MANAJEMEN GRID UNTUK RENDER ANIMASI 3 DIMENSI

Arthur Mourits Rumagit[1)], **Moch. Hariadi**[2)]
[1,2)]Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember Kampus Sukolilo, Surabaya 60111
e-mail : [1)]arthur_el@elect-eng.its.ac.id ; arthuro_el@yahoo.com, [2)]mochar@ee.its.ac.id

Abstrak
Masalah utama dalam melakukan proses *render* yaitu waktu yang dihasilkan oleh *CPU* untuk melakukan *rendering*. Untuk mengatasi masalah tersebut dikemukakannya *grid computing* dengan memanfaatkan sumber daya yang ada sebagai *suatu rendering farm*. *Grid computing* adalah pemanfaatan sumber daya atau penggabungan *resource-resource* yang terpisah secara aman dan mudah sehingga tecipta suatu lingkungan komputasi yang besar. *Rendering farm* adalah sekumpulan dari komputer yang bekerja secara bersama sama untuk melakukan proses *render*. Dibutuhkan suatu *tools* untuk melakukan proses *parallel rendering*. *Yadra* akan sangat efektif bila digunakan untuk merender suatu animasi, sebab yadra bekerja dengan cara memecah dan mendistribusikan *frame* dari *file* animasi kepada setiap komputer. Dalam penelitian ini yadra akan di implementasikan pada sistem *grid* yang telah ada di Jurusan Teknik Elektro ITS.
Keyword : *Rendering, Grid Computing, Rendering Farm*

1. PENDAHULUAN

Semakin meningkatnya kesadaran peneliti akan manfaat komputer dalam penelitian, menyebabkan semakin banyak usaha untuk menjadikan komputer sebagai alat bantu sehari-hari diberbagai bidang. Astronomi, Fisika, Biologi, mekanika dan kimia adalah beberapa contoh bidang yang paling banyak memanfaatkan komputer. Hal ini tidaklah mengherankan mengingat akan terjadi penghematan biaya dan pengurangan resiko besar-besaran ketika menggunakan komputer dibandingkan cara konvensional. Namun tidak dapat dipungkiri bahwa aplikasi-aplikasi tersebut seringkali memerlukan *resource* yang sangat besar, sehingga tidak efisien apabila hanya dikerjakan dengan sebuah *PC*. Berbagai cara dilakukan untuk mengatasi masalah ini, salah satunya dengan menggunakan *supercomputer* dan komputer *mainframe*. Ternyata cara ini dianggap belum sesuai mengingat komputer *mainframe* harganya sangat mahal. Seiring dengan berkembangnya kecepatan prosesor serta harganya yang semakin murah di pasaran dari waktu ke waktu. Dikemukakan suatu solusi *cluster computing*, dimana beberapa komputer dihubungkan menggunakan jaringan untuk dapat saling bekerja sama dalam melakukan tugas tertentu. Hal ini dimungkinkan dengan *parallel programming* dimana suatu program akan dipecah menjadi proses-proses yang lebih kecil dan selanjutnya akan dikirim ke *node-node* untuk dieksekusi secara simultan. *PVM (Parallel Virtual Machine) dan MPI (Message Passing Interface)* adalah contoh *library parallel* yang banyak digunakan. Solusi ini sangat diminati dan populer karena membutuhkan biaya yang sangat murah dan kinerja yang cukup meyakinkan. Berbagai institusi bisnis dan pendidikan banyak mengadopsi cara ini di departemennya masing-masing. Sehingga terbentuk *resource - resource* yang tersebar secara geografis. Fenomena ini menimbulkan ide bagaimana jika *resource-resource* yang banyak dan tersebar itu digabungkan? Karena bisa dibayangkan apabila *resource-resource* tersebut dapat digabungkan, maka akan terbentuk suatu lingkungan komputasi yang sangat besar. Solusi ini dikenal sebagai *grid computing*, *grid computing* menawarkan pemanfaatan bersama *resource-resource* yang terpisah secara geografis secara aman dan mudah. Setiap orang bisa menggunakannya secara bebas sesuai kebutuhan masing-masing, namun juga tetap memberikan kuasa penuh terhadap pemilik *resource* untuk mengatur kebijakannya. *Terminologi grid* ini diadopsi dari istilah *grid* dari bidang ketenagalistrikan, dimana setiap orang bisa memanfaatkan listrik dimana saja yang berasal dari beberapa sumber listrik yang tersebar luas. *Rendering* adalah suatu proses untuk mengubah model geometri menjadi suatu gambar. Proses untuk membangun sebuah gambar membutuhkan beberapa fase seperti *modeling*, pengaturan *material* dan *texture*, penempatan *virtual light*, dan proses render. Algoritma untuk melakukan proses render mendefinisikan model, geometri, pengaturan *material* dan *texture*, serta penempatan *virtual light* sebagai inputan dan menghasilkan gambar (atau sequence image untuk animasi) sebagai output. Salah satu tujuan dari penggunaan komputer grafik adalah dapat melakukan proses render dari suatu model *photorealistic* yang memiliki kualitas tinggi dengan adegan yang komplek. Dampak dari hal ini adalah dibutuhkannya suatu proses komputasi yang intensif dan membutuhkan waktu yang sangat banyak. Pada umumnya proses render yang dilakukan pada saat ini masih bekerja secara single dengan menggunakan sebuah mesin yang memiliki sumber daya yang besar. Kerugian dari proses render yang bekerja secara single adalah :

