Implementasi *Cluster Computing* Untuk Render Animasi

Lysy C. Moleong, Arthur M. Rumagit, Brave A. Sugiarso
Jurusan Teknik Elektro-FT. UNSRAT, Manado-95115, Email: licyauau@gmail.com

***Abstrak*—Komputer merupakan nama yang sudah sangat populer dalam segala aspek kehidupan. Pada saat ini kebutuhan akan komputasi yang besar sudah meningkat sedangkan jika hanya menggunakan satu buah *personal computer* (PC), sudah tidaklah memungkinkan. Untuk memenuhi kebutuhan komputasi yang besar maka dibentuklah *cluster computing* yang menghubungkan sejumlah komputer individual kedalam suatu jaringan.**

Sistem operasi komputer anggota cluster yang digunakan adalah sistem operasi Linux. Distribusi Linux yang dipakai adalah distribusi Debian GNU/Linux versi Sarge. Sistem ini di instal pada setiap komputer yang akan digunakan sebagai node. *Rendering engine* yaitu *blender* juga diinstal pada tiap node untuk melakukan proses *render*. Proses *render* dimulai dengan mengeksekusi komputer master dari yadra, setelah itu yadra akan membuka sebuah part yang digunakan untuk komunikasi dengan *computer slave*. Selain itu yadra akan menjalankan *webserver* yang nanti dapat digunakan oleh client untuk memonitor dan mendownload *job* hasil dari proses *rendering*.

Dengan adanya komputer cluster maka pengerjaan proses rendering akan lebih muda dan dapat diselesaikan dengan waktu yang singkat dibandingkan dengan satu buah *personal computer* saja.

***Kata Kunci*: Blender, Cluster, Render, Yadra**

Abstract – Computer is a thing that has been very populer in every human life. Now, the requirement of a large computing has arise while one personal computer is not enough. To fulfill this requirement, cluster computing is built for interconnect one personal computer to others into one network.

Linux operating system is used for client cluster computing wherein the linux distribution is Debian GNU/Linux Sarge version. This operating system is installed in every node. Blender is the rendering engine that also installed at every node and make a rendering. Rendering is beginning with execute yadra master, then open a part that used for communication with slave. Yadra run webserver that will be used by client to monitoring and job downloading as result from rendering.

By cluster computer, the rendering become easier and in a short time than one personal computer.

Keyword : Blender, Cluster, Rendering, Yadra.

I. PENDAHULUAN

Komputer merupakan nama yang sudah sangat populer dalam segala aspek kehidupan. Perkembangan komputerpun terasa sangat cepat, baik dari segi perangkat lunak maupun keras, dimana kedua aspek tersebut saling berkaitan dan tidak dapat dipisahkan. Perkembangan ini membawa dampak yang positif terhadap kehidupan banyak orang, dimana berbagai aplikasi mulai banyak dikembangkan untuk mempermudah segala aktifitas manusia.

Server merupakan suatu elemen penting dalam suatu sistem informasi. Dimana server merupakan sebuah sistem komputer yang menyediakan jenis layanan tertentu dalam sebuah jaringan komputer. Sehingga sebuah server harus memiliki kemampuan yang berkali–kali lipat dari komputer pribadi dan dengan harga yang berkali–kali lipat pula. Analisis yang lebih khusus jelas akan membutuhkan daya komputasi yang lebih tinggi.

Salah satu solusi dari masalah kurangnya daya komputasi adalah dengan menjalankan aplikasi pada sejumlah komputer individual yang terhubung ke jaringan. Cara ini dalam terminologi teknis dikenal sebagai *clustering*. Teknik yang pertama kali dikembangkan pada awal era 1980-an ini, sekarang telah diaplikasikan pada berbagai pusat superkomputer, laboratorium riset, dan industri. Superkomputer tercepat di dunia saat ini terdiri dari sekumpulan mikroprosesor, sebagai contoh, sistem ASCI White di *Lawrence Livermore National Laboratory, California*, yang tersusun atas 8000 prosesor. Banyak diantara laboratorium riset yang menjalankan PC sederhana yang membentuk *cluster* untuk melakukan perhitungan dan analisis data. Teknik ini hanya memerlukan ongkos sebesar kurang dari 1 USD per megaflop tiap detiknya dengan *cluster* komputer jenis Pentium III, sebuah ongkos yang sangat murah, khususnya apabila dibandingkan dengan superkomputer yang harganya bisa mencapai jutaan dolar itu. Kemajuan ini juga tidak lepas dari dikembangkannya algoritma khusus yang dapat mengeksploitasi penggunaan ribuan prosesor secara efektif.

Dalam solusi *cluster computing*, beberapa komputer yang dihubungkan menggunakan jaringan untuk dapat saling bekerja sama dalam melakukan tugas tertentu. Hal ini dimungkinkan dengan adanya *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.

