Perancangan *PC Cluster* Untuk *Render* Animasi 3D

Brian W.K. Malubaya [(1)], Arthur M. Rumagit, ST, MT. [2)], Arie S.M. Lumenta, ST, MT.[(3)],Brave A. Sugiarso, ST, MT.[(4)]
(1)Mahasiswa (2)Pembimbing 1 (3)Pembimbing 2 (4)Pembimbing 3

brian.malubaya@gmail.com[(1)] arthurrumagit@rocketmail.com [(2)] arie.lumenta@gmail.com[(3)]
bravesugiraso@yahoo.com [(4)]

Jurusan Teknik Elektro-FT, UNSRAT, Manado-95115

Abstract

Technology is very important role in human life from day to day. Technology has become part of human beings themselves that they cannot be separated. In its scope computerized, processor the data quickly and accurately is needed as a function of time.

PC cluster is one of the answers from the statements upstairs. This cluster system allows some computers work together that are connected by a network of local so that resolving the issue can be shortened especially for completion of problem solving a large scale. Rendering is a process of transformation into a picture, models of geometry this process eat quite a long time. To address the problem of pc clusters implemented in this case. Cluster used for rendering called renderfarm by the force arising yadra as middleware. By doing so can be detected the time difference rendering use single computer and pc clusters.

Having performed testing then he got that rendering use of pc clusters with 5 slaves faster 60% than single computer. By doing so, we need the further research for subsequent development.

keywords: cluster, render, renderfarm, yadra.

Abstrak

Teknologi sangat penting perannya dalam kehidupan manusia dari hari ke hari. Teknologi telah menjadi bagian dari manusia itu sendiri yang tak dapat dipisahkan. Dalam ruang lingkup komputerisasi, pemroses data yang cepat dan akurat sangat dibutuhkan seiring berjalannya waktu.

PC cluster adalah salah satu jawaban dari pernyataan diatas. Sistem *cluster* ini memungkinkan beberapa komputer bekerja secara bersama yang dihubungkan oleh jaringan lokal sehingga penyelesaian masalah dapat dipersingkat terutama untuk penyelesaian masalah skala besar. *Rendering* adalah suatu proses pengubahan model geometri menjadi sebuah gambar, proses ini memakan waktu yang cukup lama. Untuk mengatasi masalah tersebut *PC cluster* diimplementasikan dalam kasus ini. *Cluster* yang digunakan untuk render disebut *Renderfarm* denga *yadra* sebagai *middleware*. Dengan begitu dapat diketahui perbedaan waktu render menggunakan *single computer* dan *PC cluster*.

Setelah dilakukan pengujian maka didapatkan bahwa rendering menggunakan *PC cluster* 5 *slave* lebih cepat sekitar 60% dibandingkan dengan *single computer*. Dengan begitu maka perlu adanya penelitian lebih lanjut untuk pengembangan selanjutnya.

kata kunci: cluster, render, renderfarm, yadra.

I. PENDAHULUAN

Dewasa ini peran teknologi informasi sangatlah besar pengaruhnya bagi manusia. Teknologi seakan-akan telah menjadi bagian dari manusia itu sendiri. Seiring dengan keperluan tersebut perkembangan teknologi pun berkembang dengan sangat cepat sehingga pemroses data yang cepat dan akurat menjadi target utama untuk menyelesaikan suatu permasalahan dengan sistem komputerisasi.

PC cluster pada dasarnya adalah sistem penyatuan beberapa komputer atau server ke dalam satu kesatuan sehingga dapat bekerja secara bersama-sama. Masing-masing komputer tersebut akan melakukan tugas secara paralel. Dengan demikian beban untuk masing-masing komputer akan menjadi jauh berkurang dan dapat meminimalisasi waktu pengerjaan.

II. LANDASAN TEORI

A. Teknologi Cluster

Secara umum, saat orang membicarakan mengenai "*clustering*", mereka akan mengacu pada suatu teknologi yang memungkinkan sejumlah komputer untuk bekerja sama menyelesaikan permasalahan komputasi biasa. Permasalahan komputasi tersebut bisa berupa komputasi sains (dengan pemakaian *CPU* yang intensif) sampai bermacam-macam proses yang tidak memiliki kesamaan. Teknologi *cluster* ini dibutuhkan untuk meningkatkan kinerja beberapa komputer agar menjadi suatu sistem tunggal sumber daya komputasi yang melakukan pekerjaan besar. *Cluster* terhubung pada suatu jaringan, biasanya jaringan lokal atau *LAN (Local Area Network)*, melalui jaringan komputer ini dimanfaatkan kekuatan pemrosesan paralel dari perangkat komputer. Selain kekuatan pemrosesan meningkat, share sumber daya komputasi dalam kelompok jaringan juga dapat memberikan skalabilitas, *high availability* dan *failover*. Teknologi ini menggunakan alamat *IP (internet protocol)* sebagai sarana komunikasi dalam jaringan serta *middleware* yang merupakan sarana komunikasi non-fisik.

B. Arsitektur Clustering

Suatu komputer yang ter-*cluster* memiliki suatu arsitektur tertentu, dimana arsitektur tersebut memungkinkan suatu komputer dapat berkomunikasi antar komputer satu dengan lainnya. Arsitektur tersebut terdiri dari dua komponen yaitu komponen *hardware* dan *software*. Sekumpulan komputer (pada suatu jaringan komputer) independen yang beroperasi dan terlihat oleh *client* jaringan seolah-olah komputer-komputer tersebut adalah satu buah unit komputer. *Clustering* dirancang untuk meningkatkan kemampuan kinerja dari komputer-komputer yang berada pada suatu jaringan komputer untuk dapat meningkatkan hal-hal berikut:

a) Toleransi kesalahan (*fault tolerance*), yang dapat menyebabkan *node* lainnya (misal komputer B) akan mengambil alih kerja *node* utama (sebutan untuk *node* yang melakukan eksekusi program tertentu, misal komputer A) ketika *node* 4 tersebut mengalami kegagalan. *Client* tidak akan melihat pergantian peran ini. Dengan begitu, *downtime* pun dapat dikurangi secara drastis.
b) Penyerataan beban (*load-balancing*), yang dapat mendistribusikan beban satu *node* ke semua *node* anggota *cluster*. Dengan begitu, kinerja dan skalabilitas *node* utama pun menjadi relatif lebih baik. Bagian terpenting dari komputer *cluster* adalah adanya sebuah aplikasi *middleware* yang dapat menggabungkan seluruh anggota dalam *cluster* sehingga dapat bekerja sama. Tugas utama dari aplikasi *middleware* ini adalah untuk komunikasi dan sinkronisasi antar komputer.