1. Waktu yang diperlukan untuk melakukan proses ini masih dirasakan cukup lama walaupun mesin yang digunakan memiliki sumber daya yang cukup besar
2. Selama proses rendering, mesin tidak dapat digunakan untuk keperluan lain. Hal ini dikarenakan rendering membutuhkan suatu proses komputasi yang intensif dan sumber daya yang cukup besar.

Pada dasarnya penelitian ini mempunyai tujuan yaitu memanfaatkan lingkungan *grid* yang sudah ada sebagai suatu *rendering farm* yang nantinya dapat digunakan untu proses rendering dengan waktu yang relative cepat.

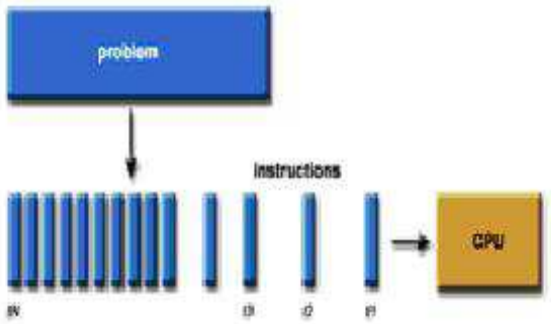
2. TINJAUAN PUSTAKA

a. Parallel Programing

Parallel computing adalah penggunaan lebih dari satu sumber daya komputasi secara simultan untuk memecahkan persoalan komputasi. *Software* tradisional umumnya dibuat untuk komputasi serial (Gambar 1) yang dijalankan oleh prosesor tunggal dimana sebuah persoalan dipecah ke dalam instruksi yang dieksekusi secara berurutan dan hanya satu instruksi yang boleh dieksekusi pada saat yang sama. Pada komputasi paralel (Gambar 2) sebuah persoalan dipecah menjadi beberapa bagian yang dapat diselesaikan pada saat yang bersamaan. Setiap bagian selanjutnya dipecah menjadi instruksi yang berurutan dan masing-masing bagian dikerjakan oleh prosesor yang berbeda. Sumber daya komputasi yang dimaksud dapat berupa : sebuah komputer tunggal dengan beberapa prosesor; beberapa komputer yang terhubung dalam jaringan; maupun kombinasi dari keduanya.

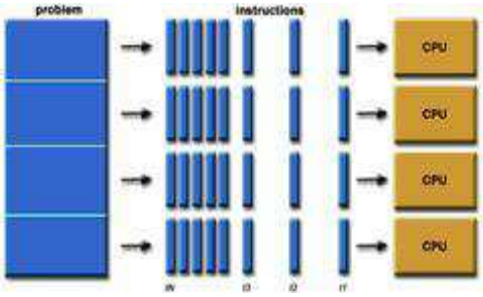
Alasan utama penggunaan komputasi paralel adalah :

- Dapat menghemat waktu

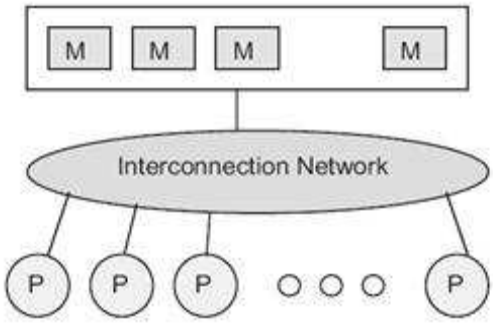


Gambar 1: Komputasi Serial

- Dapat memecahkan persoalan yang lebih besar
- Dapat menghemat biaya
- Dapat mengatasi keterbatasan fisik dari komputasi serial.



