ANALISIS DAN IMPLEMENTASI RENDER FARM

Konsentrasi : Jaringan Komputer

SKRIPSI

Untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

Firman Jati Pamungkas

NIM. 0710683045

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

ANALISIS DAN IMPLEMENTASI RENDER FARM

Konsentrasi : Jaringan Komputer

SKRIPSI

Untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

Firman Jati Pamungkas

NIM. 0710683045

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Adharul Muttaqin, S.T, M.T	**Eriq Muhammad Adams J.,S.T,M.Kom.**
NIP 197601212005011001	**NIP 85041006110027**

LEMBAR PENGESAHAN

ANALISIS DAN IMPLEMENTASI RENDER FARM

Konsentrasi : Jaringan Komputer

SKRIPSI

Untuk memenuhi sebagian persyaratanmemperoleh gelar Sarjana Komputer

Disusun Oleh :

Firman Jati Pamungkas

NIM. 0710683045

Skrisi ini telah diuji dan dinyatakan lulus pada

tanggal 12 Juli 2013

Penguji I	Penguji II
Eko Sakti Pramukantoro,S.Kom,.MKom.	**Sabriansyah Rizqika Akbar,ST,.MEng**
NIK. 86080506110252	**NIK.82080906110084**

Penguji III

Aswin Suharsono, ST., MT.

NIK.84091906110251

Mengetahui

Ketua Program StudiTeknikInformatika

Drs. Marji, MT.

NIP. 19670801 199203 1 001

PERNYATAAN

ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 21 Juli 2013

Mahasiswa,

Firman Jati Pamungkas

NIM 0710683045

repository.ub.ac

ABSTRACT

Firman Jati Pamungkas, 2013. The Analysis and Implementation of Render Farm. Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Adharul Muttaqin, ST., MT dan Eriq Muhammad Adams, ST., M.Kom.

The main problem using computer to render 3D animation is the time used by the computer in rendering process. Time problem in rendering process has a strong relationship with the computation resources owned by the computer. The increase of 3D animation quality often makes the rendering process done takes longer time. To resolve this problem, parallel computing is used by combining the resources available as rendering farm.

Computation resources owned by computer are interconnected with a LAN network and able to work with the help of additional tool named yadra. Yadra will be be effective when be used to rendering some animation, because yadra is working by split and distribute frame of animation file to every computer.

There are 2 main components in this system, Master server which is the main of the system functioned as the supervisor and distributor of the work of rendering process. Slave Master is the second component, which does the rendering all of process from master server. Yadra tool has a weakness on the absence user interface in its use. It will be make the people who use this system hardness to using this system. The addition of user interface in GUI from using Java Language Programming is beneficial to facilitate the use of the system. The result show that the speed of rendering process using render farm is faster than the speed of rendering process using only one computer.

Keywords : *render, parallel computing, render farm.*

UNIVERSITAS BRAWIJAYA

repository.ub.ac

ABSTRAK

Firman Jati Pamungkas, 2013. Analisis dan Implementasi Render Farm. Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmo Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Adharul Muttaqin, ST., MT dan Eriq Muhammad Adams, ST., M.Kom.

Masalah utama dalam melakukan proses render animasi yaitu waktu yang dihasilkan oleh komputer dalam melakukan rendering. Permasalahan waktu dalam proses render berhubungan erat dengan sumber daya komputasi yang dimiliki. Seringkali peningkatan kualitas animasi 3D justru membuat proses render yang dilakukan membutuhkan waktu yang semakin lama. Untuk mengatasi masalah tersebut digunakan solusi berupa *parallel computing* dengan menggabungkan sumber daya yang ada sebagai suatu rendering farm.

Sumber daya komputasi yang dimiliki saling berhubungan melalui sebuah jaringan LAN dan bekerja menggunakan bantuan perangkat render farm bernama yadra. Sistem dibedakan menjadi 2 yaitu komputer master sebagai induk dari sistem yang berfungsi melakukan pengawasan dan pembagian kerja dari proses render serta komputer slave yang bekerja melakukan proses render. Perangkat yadra memiliki kelemahan dengan tidak adanya user interface dalam penggunaannya. Penambahan user interface dalam bentuk GUI menggunakan bahasa pemograman Java berfungsi untuk mempermudah penggunaan sistem. Pengujian yang dilakukan terhadap sistem memperlihatkan adanya peningkatan kecepatan dari proses render dibanding dengan menggunakan 1 buah komputer.

Kata Kunci : *render, parallel computing, render farm.*

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Puji syukur kepada Allah SWT atas berkat dan rahmat, sehingga penulis dapat menyelesaikan skripsi dengan judul "Analisis dan Implementasi Render Farm" dengan baik. Skripsi ini diajukan untuk memenuhi persyaratan kelulusan dalam menyelesaikan pendidikan dan memperoleh gelar Sarjana Komputer konsentrasi komputasi berbasis jaringan pada Program Studi Teknik Informatika PTIIK.

Dalam menyelesaikan skripsi ini, penulis banyak mendapat bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Ayahanda Drs.Bambang Sulistyono, SE., Ibunda Atik Diah Hermayanti serta Adinda Kunto Jati Prakoso atas kasih sayang, pengertian, dukungan baik moril maupun materiil, dan segala doa serta pengorbanan yang tidak mungkin terbalaskan.
2. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwarsito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, Bapak Eddy Santoso, S.Kom selaku ketua program, wakil 1, wakil 2, dan wakil 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Adharul Muttaqin, ST., MT selaku Dosen Pembimbing I yang telah banyak memberikan arahan, masukan, dan bimbingan kepada penulis selama penyusunan skripsi ini.
4. Bapak Eriq Muhammad Adams,ST M.Kom selaku Dosen Pembimbing II yang telah banyak memberikan arahan, masukan, dan bimbingan kepada penulis selama penyusunan skripsi ini.
5. Bapak Ir. Sutrisno, M.T selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
6. Segenap bapak dan ibu dosen program studi teknik informatika beserta seluruh staff administrasi yang telah membantu selama kegiatan belajar.

7. Adam Hendra Brata beserta keluarga, Farid Angga Pribadi beserta keluarga yang telah begitu banyak membantu penulis dan memberikan masukan – masukan yang bermanfaat.
8. Manajemen dan staff Diery, manajemen CV Pabrik Teknologi, yang telah memberikan banyak pengalaman dan pembelajaran kehidupan.
9. Teman-teman 64 , KPMB , LEGENDARY TIF, seluruh angkatan 2007 atas semangat dan bantuannya.
10. Semua pihak yang telah membantu dan memberi semangat dalam proses penulisan skripsi ini.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan. Oleh karena itu, penulis mengharapkan saran dan kritik yang membangun untuk mengisi kekurangan tersebut. Akhirnya penulis berharap semoga skripsi ini dapat memberikan manfaat bagi kita semua, Amin.

Malang, Juni 2013

Penulis

DAFTAR ISI

ABSTRACT ..... **i**
ABSTRAK ..... **ii**
KATA PENGANTAR ..... **iii**
DAFTAR ISI ..... **v**
DAFTAR TABEL..... **vii**
DAFTAR GAMBAR..... **viii**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 1
1.3 Batasan Masalah 2
1.4 Tujuan Penulisan 2
1.5 Manfaat Penulisan 2
1.6 Sistematika Penulisan 3
BAB II DASAR TEORI **4**
2.1 Kebutuhan Akan Kecepetan Komputasi..... 4
2.2 Paralel Computing 6
2.2.1 Pengertian..... 6
2.2.2 Potensi Peningkatan Kecepatan Komputasi 8
2.2.2.1Faktor Pemercepat..... 8
2.3 Rendering 13
2.4 Blender 13
2.5 *Yadra* 14
2.6 Pengukuran Kinerja Komputasi 17
BAB III METODOLOGI **19**
3.1 Studi Literatur 20
3.2 Analisis Kebutuhan 20
3.3 Perancangan 21
3.4 Implementasi 23
3.5 Pengujian dan Analisis..... 23

BAB IV PERANCANGAN DAN IMPLEMENTASI ……. 26
4.1 Perancangan……. 26
4.1.1 Perancangan Umum Sistem……..... 27
4.1.2 Topologi Jaringan……..... 26
4.1.3 Kebutuhan Hardware……..... 31
4.1.4 Kebutuhan Software…….....32
4.1.5 Perancangan Perangkat Lunak ……..... 34
4.1.6 Perancangan Algoritma Aplikasi 34
4.1.6 Perancangan Layout Aplikasi 31
4.1.6 Penggunaan Sistem Render Farm 40
4.2 Batasan-Batasan Implementasi 40
4.3 Implementasi Perangkat Pendukung 41
4.4 Implementasi Algoritma 45
4.4.1 Implementasi Algoritma Proses Instalasi 46
4.4.2 Implementasi Algoritma Proses Aktivasi 47
4.4.3 Implementasi Algoritma Proses Submit Job 47
4.4.4 Implementasi Algoritma Remote Server 48
4.5 Implementasi Antarmuka 49
4.5.4 Implementasi Antarmuka Proses Instalasi 49
4.5.5 Implementasi Antarmuka Proses Aktivasi 50
4.5.6 Implementasi Antarmuka Proses Submit Job 51
4.5.7 Implementasi Antarmuka Remote Server 52
BAB V PENGUJIAN DAN ANALISIS 54
5.1 Skenario Pengujian 54
5.2 Hasil Perhitungan Skenario 1.....56
5.3 Hasil Perhitungan Skenario 2 61
5.4 Hasil Perhitungan Skenario 3 64
5.5 Hasil Perhitungan Skenario 4 66
5.7 Analisis Pengujian 68
5.7.1 Performa Sistem 69

5.7.2 Performa Komputasi 78

BAB VI PENUTUP 82

6.1 Kesimpulan 82

6.2 Saran 83

DAFTAR PUSTAKA 84

DAFTAR TABEL

Tabel 4.1 Alokasi IP dari komputer yang digunakan ……..... 31
Tabel 4.2 Spesifikasi Hardware Yang Dirancang Untuk Master 31
Tabel 4.3 Spesifikasi Minimum Yang Dirancang Untuk Slave 32
Tabel 5.1 Tabel Pengujian Hasil Render Skenario 1 Dengan Single Processing
dan Render Farm 56
Tabel 5.2 Tabel Pengujian MIPS 60
Tabel 5.3 Tabel Hasil Perhitungan Pada Skenario 1 60
Tabel 5.4 Tabel Pengujian Hasil Render Skenario 2 tahap 1..... 61
Tabel 5.5 Tabel Pengujian Hasil Render Skenario 2 tahap 2..... 61
Tabel 5.6 Tabel Pengujian Hasil Render Skenario 2 tahap 3..... 61
Tabel 5.7 Tabel Pengujian MIPS 63
Tabel 5.8 Tabel Hasil Perhitungan Pada Skenario 2 64
Tabel 5.9 Tabel Pengujian Hasil Render Skenario 3 tahap 1..... 64
Tabel 5.10 Tabel Pengujian Hasil Render Skenario 3 tahap 2..... 65
Tabel 5.11 Tabel Pengujian MIPS 66
Tabel 5.12 Tabel Hasil Perhitungan Pada Skenario 3 66
Tabel 5.13 Tabel Pengujian Hasil Render Skenario 4..... 66
Tabel 5.14 Tabel Pengujian MIPS 67
Tabel 5.15 Tabel Hasil Perhitungan Pada Skenario 4 68
Tabel 5.17 Contoh Tabel Hasil Perhitungan MIPS Pada Skenario 1 79
Tabel 5.18 Contoh Hasil Perhitungan Perbandingan Pada Skenario..... 80
Tabel 5.19 Contoh Tabel Hasil Efisiensi Pada Skenario 1 81

DAFTAR GAMBAR

Gambar 2.1	Sistematika Kerja Proses Sekuensial	7
Gambar 2.2	Sistematika Kerja Proses Paralel	7
Gambar 2.3	Peningkatan Kecepatan Secara Linier dan Aktual.....	10
Gambar 2.4	Kecepatan Secara Aktual dan Optimal.....	11
Gambar 2.5	Contoh Tampilan Yadra	15
Gambar 3.1	Diagram Flowchart Metodologi	19
Gambar 3.2	Gambaran Perancangan Sistem	22
Gambar 4.1	Diagram Perancangan Sistem	26
Gambar 4.2	Perancangan Sistem Render Farm	27
Gambar 4.3	Blender dan Yadra dalam Sistem	29
Gambar 4.4	Blender dan Yadra dalam GUI Sistem	30
Gambar 4.5	Flowchart Perancangan Instalasi Sistem Render Farm.....	35
Gambar 4.6	Flowchart Perancangan Instalasi Master	36
Gambar 4.7	Flowchart Perancangan Instalasi Slave...	37
Gambar 4.8	Flowchart Perancangan Aktivasi Server Master dan Slave …	38
Gambar 4.9	Flowchart Perancangan Submit Job	39
Gambar 4.10	Flowchart Perancangan Remote Sistem	40
Gambar 4.11	Perancangan Tampilan Instalasi Sistem	41
Gambar 4.12	Perancangan Tampilan Aktivasi Sistem	42
Gambar 4.13	Perancangan Tampilan Submit Job	43
Gambar 4.14	Perancangan Tampilan Remote Sistem	44
Gambar 4.15	Implementasi Perangkat Pendukung	45
Gambar 4.16	Implementasi Algoritma Untuk Proses Instalasi	46
Gambar 4.17	Implementasi Algoritma Untuk Proses Aktivasi	47
Gambar 4.18	Implementasi Algoritma Untuk Proses Submit Job	48
Gambar 4.19	Implementasi Algoritma Untuk Proses Remote Server	49
Gambar 4.20	Gambar Tampilan Antarmuka Proses Instalasi	50
Gambar 4.21	Gambar Tampilan Antarmuka Proses Aktivasi	51
Gambar 4.22	Gambar Tampilan Antarmuka Proses Submit Job	52

Gambar 4.23 Gambar Tampilan Antarmuka Proses remote Server . ..…..... 53
Gambar 5.3 Gambar Arizona.blend pada blender 73
Gambar 5.4 Hasil render Arizona.blend .....…..... 73
Gambar 5.5 Gambar Beamsatellite.blend pada blender 74
Gambar 5.6 Hasil render Beamsatellite.blend .....…… 74
Gambar 5.7 Gambar Paddywagon.blend pada blender .....… 75
Gambar 5.8 Hasil render Paddywagon.blend.....… 75
Gambar 5.9 Gambar Policelifter.blend pada blender 76
Gambar 5.10 Hasil render Policelifter.blend..... 76
Gambar 5.11 Gambar Redexosuit.blend pada blender 77
Gambar 5.12 Hasil render Redexosuit.blend.....….. 77
Gambar 5.13 Gambar Tankbot.blend pada blender 78
Gambar 5.14 Hasil render Tankbot.blend.....…… 78

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia animasi 3D saat ini yang sangatlah pesat seringkali tidak diimbangi dengan ketersediaan sumber daya yang cukup sebagai penunjangnya. 1 buah komputer yang bekerja secara sekuensial dianggap tidaklah mencukupi untuk dapat menunjang pekerjaan dalam dunia animasi.

Salah satu tahapan yang dilakukan dalam proses pengerjaan animasi 3D yaitu proses *render*. Tanpa dirender suatu gambar animasi hanya akan berupa kumpulan *point* dan *wireframe* sederhana. Proses *render* melakukan pembungkusan tekstur pada objek yang bersesuaian sesuai cahaya yang datang pada objek tersebut. Namun proses *render* membutuhkan daya komputasi yang sangat besar karena banyaknya titik koordinat yang harus dikomputasi, terutama jika data yang diolah cukup rumit.

Salah satu solusi yang sering diambil oleh para pelaku industri animasi 3D yaitu dengan membangun sistem *render farm* yang digunakan untuk mengatasi masalah *render* tersebut. *Render farm* dibangun menggunakan teori dari proses kerja *parallel computing*. Proses ini menggabungkan beberapa unit komputer di dalam suatu jaringan yang kemudian sekelompok komputer ini yang nantinya akan bekerja sama dalam menyelesaikan proses komputasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas,maka dapat diambil rumusan masalah untuk penulisan skripsi ini yaitu :

1. Bagaimana desain dan Implementasi sistem dari *render farm*?
2. Bagaimana hasil pengujian dari *render farm* ?

3. Bagaimana hasil analisis dari sistem ditinjau dari kinerja komputasi dari sistem?

1.3 Batasan Masalah

1. Perancangan jaringan berdasar pada model jaringan *Local Area Network*.
2. Penggunaan perangkat olah 3D menggukan *software Blender dengan OS Linux CentOS*.

1.4 Tujuan

Adapun tujuan yang dapat diambil dari penulisan skripsi ini yaitu membangun, mengimplementasikan dan menganalisa suatu solusi *render* dengan menggunakan *render farm*.

1.5 Manfaat

A. Bagi Penulis
   1. Mengaplikasikan ilmu teori yang selama ini didapatkan.
   2. Menambah ilmu pengetahuan berkaitan dengan masalah *parallel computing*.
   3. Menambah ilmu pengetahuan berkaitan dengan masalah implementasi *render farm* pada *parallel computing*.

B. Bagi Pengguna
   1. Dapat dijadikan referensi bagi pembuatan *render farm* pada *parallel computing*.
   2. Dapat langsung di implementasikan dalam kehidupan sehari-hari untuk menunjang proses *rendering* gambar atau video dari *blender*.

Sistematika Penulisan

Dalam menyusun penulisan ini, sistem penulisan yang digunakan oleh penulis yaitu dengan cara membagi masalah menjadi beberapa tahapan, di mana pembahasan setiap babnya sebagai berikut :

1) **Bab I Pendahuluan**

   Bab ini membahas tentang latar belakang, pengidentifikasian masalah yang muncul, tujuan, batasan masalah, dan sistematika penulisan.

2) **Bab II Dasar Teori**

   Bab ini berisi tentang teori-teori dasar yang relevan dengan masalah yang dibahas dalam laporan akhir ini.

3) **Bab III Metodologi**

   Bab ini dijelaskan langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dibuat.

4) **Bab IV Perancangan dan Implementasi Sistem**

   Bab ini membahas analisis kebutuhan pengguna, perancangan sistem, dan implementasi system yang dibuat.

5) **Bab V Pengujian dan Analisis**

   Bab ini merupakan tindak lanjut dari masalah implementasi , melakukan pengujian terhadap produk yang dibuat dan melakukan analisis sesuai dengan parameter yang ada.

6) **Penutup**

   Bab ini berisi tentang kesimpulan dan saran yang didapat setelah melakukan analisa, desain, implementasi dan pengujian terhadap analisis dan implementasi dari sistem.

BAB II
DASAR TEORI

2.1. Kebutuhan Akan Kecepatan Komputasi

Perkembangan akan kebutuhan komputasi di berbagai aspek kebutuhan, termasuk di dalam dunia animasi 3D di dunia selalu menuntut akan selalu adanya peningkatan akan kemampuan komputasi yang lebih besar dari sistem komputer saat ini. Animasi 3D yang saat ini telah berkembang tak hanya di dalam dunia pendidikan saja, namun telah berkembang ke dalam industry kreatif, perfilman membutuhkan solusi inovatif untuk menyelesaikan masalah komputasi di dalamnya. Pencapaian kinerja tinggi dari suatu komputer merupakan salah satu solusi yang harus dicari untuk dapat menyelesaikan permasalahan tersebut. Sistem komputasi harus dapat diselesaikan dalam periode waktu yang masuk akal. Waktu yang diperlukan haruslah sesingkat mungkin agar pengguna bisa bekerja secara efektif. Ketika animasi yang di desain menjadi semakin rumit, waktu *render* akan semakin bertambah. Masalah tersebut adalah masalah yang tidak dapat diselesaikan dalam waktu yang selayaknya menggunakan komputer masa kini. Kualitas animasi di dalam dunia industri tak kan pernah bisa diimbangi dengan komputer berkemampuan sekuensial.

Komputer tradisional memiliki prosesor tunggal untuk melaksanakan tugas-tugas dari suatu program. Salah satunya cara untuk meningkatkan kecepatan komputer (seperti yang telah diusulkan bertahun-tahun) adalah menggunakan beberapa prosesor dalam satu komputer (multi prosesor) maupun beberapa komputer yang mengerjakan satu tugas. Beberapa prosesor melakukan share terhadap beberapa beban kerja. Hal ini dapat dicapai dengan menggunakan satu prosesor dalam sistem komputer tunggal atau pendistribusian tugas-tugas diantara sejumlah sistem komputer yang di link seperti cluster atau jaringan. Maksudnya adalah *p* prosesor/komputer mampu menghasilkan kemampuan hingga *p* kali kecepatan komputer dengan satu prosesor/komputer. Berapapun kecepatan

prosesor/komputer itu dengan harapan masalah yang ada dapat diselesaikan Bab pendahuluan dengan waktu 1/p. Tentu saja, situasi tersebut merupakan situasi ideal yang jarang sekali terjadi. Seringkali, masalah tidak dapat dipecah menjadi beberapa bagian kecil. Selain itu, diperlukan interaksi antara masing-masing bagian, baik untuk transfer data maupun sinkronisasi. Namun, seberapa jauh peningkatan kecepatan dapat dicapai bergantung pada masalah dan peran paralelisme. Satu hal yang menyebabkan kemampuan komputer paralel menjadi tak terbatas adalah peningkatan kecepatan eksekusi suatu prosesor secara berkelanjutan yang akan meningkatkan pula kecepatan komputer paralel. Dengan demikian, akan selalu ada masalah dengan tantangan besar yang tak bisa diselesaikan dalam waktu yang memadai menggunakan komputer saat ini.