C. Komputasi Paralel

Komputasiparalel memanfaatkan beberapa elemen pemroses secara berkesinambungan untuk menyelesaikan permasalahan, dengan cara memecah masalah menjadi bagian-bagian independen, kemudian masing-masing bagian tersebut diselesaikan oleh masing-masing elemen pemroses sesuai dengan algoritma secara serempak. Elemen pemroses dapat terdiri dari unit pemroses yang heterogen, dan dapat pula terdiri dari unit pemroses yang homogen. Elemen pemroses dapat berupa komputer tunggal dengan banyak prosesor, beberapa komputer yang terhubung dalam suatu jaringan, perangkat keras yang dikhususkan untuk melakukan komputasi paralel, ataupun kombinasi dari perangkat-perangkat yang telah disebutkan.

Untuk proses pembagian proses komputasi tersebut dilakukan secara bersamaan. Untuk proses pembagian proses komputasi tersebut dilakukan oleh suatu software yang bertugas untuk mengatur komputasi..

D. 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 10 ke sumber daya untuk dieksekusi. Dalam 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*).

E. Ethernet.

Ethernet merupakan jenis perkabelan dan pemrosesan sinyal untuk data jaringan komputer yang dikembangkan oleh Robert Metcalfe dan David Boggs di Xerox Palo Alto Research Center (PARC) pada tahun 1972. *Ethernet* merupakan sebuah teknologi yang sudah dikenal oleh masyarakat luas sebagai interface yang digunakan untuk konektivitas perangkat komputer maupun laptop, hampir di setiap jaringan *LAN (Local Area Network)* di seluruh dunia. Selain karena harganya terjangkau, teknologi Ethernet sangat mudah diadaptasi oleh perangkat seperti *modem, printer, scanner, faksimile, VoIP phone*, serta perangkat teknologi informasi lainnya. Sejalan dengan perkembangan teknologi dan senakin meningkatnya kebutuhan masyarakat akan layanan komunikasi data, teknologi Ethernet juga digunakan sebagai interface dari layanan broadband data comunication, yang lebih dikenal dengan nama *Metro Ethernet*. Arsitektur *Ethernet* diperkenalkan pada tahun 1970 oleh *Xerox*, dimana terdapat tiga jenis *Ethernet* yang dibedakan berdasarkan kecepatan daya akses datanya, yaitu *Ethernet*, *Fast Ethernet* dan *Gigabit Ethernet*.

F. Topologi Jaringan

Topologi jaringan adalah suatu aturan atau cara untuk menghubungkan komputer yang satu dengan komputer yang lainnya sehingga membentuk suatu jaringan. Topologi jaringan juga dapat didefinisikan sebagai gambaran secara fisik dari pola hubungan antara komponen jaringan, yang meliputi *Server*, *Workstation*, *Hub*, dan pengkabelannya.

G. Network File System

Salah satu *protocol* yang dipergunakan pada komputasi paralel adalah *Network File System (NFS)*, *NFS* adalah protokol yang dapat membagi sumber daya melalui jaringan. *NFS* dibuat untuk dapat independent dari jenis mesin, jenis system operasi, dan jenis protokol transport yang digunakan. Hal ini dilakukan dengan menggunakan *RPC*. *NFS* memperbolehkan pengguna yang telah diijinkan untuk mengakses *file-file* yang berada di remote host seperti mengakses *file* yang berada di lokal. Protokol yang digunakan protokol mount menentukan host remote dan jenis *file* sistem yang akan diakses dan menempatkan di suatu direktori, protokol *NFS* melakukan I/O pada *remote file system.*

Protokol *moun*t dan protokol *NFS* bekerja dengan menggunakan *RPC* dan mengirim dengan protokol *TCP* dan *UDP*. Kegunaan dari *NFS* pada komputasi parallel adalah untuk melakukan sharing data sehingga setiap *node slave* dapat mengakses program yang sama pada *node master*.

H. Secure Shell (SSH)

Secure shell (SSH) adalah protokol standar yang membentuk jalur yang aman pada komunikasi antar komputer. *SSH* menggunakan teknik enkripsi public key pada sistem authentikasi pengguna untuk mengakses komputer yang lain. *SSH* memberikan sistem enkripsi pada jalur yang digunakan, sehingga memberikan tingkat keamanan data yang tinggi. *SSH* biasa digunakan untuk melakukan *remote login* dan menjalankan perintah pada komputer remote, tetapi *SSH* juga dapat digunakan sebagai *tunnel* jaringan, melakukan penerusan pada *port TCP*, dan koneksi X11. Selain itu dapat juga digunakan untuk mentransfer suatu *file* dengan protokol *SFTP* atau *SCP*. *SSH server* bekerja pada *port* 22.

I. Pemrograman Paralel

Pemrograman paralel adalah teknik pemrograman komputer yang memungkinkan eksekusi perintah/operasi secara bersamaan (komputasi paralel), baik dalam komputer dengan satu (prosesor tunggal) ataupun banyak (prosesor ganda dengan mesin paralel) *CPU*. Bila komputer yang digunakan secara bersamaan tersebut dilakukan oleh komputer-komputer terpisah yang terhubung dalam suatu jaringan komputer lebih sering istilah yang digunakan adalah sistem terdistribusi (*distributed computing*).

