IMPLEMENTASI HIGH PERFORMANCE COMPUTING MENGGUNAKAN PELICANHPC DENGAN TEKNIK BEOWULF CLUSTERING

Andry Kurniawan **Periyadi, S.T.** **Anang Sularsa, S.T., M.T.**

andrykurniawan@tass.telkomuniversity.ac.id periyadi@tass.telkomuniversity.ac.id anang@tass.telkomuniversity.ac.id

Program Studi Teknik Komputer Jurusan Network Engineering, FIT, Telkom University
Jl. Telekomunikasi no.1, Terusan Buah Batu, Bandung 40257

Abstrak

Waktu *rendering* terkadang memberikan dampak terhadap penyedia layanan *rendering video*. Sehingga dibutuhkan sebuah metode yang dapat mempercepat waktu *rendering video*.
HPC memungkinkan sebuah komputer membagi sumber dayanya kepada komputer lain yang terhubung dalam *node* yang sama. Dengan HPC maka penyedia layanan bisa menyelesaikan masalah komputasi dengan akurat, cepat, dan efisien dengan biaya yang lebih minim. HPC merupakan salah Satu solusi terhadap permasalahan *rendering* dikarenakan dengan HPC akan mempercepat proses komputasi pada komputer karena dikerjakan secara paralel.
Penulis menggunakan sistem operasi pelicanHPC yang merupakan sistem operasi yang berbased debian. Dengan menggunakan HPC performa komputer dapat meningkat serta komputer dapat mempercepat waktu dalam proses *rendering video* dan *rendering* animasi walaupun perbedaanya tidak banyak.

Kata Kunci: HPC, *NODE*, PELICANHPC

Abstract

Rendering times sometimes give effect to rendering video provider so need ad method to accelerate the time in rendering video.
HPC enable a komputer divide the resource to other komputer that connect in same node. With HPC the provider can finish the computing problem with accurate, fast and efficient with minimal costs. HPC is one of solution for rendering problem, because HPC will accelerate computing process in komputer using paralel job.
Writers use pelicanHPC operating system that constitute debian based operating sytem. With using HPC computer performance can be improved and komputer can accelerate the process time in rendering video and rendering animation although the different time not much.

Keywords: HPC, NODE, PELICANHPC

1. Latar Belakang

Perkembangan Teknologi informasi di Indonesia sudah semakin berkembang pesat. Semua memanfaatkan fasilitas teknologi informasi sebagai sarana komunikasi baik pada *multimedia*, ekonomi, pendidikan dan juga bidang lainya.

Dalam hal *multimedia* contohnya untuk proses *rendering video*. Penyedia layanan harus memiliki kualitas yang lebih baik dibandingkan para pesaingnya. Maka dari itu penyedia layanan memerlukan komputer yang dapat mempercepat proses *rendering video*.

HPC memungkinkan sebuah komputer membagi sumber dayanya kepada komputer lain yang terhubung dalam *node* yang Sama. Dengan HPC maka penyedia layanan bisa menyelesaikan masalah komputasi dengan akurat, cepat, dan efisien dengan biaya yang lebih minim. HPC merupakan solusi terhadap permasalahan *rendering* dikarenakan dengan HPC akan mempercepat proses komputasi karena dikerjakan secara paralel.

Beowulf Clustering merupakan teknik dalam menerapkan *High Performance Computing* komputer *server* terhubung dengan beberapa komputer *node* menggunakan jaringan *Ethernet* melalui jaringan yang Sama. Dalam *Beowulf* hanya bisa dihubungkan *via Ethernet* serta dalam 1 jaringan yang Sama.

PelicanHPC merupakan salah satu sistem operasi yang dapat digunakan untuk mengimplementasikan HPC. PelicanHPC merupakan turunan dari sistem operasi debian. Untuk paralel *computing* PelicanHPC masih sangat jarang digunakan.

2. Dasar Teori

2.1 HPC

High performance computing/*cluster* untuk komputasi kinerja tinggi menggunakan prinsip dasar pengolahan data secara *paralel* dimana sebuah tugas dipecah menjadi beberapa tugas atau proses yang lebih kecil yang kemudian dijalankan secara *paralel*. Hasil dari proses yang lebih kecil ini digabungkan kembali sehingga menjadi hasil tugas besar yang aslinya. Dengan diolahnya data secara *paralel* maka sebuah pekerjaan secara teori dapat diselesaikan dengan kecepatan yang diharapkan sebanding dengan jumlah mesin (*node*) dalam mesin paralelnya [1].