II. LANDASAN TEORI

Klasifikasi Jaringan Komputer

Jaringan Broadcast memiliki saluran komunikasi tunggal yang dipakai bersama-sama oleh semua mesin yang ada pada jaringan. Pesan-pesan berukuran kecil, disebut paket, yang dikirimkan oleh suatu mesin akan diterima oleh mesin-mesin lainnya. *Field* alamat pada sebuah paket berisi keterangan tentang kepada siapa paket tersebut ditujukan. Saat menerima paket, mesin akan mencek *field* alamat. Bila paket terserbut ditujukan untuk dirinya, maka mesin akan memproses paket itu, bila paket ditujukan untuk mesin lainnya, mesin terserbut akan mengabaikannya.

Jaringan Point-To-Point terdiri dari beberapa koneksi pasangan individu dari mesin-mesin. Untuk mengirim paket dari sumber ke suatu tujuan, sebuah paket pada jaringan jenis ini mungkin harus melalui satu atau lebih mesin-mesin perantara. Seringkali harus melalui banyak *route* yang mungkin berbeda jaraknya. Karena itu algoritma *route* memegang peranan penting pada jaringan *point-to-point*.

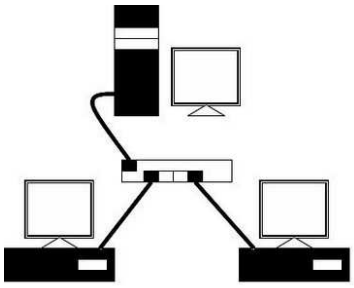
Jenis-jenis Jaringan Komputer

Suatu jaringan komputer umumnya terdiri dari Local Area Network (LAN), Metropolitan Area Network (MAN), Wide Area Network (WAN).

Local Area Network (LAN), merupakan jaringan milik pribadi di dalam sebuah gedung atau kampus yang berukuran sampai beberapa kilometer. Gambar 1 menunjukkan konfigurasi sistem dari jaringan ini. LAN seringkali digunakan untuk menghubungkan komputer-komputer pribadi dan *workstation* dalam kantor suatu perusahaan atau pabrik-pabrik untuk memakai bersama sumberdaya (*resouce*, misalnya printer) dan saling bertukar informasi.

Tabel I Klasifikasi Prosesor Interkoneksi Berdasarkan Jarak

Jarak antar prosesor	Prosesor di tempat yang sama	Contoh
0,1 m	Papan rangkaian	Data flow machine
1 m	Sistem	Multicomputer
10 m	Ruangan	Local Area Network
100 m	Gedung	
1 km	Kampus	
10 km	Kota	Metropolitan Area Network
100 km	Negara	Wide area Network
1.000 km	Benua	
10.000 km	Planet	The Internet



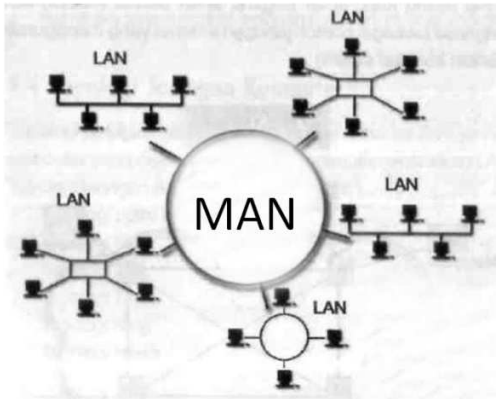
Gambar 1. Jaringan Local Area Network (LAN)

Metropolitan Area Network (MAN), pada dasarnya merupakan versi LAN yang berukuran lebih besar dan biasanya menggunakan teknologi yang sama dengan LAN. Gambar 2 menunjukkan konfigurasi sistem dari jaringan ini. MAN dapat mencakup kantor-kantor perusahaan yang letaknya berdekatan atau juga sebuah kota dan dapat dimanfaatkan untuk keperluan pribadi (swasta) atau umum. MAN mampu menunjang data dan suara, bahkan dapat berhubungan dengan jaringan televisi kabel. MAN hanya memiliki sebuah atau dua buah kabel dan mempunyai elemen switching, yang berfungsi untuk mengatur paket melalui beberapa output kabel. Adanya elemen switching membuat rancangan menjadi lebih sederhana.

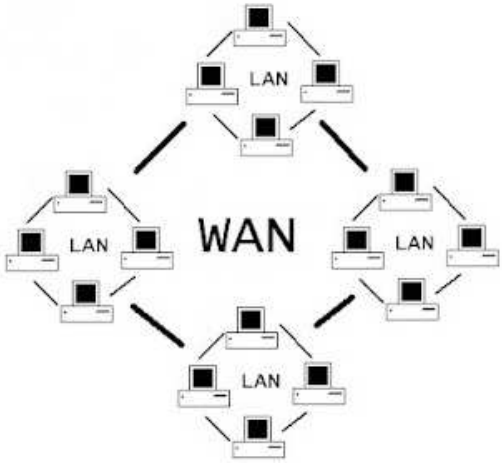
Wide Area Network (WAN), jangkauannya mencakup daerah geografis yang luas, seringkali mencakup sebuah negara bahkan benua. Gambar 3 menunjukkan konfigurasi sistem dari jaringan ini. WAN terdiri dari kumpulan mesin-mesin yang bertujuan untuk menjalankan program-program (aplikasi) pemakai.