Tujuan utama dari pemrograman paralel adalah untuk meningkatkan performa komputasi. Semakin banyak hal yang bisa dilakukan secara bersamaan (dalam waktu yang sama), semakin banyak pekerjaan yang bisa diselesaikan. Misalkan kita dapat melakukan beberapa pekerjaan sekaligus (*parallel*) daripada melakukannya secara berurutan (*serial*). Tentunya waktu yang dbutuhkan jauh berbeda.

J. Render

Render adalah proses akhir dari keseluruhan proses pemodelan ataupun animasi komputer. Dalam *rendering*, semua data-data yang sudah dimasukkan dalam proses modeling, animasi, texturing, pencahayaan dengan parameter tertentu akan diterjemahkan dalam sebuah bentuk output (tampilan akhir pada model dan animasi).

Rendering tidak hanya digunakan pada *game programming*, tetapi juga digunakan pada banyak bidang, misalnya *architecture, simulator, movie, special effect* pada tayangan televisi, dan *design visualization*. *Rendering* pada bidang-bidang tersebut memiliki perbedaan, terutama pada fitur dan teknik *rendering*-nya. Terkadang *rendering* juga diintegrasikan dengan model yang lebih besar seperti paket animasi, tetapi terkadang berdiri sendiri dan juga bisa *free open-source product*.

Rendering harus dilakukan secara cermat dan teliti. Oleh karena itu terkadang dilakukan *pre rendering* sebelum *rendering* dilaksanakan. *Pre rendering* sendiri ialah proses pengkomputeran secara intensif, biasanya digunakan untuk pembuatan film, menggunakan *graphics card* dan *3D hardware accelerator* untuk penggunaan *real time rendering*.

K. Rendering Farm

Rendering farm adalah suatu kumpulan komputer (*Computer Cluster*) yang dibangun untuk mempercepat *rendering* suatu animasi atau *image* yang biasanya digunakan untuk keperluan pembuatan film dan visual-visual efek. Render Farm menggunakan suatu sistem komputer berkinerja tinggi, seperti *cluster* komputer. *Render Farm* dibuat untuk me*render* komputer-*generated imagery (CGI)*, biasanya untuk film dan televisi efek visual. Teknik yang digunakan pada *Render Farm* adalah *Clustering*. *Clustering* merupakan teknik pada dunia komputer dimana tedapat beberapa komputer yang berhubungan satu sama lainya, sehingga menghasilkan kinerja yang maksimal. Dalam membuat *Cluster* biasanya digunakan *Personal Computer* dengan spesifikasi yang tinggi.

L. Bash Shell

Shell adalah *command executive*, artinya program yang menunggu instruksi dari pemakai, memeriksa sintak dari instruksi yang diberikan, kemudian mengeksekusi perintah tersebut. *Shell* ditandai dengan *prompt*. Untuk pemakai menggunakan *prompt* $ dan untuk *superuser* menggunakan *prompt* #.

Bash script adalah *file* yang berisi koleksi program yang dapat dieksekusi. Untuk eksekusi *bash script* menggunakan tanda **.** (dot) sebelum *file bash script* yang berarti eksekusi shell dan tanda **./** berarti *file bash script* barada pada direktori aktual. Secara *default* dalam Linux menggunakan *bash shell*.

M. Blender

Blender dikembangkan sebagai aplikasi *in-house* oleh studio animasi di Belanda *Neo Geo* dan *Not a Number Technologi (NaN)*. Blender pertama kali dibuat oleh Ton Roosendal, dan nama Blender ini sendiri terinspirasi oleh sebuah lagu. Roosendaal mendirikan NaN pada bulan juni 1998, NaN digunakan untuk mempromosikan dan mendistribusikan program Blender yang telah dibuat. Blender didistribusikan pertama kali sebagai *shareware* setelah NaN mengalami kebangkrutan pada tahun 2002.

Para *creditor* setuju untuk merilis Blender dengan syarat menggunakan GNU Genal Public Licence, untuk satu kali pembayaran sebesar 100,000. Pada tanggal 18 juli 2002 lembaga pencarian dana yang bertujuan untuk mengumpulkan donasi didirikan oleh Roosendall dan pada tanggal 7 september 2002 diumumkan bahwa dana yang telah dicari telah mencukupi dan setelah itu *source* code dari blender di-*release*. Sekarang Blender adalah *Free Software* dan secara active dikembangkan dibawah pengawasan Blender *Foundation* Blender memiliki ukuran instalasi yang relatif kecil dan dapat diimplementasikan di semua 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 modeling* terbaru.

N. 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 independent 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.

O. Putty

Putty adalah software remote console/ terminal yang digunakan untuk meremote komp dengan terhubungnya menggunakan port ssh atau sebagainya. Biasanya yang menggunakan software Putty adalah seorang administrator dan seorang Hacker. Putty juga bisa digunakan untuk menjalankan PsyBNC, telnet dan lain-lain..

III. METODOLOGI PENELITIAN

A. Tempat dan Waktu Penelitian

Dalam pelaksanaan tugas akhir ini penulis mengambil tempat penelitian pada Ruang Laboratorium Sistem Komputer (LSK), Jurusan Teknik Elektro, Fakultas Teknik Universitas Sam Ratulangi (UNSRAT), dan rumah penulis.