Terlepas dari memperoleh daya untuk meningkatkan kecepatan eksekusi atas masalah yang sudah ada, penggunaan beberapa prosesor/komputer membuat masalah yang lebih besar dan keakuratan solusi dari suatu masalah bisa dihasilkan dalam waktu yang memadai. Multikomputer atau multiprosesor seringkali membuat penyelesaian masalah dalam waktu yang ditentukan sehingga solusi yang diberikan lebih akurat. Faktor lain yang berkaitan dengan masalah tersebut adalah multikomputer seringkali memiliki jumlah memori utama yang lebih banyak daripada komputer tunggal. Hal tersebut dapat membuat terselesaikannya masalah yang membutuhkan memori utama dalam jumlah besar. [WBM-10]

Paralel komputer bukanlah ide baru, tetapi ide lama. Contohnya, Gill (1958) menulis tentang pemograman paralel pada tahun 1958, Holland (1959) menulis tentang komputer yang mampu menangani sejumlah sub-program yang berubah-ubah secara bersamaan pada tahun 1959 dan Conway (1963) menjelaskan desain komputer dan pemograman paralel pada tahun 1963. Tidak begitu lama, Flyn dan Rudd (1996) menulis tentang “arah baru menuju kemampuan sistem yang lebih dan lebih lagi ... mengantarkan kita pada suatu kesimpulan sederhana bahwa masa depan adalah paralel “.[WBM-

2.2 *Parallel Computing*

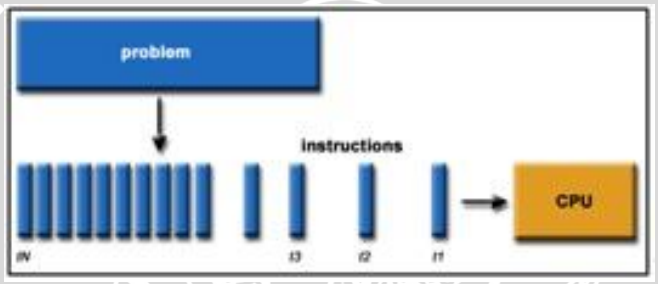
2.2.1 Pengertian

Parallel computing adalah penggunaan lebih dari satu sumber daya komputasi secara simultan untuk memecahkan persoalan komputasi. *Software* tradisional umumnya dibuat untuk komputasi serial (Gambar 2.1) 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 2.2) 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. [RMH-09]

Hal ini menunjukkan suatu teknik tingkat tinggi yang digunakan untuk menyediakan tugas-tugas pemrosesan data secara bersama dengan tujuan untuk meningkatkan kecepatan komputasi dari suatu komputer. Teknik ini menggantikan pemrosesan yang dilakukan pada komputer konvensional di mana setiap instruksi dilakukan secara sekuensial. Sistem pemrosesan paralel dapat melakukan pemrosesan data secara bersama untuk memperoleh waktu eksekusi yang lebih cepat. Tujuan utama pemrosesan paralel adalah untuk meningkatkan kemampuan komputasi komputer. Jumlah perangkat keras pada sistem akan ditambah, sehingga dapat menambah biaya untuk membangun sistem. Namun, perkembangan teknologi saat ini dapat mengurangi biaya pembelian perangkat keras tersebut dimana teknik pemrosesan paralel dapat dikatakan masih layak secara ekonomi. [SYR-10]

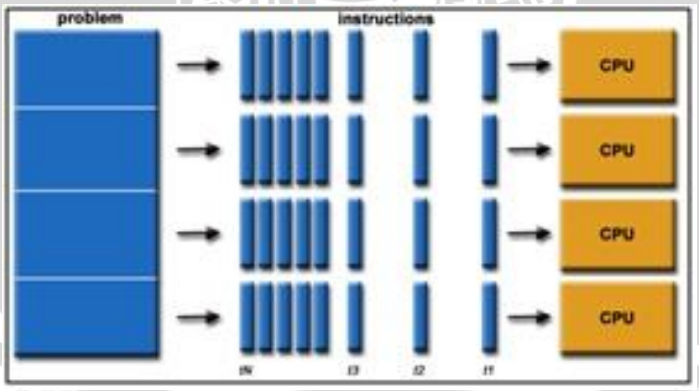
Selain alasan utama diatas, dapat ditambahkan beberapa alasan lainnya dibalik penggunaan komputasi paralel yaitu:

- Dapat menghemat waktu dibandingkan komputasi serial.
- Dapat menghemat biaya, terutama untuk kalangan pendidikan dibandingkan dengan melakukan pembelian 1 komputer sekuensial dengan kemampuan super.
- Dapat mengatasi keterbatasan fisik dari komputasi serial.
- *Fault tolerant*, sehingga ketika salah satu prosessor tidak bekerja akan digantikan yang lain meskipun dengan performa yang menurun dibandingkan performa sebelumnya



Gambar 2.1 Sistematika Kerja Proses Sekuensial

Sumber : [RMH-09]



Gambar 2.2 Sistematika Kerja Proses Paralel

Sumber : [RMH-09]

2.2.2 Potensi Peningkatan Kecepatan Komputasi

2.2.2.1 Faktor Pemercepat

Solusi waktu dapat dijadikan salah satu cara yang mudah dilakukan untuk dapat melakukan evaluasi kerja terhadap sistem komputasi yang dibangun secara

paralel. Namun dapat juga dilihat dari evaluasi analisa yang telah didapat, akan didapatkan bahwa sistem yang yang diimplemntasikan memiliki ciri-ciri tertentu. Selanjutnya ketika sistem tersebut dilakukan analisa lagi secara lebih mendalam maka dapat diperoleh ciri-ciri yang lain. Perbedaan dari hasil komputasi paralel yang di dapat dalam analisa, dapat digunakan untuk membandingkan aspek-aspek dari implementasi sistem yang berbeda dan mungkin menyediakan petunjuk bagaimana nantinya dapat dilakukan peningkatan sistem komputasi secara keseluruhan

Yang menjadi fokus utama dalam merencanakan solusi menggunakan sistem komputasi paralel adalah memperkirakan akan seberapa besar peningkatan kecepatan dari komputasi paralel ini dalam menyelesaikan suatu masalah. Pengukuran kinerja yang berguna di dalam sistem multiprosesor yaitu berkaitan dengan peningkatan kinerja komputasi. Ini berhubungan dengan waktu yang digunakan ketika dilakukan penyelesaian dari masalah yang dilakukan secara sekuensial degan waktu yang digunakan untuk menyelesaikan masalah yang sama dengan menngunakan komputasi secara paralel.

Perbandingan tersebut dapat dilakukan dengan menggunakan waktu proses *render* animasi 3D menggunakan sistem komputasi dengan waktu proses menggunakan *single processing* dengan komputasi menggunakan sistem *render farm* yang berdasarkan pada teori komputasi paralel. Faktor pemercepat, S(p)[1], adalah ukuran relatif performa yang didefinisikan sebagai berikut :

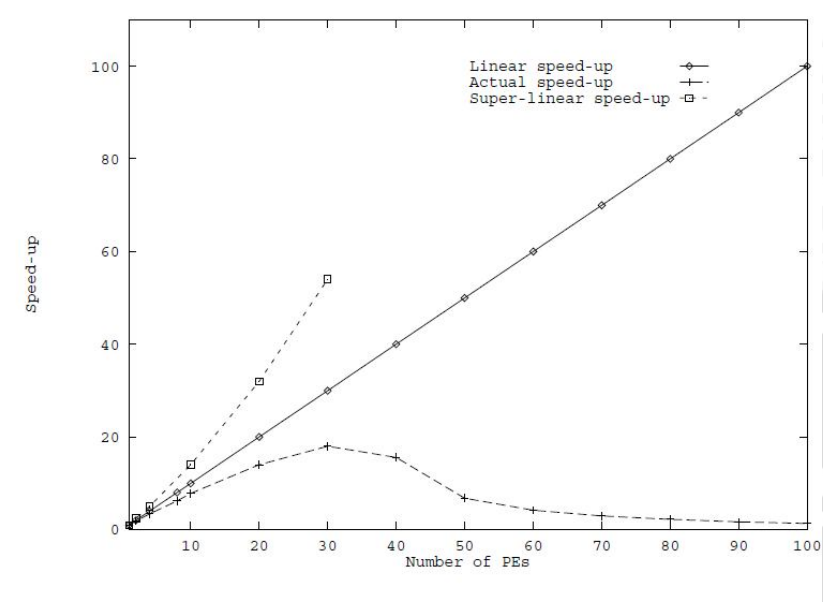
$$S\,(p) = \frac{\text{waktu proses menggunakan n sistem prosesor tunggal}}{\text{waktu proses menggunakan multi prosesor dengan p prosesor}} \qquad (2\text{-}1)$$

Dapat ditentukan t merupakan waktu proses untuk menyelesaikan masalah yang ada menggunakan sistem prosesor tunggal dan tp waktu proses untuk menyelesaikan permasalahan yang sama menggunakan sistem multiprosesor. Sehingga akan menghasilkan rumus :

$$S\,(p) = \frac{t}{tp} \qquad (2\text{-}2)$$

S(p) adalah peningkatan kecepatan jika menggunakan multiprosesor. Dapat dilihat bahwa algoritma dan proses kerja yang digunakan di dalam proses kerja menggunakan sistem multiprosesor dapat mengalami perbedaan atau tidak sama dengan algoritma yang dan proses kerja yang terjadi di dalam sistem kerja prosesor tunggal.

Istilah dari peningkatan kecepatan secara linier digunakan ketika solusi waktu proses dari sistem n prosesor memiliki nilai kecepatan n waktu lebih cepat dibandingkan solusi waktu proses menggunakan sistem komputasi paralel. Kecepatan komputasi secara linier adalah ekuivalent dari waktu yang optimal dapat disebabkan oleh bertambahnya jumlah prosesor. Waktu kecepatan komputasi yang optimal dan aktual di dalam gambar 2.2 mempresentasikan dari grafik linier dan kecepatan aktual di dalam gambar 2.3. Dapat dilihat di dalam gambar, jika grafik dari peningkatan kecepatan secara aktual akan mengalami peningkatan hingga berada pada posisi nilai tertentu dan akan mengalami penurunan. Dapat dikatakan dalam kondisi tersebut, implementasi dari sistem paralel yang dibangun mengalami penurunan kinerja komputasi.



Gambar 2.3 Peningkatan Kecepatan Secara Linier dan Aktual

Sumber : [DCH-00]

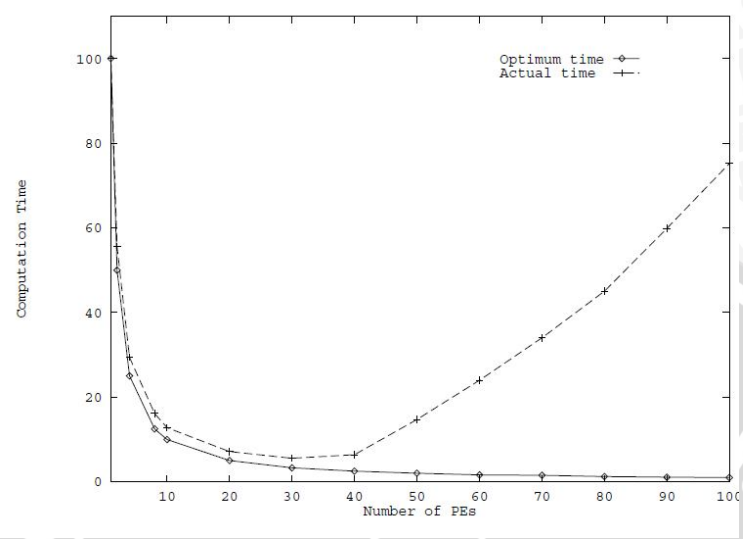
Kurva yang ketiga di dalam gambar 2.3 mempresentasikan grafik yang bernama *super-linier speed up*. Dari contoh yang digunakan, implementasi dari sistem komputasi paralel yang menggunakan prosesor berjumlah 20 buah dapat memiliki waktu proses komputasi sekitar 32 kali lebih cepat dibandingkan menggunakan sistem komputasi secara sekuensial. Hasil ini menjadi suatu hal yang berpolemik , karena tidak logis ketika kecepatan proses komputasi yang ada dapat lebih cepat dibandingkan dengan jumlah prosesor yang dimiliki oleh sistem paralel. Selama dalam percobaan, ini adalah sesuatu yang mungkin di dalam proses kerja sistem tersebut mencapai suatu kecepatan *super-linier speed up,* implementasi sistem yang seperti ini mungkin telah diimplementasikan untuk memperoleh solusi waktu seperti itu.

Sebagai contoh, prosesor paling modern memiliki angka yang terbatas dari *cache memory* dengan waktu akses yang secara signifikan lebih cepat jika dibandingkan dengan waktu akses dari memori yang standart. 2 buah prosesor dapat memiliki dua kali nilai dari cache memory.

Berdasarkan pertimbangan dari hasil analisa mengenai masalah besaran yang tetap ini berarti bahwa bagian lebih besar dari segmen masalah adalah mengenai sistem *cache memory* yang terdapat di dalam paralel dibandingkan dengan implementasi dari sistem di dalam komputasi sekuensial. Ini adalah sesuatu hal yang tidak masuk akal, untuk membayangkan sebuah situasi dimana 2 buah prosesor memiliki solusi waktu kerja lebih cepat dua kali dibanding sekuensial.

Jika kita berpikir jikalau prosesor yang sama yang digunakan di kedua sistem yang diimplementasikan, baik itu melalui komutasi sekuensial dan komputasi paralel dan mengambil kesimpulan bahwa, jumlah waktu yang digunakan untuk menyelesaikan suatu proses akan bertambah seiring dengan jumlah prosesor yang ditambah. Seperti jika kita menambahkan dua buah prosesor maka kecepatan akan menjadi dua kali kecepatan sistem sekuensial, menambahkan tiga buah prosesor menjadi tiga kali kecepatan sistem sekuensial, menambah n prosesor menjadi n kali lebih cepat. Jika n cukup besar dari proses ini, kita dapat mengharapkan skala besarnya implementasi dari komputasi paralel untuk memproduksi jawaban dari bagian kecil dari komputasi secara sekuensial yang ditunjukkan di dalam waktu optimal di dalam gambar 2.2.

Namun, di dalam kenyataan kerjanya tidak seperti mencapai waktu proses yang diharapkan ketika kita menambahkan jumlah prosesor ke dalam sistem. Skenario yang lebih realistis dapat ditunjukkan pada kurva waktu aktual pada gambar 2.2. Pada kurva tersebut menunjukkan sebuah awal penurunan dari waktu yang digunakan untuk menyelesaikan conoth masalah dari sistem paralel sampai dengan nilai tertentu dari elemen proses. Di luar dari angka ini, menambah jumlah prosesor sesungguhnya mengarahkan pada meningkatkannya waktu dari komputasi.



Gambar 2.4 Kecepatan Secara Aktual dan Optimal

Sumber : [DCH-00]

Efisiensi, berguna untuk mengetahui jumlah waktu yang diperlukan oleh prosesor untuk melakukan komputasi, hal itu bisa diketahui melalui efisiensi sistem. Dengan ini dapat diketahui seberapa besar sistem menggunakan komputasi secara paralel dapat memberikan efisiensi dalam kerjanya dibandingkan komputasi secara serial. Efisiensi, didefinisikan dengan :

$$E = \frac{\text{waktu eksekusi menggunakan satu buah prosesor}}{\text{waktu eksekusi menggunakan multiprosesor x jumlah proses}} \quad (2\text{-}2)$$

Sehingga diperoleh,

$$E = \frac{S(p)}{p} \times 100\,\% \quad (2\text{-}3)$$

2.3 Rendering

Rendering memainkan peran yang sangat vital pada proses penciptaan animasi dan gambar yang merupakan fase akhir dari pembuatan animasi 3D. *Rendering* animasi 3D melakukan prosesnya secara otomatis untuk mengkonversi dari sketsa *frame* 3D yang dibuat menjadi hasil animasi 2D dengan efek 3D ataupun tidak. Dengan mengesampingkan *style*, *rendering* dapat diselesaikan dengan menggunakan algoritma yang memang dikhususkan untuk proses *render*. Algoritma ini bisa menjadi sangat kompleks. Semakin kompleks algoritma yang digunakan maka semakin lama waktu yang dibutuhkan untuk melakukan proses kalkulasi. Kemampuan sumber daya komputer berpengaruh terhadap kecepatan proses *render* ini.

Proses *render* dapat memakan waktu yang cukup lama seperti proses *render* dari film animasi 3D sederhana dapat menghabiskan waktu antara beberapa menit sampai dengan ratusan menit untuk setiap framenya. Salah satu contoh yang dapat diambil dari film animasi 3D *Hollywood* adalah proses rendering pada adegan keramaian di film Shrek 2 dimana untuk setiap framenya membutuhkan waktu *render* yang sangat lama hingga empat puluh jam.

2.5 Blender

Blender 3D adalah sebuah perangkat *open source* grafik 3D yang dapat digunakan untuk membuat *modelling*, *rendering*, dan pembuatan animasi 3D yang awalnya dikembangkan sebagai aplikasi rumahan oleh *Dutch Animation Studio NeoGeo* dan *Not a Number Technologies (NaN)*. Pertama kali dikembangkan oleh Ton Roosendaal, yang sebelumnya dibuat oleh Ray Tracer yang sering disebut *Tracers* untuk *Amiga* pada tahun 1989.

Seperti perangkat *open source* yang lainnya, *blender* bersifat gratis untuk memakainya. Semua dapat memakainya, dan ada batasan di dalamnya. Pemakai dari *blender* terus mengalami pertumbuhan. Saat ini, banyak siswa dan desainer professional memakainya.

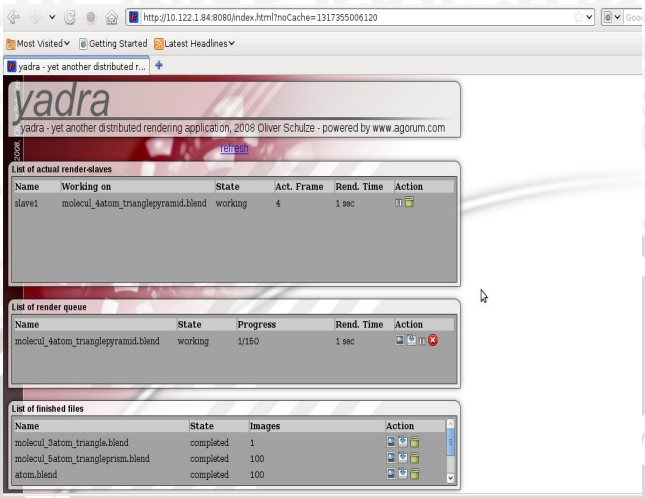
Blender sendiri tidak memerlukan perangkat keras yang memiliki kriteria *powerful.* Blender dapat digunakan oleh setiap orang yang ingin mencobanya. Andaikata pengguna ingin menggunakan *blender* untuk melakukan proses dengan hasil *rendering* yang sangat realistis, sistem dari memori RAM dan CPU akan sangat mempengaruhi ataupun dapat menggunakan sistem *render farm* untuk membantu proses *render* animasi 3D.

Kriteria minimum yang disarankan oleh *Blender Foundation* yaitu :

a. Open GL Graphics Card dengan 16 MB RAM
b. 300 MHz CPU
c. 128 MB RAM
d. 1024 x 768 *pixels display* dengan16-bit color
e. 20 MB *free hard disk Space*

2.6 Yadra

Yadra adalah suatu perangkat yang menyediakan pendistribusian untuk proses render pada animasi *blender* di suatu komputer *cluster* yang dapat digunakan dalam sistem operasi *linux* ataupun *windows*. Pengerjaan *rendering* pada jaringan membutuhkan satu *master* dan sedikitnya satu *slave*. Ketika seluruh elemen sistem telah diaktifkan maka sistem *render farm* siap untuk bekerja. *File blend* diupload dari salah satu *slave* yang bekerja kepada *master*. Tugas *master* ketika proses *render* berjalan yaitu melakukan *monitoring* proses, dan *slave* bertugas melakukan *render* terhadap *frame* yang dikirimkan *master*. Dalam pengembangannya awalnya, *yadra* menggunakan konsol atau terminal untuk penggunaannya. Salah satu dari tampilan ketika proses dari *monitoring yadra* dapat dilihat pada gambar 2.5



Gambar 2.5 Contoh Tampilan Yadra

Sumber : Dasar Teori

Adapun, fitur yang dimiliki oleh *Yadra* 1.0.1 yaitu :

1. Dapat melakukan proses render animasi *blender* di atas mesin secara bersamaan
2. Integrasi dari sistem *web interface* untuk
    - Kontrol pengerjaan *rendering*.
    - Kontrol terhadap mesin *master* ataupun *slave*.
    - Memantau antrian pekerjaan.
    - Melihat hasil dari *render*.
    - *Download* hasil dari *render*.
3. Berbasis *java*.
4. Penjadwalan dari pekerjaan di dalam prinsip *round robin*.
5. Keamanan data antara *master* dan *slave*.
6. Kemudahan instalasi.
7. *Render* dalam berbagai macam format.