Gambar 2 : Komputasi Parallel



Gambar 3: Diagram blok sistem shared-memory

Alasan lain yang juga cukup penting adalah *fault tolerant*. Jika salah satu dari prosesor mengalami kegagalan prosesor yang lain dapat menggantikannya, meskipun dengan performa yang menurun.

b. Cluster Computer dan DRM

Untuk mengimplementasikan konsep komputasi paralel dibutuhkan cluster computer dan *Distributed Resource Management (DRM)* untuk memanajemen pekerjaan dan sumber daya seperti : memori, prosesor, dan *storage.*

Cluster Computer

Cluster Computer adalah kumpulan dari komputer-komputer yang terkoneksi melalui jaringan lokal berkecepatan tinggi dan didesain untuk digunakan sebagai sumber daya komputasi yang terintegrasi atau sumber daya untuk pemrosesan data. Sebuah *cluster* memiliki beberapa karakteristik antara lain :

- Terdiri dari beberapa mesin-mesin bertipe sama
- *Tightly-coupled* menggunakan koneksi jaringan yang *dedicated*
- Semua mesin berbagi sumber daya, contohnya adalah direktori home
- Harus percaya satu sama lain, sehingga rsh maupun ssh tidak memerlukan *password*
- Harus mempunyai *software* seperti implementasi MPI yang memungkinkan program-program dijalankan di semua *node*.

Ada dua faktor yang menyebabkan *cluster* menjadi kompleks, yaitu :

1. Skala fisik yang bertambah : Sebuah *cluster* dapat terdiri dari beberapa ratus atau ribuan prosesor, hal ini menyebabkan algoritma alternatif dibutuhkan oleh suatu manajemen sumber daya dan fungsi kontrol.
2. Integrasi yang berkurang : Keinginan untuk membentuk *cluster* dari berbagai komoditas menyebabkan *cluster* tidak lebih terintegrasi dibandingkan dengan *end system.*

Cluster yang memperhatikan performa paralel dihadapkan pada masalah *coscheduling*, yaitu sebuah teknik untuk menjadwalkan proses-proses komputasi secara simultan pada prosesor yang berbeda. Pada komputer paralel yang terintegrasi, *coscheduling* dapat dilakukan dengan menggunakan sebuah *batch scheduler* yang juga bisa disebut sebagai *job scheduler*

Distributed Resource Management (DRM)

Distributed Resource Management (DRM) adalah suatu sistem yang dapat mengatur pemanfaatan sumber daya terdistribusi untuk menjalankan suatu *job*. Penggunaan sekumpulan mesin yang terhubung dalam sebuah jaringan demi penyediaan sumber daya komputasi memerlukan sebuah sistem yang dapat mengatur penggunaan sumber daya yang ada tersebut. Pengaturan diperlukan agar sumber daya yang ada dapat digunakan secara optimal. DRM ini sering juga disebut sebagai sistem penjadwalan *job (job scheduler)* karena tugasnya memang melakukan penjadwalan eksekusi *job* dalam sumber daya yang tersedia. Saat sebuah *job* dikirimkan, *job* akan masuk ke dalam sebuah antrian *job (jobs queue)*. Saat ada sumber daya yang dapat digunakan, *job* dalam antrian tadi akan diberikan ke sumber daya untuk dieksekusi. Disebutkan bahwa ada beberapa komponen dalam suatu DRM, yaitu :

1. *Batch queueing* sebagai tempat antrian *job* yang dikirimkan. *Job* akan berada pada antrian ini sampai ada mesin yang siap untuk menjalankan *job* tersebut.
2. *Resource management* yang bertugas untuk mengirimkan *job* yang berada dalam antrian ke sumber daya lalu menjalankannya.
3. *Load management* akan berurusan dengan beban dari masing-masing mesin *grid Computing, cluster computer* dan komputasi paralel yang ada. *Load management* ini harus mampu mendeteksi beban dari setiap mesin yang ada *(load measurement)* lalu mengkategorikan mesin-mesin tersebut berdasarkan bebannya *(load evaluation)*. Selain itu, pengaturan distribusi beban juga dapat dilakukan dengan cara melakukan pemindahan beban dari satu mesin ke mesin yang lain *(load migration).*

Ketiga komponen ini akan bekerja bersama dalam mendukung suatu sistem penjadwalan *job*. Contoh dari sistem penjadwalan *job* yang ada adalah *Condor* dan *Sun Grid Engine (SGE). Condor* merupakan sistem penjadwalan *job* yang didukung oleh pengembang *Globus Toolkit* sedangkan *SGE* merupakan sistem penjadwalan job komersil yang didukung oleh *Sun Microsystems.*

