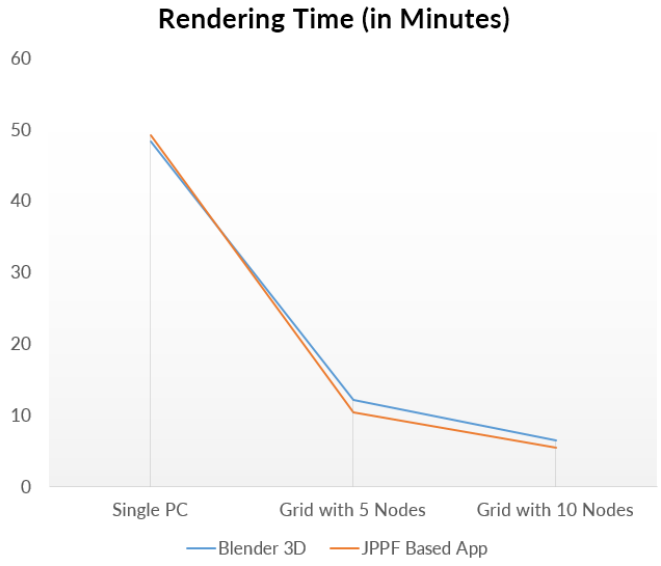


(a) (b)

Gbr. 6 Antarmuka prototipe sistem *rendering* berbasis JPPF.

TABEL II
DATA HASIL UJI COBA

Aplikasi	Skenario	*Nodes* dalam *Grid*	WC	TC	ETC	Waktu eksekusi
AJNR	Test 3d fix bgt1.blend dengan 250 *frame* dan 1 *scene*	1	1	1	1	49 m 28 s
Blender	Test 3d fix bgt1.blend dengan 250 *frame* dan 1 *scene*	1	1	1	1	48 m 29 s
AJNR	Test 3d fix bgt1.blend dengan 250 *frame* dan 1 *scene*	5	1	1	1	10 m 40 s
Blender	Test 3d fix bgt1.blend dengan 250 *frame* dan 1 *scene*	5	1	1	1	12 m 21 s
AJNR	Test 3d fix bgt1.blend dengan 250 *frame* dan 1 *scene*	10	1	1	1	5 m 41 s
Blender	Test 3d fix bgt1.blend dengan 250 *frame* dan 1 *scene*	10	1	1	1	6 m 40 s



Gbr. 7 Perbandingan waktu *rendering* antara *Net Rendering* Blender 3D dengan aplikasi berbasis JPPF, AJNR.

Data yang diambil pada penelitian ini terbagi ke dalam beberapa kasus menggunakan satu *file* Blender 3D yang digunakan mahasiswa pada mata kuliah *Advance 3D Animation*. *File* tersebut di-*render* menggunakan dua solusi, yakni menggunakan *Net Rendering* Blender 3D dan dengan menggunakan AJNR. Hasilnya kemudian dibandingkan dan dianalisis. Gbr. 7 menunjukkan perbandingan waktu *rendering* antara kedua solusi.

Menggunakan rumus *Syntetic Workloads*, nilai *WC*, *TC*, dan *ETC* secara keseluruhan menunjukkan rasio 1, seperti ditunjukkan pada Tabel II. Ini berarti tidak ada kegagalan yang terjadi saat pengerjaan *rendering* berlangsung untuk masing-masing skenario. Setiap *file* pada masing-masing percobaan dapat dieksekusi dan di-*render* dengan baik.

Jika dilihat perbandingan keduanya berdasarkan waktu eksekusi, ada sedikit selisih yang menunjukkan kinerja AJNR lebih baik daripada *Net Rendering* Blender 3D, yang dapat dikalkulasi berdasarkan (4).

$$Efficiency\ (E) = \left(1 - \frac{Grid}{Single}\right) x100 \qquad (4)$$

Perbandingan kedua solusi saat dijalankan dengan menggunakan jaringan dengan sepuluh PC (sepuluh *nodes*/*slave*) menunjukkan:

$E_{(AJNR)}$ = ~88% (10 *node*)

$E_{(Blender)}$ = ~86% (10 *node*)

Secara keseluruhan dapat dijelaskan bahwa efisiensi hasil yang ditunjukkan baik menggunakan sistem *Net Rendering* Blender 3D maupun sistem *rendering* menggunakan JPPF tidak terlalu signifikan, tetapi solusi alternatif yang ditawarkan dengan menggunakan JPPF tidak lebih buruk dari penggunaan sistem *rendering* Blender 3D. Hal ini sesuai kenyataan bahwa baik spesifikasi PC dalam jaringan dan perangkat lunak pendukung di luar sistem seperti sistem operasi yang digunakan tidak mengalami perubahan dan dalam penelitian ini juga tidak ada improvisasi terhadap algoritme *rendering* yang digunakan. Sesuai tujuan, penelitian ini telah menunjukkan bahwa solusi alternatif tersebut memiliki kelebihan, yaitu penggunaan teknik melalui pemanfaatan beberapa *framework* yang ada ke dalam sistem *rendering* dapat dilakukan, dengan syarat *Command Line Interface* (CLI) dari perangkat lunak pendukung *rendering* tersedia. Misalnya saja dapat diujicobakan dengan mengganti CLI Blender 3D dengan produk multimedia yang lain, seperti Maya ataupun 3Ds Max.