2.2 *Beowulf Clustering*

Beowulf adalah sebuah arsitektur *multi-komputer* yang digunakan untuk komputasi paralel , sistem *Beowulf* biasanya terdiri dari 1 *server node* dan beberapa client *node* yang terhubung bersama menggunakan jaringan *Ethernet* atau dalam 1 jaringan yang sama [3].

2.3 *Live CD*

CD Live linux adalah sistem operasi linux yang disimpan disebuah bootable CD atau DVD yang kemudian dapat dijalankan langsung dari CD atau DVD drive, tanpa harus meng*install*nya secara permanen dimedia penyimpanan komputer.

Sistem operasi live CD tersimpan di *media booting* yang digunakan, semua proses mulai dari *boot* sampai *shutdown*. Saat komputer *booting script* yang ada akan memuat *image kernel*. Selanjutnya, sejumlah *ramdisk* akan dibuat dalam RAM yang berguna untuk tempat penyimpanan data sementara untuk *booting*. Lalu *image rootdisk* dibongkar didalam ram dan di-*mount* sebagai *system* file root.

Sementara untuk direktori lain akan di-*mount* langsung dari *media booting*. Sehingga beban kerja komputer akan berkurang.karena tidak semua disimpan didalam memori.

Untuk penyebaranya Sekarang hampir semua distro linux seperti Slackware, Debian, Mandreva, Gentoo dan sejumlah turunanya menyediakan CD live siap pakai yang dapat di-download dan dicoba oleh para pemula dan penggunanya. live CD biasanya digunakan untuk menguji sejumlah fitur baru sehingga bisa diinstal tanpa harus mempartisi *hard disk* [2].

2.4 Komputer

Istilah komputer diambil dari Bahasa latin yaitu *computate* yang berarti menghitung. Ada beberapa definisi tentang komputer menurut para ahli di-antaranya:

Robert H. Bissmer (komputer *annual*)

"Komputer merupakan alat elektronik yang mampu melakukan beberapa tugas: menerima *input*, memproses *input* tersebut sesuai programnya, menyimpan perintah-perintah dan hasil pemrosesan, serta menyediakan *output* dalam bentuk informasi [5]."

Donald H.sanders

"Komputer merupakan sistem elektronik untuk memanipulasi data yang cepat dan tepat, dirancang dan diorganisasikan agar dapat menerima menyimpan data *input*, memprosesnya serta menghasilkan *output* dibawah pengawasan suatu langkah-langkah instruksi program yang tersimpan dimemori secara otomatis [5]."

Dari keterangan beberapa pakar tersebut dapat diambil kesimpulan bahwa Komputer adalah:

1. Alat elektronika
2. Dapat menerima *input* data
3. Dapat mengolah data
4. Dapat memberikan informasi
5. Menggunakan suatu program yang tersimpan dimemori computer
6. Dapat menyimpan program dan hasil pengolahan, dan bekerja secara otomatis [5].

2.5 *FLOPS*

Flops adalah operasi angka *floating point* yang dihasilkan dalam waktu satu detik. secara teoritis, sebuah *processor* biasanya dapat menghasilkan jumlah hingga jutaan FLOPS. Ini adalah hasil pengujian standar internasional yang biasanya digunakan untuk membandingkan kinerja dari berbagai sistem komputer.

Angka tersebut dihasilkan dalam kondisi maksimal, yaitu pada saat *processor* belum mendapatkan banyak instruksi. angka tersebut biasanya diperoleh saat komputer baru dinyalakan atau *boot* ulang [4].

3. Analisis dan Perancangan

3.1 Kebutuhan sistem

3.1.1 Kebutuhan Perangkat Keras

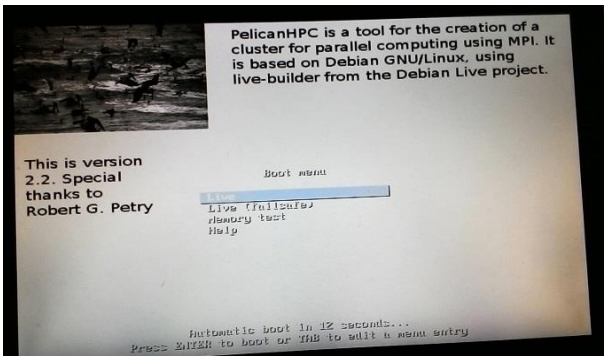
Dalam pengerjaan penelitian ini, digunakan perangkat keras dengan spesifikasi sebagai berikut :