c. Grid Computing

Kebutuhan pengaksesan informasi kapanpun dan dimanapun sekarang ini mendorong penyediaan sumber daya komputasi yang lebih *powerful*. Dunia *science* dan industri semakin hari juga semakin membutuhkan sumber daya komputasi yang sangat besar. Simulasi gempa bumi, pemodelan struktur molekul kimia, dan simulasi perubahan iklim adalah beberapa contoh aplikasi yang melibatkan data yang sangat besar mencapai satuan petabyte (1015 byte). Hal ini tentu saja membutuhkan sumber daya komputasi yang besar pula. Dengan semakin banyaknya sumber daya komputasi yang saling terhubung jaringan, *grid computing* memberikan solusi untuk mengatasi permasalahan itu. Komputasi *grid* memungkinkan penggunaan sumber daya yang melibatkan banyak

komputer yang terdistribusi dan terpisah secara geografis untuk memecahkan persoalan komputasi dalam skala besar. Setiap orang nantinya bisa menggunakan sumber daya komputasi ini sesuai kebutuhannya. Jumlah sumber daya komputasi yang ada dalam *grid* adalah tak terbatas, semua orang bisa menyumbangkan sumber dayanya untuk dipergunakan bersama-sama untuk kepentingan umat manusia. Konsep yang mendasar dari sebuah Grid adalah koordinasi sumber daya bersama dan penyelesaian masalah dalam sebuah organisasi virtual (Virtual Organization - VO) yang dinamis dan terdiri dari berbagai institusi . Ian Foster, dalam makalahnya yang berjudul *"What is the Grid? A Three Point Checklist"*, mengusulkan tiga hal yang harus ada dalam sebuah lingkungan komputer terdistribusi sehingga dapat dikatakan sebagai suatu grid. Tiga hal tersebut adalah sebagai berikut :

1. Mengkoordinasikan sumber daya - sumber daya yang tidak dikendalikan secara terpusat. Sebuah Grid mengintegrasikan dan mengkoordinasikan sumber daya dan pengguna yang berada dalam domain kendali yang berbeda, dengan kata lain berada di bawah manajemen lokal.
2. Menggunakan protokol dan antar muka yang standar, terbuka, dan dapat digunakan untuk berbagai macam hal *(general purpose)*. Sebuah *grid* dibangun menggunakan protokol dan antar muka yang menangani masalah fundamental seperti otentifikasi, otorisasi, pencarian, dan penggunaan sumber daya.
3. Memberikan *quality of service* yang canggih, yang jauh diatas kualitas layanan komponen individu dari komputasi grid tersebut, contohnya dalam hal waktu respon, *throughput*, ketersediaan sumber daya, keamanan, dan/atau penggunaan beberapa jenis sumber daya yang sesuai dengan kebutuhan pengguna.

Sedangkan menurut DR. Rajkumar Buyya, *grid* adalah salah satu bentuk sistem paralel dan terdistribusi yang memungkinkan sharing, pemilihan, dan pengumpulan sumber daya autonomous yang terdistribusi secara geografis, dengan cara dinamis saat *runtime* tergantung keberadaan, kemampuan, performansi, biaya, dan *Quality of Service* yang dibutuhkan *user*. Jadi secara umum dapat dikatakan bahwa konsep yang mendasari sebuah *grid* adalah koordinasi sumber daya bersama dan penyelesaian masalah bersama dalam sebuah lingkungan organisasi *virtual (VO)* yang dinamis yang terdiri dari berbagai institusi. Sedangkan aspek kekuatan komputasi dalam sebuah *grid* meliputi :

- Kemampuan menangani sumber daya komputasi yang berada dibawah manajemen yang berlainan.
- Kemampuan menyediakan mekanisme pemilihan sumber daya komputasi secara cerdas dan transparan terhadap *user*.
- Kemampuan memprediksi beban kerja, ketersediaan, konfigurasi dinamis serta pembagian sumber daya komputasi yang ada
- Kemampuan mendeteksi kesalahan serta mekanisme penanganannya.
- Kemampuan menangani mekanisme keamanan yang sesuai untuk pengaksesan sumber daya komputasi secara aman.

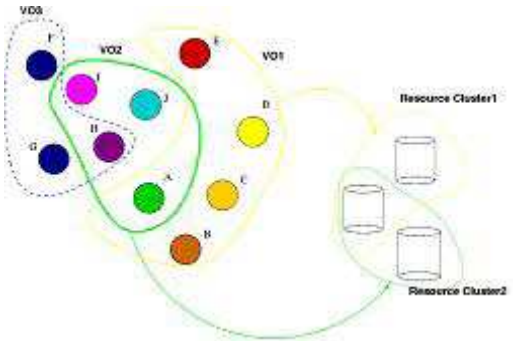
Sedangkan penanganan data yang ada dalam sebuah lingkungan *grid computing* harus mengatur secara efektif semua aspek data, meliputi lokasi data, transfer data, pengaksesan data serta keamanan data. Secara lengkap adalah sebagai berikut :