Topologi Jaringan

Topologi Jaringan adalah susunan atau pemetaan interkoneksi antara *node*, dari suatu jaringan, baik secara fisik (*riil*) dan logis (*virtual*).



Gambar 2. Jaringan Metropolitan Area Network (MAN).



Gambar 3. Jaringan Wide Area Network (WAN).

Pada sistem LAN terdapat tiga topologi utama yang paling sering digunakan yaitu topologi *star* (gambar 4a), *bus*, *ring* (gambar 4b), pohon (gambar 4c), lengkap atau gabungan (gambar 4d), cincin berinteraksi (gambar 4e), dan sembarang (gambar 4f),. Topologi jaringan ini kemudian berkembang menjadi topologi *tree*, *mesh*, dan *wireless*.

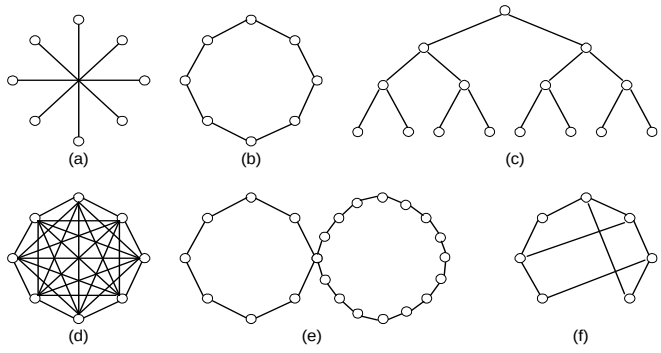
Parallel Computing

Parallel computing adalah penggunaan lebih dari satu sumber daya komputasi secara simultan untuk memecahkan persoalan komputasi. *Software* tradisional umumnya dibuat untuk komputasi serial (Gambar 5) 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 6) 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.

Cluster Computer dan DRM

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 disemua *node*. *Clustering* terbagi dalam beberapa jenis yaitu *high availability cluster*, *load balancing cluster* dan *grid computer*.

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.



Gambar 4. Bebarapa Kemungkinan Topologi. LAN Biasanya Berbentuk Simetris, Sebaliknya WAN Umumnya Bertopologi Tak Menentu.

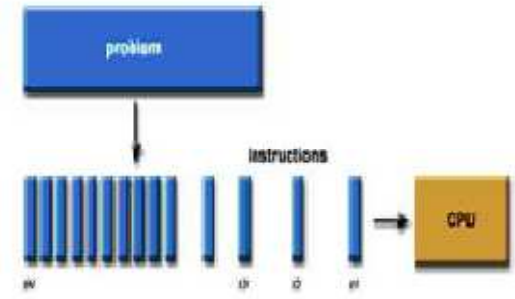
DRM ini sering juga disebut sebagai sistem penjadwalan *job (job scheduler)* karena tugasnya memang melakukan penjadwalan eksekusi *job* dalam sumber daya yang tersedia. Saat sebuah *job* dikirimkan, *job* akan masuk ke dalam sebuah antrian *job (job queue)*. Saat ada sumber daya yang dapat digunakan, *job* dalam antrian tadi akan diberikan ke sumber daya untuk dieksekusi.

Rendering

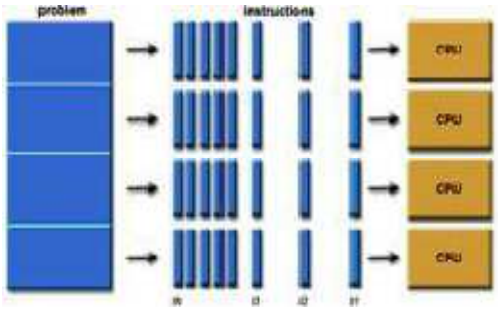
Rendering adalah suatu proses untuk mengubah model geometri menjadi suatu gambar. Proses untuk membangun sebuah gambar membutuhkan beberapa fase seperti *modelling*, pengaturan *material* dan *texture*, penempatan *virtual light*, dan proses *render*. Algoritma untuk melakukan proses *render* mendefinisikan model, geometri, pengaturan *material* dan *texture*, serta penempatan *virtual light* sebagai *input* 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 kompleks. 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 :

Waktu yang diperlukan untuk melakukan proses ini masih dirasakan cukup lama walaupun mesin yang digunakan memiliki sumber daya yang cukup besar



Gambar 5. Komputasi Serial (Blaise Barney, *Introduction to Parallel Computing*)



Gambar 6. Komputasi Paralel (Blaise Barney, *Introduction to Parallel Computing*)