B. Bahan dan Peralatan

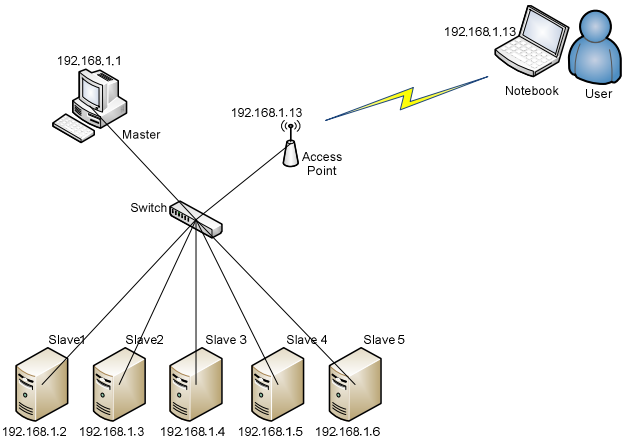
Dalam perancangan *PC cluster* ini, penulis menggunakan 6 buah *PC* dengan sistem operasi *Linux Ubuntu 12.04 LTS* ter-*install* didalamnya dengan beberapa program pendukung antara lain *blender 2.68*, *yadra 1.0.1*. Dan peralatan yang digunakan adalah *switch*, kabel UTP, konektor RJ-45 dan tang *crimper.*

C. Prosedur Penelitian

Dalam perancangan tugas akhir ini mula-mula penulis mencari referensi-referensi yang berhubungan dengan judul tugas akhir serta mencari pokok permasalahan yang harus diselesaikan. Kemudian dilakukan prancangan sistem baik *software* maupun *hardware* lalu sistem di implementasikan Setelah itu penulis melakukan penelitian tentang perbandingan waktu *render* pada *cluster* dan komputer tunggal. Dan pada akhirnya penulisan laporan serta kesimpulan dan saran.

D. Desain sistem

Pada pembuatan tugas akhir ini, penulis merancang *PC cluster* yang bertujuan untuk melakukan proses rendering secara paralel untuk dibandingkan dengan proses rendering pada komputer tunggal. *Cluster* yang difungsikan untuk paralel rendering sering disebut render farm. Topologi jaringan yang digunakan adalah topologi *star* (bintang) dan menggunakan *internet protocol (IP)* kelas C (192.168.1.0/24). Jumlah *nodes* berjumlah enam buah, lima *slaves nodes* dan satu *master node*. Pada penulisan tugas akhir ini penulis menggunakan yadra 1.0.1



Gambar 1. Perancangan Jaringan

sebagai *middleware* dari *cluster* tersebut dan blender sebagai *render engine* yang akan diinstall pada *node* pekerja (*slaves nodes*) sedangkan pada *master node* tidak perlu karena peran *master node* adalah untuk mendistribusikan pekerjaan (*job*) pada *slave*, *monitoring* proses *rendering* pada *slave* serta tempat menampung semua *file input* dan *output* (Gambar 5).

E. Perancangan Jaringan

Perancangan jaringan yang dibuat menggunakan topologi *star* dimana semua *nodes* terhubung pada *switch* yang menjadi titik pemersatu antara satu sama lain. IP yang digunakan adalah 192.168.1.0/24, alokasi IP pada tiap *node* dapat dilihat pada Gambar 1.

F. Instalasi Blender

Dalam hal ini, peran blender yaitu sebagai mesin yang melakukan proses render (*render engine*). Blender di instal hanya pada mesin pekerja (*slave node*) namun tidak ada salahnya kalau diinstal pada *node master*. Untuk menginstal blender cukup mudah karena blender sudah tersedia di repository ubuntu. Pada terminal masukan command *apt-get install blender*. Pada saat pembuatan tugas akhir ini, blender yang digunakan adalah blender 2.68.

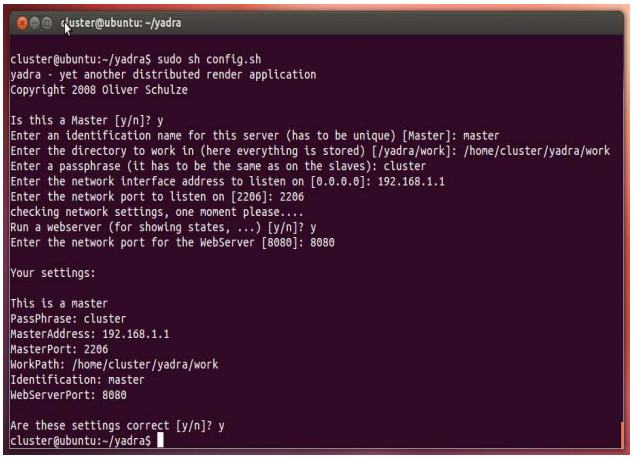
G. Instalasi dan Konfigurasi Master Yadra

Master merupakan *head node* (titik kepala) dimana semua aktifitas akan dikontrol dari *master*. *Master* berkomunikasi dengan menggunakan satu *port*. Selain itu *master* juga memiliki *web interface* yang berguna untuk memantau setiap *slave* dan untuk mendownload *file* hasil render.

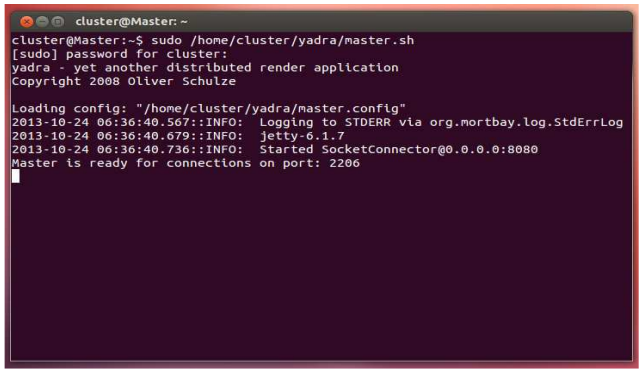
Terlebih dahulu kita memerlukan paket yadra, dalam hal ini penulis menggunakan yadra 1.0.1 yang merupakan versi stabil tanpa *bug* (*error*). Paket tersebut adalah *yadra_1_0_1 linux_java.gz*, untar paket tersebut dan kemudian eksekusi *file config.sh*. Setelah dieksekusi maka akan ada *wizard* yang meminta konfigurasi lebih lanjut hingga selesai (Gambar 2 & 3).