- Kemampuan untuk mengintegrasikan beberapa sumber data yang terdistribusi, heterogen serta *independent.*
- Kemampuan menyediakan mekanisme transfer data yang efisien untuk penyediaan data, dimana proses komputasi akan dilakukan dalam rangka mencapai skalabilitas dan efisiensi yang lebih baik.
- Kemampuan penanganan data *caching* serta mekanisme replikasi data untuk meminimalkan lalu lintas jaringan.
- Kemampuan penyediaan mekanisme pencarian data yang memungkinkan *user* mencari data sesuai dengan karakteristik data yang diinginkan.
- Kemampuan menerapkan pengenkripsian data serta pengecekannya untuk menjamin data yang dilewatkan jaringan berada dalam keadaan aman.

Kemampuan untuk menyediakan mekanisme *backup* dan *restore* untuk mencegah kehilangan data. Kebutuhan akan bentuk-bentuk layanan data pada lingkungan *grid computing* menfokuskan pada manajemen data yang diaplikasikan pada media penyimpanan data yang terpisah secara geografis. Sumber data dapat berupa *database*, *file system*, dan media penyimpanan *(storage)*. Sistem *grid* juga harus dapat menyediakan layanan visualisasi data untuk pengaksesan, penggabungan dan pemrosesan data secara transparan. Untuk mencapai hal tersebut, kebutuhan keamanan dan privasi data menjadi sesuatu yang komplek. Selanjutnya definisi tentang pembagian sumber daya pada *grid computing* terus berkembang, berdasarkan pengalaman sebelumnya, yaitu dengan lebih menfokuskan pada pembagian sumber daya melalui suatu *Virtual Organization (VO)*. Sedangkan penanganan data yang ada dalam sebuah lingkungan *grid computing* harus mengatur secara efektif semua aspek data, meliputi lokasi data, transfer data, pengaksesan data serta keamana data. Selengkapnya sebagai berikut :

- Kemampuan untuk mengintegrasikan beberapa sumber data yang terdistribusi, *heterogen* serta *independen*.
- Kemampuan menyediakan mekanisme transfer data yang efisien untuk penyediaan data dimana proses komputasi akan dilakukan dalam rangka mencapai skalabilitas dan efisiensi yang lebih baik.
- Kemampuan penanganan data *caching* serta mekanisme replikasi data untuk meminimalkan lalu lintas jaringan.

- Kemampuan penyediaan mekanisme pencarian data yang memungkinkan *user* mencari data sesuai dengan karakteristik data yang diinginkan.



Gambar 4: Virtual Organization

- Kemampuan menerapkan pengenkripsian data serta pengecekannya untuk menjamin data yang dilewatkan jaringan berada dalam keadaan aman.
- Kemampuan untuk menyediakan mekanisme *backup* dan *restore* untuk mencegah kehilangan data.

Kebutuhan akan layanan data pada lingkungan *grid computing* menfokuskan pada manajemen data yang diaplikasikan pada media penyimpanan data yang terpisah secara geografis. Sumber data dapat berupa *database, file system*, dan media penyimpanan. Sistem *grid* juga harus dapat menyediakan layanan visualisasi data untuk pengaksesan, penggabungan dan pemrosesan data secara transparan. Untuk mencapai hal tersebut, kebutuhan keamanan dan privasi data menjadi sesuatu yang komplek. Selanjutnya definisi tentang pembagian sumber daya pada *grid computing* terus berkembang, yaitu dengan lebih menfokuskan pada pembagian sumber daya melalui suatu *Virtual Organization (VO)*.

d. Gridway

Gridway Metascheduler adalah suatu *tools* yang memungkinkan untuk berbagi sumber daya komputasi dengan jangkauan yang luas, realible dan efisien, diatur oleh *system* LRM (*Local Resource Management*) yang berbeda, seperti PBS, SGE, Condor dan lainnya, didalam satu organisasi (*enterprise grid*) atau lintas *administrative domain* (*partner atau supply-chain grid*). Gridway mendukung *end user* dengan menyediakan lingkungan kerja dan fungsi yang hampir sama dengan DRM *system local*, seperti SGE, LSF, atau PBS. *End user* dapat *mensubmit*, memonitor, dan mengontrol suatu *job* dengan *command* yang hampir sama dengan DRM seperti (gwsubmit, gwwait, gwkill, gwhost...). Secara khusus gridway memiliki kelebihan :

- Dalam mengeksekusi *job* Gridway bersifat reliable maksudnya *end user* dapat melihat *job* secara transparan, selain itu *scheduler* dari gridway mampu mengatur beberapa kegagalan.
- Efisien dalam mengeksekusi *job* maksudnya *job* akan dieksekusi pada cluster yang tercepat (yang menyediakan banyak *resource*) .
- Gridway memiliki *DRM command line interface* yang hamper sama dengan dengan *command line* yang ada pada unix dan *DRM system* lain seperti SGE atau PBS.