Semua proses kerja yang dikerjakan oleh sistem *render farm* secara

konsep menggunakan prinsip dari algoritma penjadwalan *round robin*. Penjadwalan ini kerjanya bukan diinterupsi oleh proses kerja lainnya tetapi oleh penjadwal berdasarkan lama waktu berjalannya proses. Penjadwalan ini tanpa prioritas dan menganggap semua proses memiliki kepentingan yang sama. Keseluruhan proses dikerjakan secara berurutan dari atas ke bawah dalam sebuah daftar susunan dan proses akan kembali lagi ke atas dan seterusnya. Semua proses dianggap penting sehingga diberi sejumlah waktu oleh pemroses yang disebut kwanta (*quantum*) atau *time slice* dimana proses itu berjalan. Jika proses masih *running* sampai akhir *quantum*, maka CPU akan mem*preempt* proses itu dan memberikannya ke proses lain.

Algoritma kerja dalam sistem yadra menerapkan penetapan waktu pemroses atau kwanta dalam bentuk penjadwalan timer. Sama halnya dengan prinsip kerja dari round robin, sistem kerja dari yadra telah memiliki sistematika tersendiri dengan urutan kerja yang berurutan. Yang masing-masing urutan kerja tersebut telah dilakukan penetapan waktu kerjanya. Urutan kerja tersebut akan dikerjakan satu per satu secara berurutan sesuai dengan perintah kerja yang diinputkan oleh pengguna. Setiap kerja yang telah diberi waktu kerja akan segera diinterupsi proses kerjanya ketika proses kerja tersebut telah melewati batas waktu penjadwalan yang diberikan. Dan akan melanjutkan ke proses kerja selanjutnya. Ketika satu proses kerja diinterupsi, dapat dikatakan proses kerja keseluruhan sistem tidak berjalan secara sempurna dan dapat membuat terganggunya atau gagalnya jalannya sistem secara keseluruhan.

Lalu lintas pengiriman data di dalam yadra menggunakan fungsi dari pemograman java yaitu *socket* yang berfungsi sebagai abstraksi yang mana aplikasi dapat melakukan pengiriman dan menerima data sama halnya dengan membuka suatu file untuk dibaca dan ditulis pada tempat penyimpanan file. *Socket* memungkinkan untuk dapat masuk ke dalam jaringan dan berkomunikasi dengan aplikasi yang juga masuk ke dalam jaringan yang sama. Di dalam yadra, s*ocket* dibutuhkan sebagai jembatan komunikasi dan lalu lintas data antara *master server* dan *slave server*. *Socket*

mengidentifikasi masukan *port* yang diinputkan di dalam proses instalasi yadra. Sistem *port* dalam yadra termasuk ke dalam kategori *dynamically assigned port* yang merupakan *port-port* yang ditetapkan oleh sistem operasi atau aplikasi yang digunakan untuk melayani *request* dari pengguna sesuai dengan kebutuhan. *Dynamically Assigned Port* berkisar dari 1024 hingga 65536 dan dapat digunakan atau dilepaskan sesuai kebutuhan. Sehingga dapat digunakan port tersebut memang disediakan di dalam sistem *render farm.*

Ketika dilakukan proses aktivasi pada *master server* pada sistem, *socket server* akan membuka dan menunggu respons komunikasi dari *socket client* yang memiliki jalur komunikasi *port* yang sama. *Socket* akan saling berkomunikasi dan saling terhubung ketika *socket client* juga diaktifkan, kali ini yang terdapat pada *slave server*.

2.7 Pengukuran Kinerja Sistem Komputer

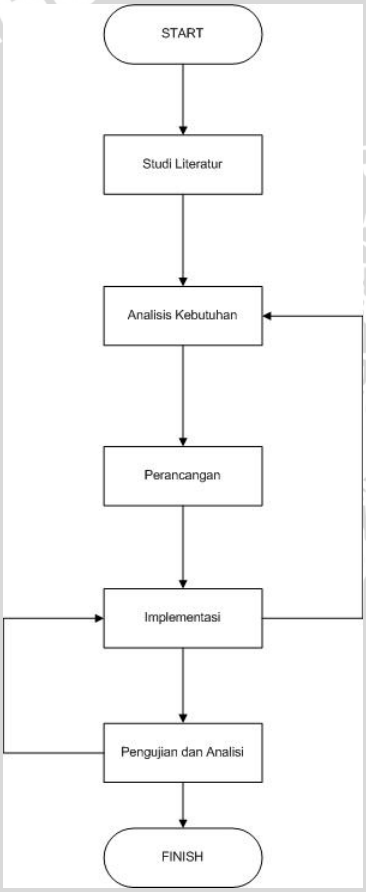
Kinerja komputer dapat dilakukan pengukuran dari jumlah waktu yang digunakan oleh komputer dalam mengeksekusi sebuah program atau perintah. Analisis kinerja akan membantu menjawab pertanyaan – pertanyaan mengenai kecepatan kinerja yang dapat dieksekusi oleh komputer. Untuk menjawab permasalahan tersebut, kita perlu menentukan waktu yang digunakan oleh suatu komputer untuk mengeksekusi. [SYR-10]

Perhitungan kinerja lainnya yang telah mendapat banyak perhatian belakangan ini adalah MIPS (laju eksekusi per unit waktu) yang dapat ditentukan dengan :

$$MIPS = \frac{jumlah\ instruksi}{waktu\ eksekusi\ x\ 10^6} \quad (2\text{-}4)$$

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dikembangkan. Metodologi penelitian selengkapnya akan ditunjukkan pada *flowchart* dibawah ini .



Gambar 3.1 Diagram flowchart metodologi

Sumber: Metodologi

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

1. *Parallel Computing*
2. Pemograman jaringan *Java* dengan *Netbeans*
3. *Render*
4. *Blender*
5. *Yadra*
6. Analisis Kinerja Komputer dan Jaringan Komputer
7. Software pendukung pengujian dan analisis kinerja

3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem. *Render Farm* memerlukan berbagai macam perangkat dan konfigurasi terhadap jaringan yang digabungkan menjadi sebuah sistem.

Untuk implementasi sistem *Render Farm* kebutuhan yang diperlukan seperti :

- Kebutuhan Perangkat Keras atau Hardware
    1. 3 buah *Personal Computer (PC)* yang akan bertindak sebagai *slave* atau *render engine*.
    2. 1 buah *Personal Computer (PC)* yang akan berfungsi sebagai *master*.
    3. *Switch*
    4. Jaringan LAN
- Kebutuhan Software Untuk Render Farm
    1. *Netbeans*
    2. *Tools Open Source Yadra*
    3. *Web Server* pada sisi *master* dan sisi *client*

4. *FTP server* untuk melakukan proses *download* hasil kerja dari sisi komputer *master*.
5. *OpenSSH* yang digunakan sebagai remote server
6. *Blender*
7. *OS Linux CentOS*

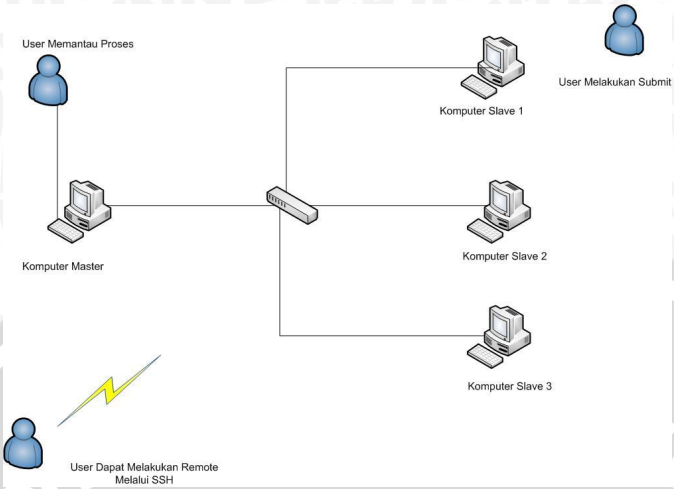
3.3 **Perancangan**

Perancangan dilakukan dengan memperhatikan berbagai aspek yang dibutuhkan untuk membangun sebuah sistem *render farm*.

Pada tahap awal perancangan dilakukan dengan pengumpulan dan pengidentifikasian terhadap berbagai macam *software* ataupun *hardware* yang telah ditentukan di dalam analisis kebutuhan.

PC atau komputer dalam sebuah LAN merupakan *node*, *node-node* ini akan terkoneksi ke jaringan melalui *Network Interface Card* (NIC) atau *network adapter* yang telah terpasang di masing-masing PC. Yang selanjutnya akan terhubung melalui *switch* untuk selanjutnya akan terus dikoneksikan dengan komputer *master* sebagai komputer utama di dalam sistem ini.

Secara umum, di dalam perancangan yang dilakukan tercantum dalam gambar 3.2.



Gambar 3.2 Gambaran perancangan sistem

Sumber : Metodologi

Berdasarkan gambar arsitektur di atas, pengguna sebagai pemakai akhir sistem akan terhubung dengan master komputer untuk melakukan *monitoring* proses dengan *web interface render farm* . Untuk proses *submit job* dapat melalui komputer *slave* (pada gambar di atas menggunakan komputer *slave* 1) untuk melakukan *submit job* ke *master* komputer. Selain itu ditambahkan pula aplikasi *SSH Execution* di dalam program aplikasi yang dibuat, sehingga pengguna nantinya dapat melakukan *remote* terhadap komputer *master* ataupun *slave* .

3.4 Implementasi

Implementasi aplikasi dilakukan dengan mengacu kepada perancangan aplikasi. *Yadra* yang dalam implementasinya menggunakan perintah *terminal* atau konsol akan dibuat menjadi program *desktop* sehingga pengguna dapat menggunakannya tanpa harus melalui perintah *terminal* atau konsol. Selanjutnya dilakukan implementasi pada sistem seperti melakukan setting konfigurasi pada bagian komputer *master*, *slave*, instalasi perangkat pendukung dan setting konfigurasi pada *jaringan* yang

digunakan. Proses implementasi tersebut bertujuan untuk menghubungkan berbagai komputer dengan jumlah lebih dari 1 unit untuk dapat bekerja di secara bersamaan dalam *parallel computing*. Selain itu juga dilakukan implementasi terhadap sistem *remote* terhadap *render farm.*

3.5 Pengujian dan Analisis

Pengujian sistem dilakukan agar dapat menunjukkan bahwa sistem yang telah dirancang dan diimplementasikan telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang melandasinya. Pengujian yang dilakukan dalam sistem ini akan memperhatikan berbagai aspek yang terdapat di dalam *parallel computing*.

Adapun parameter pengujian yang digunakan di dalamnya yaitu :

1. Kecepatan waktu *render.*
2. Kemampuan kinerja dari sumber daya yang digunakan.

Pengujian sistem *render farm* yang dilakukan dengan mendasar kepada parameter-parameter yang tersebut di atas dalam proses pengujiannya akan menggunakan skenario pengujian sebagai berikut :

- Skenario 1

  Melakukan pengukuran dari proses *render* yang dilakukan dengan menggunakan 1 PC dengan melakukan penambahan jumlah file yang di *render.*

- Skenario 2

  Melakukan pengujian sistem *render farm* dengan melakukan beberapa proses *render* secara bersamaan dengan beberapa file *blender*.

Berdasarkan skenario pengujian yang dilakukan, maka dapat dilakukan analisis secara menyeluruh terhadap sistem *render farm* yang dirancang. Adapun analisis yang dapat dilakukan di dalam sistem yaitu dengan melakukan analisis terhadap kinerja komputer, dilakukan dengan

melakukan pengukuran dan penghitungan serta perbandingan terhadap besarnya perubahan yang terjadi terhadap kinerja komputer ketika komputer melakukan proses *render* dengan menggunakan *single* komputer dengan proses yang menggunakan sistem *render farm*.

Untuk melakukan penghitungan mengenai kinerja komputer dibutuhkan juga perhitungan terhadap waktu terhadap proses yang berjalan, yang kemudian data yang dihasilkan dapat diolah di dalam rumus-rumus untuk mengukur kinerja jaringan.

Perhitungan yang dapat dilakukan dalam melakukan analisis kinerja komputer yaitu :

1. Penghitungan *MIPS*

   Pengujian *MIPS* (laju eksekusi per unit waktu) dilakukan untuk dapat mengetahui perbandingan kinerja dari tiap proses render yang dilakukan. Dapat menggunakan rumus :

$$MIPS = \frac{jumlah\ instruksi}{waktu\ eksekusi\ x\ 10^6} \quad (3\text{-}1)$$

2. Penghitungan perbandingan pemrosesan tunggal dan parallel

   Perbandingan tersebut dapat dilakukan menggunakan solusi terbaik yang dapat dilakukan prosesor tunggal, yakni waktu kerja pada sistem prosesor tunggal dibandingkan dengan waktu kerja *render* pada multiprosesor yang sedang diteliti. Dapat menggunakan rumus :

$$S\ (p) = \frac{\text{waktu proses menggunakan n sistem prosesor tunggal}}{\text{waktu proses menggunakan multi prosesor dengan p prosesor}} \quad (3\text{-}2)$$

3. Penghitungan efisiensi

   Efisiensi, sangatlah berguna untuk mengetahui jumlah waktu yang diperlukan oleh prosesor untuk melakukan komputasi. Dapat menggunakan rumus :

$$E = \frac{\text{waktu eksekusi menggunakan satu buah prosesor}}{\text{waktu eksekusi menggunakan multiprosesor x jumlah proses}} \quad (3\text{-}3)$$

Sehingga diperoleh,

$$E = \frac{S(p)}{p} \times 100\ \% \quad (3\text{-}4)$$

BAB IV
PERANCANGAN DAN IMPLEMENTASI

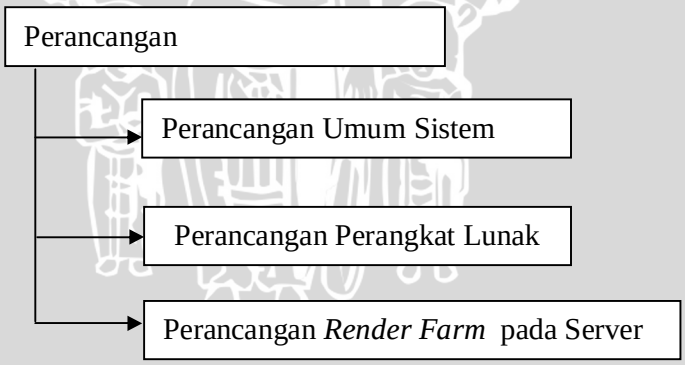
Pada bab ini akan dijelaskan mengenai mekanisme dari perancangan dan implementasi sistem *render farm* yang akan diterapkan.

4.1 Perancangan

Secara umum, di dalam perancangan yang dilakukan ini dibagi menjadi 3 bagian utama arsitektur perancangan yaitu :

1. Perancangan umum sistem
2. Perancangan aplikasi *desktop* untuk *yadra*
3. Perancangan *render farm* pada *server*

Diagram blok yang menggambarkan perancangan perangkat lunak sistem ini ditunjukkan oleh Gambar 4.1



Gambar 4.1 Diagram Perancangan Sistem

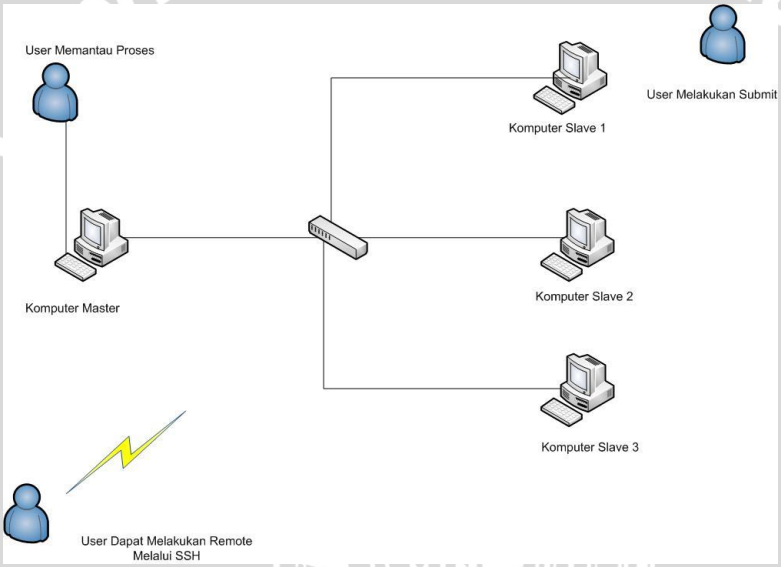
Sumber : Perancangan dan implementasi

Proses perancangan sistem secara umum dilakukan pada tahap pertama kemudian dilanjutkan dengan perancangan perangkat lunak pada

tahap selanjutnya dan tahap yang terakhir adalah proses perancangan *render farm* pada *server* yang pada tahap ini merupakan bagian penerapan dari perangkat lunak yang digunakan pada *server*.

4.1.1 Perancangan Umum Sistem

Perancangan sistem dilakukan untuk merepresentasikan arsitektur dari sistem yang dibuat secara keseluruhan. Arsitektur sistem yang dirancang ditunjukkan dalam gambar 4.2 berikut



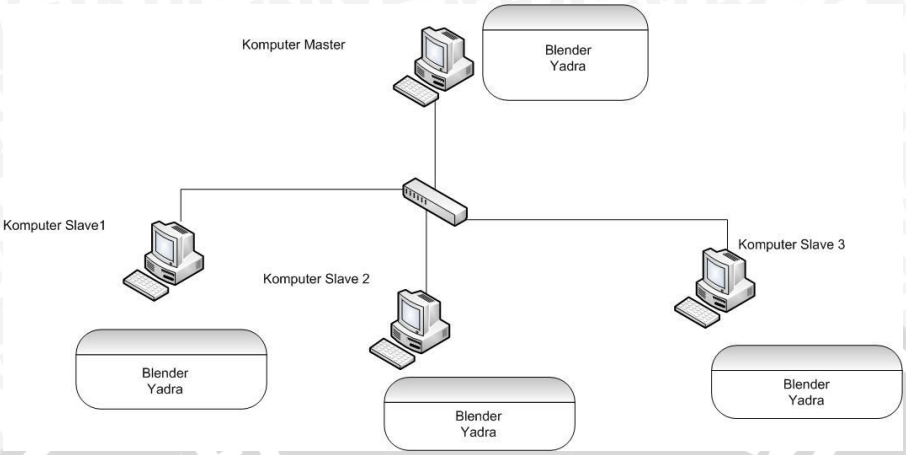
Gambar 4.2 Perancangan Sistem Render Farm

Sumber : Perancangan dan implementasi

Render farm secara singkat dapat dijabarkan adalah penggunaan beberapa komputer yang digunakan secara bersamaan untuk melakukan proses *render*. Komputer-komputer tersebut akan dihubungkan dengan jaringan LAN dengan 1 komputer yang akan menjalankan fungsi sebagai komputer master yang nantinya ketika proses berjalan dapat digunakan untuk melakukan proses *monitoring* dari kerja *render farm*. Kemudian

komputer lain akan bekerja sebagai *slave* yang menjalankan proses render secara bersamaan.

Di dalam sistem *render farm* yang dirancang,digunakan 2 buah perangkat lunak penting yang masing-masing perangkat lunak tambahan ini akan ditambahkan pada setiap bagian server, baik pada bagian *master* ataupun *slave*. *Blender* yang merupakan perangkat lunak olah data animasi 3D, di dalam sistem *render farm* ini digunakan sebagai perangkat tambahan yang digunakan yang berfungsi untuk melakukan proses *rendering* terhadap file .blend yang akan di kirimkan ke dalam sistem *render farm*. Sebagai perangkat lunak olah animasi 3D fungsi *render blender* yang akan digunakan oleh sistem *render farm* untuk menyelesaikan setiap tugas rendering yang dikirimkan oleh pengguna. Selanjutnya yaitu *Yadra*, perangkat ini yang digunakan untuk membangun sistem *render farm*. Tugas-tugas di dalam sistem render farm akan dilakukan oleh *Yadra* seperti pendistribusian *frame* yang akan dirender, melakukan koneksi antar komputer *server* dan tugas-tugas lainnya. *Yadra* sendiri tidak dapat melakukan render terhadap file .blend yang dikirimkan oleh pengguna, sehingga yadra juga membutuhkan keberadaan perangkat lunak *blender* yang juga ditanamkan ke dalam sistem. Di dalam sistem *render farm* yang dirancang, kedua perangkat penting ini akan saling berhubungan dalam bekerja saat melakukan *render*. *Yadra* yang akan mengatur segala hal yang berhubungan pada proses distribusi pekerjaan dan koneksi antar *server* didalamnya, dan *blender* yang melakukan proses *render*. Tambahan kedua perangkat yang digunakan di dalam sistem *render farm* dapat dilihat di dalam gambar 4.3.