H. Instalasi dan Konfigurasi Slave Yadra

Seperti halnya dengan *master*, yang kita butuhkan untuk menginstall *slave* yadra adalah paket yadra, yakni *yadra_1_0_1 linux_java.gz*. Untuk instalasi *slave*



Gambar 2. Konfigurasi Master Yadra



Gambar 3. Master Menunggu Koneksi dari *Slave*

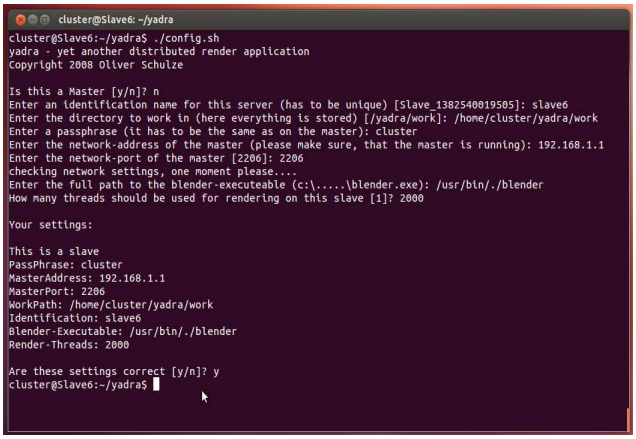
diperlukan *master* yang sedang *running*, karena nantinya*wizard* akan meminta konfirmasi dengan *master*. Seperti *master* untar paket tersebut kemudian eksekusi *config.sh*. Dalam konfigurasi *wizard* akan menanyakan apakah ini *master* atau tidak? jawablah tidak (*n*) karena komputer ini merupakan *slave* (Gambar 4).

I. Instalasi Openssh dan NFS (Network File Sharing)

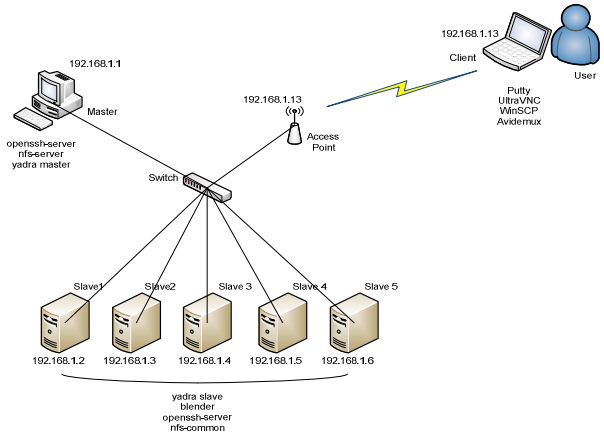
Instalasi *openssh* diperlukan agar komputer dapat berkomunikasi dalam suatu jaringan dengan aman. Karena *openssh* mempunyai jalur yang aman dengan proteksi *password* yang terenkripsi. Pada setiap mesin di instal *openssh-server* yang sudah tersedia pada *repository ubuntu*, ini dimaksudkan agar mesin lain dapat melakukan login ke mesin lain via *ssh* (*secure shell*) untuk melakukan *submiting job*. Untuk mudahkan manajemen *file* penulis juga menginstal *nfs-server* pada sisi *master node* dan *sshcommon* pada *slave node* yang kemudian folder yang berisi *file* blender yang akan dieksekusi akan di *share* pada mesin lain agar *submiting job* pada mesin selain *server* dapat dilakukan secara bersamaan.

J. Perancangan Script bantuan

Dalam proses *rendering, submiting job* dilakukan pada mesin *slave* dengan mengeksekusi *file sendToQueue_namaslave.sh* dengan format perintah *sendToQueue_nama slave.sh /tempat file blender/nama file blender.blend awal frame akhir frame format output*. Folder tempat *file* blender disimpan adalah folder yang di *share* oleh *master* yaitu folder */home/cluster/yadra/files*, yang harus dilakukan proses *mount* agar folder dapat di share.Penulis membuat dua buah *bash script* yang



Gambar 4. Konfigurasi *Slave* Yadra

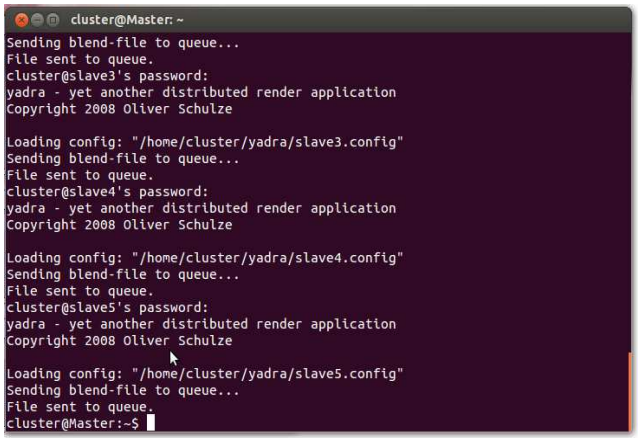


Gambar 5. Sistem Keseluruhan

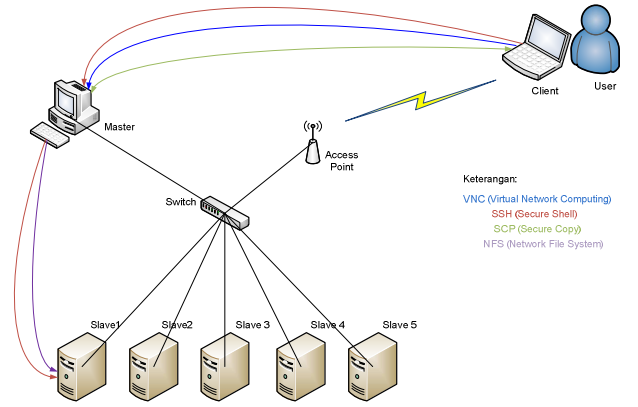
dinamakan *job.sh* dan *mount.sh*. *job.sh* berfungsi agar *submiting job* dapat dilakukan secara bersamaan pada semua mesin *slave* dan *mount.sh* berfungsi untuk dapat me-*mount* folder */home/cluster/yadra/files* ke semua *slave*