Dalam memperlakukan *job* yang telah disubmit gridway memiliki kesamaan dengan proses yang terjadi pada unix. Setiap *job* diberi nomor identitas, yang dianologikan sebagai PID dari proses. Sedangkan nilai dari angka yang diberikan disebut degan *Job Identifier*,atau disingkat dengan JID. Jika job yang disubmit adalah array job, gridway juga akan memberikan identitas yang berupa array *identifier* atau disingkat sebagai AID. Indeks dari *job* yang terdapat didalam *array* dinamakan dengan *task identifier*, atau disingkat TID.

Jobs disubmit dengan menggunakan perintah gwsubmit. Sebuah job dideskripsikan berdasarkan file templatenya, disini *user* dapat menspesifikasi *executable file*, *command line arguments, input* atau *output files* dan juga beberapa aspek lainnya. *Job* dapat dimonitor dengan menggunakan perintah gwps, selain itu *user* juga dapat mengontrol *job* dengan menggunakan perintah gwkill. *User* juga dapat mensinkronkan antara satu *job* dengan *job* yang lain dengan perintah gwwait. Perintah gwhistory digunakan untuk mengetahui *resource* mana yang telah digunakan untuk mengeksekusi suatu *job*.

Sistem monitoring *commands* dapat mengizinkan *user* untuk mengumpulkan informasi tentang sistem dari gridway. Perintah ini adalah gwuser digunakan untuk menunjukkan informasi tentang *user* yang menggunakan fasilitas gridway, *gwhost* digunakan untuk memonitoring *host* yang tersedia dan *gwacct* digunakan untuk mencetak informasi setiap *user* atau setiap *host*.

e. Rendering

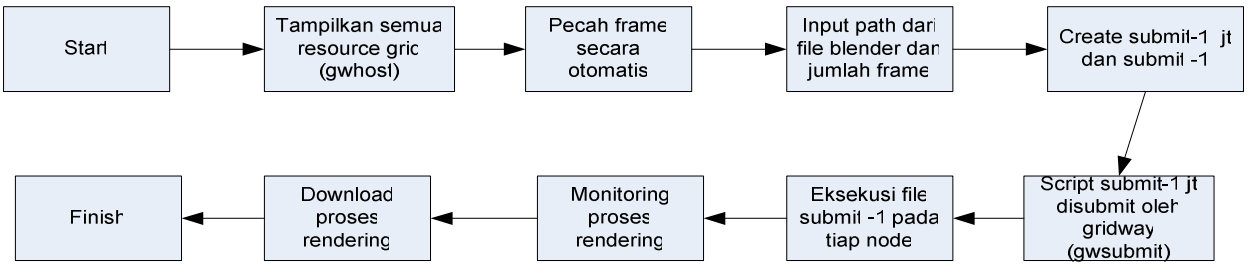
Rendering memainkan peran yang sangat vital pada proses penciptaan animasi dan gambar. Rendering dapat digunakan untuk meniru objek visual yang nyata (*photorealistic rendering*), atau *stylistic fashion (non-photorealistic renderings*) dengan baik. Dengan mengesampingkan *style,* rendering dapat diselesaikan dengan menggunakan algoritma yang memang dikhususkan untuk proses render. Algoritma ini bisa menjadi sangat kompleks. Salah satu contohnya adalah *ray tracing*, *ray tracing men-generate* citra dengan menjejaki sinar dari lampu per piksel dipandang dari kamera ke suatu adegan melalui layar virtual. Semakin kompleks algoritma yang digunakan maka semakin lama waktu yang dibutuhkan untuk melakukan proses kalkulasi. Dengan menggunakan *hardware* yang moderen proses render dari film sederhana dapat menghabiskan waktu antara beberapa menit sampai dengan dua ratus menit untuk setiap frame-nya. Salah satu contoh extreme yang dapat diambil adalah proses rendering pada adegan keramaian di film Sherk 2 dimana untuk setiap *frame*nya membutuhkan waktu render sampai dengan empat puluh jam.

f. Yadra (Yet another distributed rendering application)