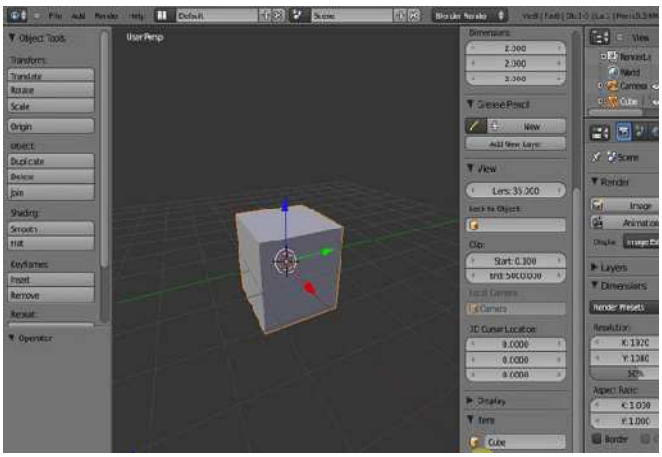
Selama proses *rendering*, mesin tidak dapat digunakan untuk keperluan lain. Hal ini dikarenakan *rendering* membutuhkan suatu proses komputasi yang intensif dan sumber daya yang cukup besar.

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 rendering*) 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.

Blender

Blender memiliki ukuran instalasi yang relatif kecil dan dapat diimplementasikan disemua *platform* komputer. Walaupun sering didistribusikan tanpa adanya dokumentasi yang cukup atau tanpa contoh yang jelas, *software* ini mengandung beberapa *feature* yang hampir sama dengan *software modelling* terbaru. Beberapa kemampuan dari *blender* adalah : Mendukung keanekaragaman dari bentuk geometri primitif, termasuk *polygon* yang tak beraturan, *fast subdivision, surfaced modeling*, kurva bezier, *metalballs* dan lain lain. Tampilan jendela utama dari program blender ini dapat dilihat pada gambar 7.
Memiliki kemampuan *rendering internal* yang cukup handal dan terintregrasi dengan *YafRay yaitu Free Software* untuk *ray tracer*.



Gambar 7. Modeling blender

Didukung dengan *keyframed animation tools* termasuk *kinematic invers, armature (skeleton), shape keys (morphing)*, animasi *nonlinier*, pemberian bobot pada *vertex*, pendeteksian *mesh colution, particle based hair*, dan partikel sistem dengan *collution detection.*
Didukung oleh *phyton scripting* untuk menciptakan *tools* baru dan *prototyping, game logic, import* dan *export* dari *format* lain seperti OBJ, FBX, DFX dan *task automation*.
Memiliki kemampuan untuk *editing* video atau audio yang *nonlinier* dan masih banyak lagi *feauture* yang lain yang merupakan teknologi *high-end*.

Yadra (Yet another distributed rendering application)

Yadra adalah suatu *tools* yang menyediakan pendistribusian untuk proses *render* pada animasi *blende*r di suatu komputer *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 di-*install* 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*.

III. PERANCANGAN SISTEM

Dalam tugas akhir ini ada beberapa metode penelitian yang dilakukan untuk bisa memperolah hasil yang tepat dan benar, sehingga tugas akhir ini bisa akurat. Selain dengan mengacu pada referensi buku yang berkaitan langsung dengan tugas akhir ini, referensi juga mengacu pada jurnal, *procceedings* dan *paper* nasional maupun *paper* internasional. Referensi jurnal, *Procceeding* dan *paper* ini diambil sumber referensi yang terbaru tahun keluarannya atau waktu penelitiannya. Selain referensi tersebut, digunakan juga sumber-sumber referensi dari internet. Dijaman digital saat ini, internet sudah merupakan sumber yang akurat dan terpercaya, asalkan pengguna tahu keabsahan situs dan isi beritanya. Sumber referensi lainnya yang sangat penting dan akurat adalah situs resmi dari pengelolanya sendiri.

Rendering farm adalah sekumpulan dari *computer cluster* yang bekerja secara bersama-sama untuk melakukan proses *rendering* guna mempersingkat waktu dari proses *render*. Pada Tugas Akhir ini penulis menggunakan Globus Toolkit versi 7-0-1 yang digunakan sebagai *tools* untuk menghubungkan beberapa *computer cluster*. *Headnode* berfungsi untuk mendistribusikan *job* kepada *worker node*. Pada *headnode* dan *worker node* diinstall yadra, yadra di *headnode* bertindak sebagai master, dan pada *worker node*

yadra bertindak sebagai *slave*. Sedangkan untuk server jaringan yang meliputi DNS, NFS, LDAP, dan NTP berada pada alamat IP 10.101.255.12

Pemilihan Sistem Operasi

Dalam Tugas Akhir ini, sistem operasi komputer anggota *cluster* adalah sistem operasi Linux. Sistem operasi ini dipilih dengan alasan kemudahan dalam mengkonfigurasi, kestabilan serta sifatnya yang terbuka dalam arti bebas dimodifikasi, didistribusikan ulang, dan dipakai oleh pihak manapun.