Gambar 4.3 Blender dan Yadra dalam Sistem

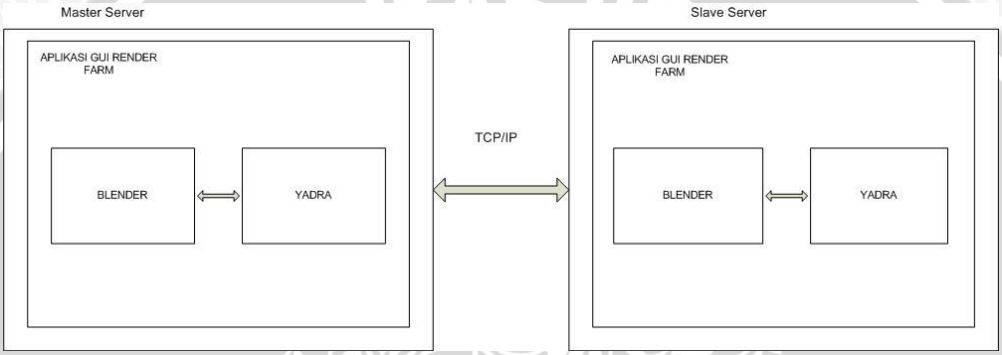
Sumber : Perancangan dan implementasi

Sistem *render farm* ketika diimplementasikan ke dalam sistem seperti pada gambar 4.3 diatas, sudah dapat membangun suatu sistem *render farm*. Sistem ini telah dapat berjalan dan digunakan oleh pengguna sistem. Penggunaan *Yadra* dari proses instalasi, aktivasi ataupun dalam melakukan proses permintaan *submit job* ke dalam *server* keseluruhannya menggunakan perintah terminal atau perintah konsol. Dan dirasakan sistem *render farm* yang masih menggunakan konsol atau terminal untuk penggunaannya dapat menyulitkan pengguna yang tidak akrab dengan penggunaan bahasa perintah dari terminal ataupun konsol baik itu perintah di windows ataupun linux.

Dan untuk mengatasi permasalahan diatas , maka ditambahkan tambahan fitur di dalam *yadra* yaitu berupa penambahan GUI atau tampilan *desktop* dari sistem *render farm.* Tambahan GUI di dalam sistem ini, akan membuat jalannya sistem *render farm* menjadi lebih mudah, dikarenakan keseluruhan proses berjalannya *render farm* dilakukan di dalam GUI tersebut. Pengguna melakukan proses kerja sistem render farm dengan mengakses tombol-tombol dan mengisi beberapa inputan di dalamnya.

Tambahan fitur GUI di dalam sistem *render farm* diimplementasikan menggunakan bahasa pemograman *java.* Keseluruhan perintah dan variabel

yang digunakan menggunakan perintah dan variabel yang disesuaikan dengan yang dimiliki oleh *yadra*. Selain itu juga tidak dilakukan perubahan terhadap jalannya sistem *render farm* yang fungsi-fungsinya dimiliki oleh *yadra*. Sehingga melalui GUI ini, pengguna dapat langsung terhubung di dalam sistem *render farm* melalui tampilan program *desktop*. Skema kerja rancangan pembuatan user interface dapat dilihat pada gambar 4.4



Gambar 4.4 Blender dan Yadra dalam GUI Sistem

Sumber : Perancangan dan implementasi

Dalam gambaran skema kerja diatas, dapat dilihat jika sistem *interface* akan membungkus tampilan dari sistem render farm. Di dalamnya *blender* dan *yadra* yang akan bekerja dalam proses *render* file .blend yang telah di *submit* ke dalam sistem.

4.1.2 Topologi Jaringan

Topologi jaringan komputer yang digunakan dalam sistem *render farm* ini, menggunakan jaringan LAN yang diterapkan pada beberapa unit komputer. Topologi ini dapat membuat jaringan yang nantinya dapat memungkinkan keseluruhan komputer untuk saling terkoneksi satu sama lain dan melakukan proses sebagaimana komputasi paralel.

4.1.3 Kebutuhan Hardware

Pada tabel 4.1 dijelaskan tentang perangkat yang digunakan, fungsi dari perangkat-perangkat tersebut, dan alokasi IP yang diberikan pada masing-masing perangkat agar dapat saling berhubungan.

Tabel 4.1 Alokasi IP dari komputer yang digunakan

Perangkat	Keterangan	Alokasi IP
PC 1	Sebagai master	10.10.10.1
PC 2	Sebagai node	10.10.10.2
PC 3	Sebagai node	10.10.10.3
PC 4	Sebagai node	10.10.10.4
PC 5	Sebagai node	10.10.10.5

Sumber : Perancangan dan implementasi

Sistem *hardware* yang dibutuhkan dalam membangun sistem ini sehingga dapat berjalan dijelaskan pada tabel 4.2 dan 4.3.

Tabel 4.2 Spesifikasi hardware minimum yang dirancang untuk master

No.	Deskripsi	Spesifikasi
1	CPU	Intel® Core™2 Duo CPUT6600 @ 2.20GHz / diatasnya
2	RAM	2 GB DDR2
3	*Platform*	Linux Centos
4	Sistem Koneksi	LAN 100Mbps / diatasnya
5	Penyimpanan Data	20 GB

Sumber : Perancangan dan implementasi

Tabel 4.3 Spesifikasi minimum yang dirancang untuk slave

No.	Deskripsi	Spesifikasi
1	CPU	Intel® Core™2 Duo CPUT6600 @ 2.20GHz / diatasnya
2	RAM	2 GB DDR2
3	*Plattform*	Linux Centos
4	Sistem Koneksi	LAN 100Mbps / diatasnya
5	Penyimpanan Data	20 GB

Sumber : Perancangan dan implementasi

Pada tabel diatas spesifikasi untuk *master server* menggunakan prosesor *Intel® Core™2 Duo CPUT6600* @ 2.20GHz / diatasnya, dengan jumlah memori *RAM* 2 GB *DDR2*, dan koneksi data pada jaringan LAN menggunakan kecepatan minimal 100Mbps. Sedangkan pada tabel kedua adalah spesifikasi untuk *node server*, adalah menggunakan prosesor *Intel® Core™2 Duo* CPUT6600 @ 2.20GHz / diatasnya dengan jumlah memori *RAM* 2 GB *DDR2*, dan koneksi data pada jaringan LAN menggunakan kecepatan minimal 100Mbps. Dan untuk kapasitas penyimpanan yang digunakan adalah 20GB untuk instalasi sistem operasi dan untuk penyimpanan data.

4.1.4 Kebutuhan Software

Adapun kebutuhan *software* dan perangkat-perangkat yang nantinya akan digunakan di dalam membangun sistem *render farm* yaitu:

1. Instalasi *CentOS*

Sistem operasi *linux* yang digunakan pada sistem *render farm* ini adalah *CentOS* 6.2 yang akan diinstal pada keseluruhan komputer yang digunakan baik *master* ataupun *node server*. Kebutuhan di dalam sistem akan disesuaikan dengan operasi sistem *CentOs*, termasuk dalam perancangan perangkat lunak.

2. Yadra

Yadra merupakan salah satu bagian penting dalam pembangunan sistem ini. *Yadra* merupakan perangkat yang memungkinkan pembangunan dari sistem *render farm* itu sendiri. Dalam implementasinya *yadra* memiliki kekurangan yaitu diharuskannya instalasi menggunakan sistem konsol dan tidak dapat dilakukan sistem *remote* terhadap *server*. Dalam sistem ini *interface* yadra yang tadinya berupa konsol digantikan oleh GUI *desktop* untuk memudahkan *pengguna*, selain itu ditambahkan fitur untuk *remote server* .

3. Blender

Blender tidak dapat dipisahkan dalam pembangunan server ini, karena *yadra* sendiri tidak dapat bekerja tanpa adanya *blender* yang telah terinstal di dalam server. Dalam proses instalasi *Yadra* dimasukkan direktori dimana *blender* diinstal di dalam server. Selain itu, *blender* sendiri juga dapat digunakan dalam proses pengujian sebagai pembanding ketika dilakukan proses *render* dengan menggunakan *render farm* dan menggunakan 1 buah PC.

4. OpenSSH Server

Salah satu fitur yang ditambahkan di dalam aplikasi yang akan dibangun yaitu penambahan fasilitas *remote server*. Dengan fitur tambahan ini pengguna dapat melakukan proses kerja dari jarak jauh dengan fasilitas *remote*. Syarat yang harus dipenuhi untuk menambahkan fasilitas ini yaitu instalasi perangkat *OpenSSH* pada *server*. Dengan adanya *OpenSSH* akan mengaktifkan *SSH tunneling* pada server yang memungkinkan

pemanfaatan fasilitas *SSHexecute library* pada aplikasi yang dibuat.

5. Web Server

Web server digunakan sebagai perangkat penunjang bagi *interface* dalam proses monitoring dari renderfarm. Proses *monitoring* dilakukan pada fasilitas *website* yang telah disediakan oleh *Yadra*. Untuk membuat *web Server* digunakan APACHE *Web Server*.

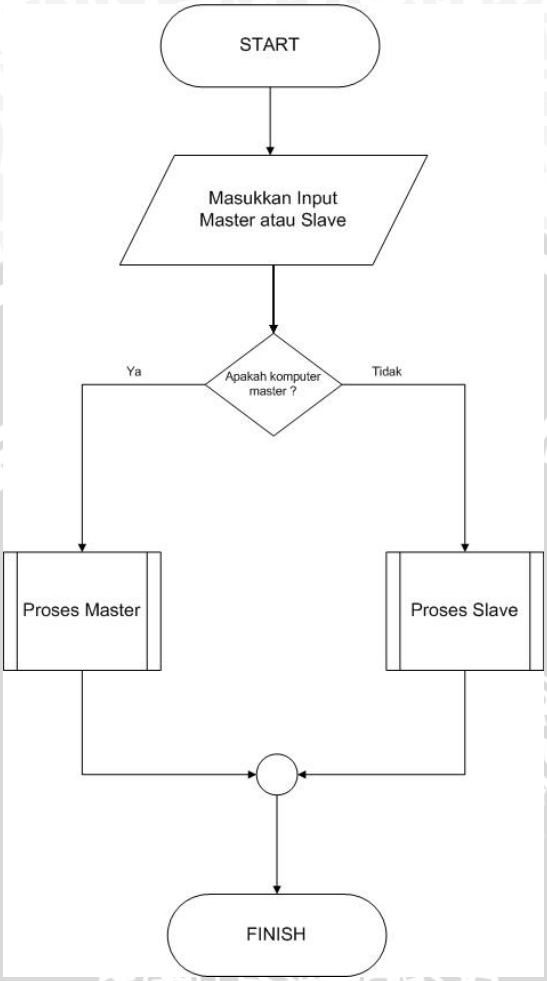
4.1.5 Perancangan Perangkat Lunak

Perancangan sistem program *render farm* merupakan tahap awal yang bertujuan untuk mengetahui gambaran program yang akan dibuat. Perancangan sistem program dititikberatkan pada bagaimana nantinya *render farm* yang dibangun menggunakan perangkat *Yadra* dapat dijalankan menggunakan aplikasi melalui *GUI desktop*.

4.1.6 Perancangan Algoritma Aplikasi

Program perancangan dibagi menjadi tiga proses perancangan algoritma yaitu perancangan algoritma untuk instalasi sistem, algoritma untuk aktivasi sistem, algoritma untuk melakukan *submit job* dan algoritma untuk *remote*.

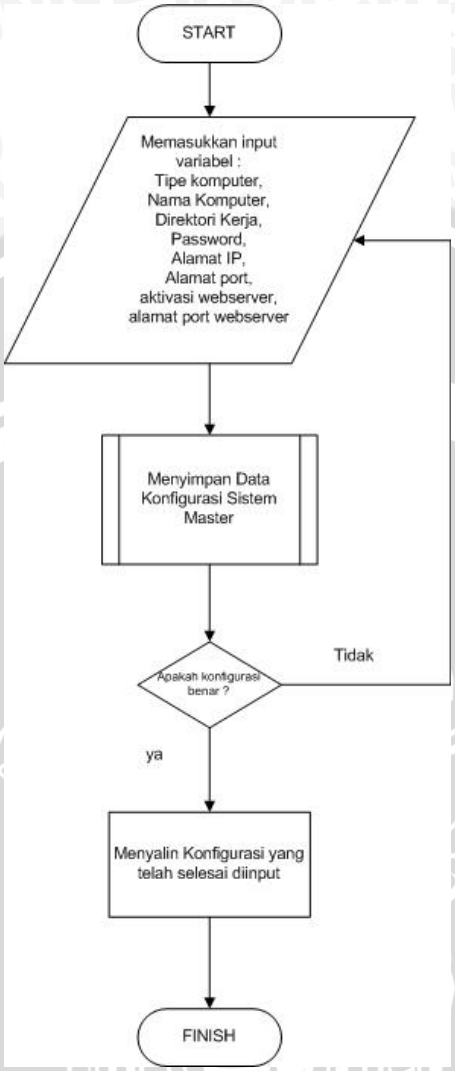
Perancangan perangkat lunak memiliki diagram *flowchart* yang ditunjukkan oleh gambar 4.5 hingga gambar 4.10.



Gambar 4.5 Flowchart Instalasi Sistem Render Farm

Sumber : Perancangan dan implementasi

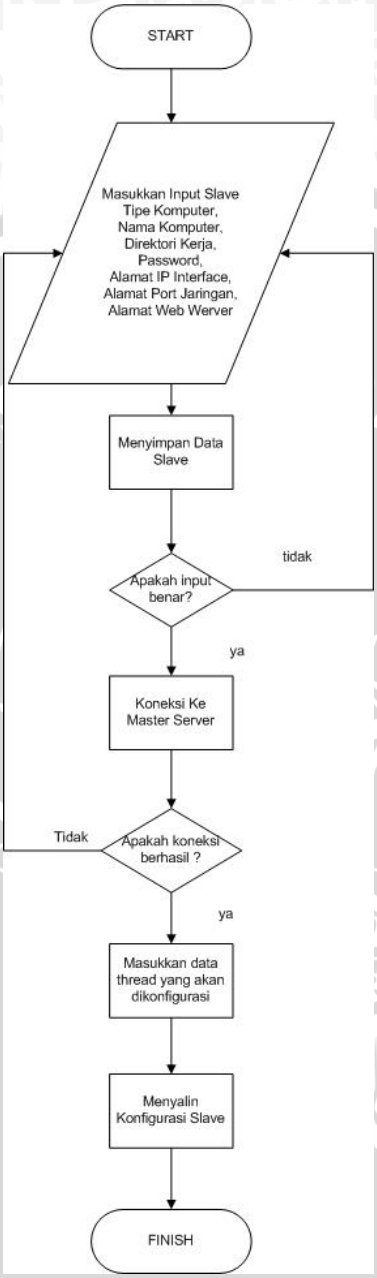
Pada gambar 4.5 dijelaskan mengenai *flowchart* dari proses instalasi sistem secara keseluruhan. Dikarenakan ada 2 jenis komputer berbeda yang digunakan dan masing-masing memiliki fungsi tersendiri. Pengguna akan diberi 2 pilihan proses instalasi yaitu *master* atau *slave*. Setiap komputer *master* ataupun *slave* memiliki proses dan masukan yang berbeda.



Gambar 4.6 Flowchart Perancangan Instalasi Master

Sumber : Perancangan dan implementasi

Pada *flowchart* 4.6, telah di titikberatkan pada proses instalasi pada komputer *master*. Pada proses ini, pengguna memasukkan beberapa masukan yang kemudian masukan tersebut akan disimpan dan diproses. Hasil masukan pada komputer *master* yang telah disimpan dan diproses menjadi langkah awal dari membangun sistem *render farm*, karena beberapa masukan dari *slave* harus memiliki kesamaan dari masukan *master*.

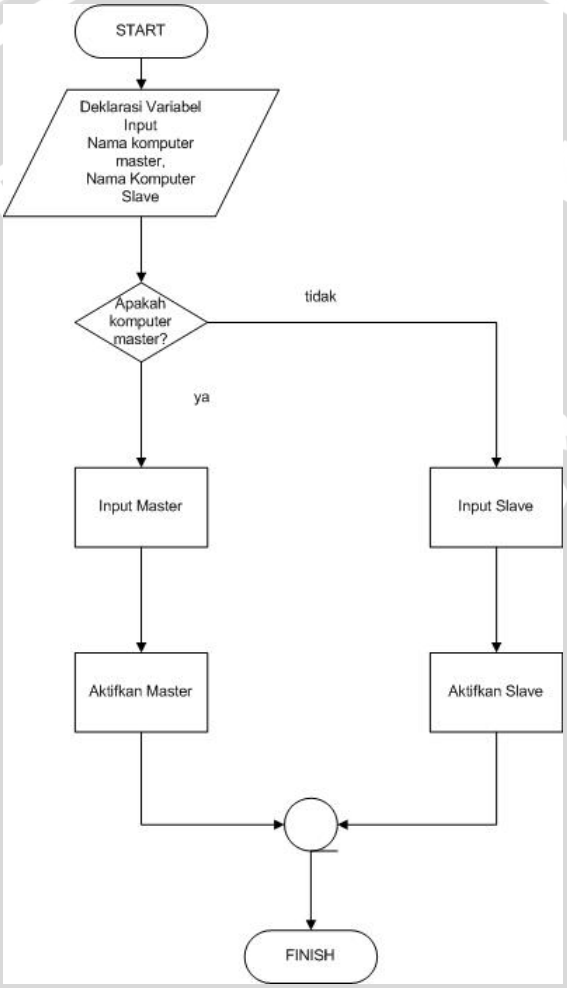


Gambar 4.7 Flowchart Perancangan Instalasi Slave

Sumber : Perancangan dan implementasi

Setelah proses instalasi pada *master* selesai, maka dilanjutkan proses pada *slave.* Pada proses ini, pengguna juga memasukkan beberapa masukan

yang kemudian masukan tersebut akan disimpan dan diproses. Komputer *slave* akan mengecek koneksi jaringan ke *master*, apabila koneksi terputus proses tidak dapat dilanjutkan. Beberapa perintah masukan diharuskan memasukkan masukan yang sama dengan masukan dari *master*.

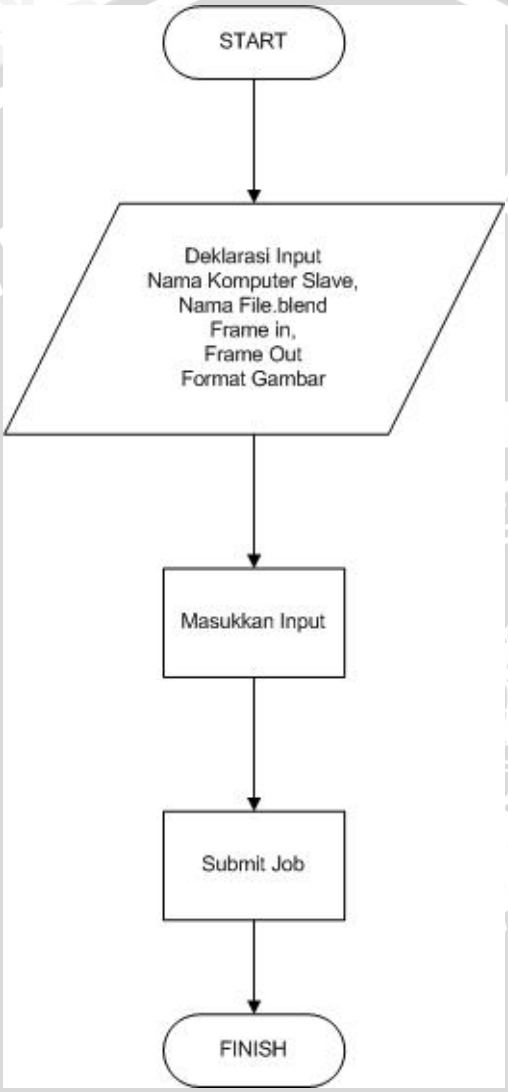


Gambar 4.8 Flowchart Perancangan Aktivasi Server Master dan Slave

Sumber : Perancangan dan implementasi

Aktivasi sistem dilakukan untuk membuat sistem menjadi siap untuk

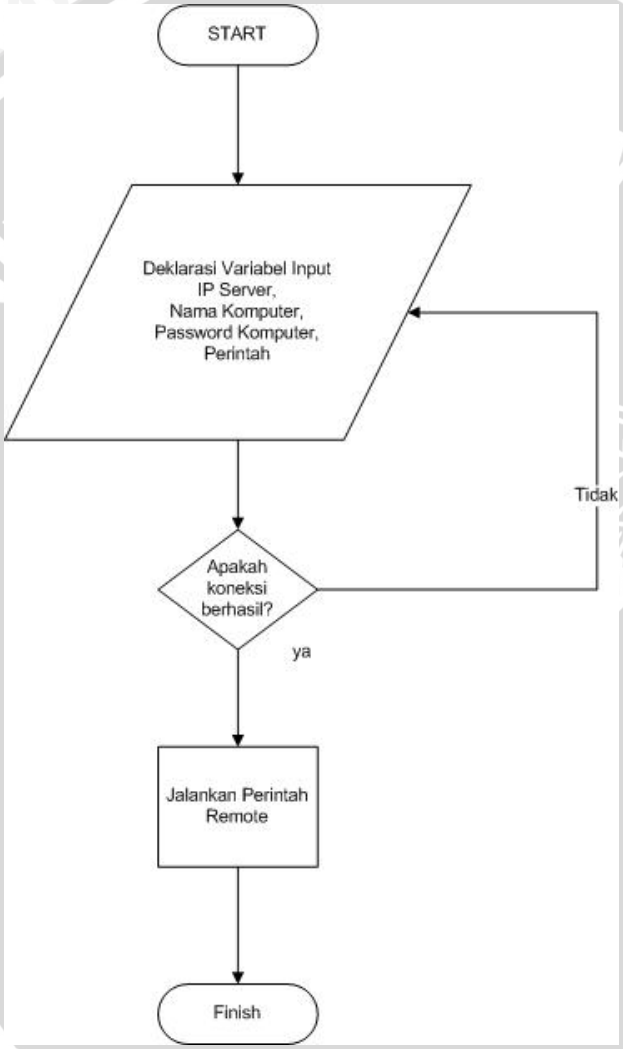
melakukan proses *render*. Aktivasi dilakukan satu per satu dari komputer *master* dan *slave*. Pengguna memilih *master* atau *slave* dan kemudian memasukkan nama komputer yang diaktifkan.