V. KESIMPULAN

Ada beberapa kesimpulan yang dapat diambil dari hasil penelitian ini, yaitu sebagai berikut. Konfigurasi *rendering farm* (di luar pembuatan konfigurasi pendukung seperti rak *server*/PC dan kebutuhan pendukung perangkat keras lain) menggunakan Blender 3D sangat mudah dilakukan, karena Blender 3D sudah menyediakan fitur untuk melakukan *rendering* dalam jaringan. Yang dibutuhkan hanyalah instalasi Blender 3D di masing-masing PC dan penyediaan koneksi jaringan LAN/internet yang stabil. Selain itu, konfigurasi Blender 3D membutuhkan PC yang didedikasikan untuk proses *rendering*, sebab PC tersebut tidak dapat digunakan untuk kepentingan lain saat proses *rendering* berlangsung. Kemudian, konfigurasi *rendering file* Blender 3D alternatif berbasis *grid computing* dapat dihasilkan dengan

mengintegrasikan *framework* JPPF, Hazelcast, dan FFmpeg ke dalam sebuah aplikasi berbasis Java dengan memanggil proses eksternal Blender 3D yang disebut dengan istilah Blender CLI ke dalamnya. Program ini dapat dikembangkan lebih jauh untuk dibuat menjadi program yang dapat dipasang dan dijalankan secara otomatis saat PC dinyalakan. Program ini juga dapat dibuat berjalan sebagai *background process*, sehingga PC tetap memungkinkan digunakan untuk kepentingan yang lain. Selanjutnya, hasil kinerja prototipe sistem AJNR dengan menggunakan teknik *grid computing* dengan menggabungkan beberapa teknologi untuk mempercepat proses *rendering* dan melakukan konversi keluaran secara otomatis memberikan hasil yang cukup menjanjikan dengan tingkat efisiensi sebesar 88%. *Rendering* alternatif menggunakan JPPF memberikan efisiensi yang cukup baik dibandingkan dengan *rendering* menggunakan Blender 3D.

REFERENSI

[1] (2018) WBDL - Web Based Digital Library. [Online], http://pti.undiksha.ac.id/digilib/, tanggal akses: 13 Januari 2018.

[2] J. M. Blain, *The Complete Guide to Blender Graphics*, CRC Press, 2013.

[3] P. Bui, T. Boettcher, N. Jaeger, and J. Westphal, "Using Clusters in Undergraduate Research: Distributed Animation Rendering, Photo Processing, and Image Transcoding," *Proc. - IEEE Int. Conf. Clust. Comput. ICCC*, 2013.

[4] F. Berman, G. Fox, and T. Hey, *Grid Computing (Making the Global Infrastructure a Reality)*. John Wiley and Sons, 2003.

[5] (2017) JPPF Framework, [Online], http://jppf.org/, tanggal akses: 19 Juli 2017.

[6] J. Xiong, J. Wang, and J. Xu, "Research of Distributed Parallel Information Retrieval Based on JPPF," *Proc. - 2010 Int. Conf. Inf. Sci. Manag. Eng. ISME 2010*, Vol. 1, 2010, hlm. 109–111.

[7] B. . Argarwal, S. . Tayal, and M. Gupta, *Software Engineering & Testing (An Introduction)*. Jones And Bartlett Publishers, 2010.

[8] R. K. Yin, *Case Study Research: Design and Methods*, 4th ed. SAGE Publications, Inc., 2009.

[9] D. H. J. Epema, C. Franke, A. Iosup, L. Schley, B. Song, and R. Yahyapour, *On Grid Performance Evaluation Using Synthetic Workloads*, part of Lecture Notes in Computer Science, Berlin, Heidelb.: Springer, 2006.

[10] K. Trammell. (2016) Setting Up a Renderfarm, [Online], https://cgcookie.com/tutorial/setting-up-a-renderfarm, tanggal akses: 19 Juli 2017.

[11] (2017) FFmpeg. [Online], http://ffmpeg.org/, tanggal akses: 19 Juli 2017.

[12] (2017) Hazelcast.org - The Leading Open Source In-Memory Data Grid, [Online], https://hazelcast.org/, tanggal akses: 19-Jul-2017.