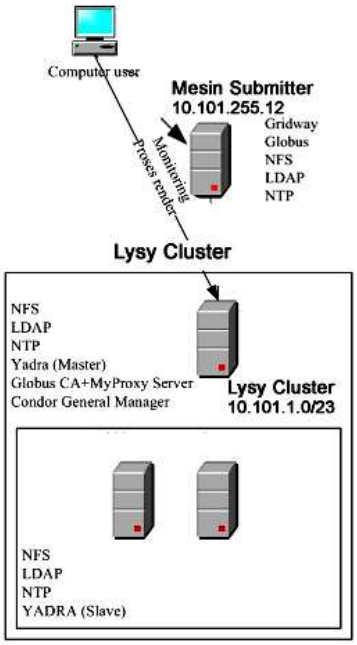
Distribusi Linux yang akan dipakai adalah distribusi Debian GNU /Linux versi Sarge. Distribusi Debian dipilih karena Debian merupakan sebuah distribusi Linux yang bebas. Bebas disini artinya semua aplikasi yang terdapat didalamnya menganut lisensi jenis General Public License (GPL). Didalam GPL disebutkan bahwa semua orang dapat memodifikasi, memakai, dan mendistribusikan ulang *software-software* yang memakai lisensi jenis ini. Tidak semua jenis distribusi Linux adalah *free*, dan Debian tergolong *free*.

Selain Debian, distribusi Linux yang dipakai adalah Ubuntu. Sebenarnya distribusi ini masih termasuk varian dari Debian, hanya saja versi Ubuntu lebih menitikberatkan pada tampilan dan kemudahan pengoperasian dari segi *user*. Karena itu, pengguna Ubuntu saat ini lebih besar dan memiliki tutorial yang lebih lengkap di internet.

Pemilihan Perangkat Lunak

Ada banyak perangkat lunak yang dikenal untuk implementasi dari MPI, antara lain : MPICH, FT-MPI, LA-MPI, PACX-MPI, LAM/MPI dan OpenMPI.

Pada Tugas Akhir ini, perangkat lunak untuk implementasi dari MPI yang digunakan adalah MPICH, karena MPICH mempunyai beberapa kelebihan daripada perangkat lunak yang lain yaitu yang pertama adalah MPICH, bisa didapatkan



Gambar 8. *Design* Jaringan *Rendering Farm*

dari binari maupun *source*-nya dengan gratis, dan mudah didapat karena bisa langsung di-*download* pada situs utama maupun pada mirrornya, dan sudah banyak distro linux yang menyertakan LAM/MPI sebagai salah satu paketnya. Kelebihan selanjutnya adalah kelengkapan dalam implementasi dari MPI, karena MPICH mengimplementasikan seluruh MPI-1 dan banyak MPI-2. Dan yang terakhir yaitu MPICH ini dapat berjalan pada platform Unix yang heterogen dengan performa yang sangat bagus.

Koneksi Antar Node

Komputer-komputer yang akan digunakan sebagai *node* pada tugas akhir ini adalah komputer-komputer yang terletak pada satu *network* 10.101.1.0/23 (LAN), dengan alasan untuk mencapai kinerja yang tinggi pada komunikasi antar *node* melalui jaringan TCP/IP. Media komunikasi yang digunakan adalah menggunakan media kabel UTP. Diusahakan *node-node* yang terhubung pada jaringan yang memiliki transfer data 100 Mbps. Pada tugas akhir ini menggunakan 3 *node*, dengan node pertama (n0) sebagai *node* master, dan 2 lainnya sebagai *slave*.

Shared Storage menggunakan NFS

Network File System digunakan untuk membuat *shared storage*, sehingga seolah mempunyai tempat penyimpanan bersama yang bisa diakses, dibaca ataupun ditulis oleh semua *node*, NFS dipilih karena kesederhanaannya dalam komunikasi yang terjadi pada *server* dan *client*. *Shared storage* akan digunakan untuk meletakkan file binari yang akan dieksekusi oleh semua *node*.

NFS server dipasang pada master node dan NFS client digunakan pada *slave node*. Master node sebagai NFS server akan melakukan *exporting directory* untuk kemudian dimount oleh slave node sebagai NFS client.

SSH sebagai Remote Shell

Remote Shell diperlukan LAM/MPI untuk menjalankan lamd (lam daemon) pada *slave node*, ketika *master node* melakukan LAM *booting*. Ada beberapa alternatif pilihan dalam memilih *remote shell* yang akan digunakan, yaitu : SSH dan rsh. Dalam tugas akhir ini dipilih SSH untuk *remote shell* dengan alasan keamanan. Paket-paket yang dikirim SSH sudah terenkripsi. Semua *slave node* pada LAM/MPI, dipasang ssh-*server*. SSH adalah singkatan dari *Secure Shell.*

SSH pertama kali dikembangkanoleh Sistem V varian. Tidak lama kemudian SSH juga diadopsi oleh sistem operasi Linux. SSH merupakan *remote shell* seperti halnya rsh, rlogin, dan rexec. Perbedaan mendasar dari SSH bila dibandingkan dengan *remote shell* yang lain adalah bahwa data yang dikirimkan sudah dienkripsi dengan menggunakan *Secure Socket Layer* (SSL). Oleh karena itu, *remote shell* dari OpenBSD ini disebut dengan *Secure Shell*.