Gambar 4.9 Flowchart Perancangan *Submit Job*

Sumber : Perancangan dan implementasi

Proses *submit job* berfungsi untuk mengirimkan *request* file .blend yang akan dirender dari komputer *slave* ke komputer *master*. Proses ini dapat dilakukan di semua komputer *slave*, sehingga pengguna diharuskan memasukkan nama komputer yang melakukan proses *submit job*.



Gambar 4.10 Flowchart Perancangan Remote Sistem

Sumber : Perancangan dan implementasi

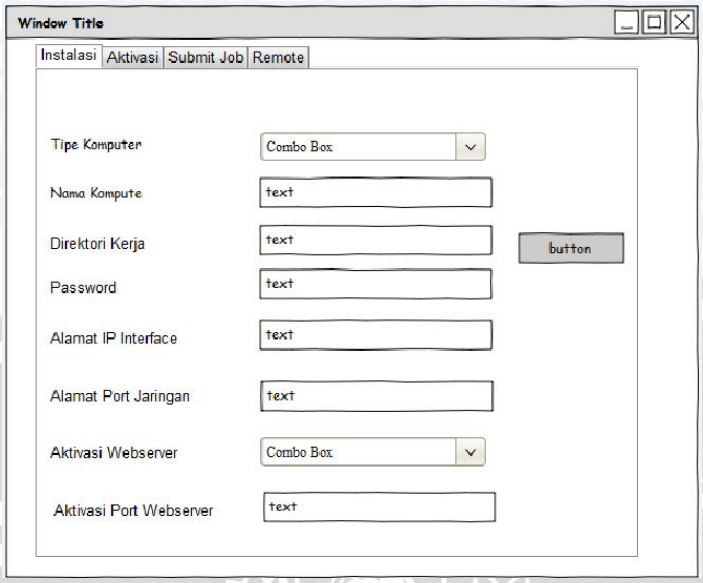
Flowchart yang terakhir menggambarkan proses dari *remote* terhadap komputer *server*. Dalam proses ini, pengguna memasukkan masukan seperti alamat IP yang dituju, nama komputer dan perintah yang dijalankan. Sistem akan melakukan koneksi terhadap komputer yang dituju, apabila koneksi gagal sistem akan mengembalikan ke pengguna untuk melakukan masukan kembali.

4.1.7 Perancangan Layout Aplikasi

Program yang dibangun terdiri dari tiga bagian yang dibagi menggunakan tab. Bagian pertama merupakan *layout* untuk melakukan instalasi sistem. Pada bagian ini terdapat beberapa *textfield* kosong yang tiap *textfield* tersebut memiliki fungsi masing-masing yang harus diisi oleh *pengguna*. Textfield itu terdiri dari :

1. Pemilihan komputer master atau komputer *slave*
2. Nama komputer *server*
3. Direktori kerja
4. *Password* atau *passphrase*
5. Alamat IP *interface*
6. Alamat *port* jaringan
7. Pemilihan aktivasi *web server*
8. Alamat *port webserver*

Tampilan *layout* dari bagian instalasi sistem ditunjukkan pada gambar 4.11.

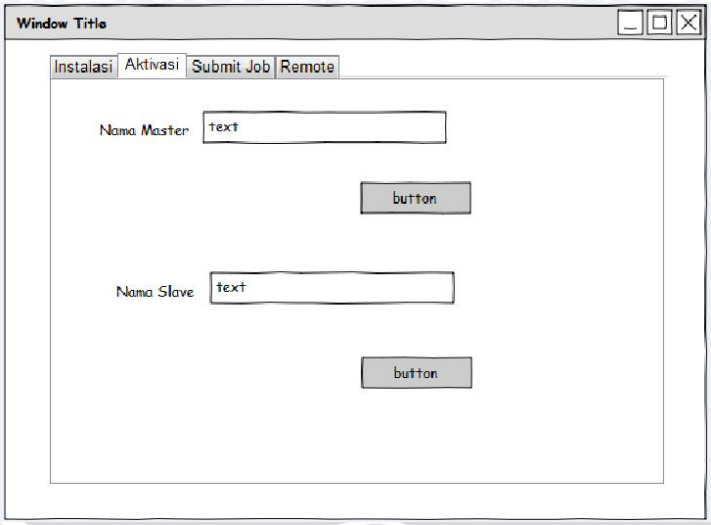


Gambar 4.11 Perancangan Tampilan Instalasi Sistem

Sumber : Perancangan dan implementasi

Pada bagian kedua merupakan layout untuk melakukan aktivasi terhadap sistem. Ada 2 jenis aktivasi yang dapat dilakukan yaitu aktivasi pada komputer *master* dan aktivasi pada komputer *slave*. Aktivasi ini diperlukan untuk mengaktifkan sistem pada *master* dan pada *slave* sehingga kedua sistem tersebut menjadi dalam keadaan siap untuk menerima data .blend untuk di *render*. Pada bagian ini akan terdapat *textfield* yang diisi yang merupakan nama dari *server* yang akan diaktifkan. Ada 2 *textfield* yang digunakan yaitu *textfield* master dan server serta 2 tombol *submit* untuk mengaktifkan.

Tampilan *layout* dari bagian aktivasi sistem ditunjukkan pada gambar 4.12



Gambar 4.12 Perancangan Tampilan Aktivasi Sistem

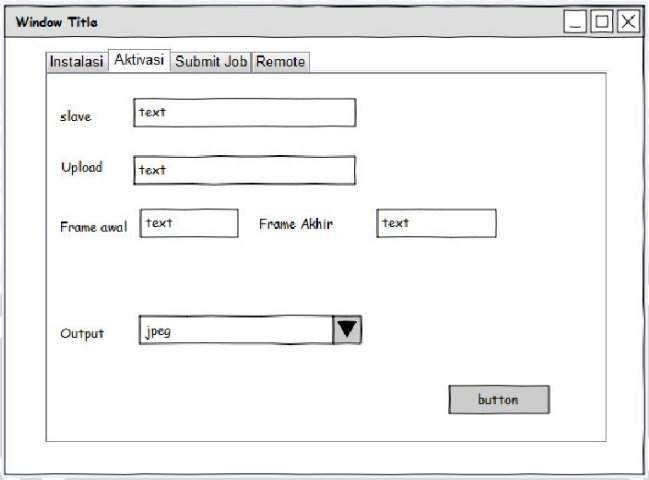
Sumber : Perancangan dan implementasi

Bagian ketiga merupakan bagian dimana pengguna dapat melakukan *submit job* dari komputer *slave* ke komputer *master*. Terdapat beberapa *textfield* yang juga harus diisi oleh pengguna.

Textfield itu terdiri dari :

1. Nama komputer *slave*
2. *Upload file* .blend yang akan direnderr
3. *Frame* awal dari file
4. *Frame* akhir dari file
5. *Textfield* akhir gambar setelah direnderr seperti jpeg atau png.

Tampilan *layout* dari bagian *submit job* ditunjukkan pada gambar 4.13.



Gambar 4.13 Perancangan Tampilan Submit Job

Sumber : Perancangan dan implementasi

Bagian keempat merupakan bagian dimana pengguna dapat melakukan *remote* jarak jauh terhadap komputer *slave* ataupun komputer *master* terdapat beberapa *textfield* yang juga harus diisi oleh pengguna.

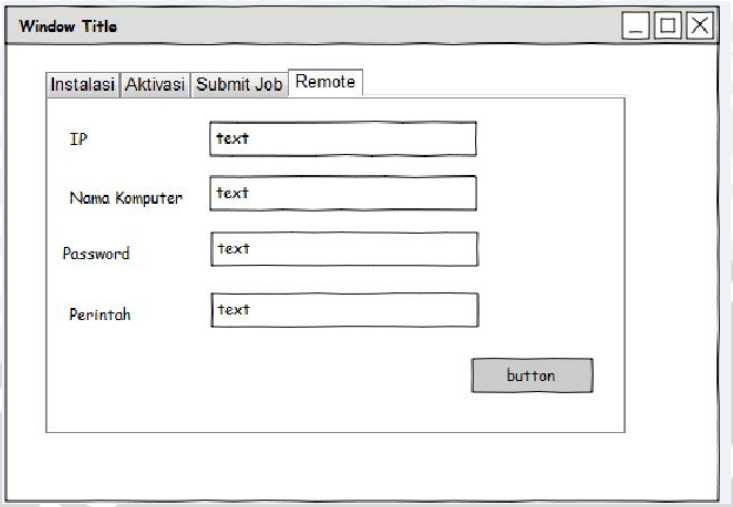
Textfield itu terdiri dari :

1 IP dan nama komputer yang dituju.

2 *Password* komputer yang dituju.

3 Perintah *remote*

Tampilan *layout* dari bagian *submit job* ditunjukkan pada gambar 4.14.



Gambar 4.14 Perancangan Tampilan Remote Sistem

Sumber : Perancangan dan implementasi

4.1.8 Penggunaan Sistem Render Farm

Untuk perancangan dari sistem *render farm*, difokuskan terhadap implementasi menggunakan perangkat *Yadra*. Sistem *render farm* dibangun setelah selesainya program *interface desktop yadra* dibangun. Untuk sistem ini digunakan aplikasi *Yadra* versi 1.0.1.

Render farm dapat dikerjakan ketika implementasi dari sisi topologi jaringan dan aplikasi *interface yadra* telah selesai dikerjakan.

4.2 Batasan – Batasan Implementasi

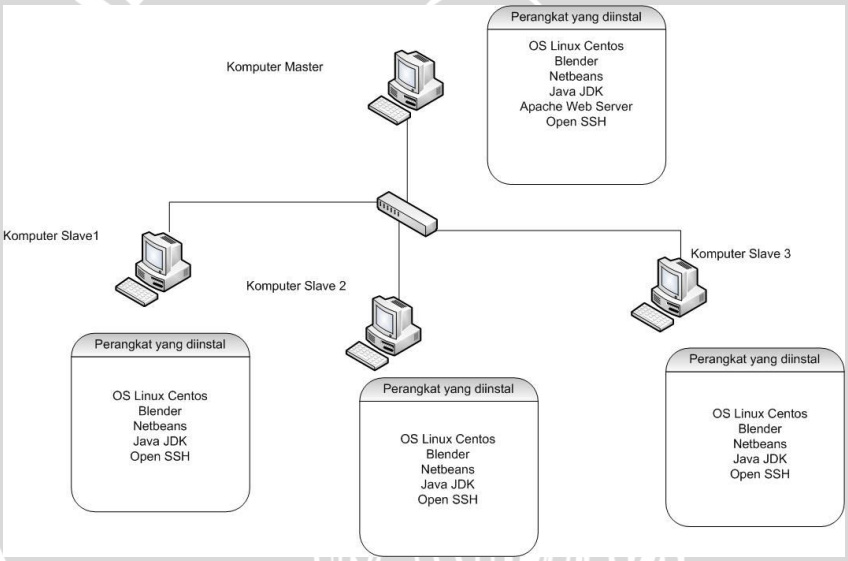
Beberapa batasan dalam mengimplementasikan sistem *render farm* adalah sebagai berikut :

1. Operasi sistem yang digunakan di dalam sistem render farm yaitu *CentOS* 6.
2. Topologi jaringan menggunakan jaringan LAN sebagai interkoneksi antar PC.
3. Sistem *GUI render farm* dirancang dan dijalankan dengan menggunakan *Java Desktop Application*.

4. Perangkat lunak 3D yang digunakan yaitu blender 3D.
5. *File* 3D yang dirender memiliki format *.blend.*

4.3 Implementasi Perangkat Pendukung

Dalam implementasi sistem *render farm* juga dilakukan instalasi terhadap beberapa perangkat pendukung yang dibutuhkan di dalam sistem *render farm* agar sistem dapat berjalan dengan baik. Implentasi perangkat pendukung ditunjukkan pada gambar 4.15.



Gambar 4.15 Implementasi Perangkat Pendukung

Sumber : Perancangan dan implementasi

4.4 Implementasi Algoritma

Perangkat lunak yang dibuat memiliki beberapa proses di dalamnya. Beberapa proses yang dicantumkan dalam penulisan makalah skripsi ini hanya untuk algoritma dari beberapa proses (operasi) saja sehingga tidak semua *method* akan dicantumkan. Implementasi algoritma ini

direpresentasikan dalam bentuk *pseudocode* dari algoritma – algoritma tersebut.

4.4.1 Implementasi Algoritma Proses Instalasi

Proses instalasi berfungsi untuk melakukan instalasi pada komputer master dan slave. Implementasi algoritma untuk melakukan insatalasi dijelaskan pada Gambar 4.16

Nama Algoritma: Instalasi

Deklarasi :

- String → Tipe komputer
- String → Nama komputer
- String → Direktori kerja
- String → Password
- String → Alamat IP
- Integer → Alamat port jaringan
- String → Web server
- Integer → Alamat port webserver

Deskripsi :

- Masukan : Tipe komputer, nama komputer, direktori kerja, password, alamat IP, alamat por jaringan, ,web server, alamat port webserver
- Proses
  1. Memasukkan variabel masukan.
  2. Variabel masukan yang telah dimasukkan akan disimpan.
  3. Akan dilakukan pengecekan konfigurasi masukan yang dimasukkan .
  4. Jika data masukan telah benar akan dilakukan proses *copyscript* dalam bentuk file .sh ataupun file .bat

Keluaran : -

Gambar 4.16 Implementasi algoritma untuk proses instalasi

Sumber : Perancangan dan implementasi

4.4.2 Implementasi Algoritma Proses Aktivasi

Proses instalasi berfungsi untuk melakukan aktivasi pada komputer *master* dan *slave*. Aktivasi dilakukan untuk membuat komputer *master* dan *slave* menjadi aktif sebelum keduanya siap melakukan pekerjaan. Implementasi algoritma untuk melakukan aktivasi dijelaskan pada Gambar 4.17 .

Nama Algoritma: Aktivasi

Deklarasi :

- String → Nama komputer

Deskripsi :

- Masukan : Nama komputer.
- Proses
    1. Memasukkan variabel masukan.
    2. Variabel masukan yang telah dimasukkan akan disimpan.
    3. Masukan nama komputer yang dimasukkan akan dijalankan bersamaan perintah terminal untuk mengaktifkan server master atau slave.

Keluaran :-

Gambar 4.17 Implementasi algoritma untuk proses instalasi

Sumber : Perancangan dan implementasi

4.4.3 Implementasi Algoritma Proses Submit Job

Proses instalasi berfungsi untuk melakukan *submit job* pada komputer *slave* dan ditujukan pada komputer master. *Submit job* dapat dilakukan di semua komputer *slave*. Implementasi algoritma untuk melakukan *submit job* dijelaskan pada Gambar 4.18 .

Nama Algoritma: Submit Job

Deklarasi :

- String → Nama komputer
- String → Nama file
- String → Lokasi file

Deskripsi :

- Masukan : Nama komputer, nama file, lokasi file
- Proses
    1. Memasukkan variable masukan
    2. Variabel masukan akan digabungkan ke dalam baris perintah terminal
    3. Perintah terminal akan menjalankan perintah file.sh untuk melakukan submit job.

Keluaran :-

Gambar 4.18 Implementasi algoritma untuk proses submit job

Sumber : Perancangan dan implementasi

4.4.4 Implementasi Algoritma *Remote Server*

Proses instalasi berfungsi untuk remote jarak jauh terhadap komputer *master* ataupun *slave*. Proses *remote* dapat dilakukan jika komputer yang digunakan dan komputer yang dituju telah terhubung dan saling terkoneksi Implementasi algoritma untuk melakukan remote server dijelaskan pada Gambar 4.19 .

Nama Algoritma: Remote Server

Deklarasi :

- String → IP komputer
- String → Penggunaname komputer
- String → Password komputer
- String → Perintah

Deskripsi :

- Masukan : Nama komputer.
- Proses
    1. Masukkan variable masukan
    2. Melakukan koneksi terhadap komputer yang dituju.
    3. Koneksi berhasil, perintah akan masuk ke komputer yang dituju.
- Keluaran :-

Gambar 4.19 Implementasi algoritma untuk proses remote server

Sumber : Perancangan dan implementasi

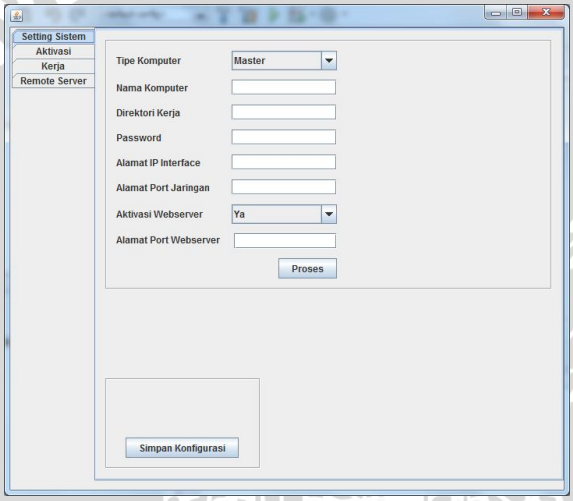
4.5 Implementasi Antarmuka

Implementasi antarmuka aplikasi *render farm* terdiri dari 4 bagian, yaitu implementasi antarmuka aplikasi untuk instalasi, aktivasi server, *submit job* dan *remote server*. Implementasi antarmuka aplikasi administrator dilakukan dengan *komponen graphical pengguna interface Java Swing*.

4.5.1 Implementasi Antarmuka Proses Instalasi

Antarmuka proses instalasi merupakan halaman yang digunakan untuk memasukkan masukan dibutuhkan dalam proses instalasi sistem *render farm*. Antarmuka proses instalasi ini menggunakan komponen *graphical* pengguna *interface* Java Swing.

Di dalam tampilan antar muka halaman ini berisi *textfield* dan *combo box* yang digunakan sebagai tempat untuk memasukkan masukan. Implementasi antarmuka proses instalasi ditunjukkan pada gambar 4.20.



Gambar 4.20 Gambar tampilan antarmuka proses instalasi

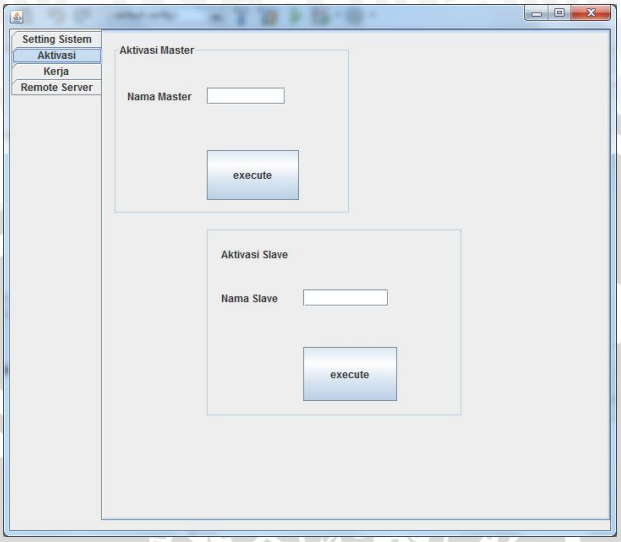
Sumber : Perancangan dan implementasi

4.5.2 Implementasi Antarmuka Proses Aktivasi

Antarmuka proses aktivasi digunakan untuk memasukkan masukan yang diperlukan ketika pengguna akan melakukan aktivasi terhadap komputer *master* dan *slave.* Antarmuka pengguna aplikasi instalasi ini menggunakan komponen *graphical pengguna interface Java Swing*.