Yadra adalah suatu *tools* yang menyediakan pendistribusian untuk proses render pada animasi blender di suatu computer cluster. Yadra mudah untuk dikonfigurasi dan operasinya stabil. Tidak membutuhkan konfigurasi pada *share jaringan* (CIFS/SMB membuat proses render pada jaringan sedikit ada kesulitan). Yadra digunakan dengan *java - independen platform*, direkemondasikan untuk menggunakan *java - independent platform* versi-5 keatas . Hal ini berlaku pada setiap *platform, platform* yang digunakan adalah windows dan linux.
Pengerjaan rendering pada jaringan membutuhkan satu *master* dan sedikitnya satu *slave*. Satu *Master* dan Satu *slave* dapat *diinstall* pada satu mesin. Namun pada umumnya penginstalan *master* dan *slave* diletakkan pada mesin yang berbeda. *Master* hanya mengontrol distribusi proses rendering pada jaringan (cluster). *Slave* bertanggung jawab pada proses pengerjaan rendering. Setiap *slave* melakukan proses render pada *frame* yang menjadi job setiap *slave*. Semua hasil akhir render akan dikirimkan ke *master* dan dapat di *download* melalui *web interface* yadra.

3. METODE PENELITIAN

Hal pertama yang dilakukan adalah pemilihan rendering engine pemilihan ini didasarkan pada kemampuan dan feature yang dimiki oleh rendering engine tersebut, setelah itu proses pencarian literatur dan informasi yang berakaitan dengan rendering engine dilakukan, hal yang perlu diperhatikan adalah kemampuan dari rendering engine untuk melakukan proses rendering secara parallel (dengan menggunakan computer cluster). Langkah selanjutnya adalah mengimplementasikan rendering engine pada lingkungan grid yang sudah ada, dalam hal ini adalah grid elektro. Setelah proses implementasi maka dilakukan tahap pengujian, proses pengujian dilakukan dengan melihat performance dari rendering farm.



Gambar 5: Alur eksekusi proses render pada grid sebagai rendering farm

4. HASIL DAN PEMBAHASAN

Dengan memanfaatkan *grid computing* sebagai suatu *rendering farm* maka dapat melakukan proses render dengan menggunakan *tools* bantuan yaitu yadra yang dapat melakukan proses render untuk file animasi dengan waktu yang relative cepat.

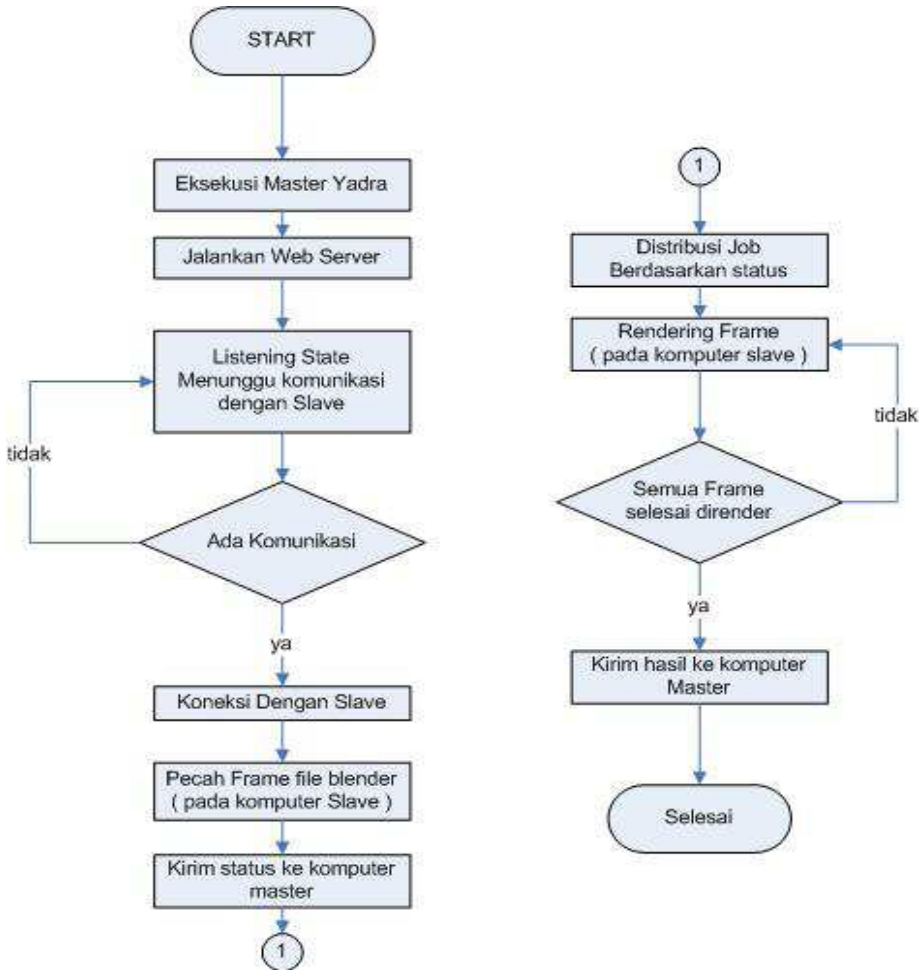
Yadra (Yet another distributed rendering application)