K. Pengaturan File Sharing

Untuk *sharing file* kita memerlukan *nfs-server* dan *nfs-common* yang telah terinstal pada *master node* dan *slave node*. *Master node* akan men*share file* pada semua *slave node*. *File* yang akan di *share* adalah *file blender* yang akan di*render*, dalam hal ini penulis men*share file 100.blend, 200.blend dan 300.blend* yang berada di */home/cluster/yadra/files* pada semua *slave node* pada direktori */home/cluster/yadra/files*. Nantinya *slave node* dapat mengakses semua *file*s yang di *share* oleh *master node*. Pertama, gantilah kepemilikan dari folder yang akan di share dengan cara mengetikan command ini pada terminal *chown nobody:nogroup/home/cluster/yadra/files* kemudian daftarkan folder yang akan dishare bersama dengan ip mesin pada *file* export yang berada pada */etc/exports*. Penulis menggunakan *nano text editor*, *nano /etc/exports*. kemudian tambahkan */home/cluster/yadra 192.168.1.1 (rw, sync, no_subtree_check*. Setelah itu ketikan *exports -a*. Kemudian pada sisi *slave*, ketikan *mount 192.168.1.1: /home/cluster/yadra/files /home/cluster/yadra/files*. Dengan begitu *slave node* dapat mengakses folder */home/cluster/yadra/files* yang ada pada



Gambar 6. *File* Send To Queue



Gambar 7. Submitting *job* via notebook

master melalui folder */home/cluster/yadra/files* pada *slave node*.

L. *Submitting Job*

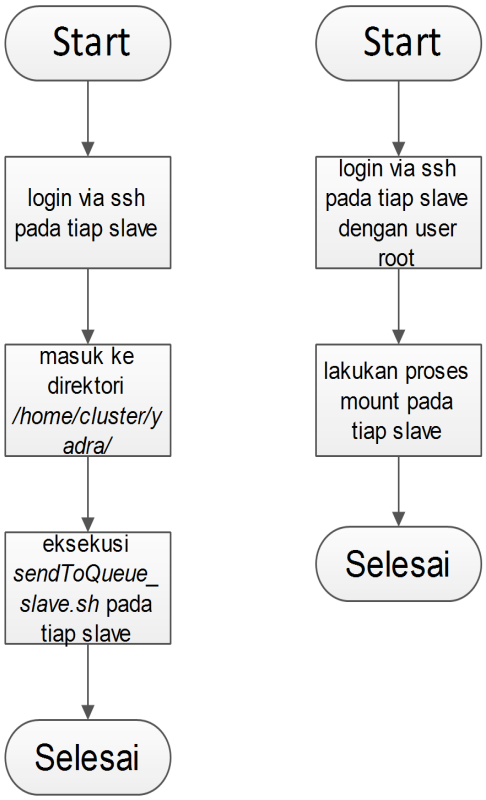
Awalnya *file blender* yang akan di-*render* di-*copy* terlebih dahulu dari *client* ke *master* menggunakan WinSCP ke *folder /home/cluster/yadra/files* yang kemudian akan di-*share* secara otomatis pada *slave node*. Proses login WinSCP yaitu dengan cara memasukan IP *master node* serta *username* dan *password* (Gambar 6 & 7)

M. *Alur Proses Script Bantuan*

Script bantuan (gambar 8), dalam hal ini *job.sh* dan *mount.sh* dimaksudkan agar penulis tidak perlu melakukan *ssh* pada *slave* satu per satu untuk *submiting job* dan melakukan proses *mount*. *Script* ini berisi beberapa *command* yang akan tereksekusi secara bersamaan bila *script* tersebut dieksekusi. Sebelumnya kita perlu mengatur secara manual pembagian *job* pada setiap *slave node* sebelum mengeksekusi *job.sh*. Pengaturan tersebut dilakukan dengan mengedit *script* yg ada pada *file job.sh*

N. *Alur Proses Rendering Yadra*

Proses *rendering* pada yadra dimulai dengan mengeksekusi *master node* sehingga *port* akan terbuka bagi *slave node* untuk berkomunikasi. Penulis menggunakan *port default* yaitu *port 2206*. Jalankan *web interface* yadra dengan cara memasukan *address 192.168.1.1:8080* pada *browser* untuk melihat status dari



Gambar 8. *Flowchart Script* Bantuan *sending job* dan *sharing folder*

slave apakah telah terhubung atau tidak. *Webserver* ini dapat digunakan untuk memonitor proses *render* dan untuk men*dowload* hasil *render* bila semua *job* telah selesai di*render*. *File* hasil *render* akan disimpan pada komputer *master* dalam bentuk *zip file*. Pada *web server* yadra tidak memiliki fitur untuk *real-time update* dimana bila kita ingin melihat *update* dari status pada *slave* harus menekan tombol *refresh* yang ada pada bagian atas tengah halaman atau menekan tombol F5 pada *keyboard*.Maka dari itu penulis menambahkan *script auto-refresh* pada bagian depan halaman *web server* tersebut. Secara keseluruhan proses yang dilakukan oleh *cluster* untuk proses rendering dapat dilihat pada flowchart berikut (Gambar 9 &10).