Di dalam tampilan antar muka halaman ini, terdapat *text field* untuk mengisi nama komputer yang akan diaktivasi. Implementasi antarmuka

proses aktivasi ditunjukkan pada gambar 4.21.



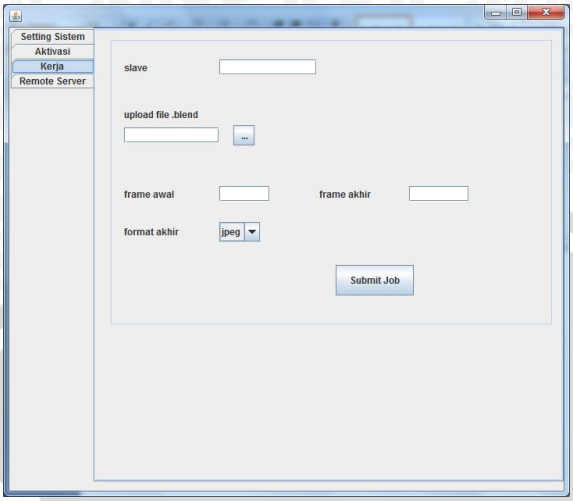
Gambar 4.21 Gambar tampilan antarmuka proses aktivasi

Sumber : Perancangan dan implementasi

4.5.3 Implementasi Antarmuka Proses Submit Job

Antarmuka proses *submit job* merupakan halaman yang dapat digunakan untuk memasukkan masukan yang diperlukan ketika pengguna melakukan proses pengiriman data untuk permintaan pengerjaan *render* terhadap komputer master dari komputer *server*. Antarmuka pengguna aplikasi instalasi ini diimplementasikan menggunakan komponen *graphical pengguna interface Java Swing*.

Di dalam tampilan antar muka halaman ini, terdapat *text field* untuk mengisi nama komputer *slave* tempat pengiriman permintaan render, memasukkan *frame* gambar, *upload file blend* dan format gambar. Implementasi antarmuka proses instalasi ditunjukkan pada gambar 4.22.

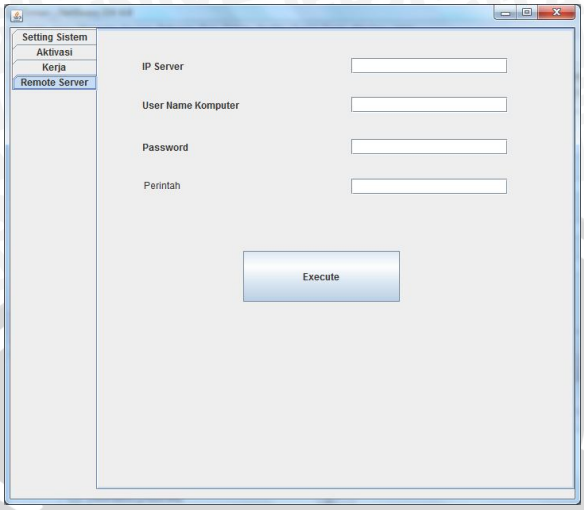


Gambar 4.22 Gambar tampilan antarmuka proses submit job

Sumber : Perancangan dan implementasi

4.5.4 Implementasi Antarmuka Remote Server

Antarmuka pengguna proses *remote server* merupakan halaman yang digunakan untuk memasukkan masukan yang diperlukan ketika pengguna akan melakukan proses *remote* terhadap komputer *master* atau *slave*. Antarmuka pengguna aplikasi instalasi ini diimplementasikan menggunakan komponen *graphical pengguna interface Java Swing*. Implementasi *remote server* ditunjukkan pada gambar 4.23.



Gambar 4.23 Gambar tampilan antarmuka proses remote server

Sumber : Perancangan dan implementasi

BAB V
PENGUJIAN DAN ANALISIS

Secara umum, pengujian dilakukan untuk dapat menunjukkan bahwa sistem telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang dibutuhkan oleh sistem dan pengguna. Pengujian pada sebuah sistem adalah untuk melihat keberhasilan sistem menjalankan fungsionalitasnya dan mengetahui tingkat reliabilitas dari sistem ketika memproses masukan. Pengujian sistem *render farm* ini terdari dari dua pengujian yaitu, pengujian terhadap kinerja komputasi. Pengujian dilakukan pada saat adanya proses render pada sistem dengan melakukan pengamatan.

Untuk pengujian kali ini , terdapat 6 file blend yang akan di render yaitu :

1. Arizona.blend yang memiliki 60 frame di dalamnya.
2. Beamsattelite.blend yang memiliki 250 frame di dalamnya.
3. Paddywagon.blend yang memiliki 148 frame di dalamnya.
4. Policelifter.blend yang memiliki 250 frame di dalamnya.
5. Redexosuit.blend yang memiliki 65 frame di dalamnya.
6. Tankbot.blend yang memiliki 200 frame di dalamnya.

5.1 Skenario Pengujian

Untuk pengujian kinerja komputasi pengujian dilakukan dengan melakukan proses render dengan menggunakan sistem *render farm* dan juga hanya menggunakan *single processing*. Skenario ini dibutuhkan untuk mendapatkan data kinerja yang nantinya dapat data tersebut dapat diolah kembali menggunakan rumus komputasi yang ada sehingga data yang ada dapat memunculkan kinerja komputasi yang terjadi pada sistem *render farm*.

Skenario render yang dirancang dalam pengujian terdiri atas 4

skenario , yaitu :

1. Skenario 1

   Skenario 1 akan melakukan proses render 1 file *blender* per proses yang akan di render pada *single processing* dan pada *render farm*

2. Skenario 2

   Skenario 2 akan melakukan proses render 2 file blender per proses yang akan di render pada *single processing* dan pada *render farm*.

3. Skenario 3

   Skenario 3 akan melakukan proses render 3 file blender per proses yang akan di render pada *single processing* dan pada *render farm*.

4. Skenario 4

   Skenario 4 akan melakukan proses render 6 file blender per proses yang akan di render pada *single processing* dan pada *render farm*.

Dari hasil render dengan skenario tersebut dapat diolah lebih lanjut untuk selanjutnya dapat dimasukkan ke dalam rumus perhitungan analisis kinerja komputasi.

1. Penghitungan MIPS

   Pengujian MIPS (laju eksekusi per unit waktu) dilakukan untuk dapat mengetahui perbandingan kinerja dari tiap proses *render* yang dilakukan.Dapat menggunakan rumus :

$$MIPS = \frac{jumlah\ instruksi}{waktu\ eksekusi\ x\ 10^6} \quad (5\text{-}1)$$

2. Penghitungan perbandingan pemrosesan tunggal dan parallel

   Perbandingan tersebut dapat dilakukan membandingkan hasil kerja sistem prosesor tunggal dibandingkan dengan multiprosesor. Dapat menggunakan rumus :

$$S\,(p) = \frac{\text{waktu proses menggunakan n sistem prosesor tunggal}}{\text{waktu proses menggunakan multi prosesor dengan p prosesor}} \quad (5\text{-}2)$$

3. Perhitungan efisiensi

Efisiensi,kadangkala sangatlah berguna untuk mengetahui jumlah waktu yang diperlukan oleh prosesor untuk melakukan komputasi. Dapat menggunakan rumus :

$$E = \frac{\text{S (p)}}{p} \text{ x } 100\,\% \quad (5\text{-}3)$$

5.2 Hasil Perhitungan Skenario 1

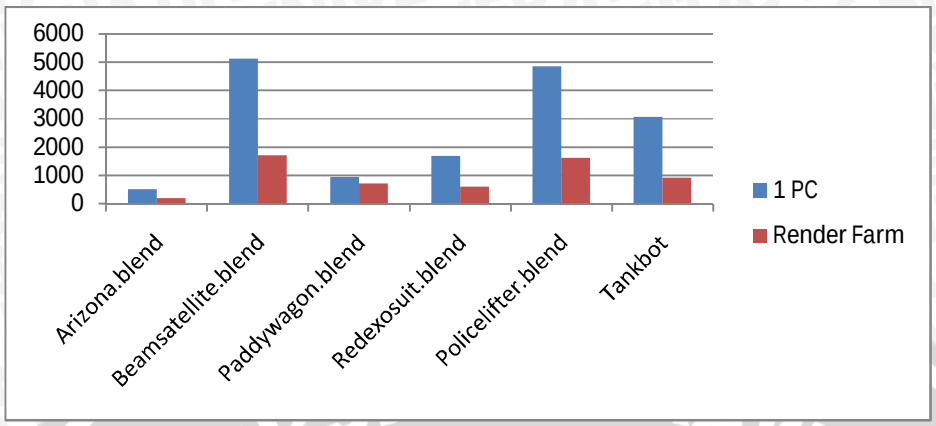
Perbandingan 1 file per proses yang dikerjakan pada *single processing* dan *render farm* dilakukan dengan melakukan melakukan proses *render* sehingga diketahui waktu berjalannya proses dimasing-masing sistem. Hasil perhitungan proses render ditunjukkan pada tabel 5.1.

Tabel 5.1 Tabel pengujian hasil render skenario 1 dengan single processing dan render farm

Nama File	Jumlah Frame	Waktu (detik) single processing	Waktu (detik) render farm
Arizona.blend	60 frame	495	194
Beamsattelite.blend	250 frame	5097	1695
Paddywagon.blend	148 frame	935	707
Redexosuit.blend	65 frame	1691	600
Police Lifter	250 frame	4848	1610
Tankbot.blend	200 frame	3055	915

Sumber : Pengujian dan analisis

Grafik perbandingan render skenario 1



Grafik 5.1 Grafik pengujian hasil render pada skenario 1

Sumber : Pengujian dan analisis

Setelah proses *render* selesai dilakukan dan mendapatkan hasil waktu dari proses *render* yang dijalankan, waktu *render* yang dihasilkan dapat dimasukkan ke dalam rumus-rumus-rumus perhitungan analisis kinerja komputasi.

Pada perhitungan MIPS menggunakan rumus

$$MIPS = \frac{jumlah\ instruksi}{waktu\ eksekusi\ x\ 10^6} \quad (5\text{-}4)$$

Sebagai contoh pada file arizona.blend memiliki waktu *render* sebesar 495 detik, jumlah instruksi tidak diketahui secara pasti, sehingga menggunakan n dalam persamaannya, apabila kedua nilai yang didapatkan ke dalam rumus MIPS maka persamaan menjadi

$$MIPS = \frac{n}{495\ x\ 10^6}$$

Selanjutnya pada perbandingan proses dengan *single processing* dengan *render farm* menggunakan rumus

$$S\ (p) = \frac{\text{waktu proses menggunakan n sistem prosesor tunggal}}{\text{waktu proses menggunakan multi prosesor dengan p prosesor}} \quad (5\text{-}5)$$

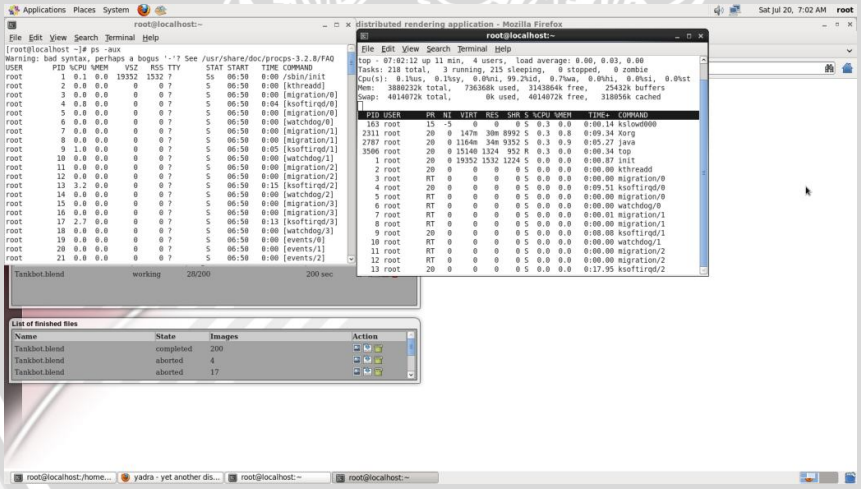
Pada file tankbot.blend memiliki waktu proses pada *single processing* sebesar 3055 detik dan render farm sebesar 915 detik, jika dimasukkan ke dalam rumus persamaan menjadi

$$S\,(p) = \frac{3055}{915} = 3.33$$

Selanjutnya pada perhitungan efisiensi menggunakan rumus

$$E = \frac{\mathrm{S\,(p)}}{p} \times 100\,\% \qquad (5\text{-}6)$$

Pada file tankbot.blend memiliki sp = 3,33, jumlah proses dihitung berdasarkan nilai yang dikeluarkan melalui perangkat bantuan dari perintah *linux* yaitu perintah *top*. Dengan menggunakan perintah *top*, dapat dilihat berapa jumlah proses yang berjalan ketika sistem render farm ini berjalan. Proses yang berjalan ditunjukkan dalam gambar 5.1

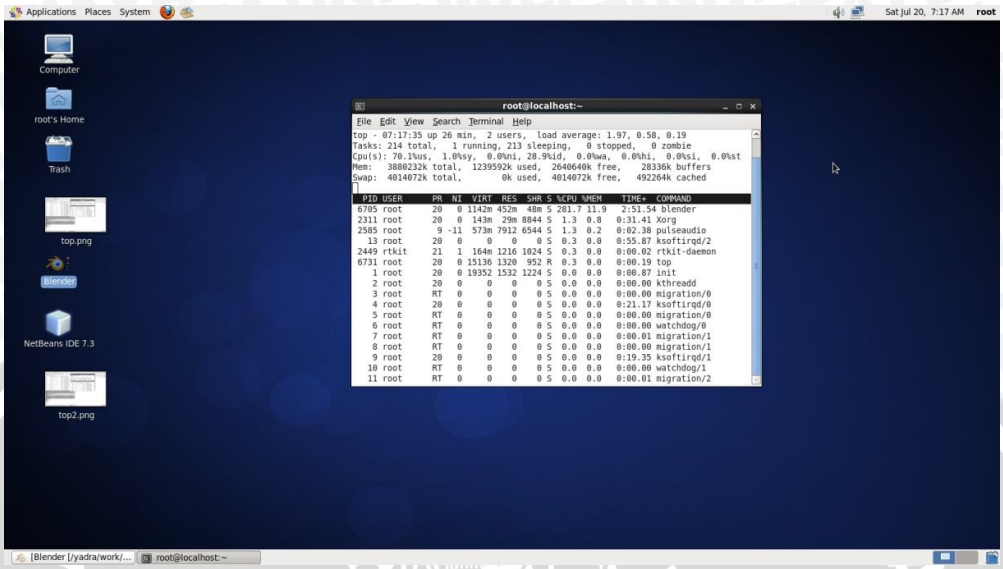


Gambar 5.1 Proses paralel yang berjalan dalam perintah top

Sumber : Pengujian dan analisis

Dalam perintah *top*, ditunjukkan bahwa proses yang berjalan ketika proses *render* melalui sistem *render farm* ada 3 proses yang berjalan ketika proses *render farm* berjalan. Dapat dijadikan perbandingan, ketika proses *render* yang dilakukan oleh 1 buah komputer jumlah proses yang berjalan

hanya 1 buah. Ditunjukkan pada gambar 5.2



Gambar 5.2 Proses single yang berjalan dalam perintah top

Sumber : Pengujian dan analisis

Jumlah proses yang berjalan pada sistem *render farm* tidak terpengaruh dengan jumlah file .blend yang dirender secara bersamaan. Jumlah proses yang berjalan tetap dan tidak berubah. Sehingga jika dimasukkan ke dalam rumus efisiensi menjadi:

$$E = \frac{3{,}33}{3} x100\% = 1{,}11$$

Tabel 5.2 Tabel pengujian MIPS

Nama File	MIPS single processing	MIPS pada render farm	Perbandingan MIPS render farm dan single processing
Arizona.blend	$\frac{n}{495\ x\ 10^6}$	$\frac{n}{194\ x\ 10^6}$	0.39
Beamsatellite.blend	$\frac{n}{5097\ x\ 10^6}$	$\frac{n}{1695x\ 10^6}$	0.33

Paddywagon.blend	$\frac{n}{935\ x\ 10^6}$	$\frac{n}{707\ x\ 10^6}$	0.76
Redexosuit.blend	$\frac{n}{1691\ x\ 10^6}$	$\frac{n}{600\ x\ 10^6}$	0.35
Policelifter.blend	$\frac{n}{4848x\ 10^6}$	$\frac{n}{1610\ x\ 10^6}$	0.33
Tankbot.blend	$\frac{n}{3055\ x\ 10^6}$	$\frac{n}{915\ x\ 10^6}$	0.30

Sumber : Pengujian dan analisis

Tabel 5.3 Tabel hasil perhitungan pada skenario 1

Nama File	Perbandingan Proses tunggal dan parallel	Efisiensi render farm
Tankbot	3.33	1.11%
Arizona	2.55	0.85%
Paddywagon	1.32	0.44%
Policelifter	3.01	1%
Beamsatellite	3	1%
Redexosuit	2.81	0.93%

Sumber : Pengujian dan analisis

5.3 Hasil Perhitungan Skenario 2

Pada pengujian skenario 2, data yang diuji dikelompokkan menjadi masing-masing 2 file yang akan diuji bersama-sama. Pembagian kelompok pengujian pada skenario 2 dibagi menjadi beberapa tahap pengujian yaitu:

1. Tahap 1 terdiri atas tankbot.blend dan redexosuit.blend
2. Tahap 2 terdiri atas policelifter.blend dan paddywagon.blend
3. Tahap 3 terdiri atas Arizona.blend dan beamsatellite.blend

Perbandingan 2 file per proses render yang dikerjakan single processing dan render farm ditunjukkan pada tabel 5.3, tabel 5.4 dan tabel 5.5

Tabel 5.4 Tabel pengujian hasil render skenario 2 tahap 1

Nama File	Jumlah Frame	Waktu (detik) single processing	Waktu (detik) render farm
Tankbot.blend	200 frame	4557	1639
Redexosuit.blend	65 frame	2735	972

Sumber : Pengujian dan analisis

Tabel 5.5 Tabel pengujian hasil render skenario 2 tahap 2

Nama File	Jumlah Frame	Waktu (detik) single processing	Waktu (detik) render farm
Policelifter.blend	60 frame	5212	2298
Paddywagon.blend	250 frame	2177	1655

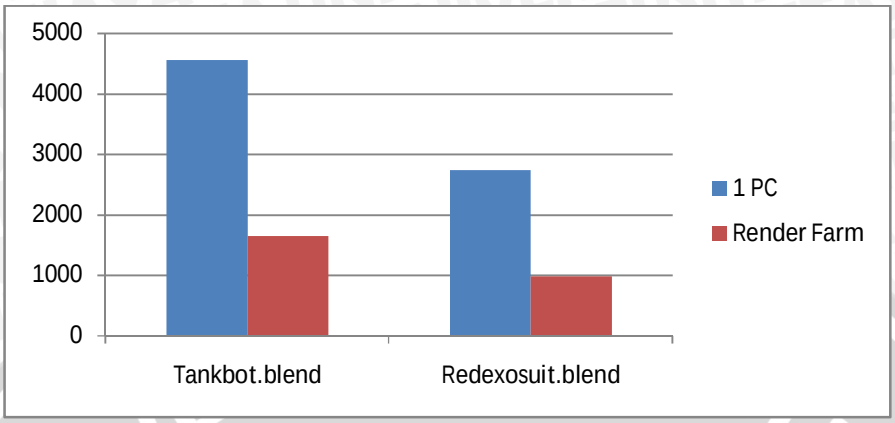
Sumber : Pengujian dan analisis

Tabel 5.6 Tabel pengujian hasil render skenario 2 tahap 3

Nama File	Jumlah Frame	Waktu (detik) single processing	Waktu (detik) render farm
Arizona.blend	60 frame	809	400
Beamsatellite.blend	250 frame	5792	1871