Blender

Pada tugas akhir ini *blender* sebagai *rendering engine* akan diinstal pada tiap *node*, dimana setiap *node* ini nantinya akan dijadikan sebagai *slave* dari *yadra*. Agar dapat berjalan dengan baik maka paket pendukung dari *blender* harus diinstal, paket paket yang dibutuhkan adalah :

libavcodec1d (>= 0.cvs20070307) _mpeg codec library
libavformat1d (>= 0.cvs20070307) _mpeg _le format library
libavutil1d (>= 0.cvs20070307) _mpeg utility library
libc6 (>= 2.6-1) [mips, sparc] GNU C Library: Shared librariesalso a virtual package provided by libc6-udeb
libc6 (>= 2.6.1-1) [not alpha, ia64, mips, sparc

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 umum nya 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 *rendering farm* 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* di enkripsikan melalui passphrase. *Master* dan *slave* dariYadra 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 direnderakan disimpan di komputer master dan dapat di*download* oleh *user*.

Instalasi dan konfigurasi Master Yadra

Master bekerja pada satu komputer yang telah terhubung pada setiap *slave*, untuk komunikasinya menggunakan TCP *connection* pada satu *port* (akan ditentukan pada saat konfigurasi). Master juga menjalankan webserver yang dapat digunakan untuk mengakses web *interface* dari yadra, Web *interface* ini berguna untuk memperlihatkan status pada proses distribusi *rendering* atau file - file yang dapat didownload. Paket yang dibutuhkan untuk instalasi adalah yadra_1_0_1 linux_java.gz

Untuk proses instalasi dilakukan unzip pada paket tersebut. Setelah paket diekstrak maka konfigurasi master dapat dilakukan. Yang perlu diperhatikan adalah tiap *node* harus diinstall blender dan yadra. Setelah sebelumnya melakukan proses instalasi, selanjutnya dilakukan proses konfigurasi pada direktori (disarankan untuk *running script* pada direktory dimana paket terinstalasi). Eksekusi file *config.sh*. Setelah perintah ini dieksekusi maka selanjutnya akan muncul wizard yang akan menjadi pembimbing dalam proses instalasi, hal pertama yang ditanyakan oleh wizard adalah kedudukan dari komputer yang akan dikonfigurasi apakah sebagai master atau *slave*.

Selanjutnya wizard akan menanyakan nama dari master yang sedang dikonfigurasi dimana sebaiknya nama dari master harus unik. Setelah itu wizard akan menanyakan direktori yang dapat digunakan untuk menampung semua file hasil dari proses *rendering*. Kemudian wizard akan menanyakan tentang *passpharase*, antara master dan *slave passpharase* harus sama. Setelah itu alamat IP yang digunakan untuk *listening* pada koneksi ditanyakan oleh wizard dalah hal ini alamat IP yang dipakai adalah alamat IP pada master.

Selain alamat IP wizard juga akan menanyakan *network port* yang digunakan untuk *listening*, *default port* dari yadra adalah port 2208. Setelah konfigurasi pada tahap ini selesai sebenarnya master telah siap untuk digunakan, namun agar *user* dapat memantau status dari proses *rendering* dengan lebih mudah maka instalasi web server yadra merupakan salah satu opsi yang sangat tepat. Instalasi web server ini juga diatur oleh wizard, pertama kali yang ditanyakan oleh wizard adalah apakah komputer master yang sudah dikonfigurasi akan diinstal web server, setelah itu wizard akan menanyakan *port* mana yang bisa dipakai untuk keperluan dari web server, setelah kita menetukan *port* web server maka konfigurasi web server telah selesai.

Instalasi dan konfigurasi Slave Yadra

Sebagaimana komputer master paket yang dibutuhkan untuk instalasi komputer *slave* adalah yadra_1_0_1_linux_java.gz Untuk proses instalasi dilakukan unzip pada paket tersebut . Setelah paket diekstrak maka konfigurasi *slave* dapat dilakukan. Untuk melakukan konfigurasi pada *slave* dibutuhkan master yang sedang aktif, karena ada sebagian konfigurasi *slave* yang berdasarkan
pada master untuk menghubungkan komunikasinya. Disetiap *computer slave* membutuhkan *blender* sebagai *rendering engine*. Seperti halnya komputer master konfigurasi komputer *slave* dilakukan dengan megeksekusi file *configg.sh.*

Selanjutnya akan muncul wizard yang akan menjadi pembimbing dalam pengkonfigurasian komputer *slave.* Pada umumnya konfigurasi *slave* ini hampir sama dengan konfigurasi master, namun ada sedikit perbedaan. Pada komputer master wizard tidak menanyakan tempat file executable dari blender, sedangkan pada komputer *slave*

wizard menanyakan hal ini, hal ini dikarenakan komputer *slave* membutuhkan blender untuk proses *rendering.*