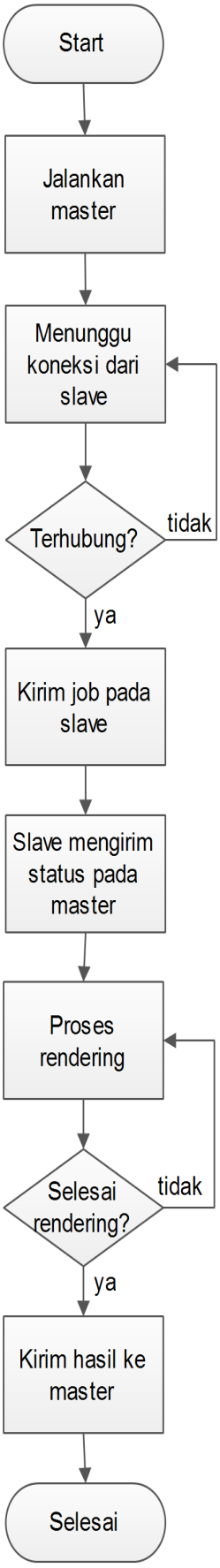
IV. PENGUJIAN DAN ANALISA

A. *Pengujian Rendering*

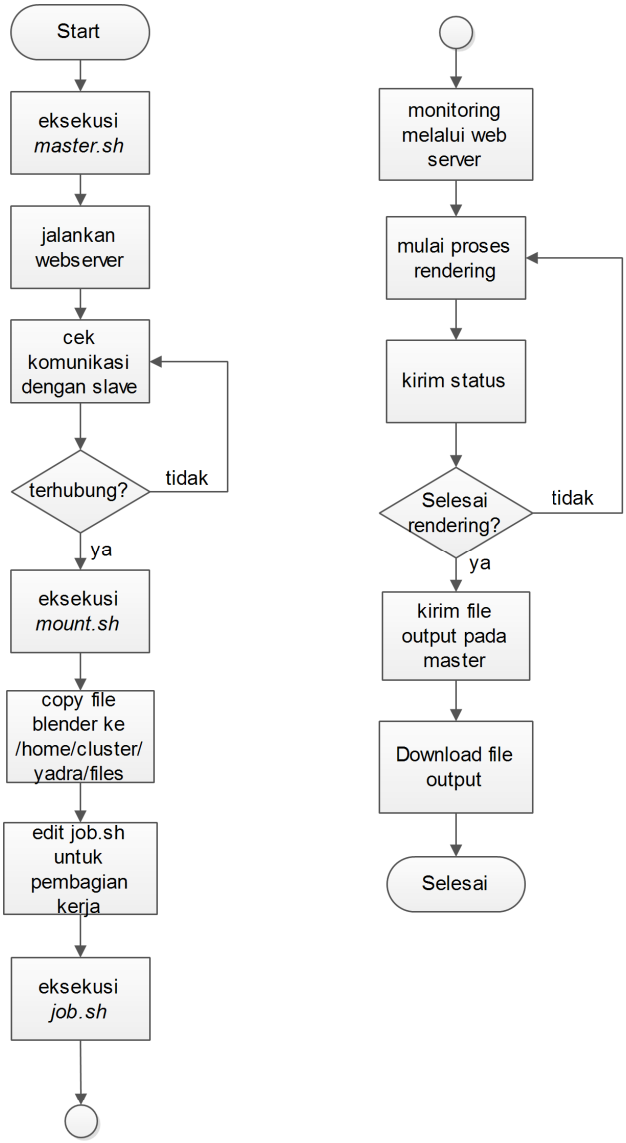
Pada BAB ini penulis akan melakukan pengujian pada *cluster* dengan cara me*render* lima buah *file* yang masing-masing memiliki jumlah *frame* yang berbeda. Pengujian tersebut dimaksudkan agar dapat diketahui perbandingan waktu rendering antara *single computer* dan *PC cluster* serta peningkatan *performance* pada *cluster* seiring bertambahnya jumlah *node*.

Dengan menggunakan persamaan $S = \frac{T1}{Tn}$ kita akan mendapatkan peningkatan yang terjadi pada *cluster*. Dimana,

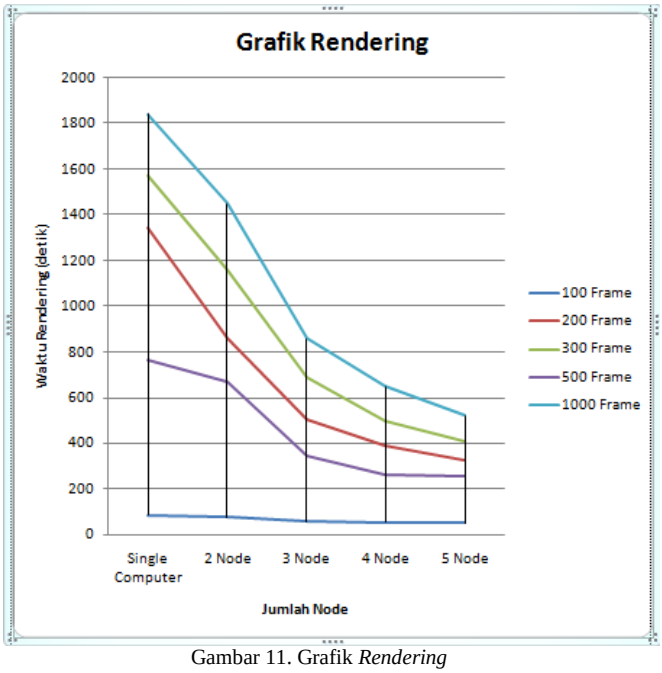
S= Peningkatan kecepatan
T1= Waktu *render* pada *single computer*
Tn= Waktu *render* pada *cluster n node*



Gambar 9. *Flowchart Rendering* Yadra



Gambar 10. *Flowchart Rendering Cluster*



Gambar 11. Grafik *Rendering*

B. Analisa

Setelah dilakukan beberapa pengujian pada *cluster*, penulis akan membandingkan perbandingan *rendering* dengan *single computer* dan *rendering* dengan *cluster* serta menganalisa terhadap data yang telah didapat dari pengujian tersebut.