1. *Switch*
   Menggunakan *Switch* D-link DES-1008A.
2. 1 buah komputer *frontend node.*
   a. *Processor* : Intel *core* i3
   b. RAM : 3 GB
3. 3 buah komputer *node.*
   A. *processor* : 2 komputer *core* i3 dan 1 komputer *core* 2 duo.
   b. RAM : 2 komputer 3 GB dan 1 komputer 1 GB.

3.1.2 Kebutuhan Perangkat Lunak

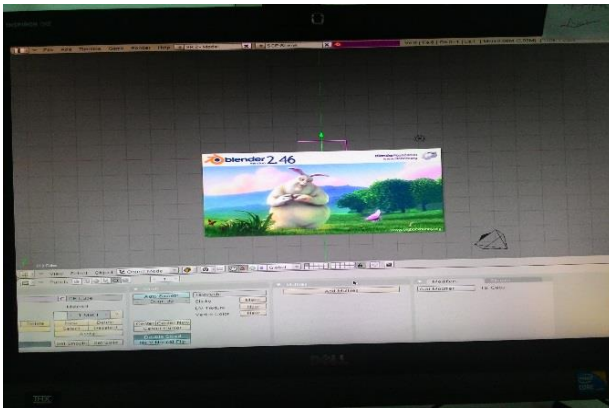
Berikut merupakan spesifikasi perangkat lunak yang digunakan :

1. Sistem operasi PelicanHPC v2.2.



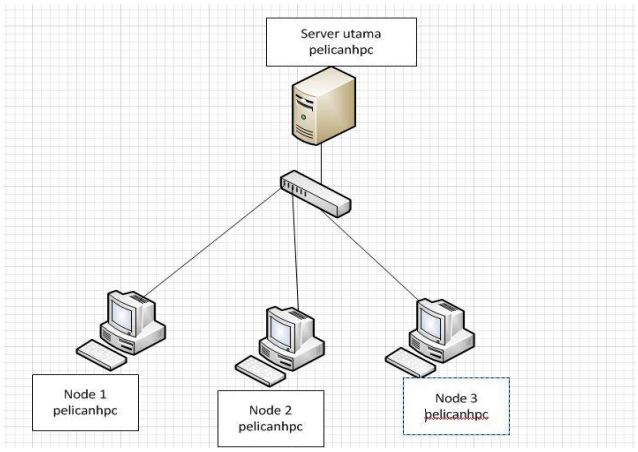
Gambar 3-1 Pelican

2. Aplikasi *render* menggunakan blender v 2.46.



Gambar 3-2 Blender

3.1.3 Perancangan Jaringan



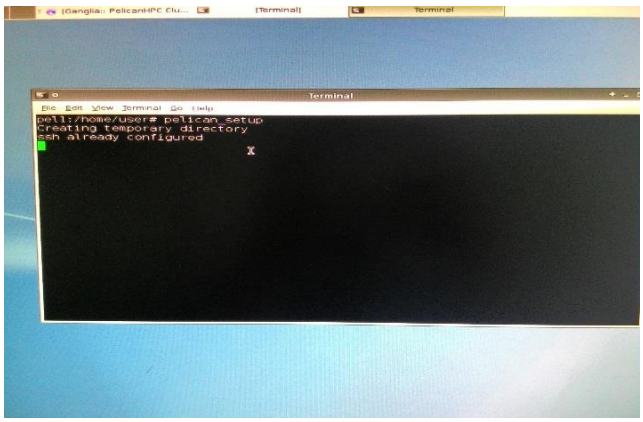
Gambar 3-3 Topologi

Frontend node akan mencari *node* yang terhubung dalam 1 jaringan sehingga beban komputasi yang dikerjakan *frontend node* akan dikerjakan secara paralel.