Penggunaan Yadra

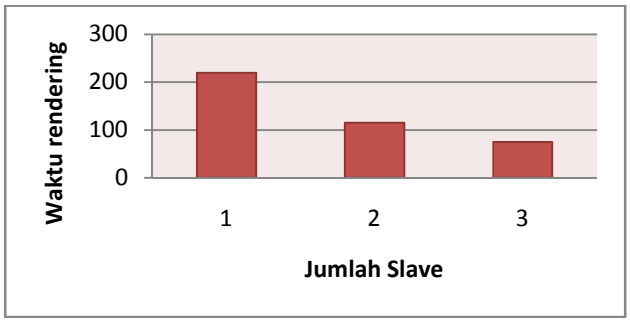
Proses *submitting job* pada yadra perlu dilakukan sebelum proses *rendering* dimulai. Untuk mensubmit *job* pada yadra digunakan perintah sendToQueue_nama dari *slave*. Format lengkap dari perintah diatas adalah sendToQueue_nama slave.sh /tempat_file_blender/ nama__file_blender.blend awal_frame akhir *frame* format output hasil *render*. Contoh kasus, file blender berada pada directory /home/lisy dengan nama *cartoon.blend* memiliki lima puluh *frame*. Apabila diinginkan komputer *slave* untuk merender antara *frame* satu sampai dengan sepuluh dengan output JPEG maka perintah yang ditulis adalah _sendToQueue_slave.sh /home/lisy/cartoon.blend 1 10 JPEG_. Nilai pada perintah diatas menunjukkan mulai *frame* animasi dan akhir *frame*. JPEG menunjukkan output format (Format lain yangmungkin adalah TGA, IRIS, HAMX, FTYPE, JPEG, IRIZ, RAWTGa, PNG, BMP dan semua format yang tersedia dalam blender. Jika kita menuliskan output formatnya dengan benar maka kita mendapat suat pesan dimana file tersebut berhasil diupload.

IV. HASIL DAN PEMBAHASAN

Spesifikasi sumber daya yang dipakai pada pengujian ini terdiri dari dua bagian besar yaitu mesin *submitter* dan lisy *cluster*. Untuk mesin *submitter* memiliki spesifikasi yaitu prosesor Intel(R) Pentium(R) 4 CPU 3.00GHz, memori 1 GB, IP-Address 10.101.255.12 dengan Debian GNU/Linux sebagai sistem operasinya . Lysy *Cluster* memiliki spesifikasi prosesor Intel(R) Pentium(R) 4 CPU 3.00 GHz, memori RAM 1 GB, alokasi IP : 10.101.1.0/23 dengan sistem operasi Debian GNU/Linux.

Tabel II Pengujian *Performance Yadra* Pada Lisy *Cluster* (100 frame)

Jumlah Slave (Node)	Spesifikasi Yang Direnderr (Frame)	Waktu Rendering (Detik)
1	100	220
2	100	115
3	100	75



Gambar 9. Gambar Grafik *Performance Yadra* Pada Lisy *Cluster* (100 *Frame*)

Pengujian dilakukan menggunakan *cluster* yang ada di Jurusan Teknik Elektro UNSRAT, yaitu Lisy c*luster*.

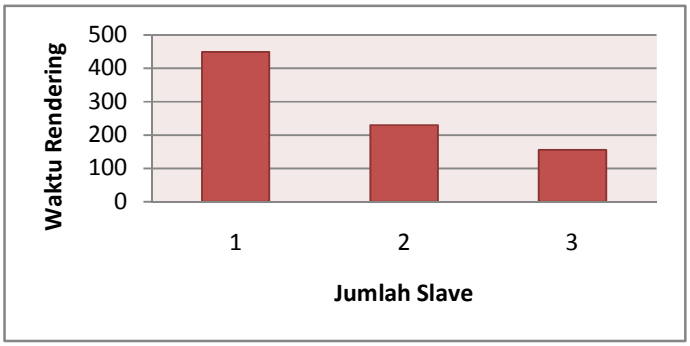
Pengujian

Pengujian Terhadap Performance dari Yadra

Pengujian ini dilakukan dengan memberikan perintah pada *yadra* untuk me-*render file Cartoon.blend* yang merupakan *file* animasi yang terdiri dari seratus sampai empatratus *frame*. Untuk merender *file*, yadra memerlukan *slave* yang terdapat pada setiap *node cluster* dan untuk mengetahui *performance* dari *yadra* maka jumlah *slave* akan ditingkatkan dari satu sampai mencapai batas maksimum dari *slave* yang tersedia. Pengujian ini dilakukan secara manual dengan cara melakukan *ssh* pada tiap *cluster* untuk mengeksekusi tiap *slave* dari *yadra*. Tabel II menunjukkan pengujian yang menggunakan tiga *slave* masing-masing memakai 100 frame yang dirender. Ketiga *slave* ini memiliki perbedaan waktu rendering. Perbedaan waktu render hanya dalam hitungan detik. Gambar 9 merupakan gambar grafik *Performance Yadra* pada Lisy *Cluster* (100 *Frame*) yang mengacu pada tabel II. Tabel III menunjukkan pengujian yang menggunakan tiga *slave* yang memakai 200 frame. Ketiga *slave* ini memiliki perbedaan waktu rendering. Gambar 10 merupakan gambar grafik *Performance* Yadra pada Lisy *Cluster* (200 *Frame*) yang mengacu pada tabel III. Tabel IV menunjukkan pengujian yang menggunakan tiga *slave* yang memiliki 300 frame. Ketiga *slave* ini memiliki perbedaan waktu rendering. Gambar 11 merupakan gambar grafik *Performance Yadra* pada Lisy *Cluster* (300 *Frame*) yang mengacu pada tabel IV. Tabel V dan gambar 12 diperoleh pada *slave* yang menggunakan 400 frame.