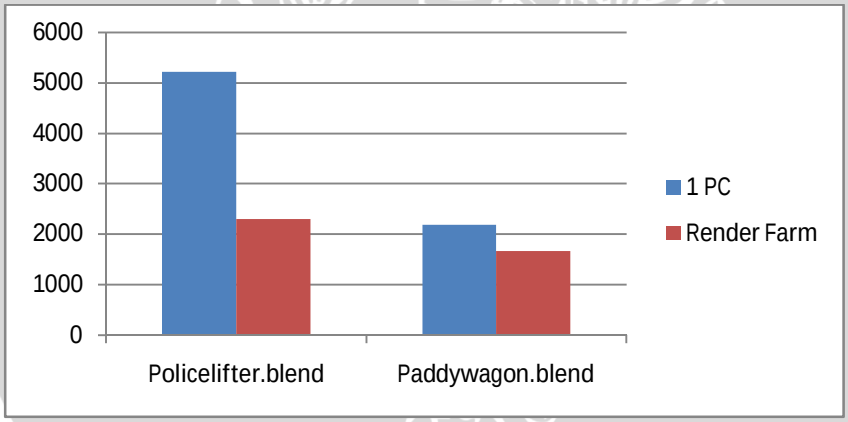
Sumber : Pengujian dan analisis

Grafik perbandingan render skenario 2



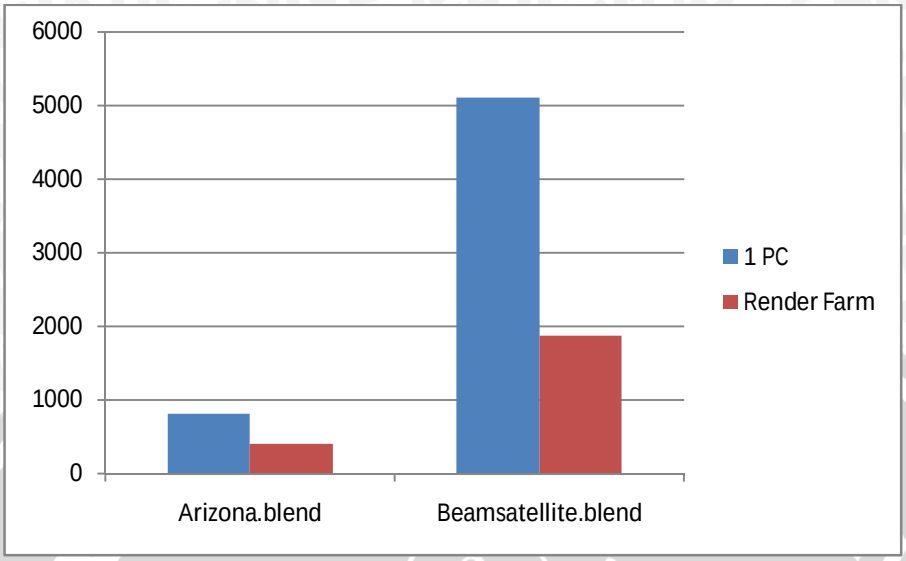
Grafik 5.2 Grafik pengujian hasil render skenario 2 tahap 1

Sumber : Pengujian dan analisis



Grafik 5.3 Grafik pengujian hasil render skenario 2 tahap 2

Sumber : Pengujian dan analisis



Grafik 5.4 Grafik pengujian hasil render skenario 2 tahap 3

Sumber : Pengujian dan analisis

Penghitungan dari hasil pengujian skenario 2

Nama File	MIPS 1 PC	MIPS pada render farm	Perbandingan MIPS
tankbot.blend	$\frac{n}{4557\ x\ 10^6}$	$\frac{n}{1639\ x\ 10^6}$	0.36
Redexosuit.blend	$\frac{n}{2735\ x\ 10^6}$	$\frac{n}{972x\ 10^6}$	0.36
Policelifter.blend	$\frac{n}{5212\ x\ 10^6}$	$\frac{n}{2298\ x\ 10^6}$	0.44
Paddywagon.blend	$\frac{n}{2177\ x\ 10^6}$	$\frac{n}{1655\ x\ 10^6}$	0.76
Arizona.blend	$\frac{n}{809x\ 10^6}$	$\frac{n}{400\ x\ 10^6}$	0.49

Beamsatellite.blend	$\frac{n}{5792\ x\ 10^6}$	$\frac{n}{1871\ x\ 10^6}$	0.32

Tabel 5.7 Tabel pengujian MIPS

Sumber : Pengujian dan analisis

Tabel 5.8 Tabel hasil perhitungan pada skenario 2

Nama File	Perbandingan proses tunggal dan parallel	Efisiensi render farm
Tankbot	2.78	0.92%
Arizona	2.02	0.67%
Paddywagon	1.31	0.43%
Policelifter	2.26	0.75%
Beamsatellite	3.09	1.03%
Redexosuit	2.81	0.93%

Sumber : Pengujian dan analisis

5.4 Hasil Perhitungan Skenario 3

Pada pengujian skenario 3, data yang diuji dikelompokkan menjadi masing-masing 3 file yang akan diuji bersama-sama. Pembagian kelompok pengujian pada skenario 3 dibagi menjadi beberapa tahap pengujian yaitu:

1. Tahap 1 terdiri atas tankbot.blend, redexosuit.blend dan police lifter.blend
2. Tahap 2 terdiri atas paddywagon.blend, arizona.blend dan beamsatellite.blend

Perbandingan 3 file per proses render yang dikerjakan single processing dan render farm ditunjukkan pada tabel 5.9 dan tabel 5.10.

Tabel 5.9 Tabel pengujian hasil render skenario 3 dengan

Nama File	Jumlah Frame	Waktu (detik) single processing	Waktu (detik) render farm
Tankbot.blend	200 frame	6332	2884
Redexosuit.blend	65 frame	4285	1386
Policelifter.blend	250 frame	6939	3217

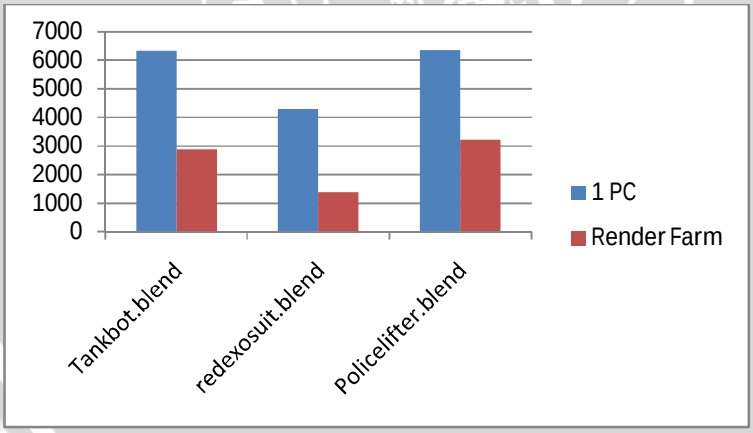
Sumber : Pengujian dan analisis

Tabel 5.10 Tabel pengujian hasil render skenario 3 dengan tahap 2

Nama File	Jumlah Frame	Waktu (detik) single processing	Waktu (detik) render farm
Paddywagon.blend	148 frame	2489	1860
Arizona.blend	60 frame	1146	900
Beamsatellite.blend	250 frame	5872	2200

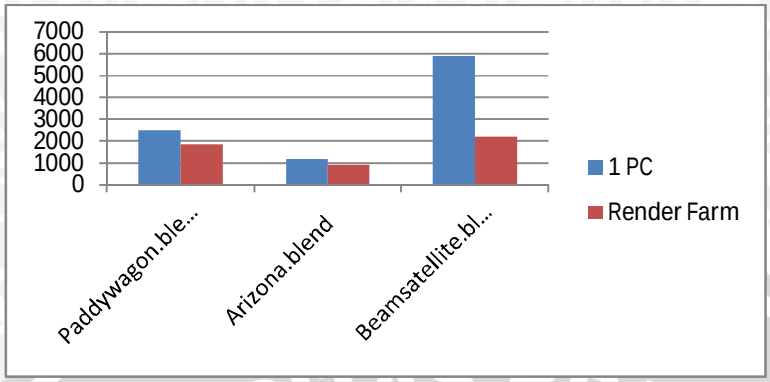
Sumber : Pengujian dan analisis

Grafik perbandingan render skenario 3



Grafik 5.5 Grafik pengujian hasil render skenario 3 tahap 1

Sumber : Pengujian dan analisis



Grafik 5.6 Grafik pengujian hasil render skenario 3 processing tahap 2

Sumber : Pengujian dan analisis

Penghitungan dari hasil pengujian skenario 3

Tabel 5.11 Tabel pengujian MIPS

Nama File	MIPS 1 PC	MIPS pada render farm	Perbandingan MIPS
tankbot.blend	$\frac{6332}{10^6}$	$\frac{2884}{10^6}$	0.46
Redexosuit.blend	$\frac{4285}{10^6}$	$\frac{1386}{10^6}$	0.32
Policelifter.blend	$\frac{6939}{10^6}$	$\frac{3217}{10^6}$	0.46
Paddywagon.blend	$\frac{2489}{10^6}$	$\frac{1860}{10^6}$	0.75
Arizona.blend	$\frac{1146}{10^6}$	$\frac{900}{10^6}$	0.79
Beamsatellite.blend	$\frac{5872}{10^6}$	$\frac{2200}{10^6}$	0.37

Sumber : Pengujian dan analisis

Tabel 5.12 Tabel hasil perhitungan pada skenario 3

Nama File	Perbandingan Tunggal dan Paralal	Efisiensi
Tankbot.	2.19	0.73%
Arizona.	1.27	0.42%
Paddywagon	1.33	0.45%
Policelifter	2.15	0.72%
Beamsatellite	2.66	0.89%
Redexosuit	3.091	1.03%

Sumber : Pengujian dan analisis

5.5 Hasil Perhitungan Skenario 4

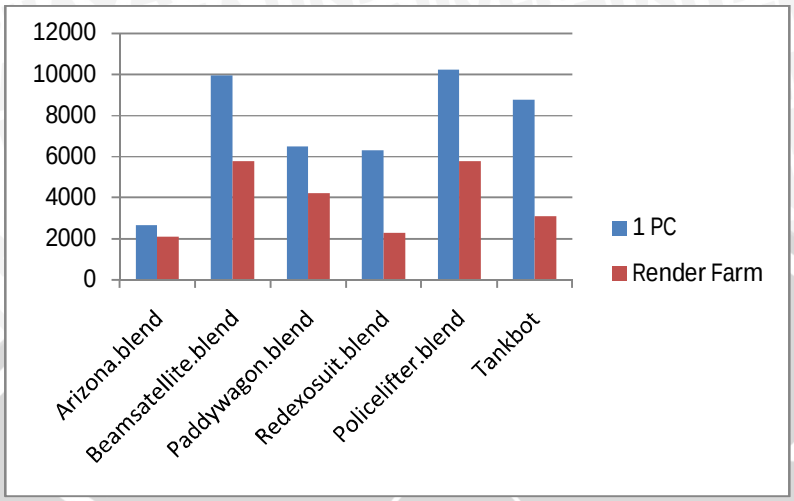
Perbandingan 1 file per proses yang dikerjakan pada *single processing* dan *render farm* dilakukan dengan melakukan melakukan proses *render* sehingga diketahui waktu berjalannya proses dimasing-masing sistem. Hasil perhitungan proses *render* ditunjukkan pada tabel 5.10.

Tabel 5.13 Tabel pengujian hasil render skenario 4 dengan single processing dan render farm

Nama File	Jumlah Frame	Waktu (detik) single processing	Waktu (detik) render farm
Arizona.blend	60 frame	2675	2103
Beamsattelite.blend	250 frame	9930	5792
Paddywagon.blend	148 frame	6492	4220
Redexosuit.blend	65 frame	6320	2310
Policelifter	250 frame	10.220	5772
Tankbot.blend	200 frame	8750	3111

Sumber : Pengujian dan analisis

Grafik perbandingan render skenario 4



Grafik 5.7 Grafik pengujian hasil render skenario 4

Sumber : Pengujian dan analisis

Penghitungan dari hasil pengujian skenario 4

Tabel 5.14 Tabel pengujian MIPS

Nama File	MIPS 1 PC	MIPS pada render farm	Perbandingan MIPS
Arizona.blend	$\frac{2675}{10^6}$	$\frac{2103}{10^6}$	0.79
Beamsatellite.blend	$\frac{9930}{10^6}$	$\frac{5792}{10^6}$	0.58
Paddywagon.blend	$\frac{6492}{10^6}$	$\frac{4220}{10^6}$	0.65
Redexosuit.blend	$\frac{6320}{10^6}$	$\frac{2310}{10^6}$	0.37
Policelifter.blend	$\frac{10220}{10^6}$	$\frac{5772}{10^6}$	0.58

Tankbot.blend	$\overline{8750 \quad 10^6}$	$\overline{3111 \quad 10^6}$	0.36

Sumber : Pengujian dan analisis

Tabel 5.15 Tabel hasil perhitungan pada skenario 4

Nama File	Perbandingan proses tunggal dan paralel	Efisiensi
Tankbot	2.81	0.93%
Arizona	1.27	0.42%
Paddywagon	1.53	0.51%
Policelifter	1.77	0.59%
Beamsatellite	1.71	0.57%
Redexosuit	2.73	0.91%

Sumber : Pengujian dan analisis

5.6 Analisis Pengujian

Berdasarkan dari hasil perhitungan komputasi ataupun terhadap hasil perhitungan jaringan di atas dapat dilakukan analisis terhadap sistem *render farm* yang telah diimplementasikan.

5.6.1 Performa Sistem

Dari performa kemampuan proses render dapat dilihat , proses render yang menggunakan sisten *render farm* memiliki tingkat kecepatan untuk menyelesaikan proses *render* yang lebih cepat dibandingkan menggunakan *single processing.*

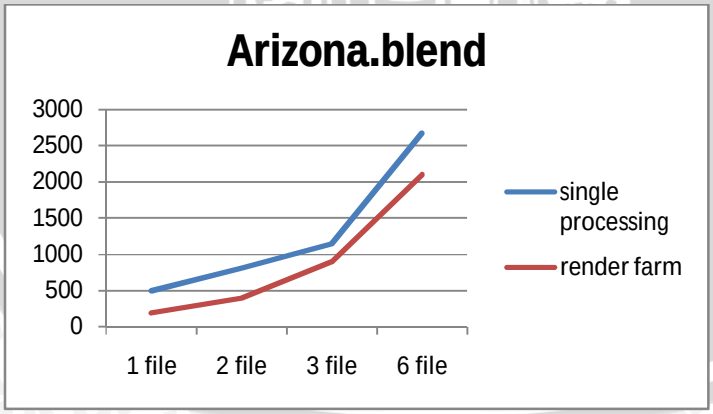
Kecepatan render baik menggunakan *single processing* ataupun menggunakan *render farm* akan semakin menurun seiring semakin banyaknya jumlah file yang diproses. Untuk hasil *render* sendiri tidak memiliki perbedaan kualitas baik menggunakan *single processing* ataupun sistem *render farm.* Sehingga dapat di analisis jika perbedaan dari sistem

single processing dan sistem *render farm* terdapat pada kecepatan waktu dari proses render gambar.

Sistem *render farm* bekerja dengan melakukan pembagian *frame* kepada komputer *slave* yang bekerja untuk merender gambar tersebut.. Sistem ini akan membagi *frame* yang akan dirender ke dalam antrian yang tidak memiliki prioritas di dalamnya, ketika proses habis maka proses akan dialihkan ke dalam proses selanjutnya. Kelemahan sistem ini terkadang kualitas tiap-tiap *frame* memiliki perbedaan. Dengan tidak adanya prioritas, *frame* yang lebih ringan dalam proses rendernya bisa saja diletakkan di proses terakhir dan lebih didahulukan *frame* yang lebih berat pengerjaannya.

Selain itu sistem ini akan berjalan efektif jika melakukan proses *render* pada file animasi yang memiliki jumlah frame lebih dari 1. Karena jumlah *frame* yang banyak tersebut akan dikerjakan secara bersamaan dengan membagi *frame* kepada komputer slave yang bekerja. Namun, untuk file animasi yang hanya memiliki jumlah frame hanya 1 buah, sistem ini akan menjadi tidak efisien. Karena hanya mengerjakan 1 frame yang waktu pengerjaannya sama jika dikerjakan jika menggunakan *single processing*.

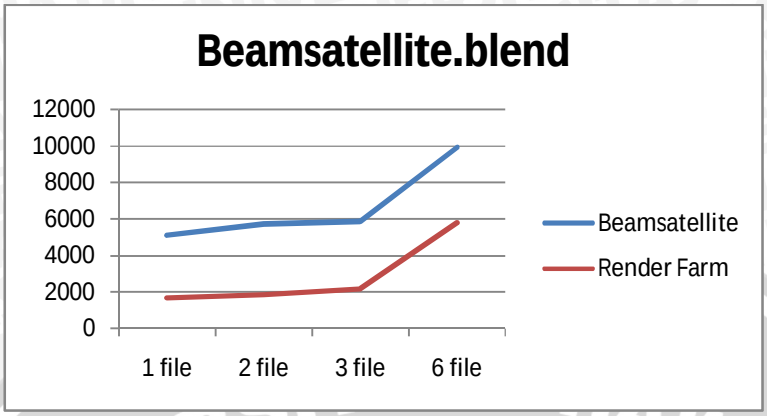
Grafik Render proses pada file Arizona.blend dengan *single processing* dan render farm ditunjukkan pada grafik 5.8



Grafik 5.8 Grafik pengujian hasil render Arizona.blend

Sumber : Pengujian dan analisis

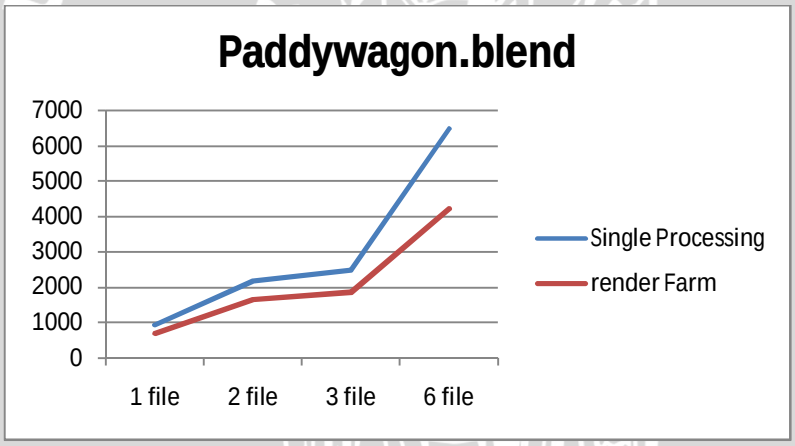
Grafik Render proses pada file beamsatellite.blend dengan *single processing* dan *render farm* ditunjukkan pada grafik 5.9



Grafik 5.9 Grafik pengujian hasil render beamsatellite.blend

Sumber : Pengujian dan analisis

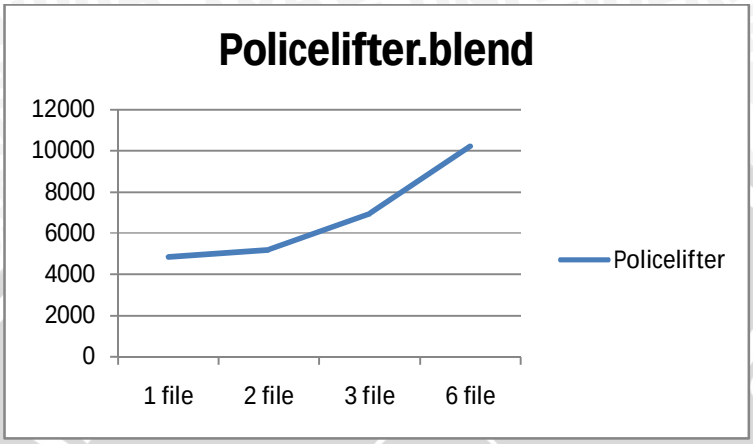
Grafik Render proses pada file paddywagon.blend dengan single *processing* dan *render farm* ditunjukkan pada grafik 5.10



Grafik 5.10 Grafik pengujian hasil render paddywagon.blend

Sumber : Pengujian dan analisis

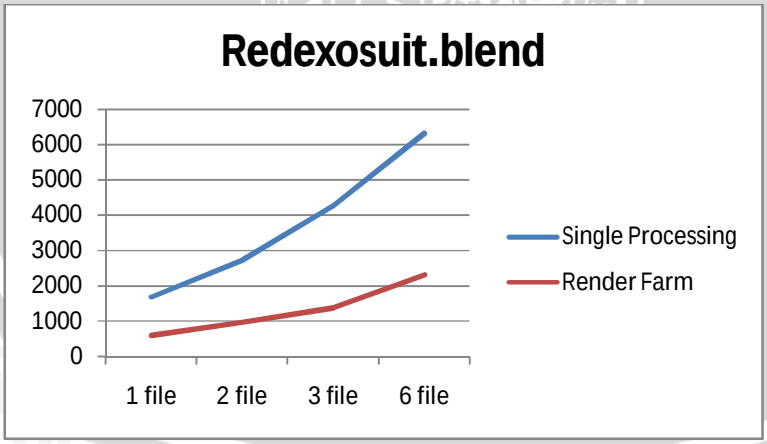
Grafik Render proses pada file policelifter.blend dengan *single processing* dan *render farm* ditunjukkan pada grafik 5.11



Grafik 5.11 Grafik pengujian hasil render policelifter.blend pada single processing