4. Pengujian

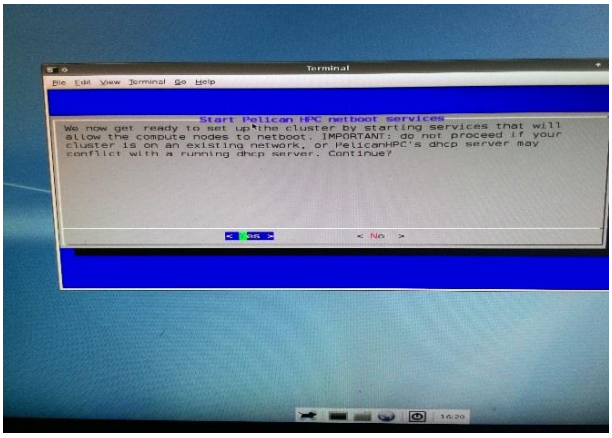
4.1 Menguji cluster

Pelican setup digunakan untuk menjalankan proses pencarian *cluster* pada pelicanHPC.



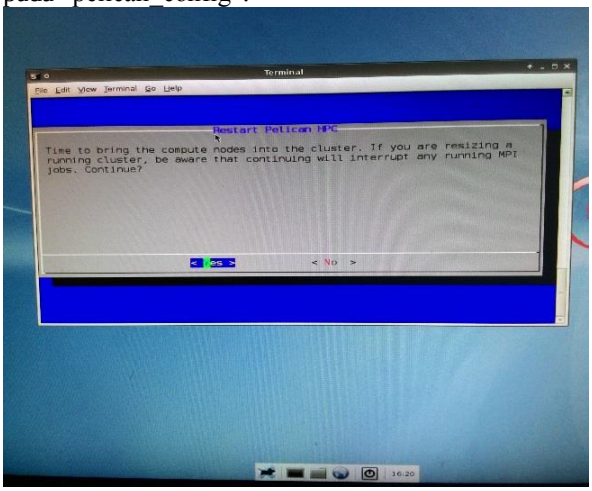
Gambar 4-1 Pelican Setup

Pastikan jika *cluster* tidak terhubung dengan DHCP *server* yang lain atau *frontend node* sedang bergabung dengan jaringan yang lain. Bagian ini juga digunakan untuk *clustering* otomatis. *Media* penghubungnya melalui *booting* jaringan.

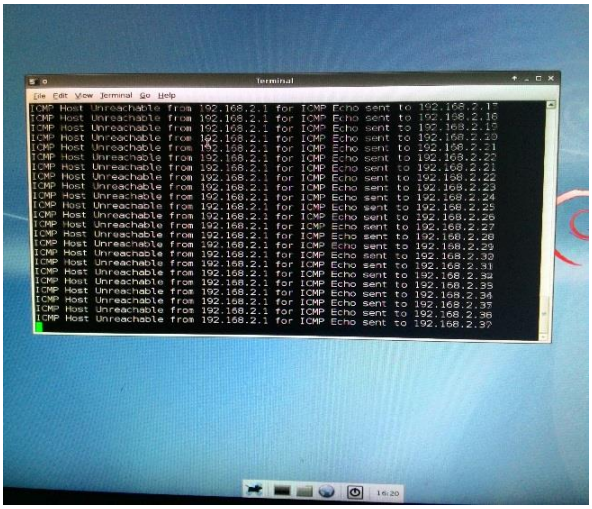


Gambar 4-2 Net*booting*

Selanjutnya pelicanHPC akan mencari *node* yang berada dalam 1 jaringan dengan *frontend node*. Ip address yang dicari adalah ip address yang berada pada "pelican_config".



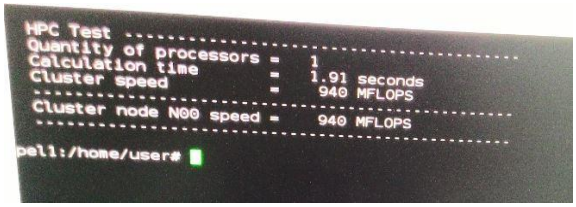
Gambar 4-3 memulai pencarian *node*



Gambar 4-4 Pencarian *node*

Hasil dari pengujian ini adalah sebagai berikut:

a. Tanpa *node*



Gambar 4-5 Flops tanpa *node*

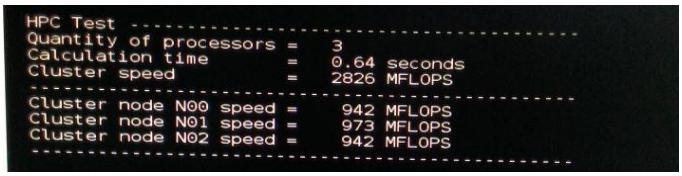
Dapat dilihati pada pengujian pertama jumlah *processor*s yang terbaca adalah 1 dan flops total adalah 940 MFLOPS.

b. 1 *node*

Gambar 4-6 Flops 1 *node*

Dapat dilihat pada pengujian 1 *node*. *Processor*s yanbg terbaca adalah 2 dan flops total adalah 1880 MFLOPS.