Tabel III Pengujian *Performance Yadra* Pada Lisy *Cluster* (200 frame)

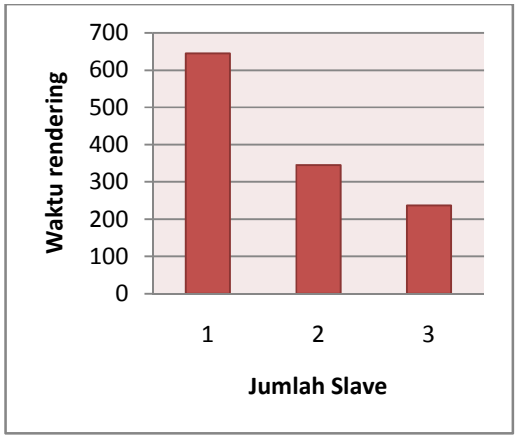
Jumlah Slave (Node)	Spesifikasi Yang Direnderr (Frame)	Waktu Rendering (Detik)
1	200	450
2	200	230
3	200	155



Gambar 10. Gambar Grafik *Performance Yadra* Pada Lisy *Cluster* (200 *Frame*)

Tabel IV Pengujian *Performance Yadra* Pada Lisy *Cluster* (300 frame)

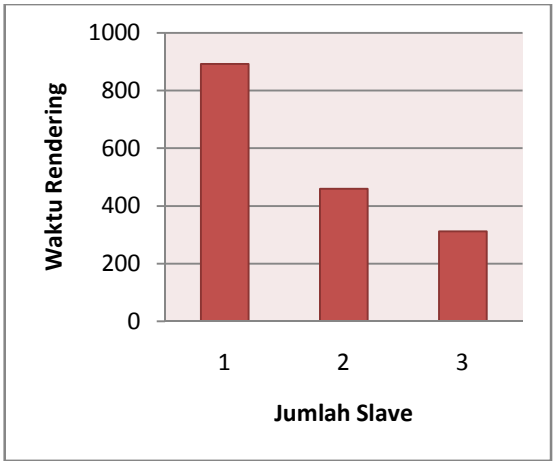
Jumlah Slave (Node)	Spesifikasi Yang Dirender (Frame)	Waktu Rendering (Detik)
1	300	670
2	300	335
3	300	210



Gambar 11. Gambar Grafik *Performance Yadra* Pada Lisy *Cluster* (300 *Frame*)

Tabel V Pengujian *Performance Yadra* Pada Lisy *Cluster* (400 frame)

Jumlah Slave (Node)	Spesifikasi Yang Dirender (Frame)	Waktu Rendering (Detik)
1	400	950
2	400	445
3	400	300



Gambar 12. Gambar Grafik *Performance Yadra* Pada Lisy *Cluster* (400 *Frame*)

V. PENUTUP

Kesimpulan:

Setelah melalui tahapan implementasi dan pengujian sistem, maka diperoleh beberapa kesimpulan antara lain :

1. Komputer *Cluster* berhasil di bangun di lingkungan Elektro UNSRAT.
2. Berhasil melakukan *render* dalam sistem *Cluster* dengan waktu yang singkat.

Saran

Perlu adanya *cluster* khusus yang memang didedikasikan untuk *rendering farm.*

DAFTAR PUSTAKA

[1] B. Barney, 2008., "*Introduction to Parallel Computing* ", tersedia di : https: //computing.llnl.gov/tutorials/parallel_comp/.

[2] F. I. Rusadi, "*Evaluasi Lingkungan Pemrograman Paralel Berbasis Message Passing Interface dalam Infrastruktur Komputasi Grid di Universitas Indonesia*", 2006.

[3] Gridway Team , *Gridway 5.2 Documentation: User Guide*, Universidad Complutense de Madrid, 2007.

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

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

[6] I. Juliansyah, Perancangan Video Live Streaming, Tampilan LED Screen, Berjalan di Jaringan Kampus Universitas Sam Ratulangi,. *Skripsi* Universitas sam ratulangi Fakultas Teknik Jurusan Teknik Elektro Manado, 2013.

[7] J. Joseph and C. Fellenstein, "*Grid Computing*", Prentice Hall, 2003.

[8] S. L. Gooding, L. Arns, P. Smith, and J. Tillotson, "*Implementation of a Distributed Rendering Environment for the TeraGrid*", Purdue University.

[9] Ursula Maier dan Georg Stellner, 2008, "*Distributed Resource Management for Parallel Applications in Networks of Workstations*", tersedia : http://citeseer.ist.psu.edu/maier97 distributed.html.