Pada pengujian sebelumnya telah didapatkan data dari hasil *rendering* pada lima *file* berbeda dengan jumlah *frame* yang berbeda pula. Dengan begitu didapatkan grafik gabungan dari pengujian tersebut (Gambar 11).

Pada grafik diatas dapat diketahui perbandingan antara *rendering* dengan *single computer* dan *rendering* dengan *PC cluster*. Setiap *file* yang di *render* dengan *PC cluster* memakan waktu yang lebih sedikit dibandingkan di *render* dengan *single computer*. *File* yang memiliki *frame* lebih banyak belum tentu akan memakan waktu yang lebih lama, ini dibuktikan dari hasil pengujian sebelumnya dimana *file* dengan 500 *frame* di *render* hanya dalam waktu 769 detik sedangkan *file* 200 *frame* memakan waktu 1343 detik dan *file* 300 *frame* memakan waktu 1572 bila di *render* dengan *single computer*. Hal ini disebabkan dari karakteristik *file* itu sendiri, banyaknya objek yang ada pada *file* tersbut membuat waktu rendering jadi lebih lama dan membuat ukuran *file* jadi lebih besar. Pada umumnya jumlah objek yang lebih banyak akan memakan waktu yang lebih lama walaupun memiliki jumlah *frame* yang sama (Gambar 12).

Dari pengujian serta data yang didapatkan diatas maka perancangan *PC cluster* dinyatakan berhasil dengan persentase peningkatan *performance* saat *rendering* dari *single computer* ke *PC cluster* dapat dilihat pada tabel dan grafik berikut:

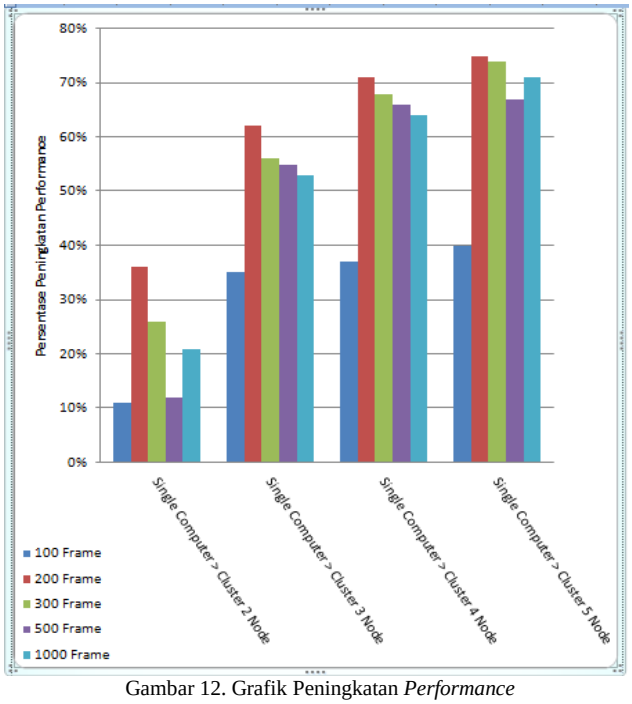
rumus peningkatan performance $\frac{TSc-TCn}{TSc} \times 100\% = PP\ (\%)$

dimana, TSC = waktu *render* dengan *single computer*
TCn = waktu *render* dengan *cluster n node*
PP (%) = persentase peningkatan *performance* dari *single computer* ke *PC cluster*

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Setelah melakukan perancangan, implementasi hingga pengujian penulis dapat menyimpulkan beberapa hal, yaitu *rendering* dengan PC *cluster* lebih cepat dibandingkan dengan single *computer*. Terjadi peningkatan *performance* pada PC *cluster* seiring bertambahnya jumlah *node*. *Rendering* pada *file* yang berbeda dapat menyebabkan perbedaan waktu *rendering* ini disebabkan oleh perbedaan karakteristik dari *file* itu sendiri. Gangguan pada *PC cluster*, dalam hal ini keterlambatan/*delay* pada saat menerima *job* akan sangat berpengaruh pada percepatan yang dihasilkan oleh *PC cluster.*



Gambar 12. Grafik Peningkatan *Performance*

B. Saran

Perlu adanya penelitian lebih lanjut tentang *PC cluster* dengan topik dan judul yang berbeda mengingat teknologi *cluster* semkin diperlukan untuk beban kerja yang lebih besar.

DAFTAR PUSTAKA

[1] A.M . Rumagit, Manajemen Grid Untuk Render Animasi 3 Dimensi, *Tesis* Program Studi S2 Teknik Elektro Institut Sepuluh Nopember, Surabaya, 2009.

[2] D.C. Suhendra, Implementasi Sistem PC *Cluster* Pada Operasi Perkalian Matriks, *Skripsi* Program Studi Teknik Elektro Universitas Sam Ratulangi, Manado, 2010

[3] I.D. Alam, Implementasi Paralel 3D Rendering Pada Lingkungan Grid Elektro, Skripsi Program Studi Teknik Elektro Institut Sepuluh Nopember, Surabaya, 2009.

[4] M.E.I. Najoan, Program Simulasi Untuk Arsitektur Message-Passing Multi*computer*, *Tesis* Program Studi S2 Teknik Sistem Komputer Institut Teknologi Bandung, Bandung, 2002.

[5] Wilkinson, B & Allen, M, *Parallel Programming Teknik Dan Aplikasi Menggunakan Jaringan Workstation Dan Komputer Paralel* Edisi 2, Cetakan Pertama, Penerbit ANDI, Yogyakarta, 2005.