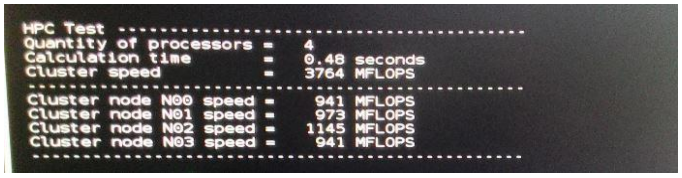
c. 2 *node*



Gambar 4-7 Flops 2 *node*

Dapat dlihat pada pengujian 2 *node*. *Processor*s yang terbaca adalah 3 dan flops total adalah 2826 MFLOPS.

d. 3 *node*



Gambar 4-8 Flops 3 *node*

Dapat dilihat pada pengujian 3 *node*. *Processor*s yang terbaca adalah 4 dan flops total adalah 3764 MFLOPS.

Tabel 4-1 Flops

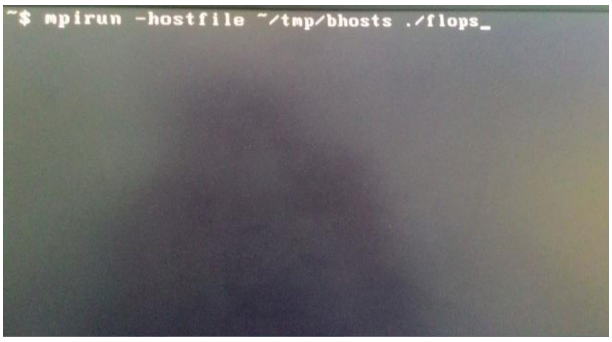
node	Calculation time	*Cluster node* flops
0	1.91 seconds	940 mflops
1	0.96 seconds	1860 mflops
2	0.64 seconds	2826 mflops
3	0.48 seconds	3764 mflops

Dari pengujian ini terlihat flops bertambah ketika *node* semakin banyak aktif. Yang dapat diartikan bahwa kecepatan *cluster* dalam mengeksekusi semakin cepat juga.

4.2 Pengujian Job Flops pada Cluster

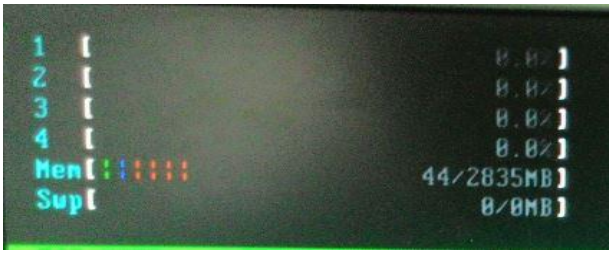
pada tahap ini *frontend node* akan memberikan tugas kepada *node* untuk melakukan perintah flops kembali. hampir sama dengan pengujian pertama tetapi pengujian kali ini dilakukan pada saat *cluster* sudah terbentuk.

a. Pada *frontend node* ketikan perintah "mpirun –hostfile ~/tmp/bhosts. /flops" untuk memulai pemberian *job* pada *node* yang ter-identifikasi pada "~/tmp/bhosts".

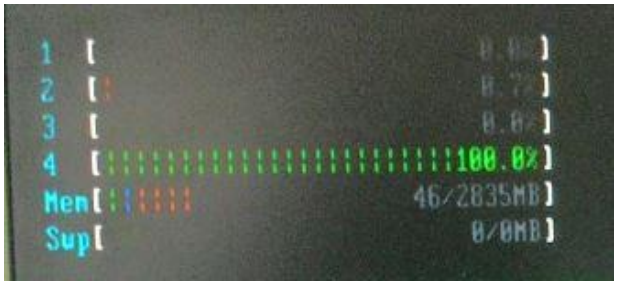


Gambar 4-9 Pengujian mpirun

b. Setelah itu pada *node* ketikan perintah "htop" untuk melihat beban yang diterima CPU pada masing-masing *node*.