Yadra adalah suatu *tools* yang menyediakan pendistribusian untuk proses render pada animasi blender disuatu computer cluster. Yadra mudah untuk dikonfigurasi dan operasinya stabil. Tidak membutuhkan konfigurasi pada share jaringan (CIFS/SMB membuat proses render pada jaringan sedikit ada kesulitan). Yadra digunakan dengan *java - independen platform*, direkemondasikan untuk menggunakan *java _ independent platform* versi-5 keatas . Hal ini berlaku pada setiap *platform*, *platform* yang digunakan adalah windows dan linux. Pengerjaan rendering pada jaringan membutuhkan satu master dan sedikitnya satu slave. Satu Master dan Satu slave dapat diinstall

pada satu mesin. Namun pada umumnya penginstalan master dan slave diletakkan pada mesin yang berbeda. Master hanya mengontrol distribusi proses rendering pada jaringan (cluster). Slave bertanggung jawab pada proses pengerjaan rendering. Setiap slave melakukan proses render pada frame yang menjadi *job* setiap slave. Semua hasil akhir render akan dikirimkan ke master dan dapat di *download* melalui *web interface* yadra. Yang pertama kali perlu dilakukan untuk membangun sebuah *renderfarm* berbasis yadra adalah instalasi master dari Yadra, setelah itu instalasi slave baru dapat dilakukan. Urutan instalasi ini tidak dapat dibalik karena konfigurasi pada slave baru dapat dilakukan apabila master dari yadra dalam keadaan aktif. Komunikasi antara slave dan master dienkripsikan melalui *passphrase*. Master dan slave dari Yadra harus memiliki *passphrase* yang sama jika tidak komunikasi antara master dan slave tidak dapat terlaksana.

Alur proses rendering Yadra

Proses rendering dimulai dengan mengeksekusi komputer master dari yadra, setelah komputer master dieksekusi maka yadra akan membuka sebuah port yang digunakan untuk komunikasi dengan computer slave. Selain itu yadra juga akan menjalankan *webserver* yang nanti dapat digunakan oleh *client* untuk memonitor dan *mendownload job* hasil dari proses rendering. Pada saat ini master berada pada posisi *listening*. Pada setiap komputer slave frame dari file blender dipecah menjadi beberapa bagian dan hasil dari proses tadi dikirim pada komputer master untuk dijadikan sebagai status dari slave, komputer master mengatur pendistribusian dari render-jobs berdasarkan status yang dikirim oleh setiap slave. Untuk proses rendering dilakukan sepenuhnya oleh komputer slave. Semua frame yang telah selesai di render akan disimpan di komputer master dan dapat di*download* oleh *user*. Untuk lebih jelasnya dapat dilihat pada gambar 6.



Gambar 5: Flow chart proses render yadra

5. KESIMPULAN

Setelah melalui tahapan implementasi dan pengujian sistem, maka diperoleh beberapa kesimpulan antara lain : *Rendering farm* dengan Yadra sebagai *tools* untuk rendering telah berhasil diimplementasikan pada linkungan komputasi *grid* di Jurusan Teknik Elektro ITS. Dengan melakukan proses render pada *grid computing* sebagai

suatu *rendering farm* terdapat hasil yang sangat signifikan (lebih cepat) dibandingkan dengan merender di *single machine*.

6. DAFTAR PUSTAKA

Blaise Barney, 2008. Introduction to Parallel Computing, https://computing.llnl.gov/tutorials/parallel_comp/.

I. Foster dan C. Kesselman, 2004. The Grid : Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers

Fajran Iman Rusadi, 2006. Evaluasi Lingkungan Pemrograman Paralel Berbasis Message Passing Interface dalam Infrastruktur Komputasi Grid di Universitas Indonesia.

Ursula Maier dan Georg Stellner, 2008. Distributed Resource Management for Parallel Applications in Networks of Workstations, http://citeseer.ist.psu.edu/maier97distributed.html.

Ian Foster, 2002. What is the Grid? A Three Point Checklist. Grid Today, http://www-fp.mcs.anl.gov/~foster/Articles/ WhatIsTheGrid.pdf.

Joshy Joseph and Craig Fellenstein, 2003. Grid Computing. Prentice Hall.

GridWay Team, 2007 GridWay 5.2 Documentation: User Guide, Universidad Complutense de Madrid

S. Lee Gooding, Laura Arns, Preston Smith, and Jenett Tillotson, Implementation of a Distributed Rendering Environment for the TeraGrid, Purdue University