Sumber : Pengujian dan analisis

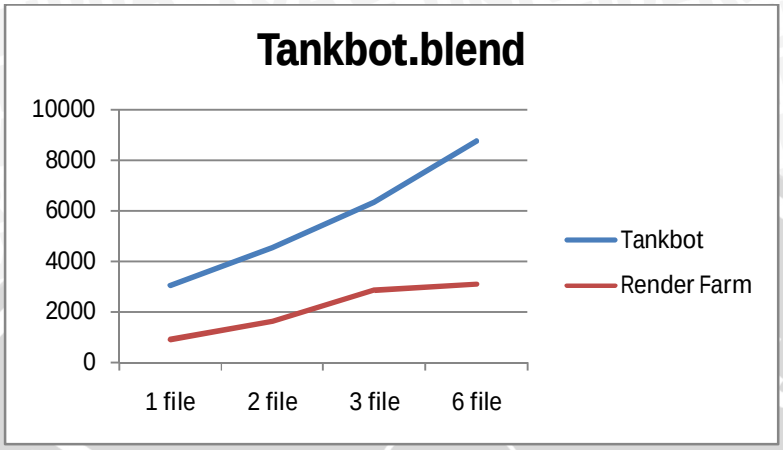
Grafik Render proses pada file redexosuit.blend dengan *single processing* dan *render farm* ditunjukkan pada grafik 5.12



Grafik 5.12 Grafik pengujian hasil render redexosuit.blend pada single processing

Sumber : Pengujian dan analisis

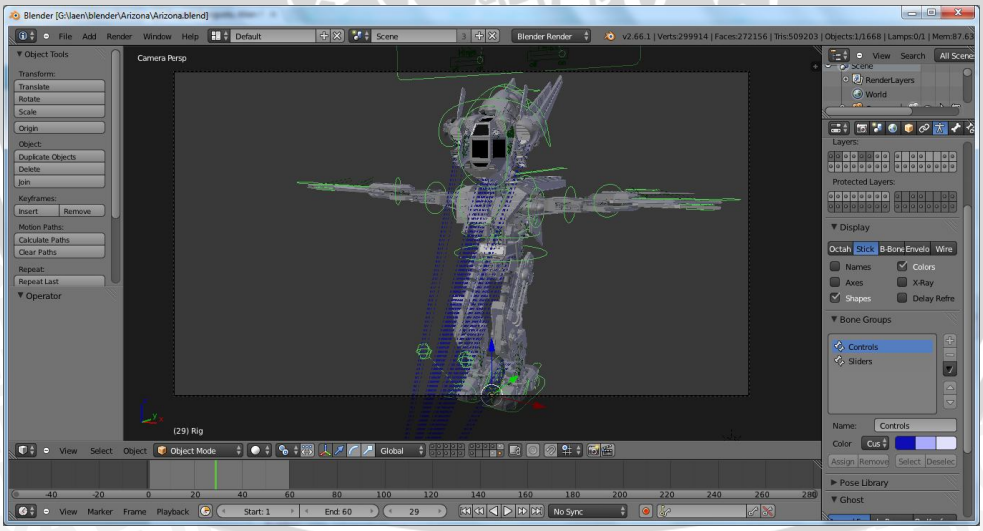
Grafik Render proses pada file redexosuit.blend dengan *single processing* dan *render farm* ditunjukkan pada grafik 5.13



Grafik 5.13 Grafik pengujian hasil render tankbot.blend pada single processing

Sumber : Pengujian dan analisis

- Hasil render gambar

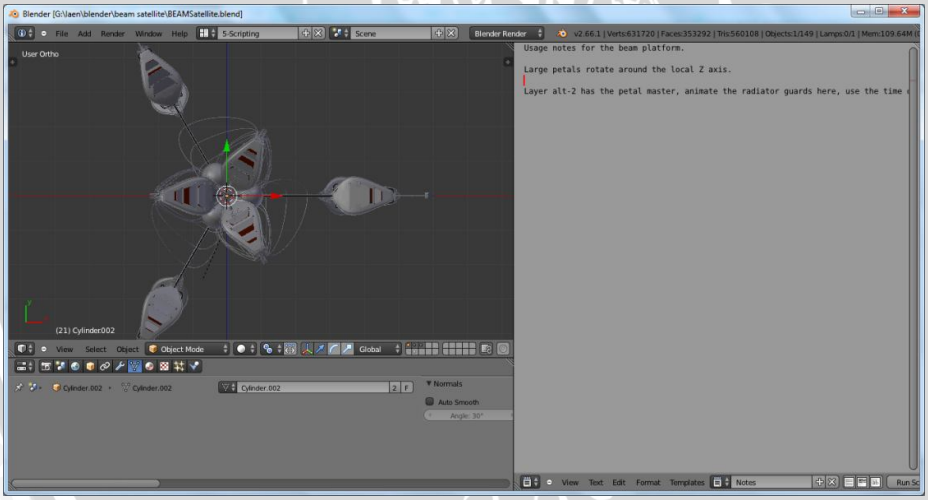


Gambar 5.3 Gambar Arizona.blend pada blender

Sumber : Pengujian dan analisis

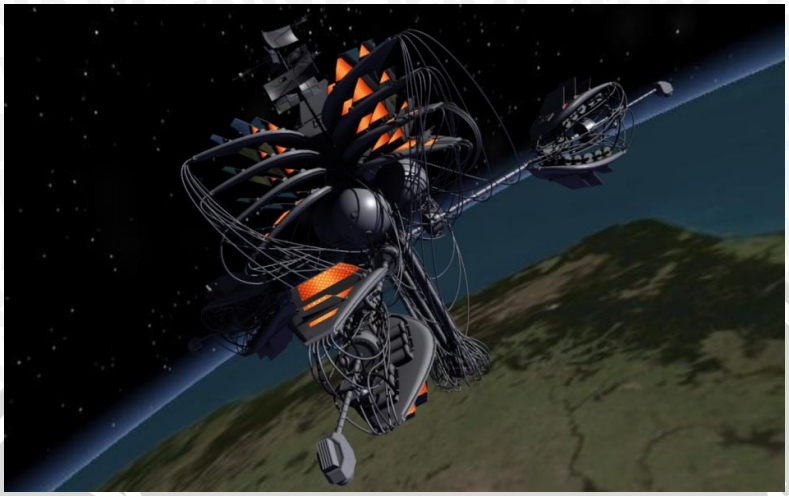
Gambar 5.4 Hasil render Arizona.blend

Sumber : Pengujian dan analisis



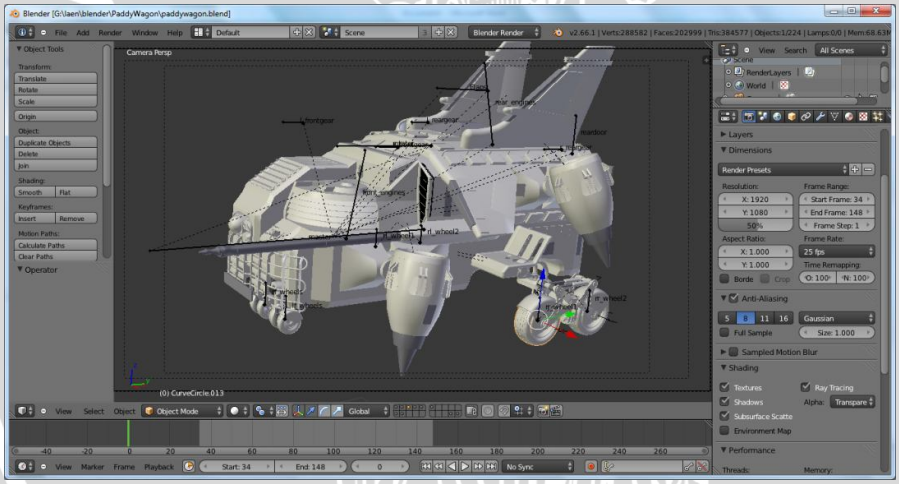
Gambar 5.5 Gambar Beamsatellite.blend pada blender

Sumber : Pengujian dan analisis



Gambar 5.6 Hasil render beamsatellite

Sumber : Pengujian dan analisis

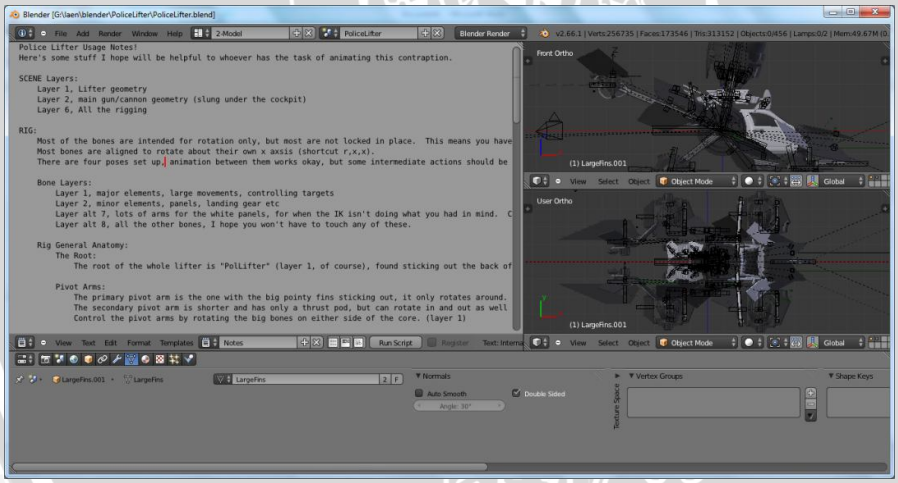


Gambar 5.7 Gambar paddywagon.blend pada blender

Sumber : Pengujian dan analisis

Gambar 5.8 Hasil render paddywagon.blend

Sumber : Pengujian dan analisis

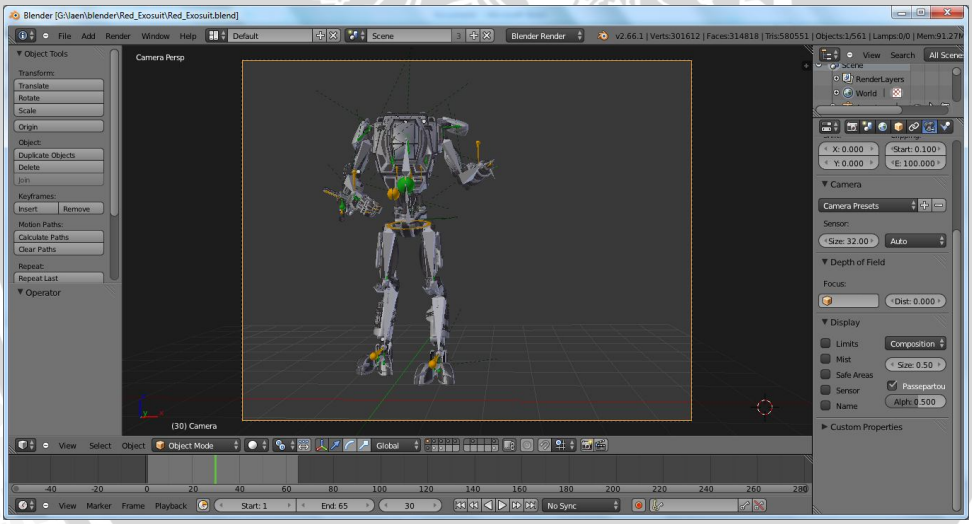


Gambar 5.9 Gambar police lifter pada blender

Sumber : Pengujian dan analisis

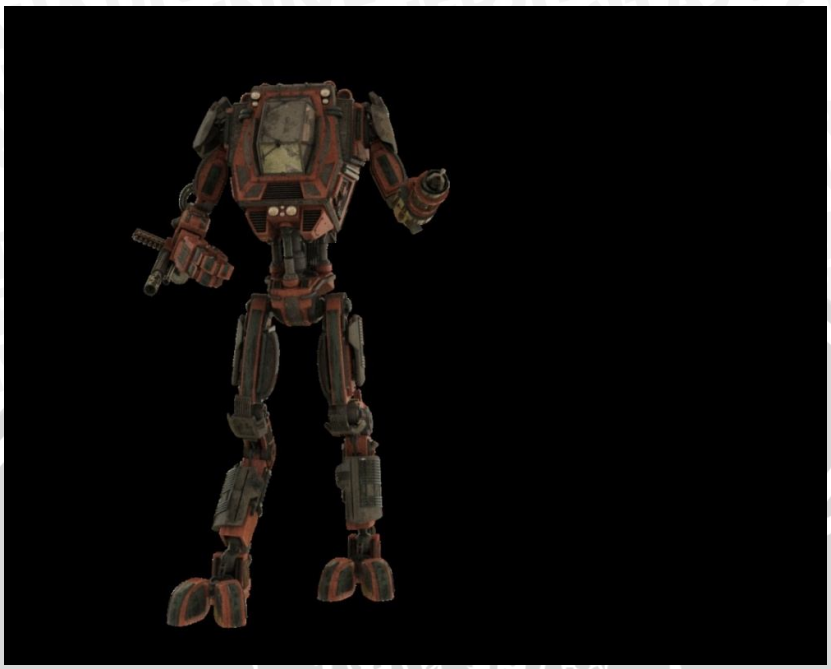
Gambar 5.10 Hasil render police lifter.blend

Sumber : Pengujian dan analisis



Gambar 5.11 Gambar redexosuit.blend pada blender

Sumber : Pengujian dan analisis



Gambar 5.12 Hasil render redexosuit.blend

Sumber : Pengujian dan analisis

Gambar 5.13 Gambar tankbot.blend pada blender

Sumber : Pengujian dan analisis

Gambar 5.14 Hasil pengujian tankbot.blend

Sumber : Pengujian dan analisis

5.6.2 Performa Komputasi

Pengujian yang dihasilkan dalam sistem *render farm* menghasilkan data hasil kecepatan *rende*r yang selanjutnya dapat diolah kembali ke dalam rumus MIPS, rumus perbandingan pemrosesan *single* dan paralel serta mendapatkan angka dari perhitungan efisiensi kerja komputasi.

MIPS (*Millions of Instruction executed Per Seconds*) digunakan untuk menunjukan kecepatan sebuah sistem komputer. Karena waktu instruksi yang berjalan berbeda untuk isntruksi yang berbeda , maka nilai MIPS pada setiap proses yang berjalan dapat berbeda. Begitu pula di dalam sistem render farm kali ini . MIPS dari suatu sistem komputer merupakan nilai kasar untuk dapat menilai kinerja dari suatu sistem komputer.

Dalam pengujian sistem kali ini dibuat perbandingan berapa nilai MIPS dari proses render dari kinerja yang berjalan di *single processing* dan paralel komputing. Dapat di ambil contoh pada tabel 5.21 hasil perhitungan MIPS skenario 1

Tabel 5.17 Contoh tabel hasil perhitungan MIPS pada skenario 1

Nama File	MIPS single processing	MIPS pada render farm	Perbandingan MIPS render farm dan single processing
Arizona.blend	$\frac{495 \quad 10^6}{}$	$\frac{194 \quad 10^6}{}$	0.39
Beamsatellite.blend	$\frac{5097 \quad 10^6}{}$	$\frac{1695 \quad 10^6}{}$	0.33
Paddywagon.blend	$\frac{935 \quad 10^6}{}$	$\frac{707 \quad 10^6}{}$	0.76
Redexosuit.blend	$\frac{1691 \quad 10^6}{}$	$\frac{600 \quad 10^6}{}$	0.35
Policelifter.blend	$\frac{4848 \quad 10^6}{}$	$\frac{1610 \quad 10^6}{}$	0.33
Tankbot.blend	$\frac{3055 \quad 10^6}{}$	$\frac{915 \quad 10^6}{}$	0.30

Sumber : Pengujian dan analisis

Dapat dilihat pada data dari tabel 5.29, dibandingkan antara perhitungan dari *single processing* dan *render farm* perbedaan nilai dari hasil perhitungan MIPS yang didapat. Dari hasil pengujian menggunakan file tankbot.blend dilihat dari nilai pada *single processing* dibandingkan dengan nilai MIPS dari *render farm* sebesar. Disimpulkan bahwa nilai MIPS *render farm* lebih besar dibanding pada *single processing*.

Dalam implementasi sistem di atas , dapat dilihat data hasil perhitungan ,terjadi percepatan sebanyak *Sp* dari proses *render farm*, dapat diartikan bahwa ketika proses berjalan di dalam sistem render farm maka kecepatan komputasi dari sistem berjalan melebihi beberapa kali lipat dari proses yang berjalan di dalam sistem s*ingle* prosesor sebagai contoh hasil perhitungan dari skenario pertama dalam tabel 5.22

Tabel 5.18 Contoh Tabel hasil perhitungan perbandingan proses pada skenario 1

Nama File	Hasil Perhitungan
Tankbot.blend	3.33
Arizona.blend	2.55
Paddywagon.blend	1.32
Policelifter.blend	3.01
Beamsatellite.blend	3.00
Redexosuit.blend	2.81

Sumber : Pengujian dan analisis

Dalam perhitungan efisiensi , hasil akhir menggunakan angka dalam bentuk persen, dari sini menandakan ketika proses yang berjalan di dalam sistem *render farm* hanya menggunakan sekian persen dari waktu yang dihasilkan dari proses rata-rata yang dibutuhkan oleh komputasi yang menggunakan *single* prosesor . Sebagai contoh dapat dilihat pada data efisiensi dari skenario 1 pada tabel 5.23.

Tabel 5.19 Tabel hasil perhitungan efisiensi pada skenario 1

Nama File	Hasil Perhitungan
Arizona.blend	1.11%
Beamsattelite.blend	0.85%
Paddywagon.blend	0.44%
Redexosuit.blend	1%
Policelifter	1%
Tankbot.blend	0.93%

Sumber : Pengujian dan analisis

BAB VI
PENUTUP

6.1 Kesimpulan

Dari hasil pengamatan selama perancangan, implementasi, dan proses pengujian perangkat lunak yang digunakan, diambil kesimpulan sebagai berikut:

1. Sistem render farm yang diimplementasikan dapat dijadikan solusi dari permasalahan sumber daya komputasi dalam pemenuhan proses *render* 3D
2. Berdasarkan dari hasil perhitungan kinerja komputasi melalui proses render farm didapatkan didapatkan hasil MIPS lebih besar, terjadi efisiensi kinerja komputasi dan adanya peningkatan kinerja dari komputasi
3. Dengan adanya tambahan tambahan GUI pada sistem *render farm* yang digunakan pada sistem mempermudah dan mempercepat dari proses instalasi, *submit job* dan aktivasi sistem pada server. Hal ini dapat dibandingkan dengan penggunaan sistem sebelumnya yang diharuskan menggunakan terminal atau konsol, yang bisa saja menyulitkan pengguna terutama pengguna yang tidak akrab dengan penggunaan bahasa terminal atau konsol
4. Sistem *render farm* memiliki tingkat keefektifan jika digunakan untuk melakukan *render* data animasi 3D yang memiliki jumlah render lebih dari 1, karena sistem kerjanya yang membagi *frame* animasi ke komputer yang ada. Sistem ini menjadi tidak efektif jika hanya melakukan proses *render* pada data animasi yang hanya memiliki 1 buah *frame*.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan sistem ini antara lain :

1. Disarankan untuk dapat menambahkan pengolah animasi 3D lain selain *blender* seperti *3D Max* dll.
2. Ditambahkan nya fitur *auto refresh* pada *web monitoring* sehingga *monitoring* lebih *real time*.

DAFTAR PUSTAKA

[WBM-10] Wilkinson,Barry Allen,Michael.2010.*Parallel Programming Teknik dan Aplikasi Menggunakan Jaringan Workstation dan Komputer Paralel*.Andi.Yogyakarta

[DCH-00] Davis,Tim Chalmers,Alan Jensen,Henrik Wann.2000. Practical Parallel Processing for Realistic Rendering.New Orleans,

[RMH-09] Rumagit,Mourits,Arthur Hariadi,Moch.2009.*Manajemen Grid Untuk Render Animasi 3 Dimensi*.Jurusan Teknik Elektro Institut Teknologi Sepuluh November.Surabaya

[BAL-08] Britto,Allan.2008. *Blender 3D: Architecture, Buildings, and Scenery*.PACKT Publishing. Birmingham,United Kingdom

[HRL-09] Hess,Roland.2009.*Animating With Blender*.Focal Press.Oxford.United Kingdom

[AAH-04] Abbas, Ahmar.2004. *Grid Computing: A Practical Guide to Technology and Applications.* Charles River Media, Inc. Massachusetts.United States of America

[GOF] Nugraha,Giovanni.Ekwardo,Odit.Mukhlish,Fata.*Penggunaan Algoritma Divide and Conquer Dalam Parallel Computing Untuk Melakukan 3D Rendering*.Institut Teknologi Bandung.Bandung

[SYR-10] Syahrul.2010. Organisasi dan Arsitektur Komputer. Andi,Yogyakarta

[DIK-11] Diktat TI Telkom, BAB XVI QUALITY OF SERVICE (QoS) DAN PENGUKURANNYA

repository.ub.ac
UNIVERSITAS BRAWIJAYA