Gambar 4-10 Okupansi sebelum diuji



Gambar 4-11 Okupansi setelah diuji

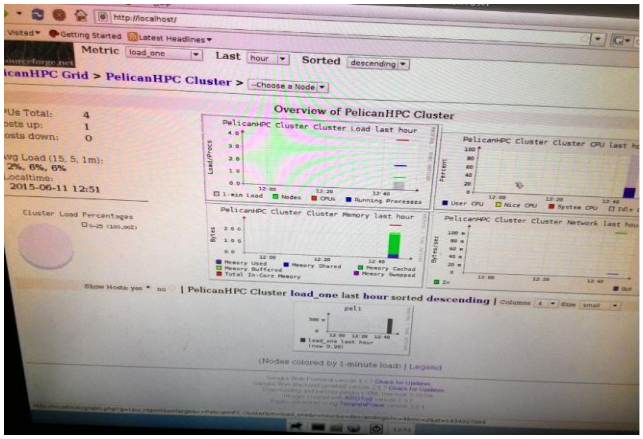
4.3 Pengujian Ganglia

Pengujian kedua penulis lakukan dengan aplikasi monitoring ganglia. Penulis akan melihat keadaan *cluster* contohnya adalah jumlah memori total ataupun memori per-*user*.

Ganglia juga menampilkan informasi dalam bentuk *graphic*.

Penulis mendapatkan hasil:

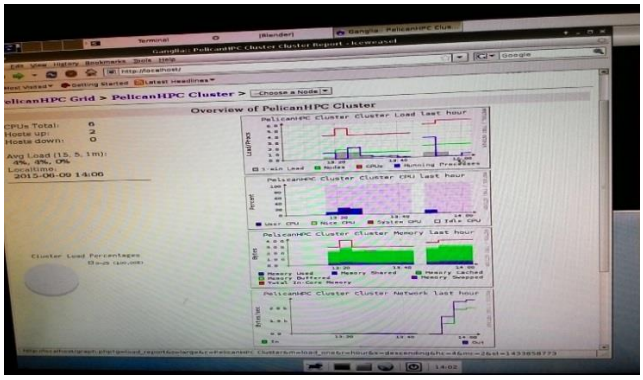
a. Tanpa *node*



Gambar 4-12 Ganglia tanpa *node*

Ketika hanya *frontend node* saja yang aktif pada bagian h*ost up* hanya muncul 1 saja karena *node* yang lain belom diaktifkan.

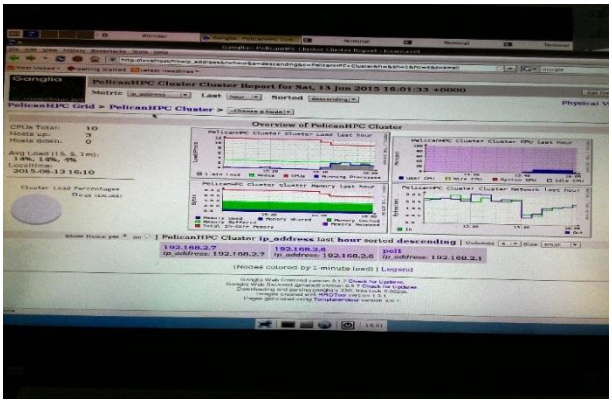
b. 1 *node*



Gambar 4-13 Ganglia 1 *node*

Setelah *node* pertama diaktifkan Jumlah *host up* dan *core* dalam CPU total akan bertambah. Terlihat pula peningkatan pada RAM.

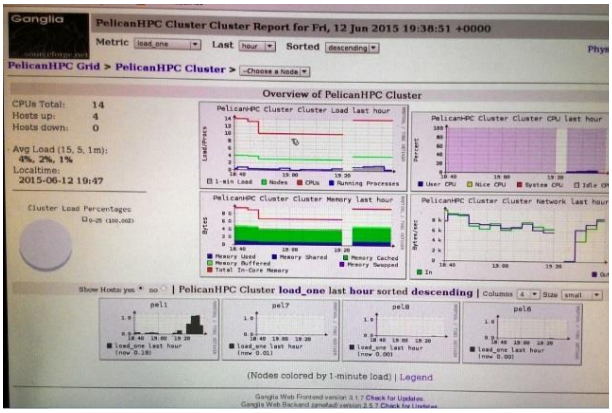
c. 2 *node*



Gambar 4-14 Ganglia 2 *node*

Setelah *node* kedua diaktifkan Jumlah *host up* dan *core* dalam CPU total akan bertambah. Terlihat pula peningkatan pada RAM.

d. 3 *node*



Gambar 4-15 Ganglia 3 *node*

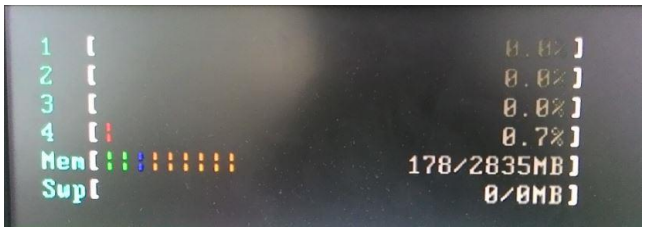
Setelah *node* ketiga diaktifkan Jumlah host up dan *core* dalam CPU total akan bertambah. Terlihat pula peningkatan pada RAM.

Kesimpulan dari pengujian ini adalah *graphic* trus meningkat ketika *host up* bertambah dari julmah *core* sampai total memori yang ada terdapat peningkatan.

4.4 Pengujian *Rendering video* dan animasi

a. Parameter yang akan diukur adalah seberapa cepat waktu *frontend node* untuk me-*render video* dan animasi.
b. *Video* yang digunakan adalah *video* film divergent 720 p *Blue ray* dengan frame yang akan diambil antara frame 1-2000
c. Animasi yang di*render* adalah animasi pemtantulan bola dengan resolusi 1920:1280.
d. Hasil yang diharapkan adalah adanya perbedaan waktu *render* saat *node* non aktif dengan *node* yang aktif.
e. Hasil pengukuran adalah sebagai berikut.

Berikut ini merupakan tampilan okupansi CPU *node* ketika melakukan *render* dimana okupansi terlihat kecil dikarenakan tidak ada pembagian *render* menggunakan net *render*.



Gambar 4-16 Okupansi *render*

Tabel 4-2 Waktu *render video*

NODE	Frame	Waktu *render*
0	1-2000	4:02:73
1	1-2000	4:01:62
2	1-2000	3:59:13
3	1-2000	3:58:26

Tabel 4-3 Waktu *render* animasi

NODE	Frame	Waktu *render*
0	1-250	13:18:23
1	1-250	13:16:31
2	1-250	13:14:24
3	1-250	13:11:78

f. Berdasarkan hasil diatas terlihat jika tanpa *node* akan membutuhkan waktu lebih lama daripada menggunakan *node*. Tetapi perbedaan waktu terlihat tidak banyak dikarenakan okupansi pada CPU *node* sangat kecil.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Dari pengujian diatas, maka penulis dapat mengambil kesimpulan sebagai berikut :

1. Performa komputer meningkat terlihat dari pengujian rata-rata peningkatan flops adalah 900 MFLOPS per-*node* yang aktif (sesuai dengan spesifikasi *processor* yang digunakan).
2. Waktu *rendering* berukang walaupun tidak banyaj perbedaan antara 1-3 detik per-*node* yang aktif..

5.2 Saran

Saran yang dapat penulis berikan pada penelitian ini diantaranya :

1. Proyek akhir ini bisa dikembangkan dengan *media wireless*.
2. Tambah lagi dalam banyaknya *node* agar efek komputasi semakin terasa..
3. Agar pembagian kinerja *render* jelas gunakan mode net *render* blender.

DAFTAR PUSTAKA

[1] e. dahniar, “Perancangan Dan Realisasi High Performance Cluster menggunakan MPICH2 Dengan Studi kasus Portal Mahasiswa Politeknik Telkom,” *Proyek Akhir,* p. 7, 2012.

[2] k. y. antonius, “Tutorial Praktis :membuat CD live linux dengan kernel sendiri,” *Live CD linux,* p. 1, 8 Agustus 2006.

[3] k. swendson, “the beowulf HOWTO,” *beowulf,* p. 1, 17 mei 2004.

[4] e. sutanta, “PEMANFAATAN METODE ITERASI MATEMATIS,” *Kinerja processor,* p. 3, 5 maret 2005.

[5] g. a. mutiara dan periyadi. , Sistem Komputer, yogyakarta: deepublish, 2013.