# APLIKASI SISTEM MONITORING BERBASIS WEB UNTUK OPEN CLUSTER

*(APPLICATION OF MONITORING SYSTEM WEB BASED FOR OPEN CLUSTER)*

**TUGAS AKHIR**

***Diajukan sebagai syarat untuk memperoleh gelar Sarjana Teknik***

***Di Jurusan Teknik Elektro Sekolah Tinggi Teknologi Telkom***

**Disusun Oleh :**

**GHEYB JHUANA OHARA**

**111000168**


**JURUSAN TEKNIK ELEKTRO**

**SEKOLAH TINGGI TEKNOLOGI TELKOM**

**BANDUNG**

**2005**

# ABSTRAKSI

Open cluster yang dikembangkan oleh LIPI adalah alternatif pembentukan komputasi parallel fisik berbiaya murah yang dibentuk dari sejumlah komputer dan dibuka untuk publik. Pada umumnya pemakaian sebuah cluster cenderung hanya untuk jaringan tertutup dengan akses dari luar yang terbatas. Seperti kita ketahui algoritma parallel mungkin tampak memiliki speed-up yang tinggi secara teoritis, tetapi saat diimplementasikan ke dalam sistem yang nyata memberikan hasil yang lebih rendah sehingga terdapat beberapa masalah yang dapat membatasi kinerja program parallel diantaranya *memory contension, Kode sekuensial yang berlebihan, Delay komunikasi, Delay sinkronisasi*, dan lain-lain.

Sebagai sistem yang baru, *open cluster* ini membutuhkan sistem penunjang lain yang mampu mendukung operasional sistem tersebut seperti sistem monitoring dan manajemen, karena monitoring merupakan jantung dari managemen sistem cluster. Pada tugas akhir ini dikembangkan sistem *monitoring* untuk sumberdaya *hardwar*e seperti *ethernet traffic, memory usage, cpu resource* dan *system statistic* untuk masing-masing node di dalam *cluster*. Selain itu dibutuhkan juga manajemen untuk pengaturan node, item monitoring, *power* kontol dan juga pengaturan *user*. Untuk pengumpulan data digunakan SNMP (*Simple Network Management Protokol*) dan tools lain yang mendukung. Karena cluster ini dibuka untuk publik, maka sistem *monitoring* yang dikembangkan adalah sistem yang dapat diakses oleh publik melalui media *web*.

# ABSTRACT

Open cluster that being developed by LIPI is one of the alternative ways to built a parallel computation with low cost budget, and it's build from several computers and open for public. Usually a cluster is closed for public, and only have limitted acces link to the outside. We all knows the algorithm of parallel looks to have speed-up in theoritical, but when it's implement to the real system, it give a less result/output and it give some problems that can limmited the works of parallel programing, such as : *memory contension, to much sequential code, delay communication, delay sinchronyzation, etc.*

As a new system, open cluster need another benefactor system that can support the operational system such as monitoring system, because monitoring is the heart of cluster system management. For my thesis, I develop a monitoring system for hardware like ethernet trafic, memory usage, cpu resource and system static for each node in the cluster. Beside that management to arrange the node, monitoring item, power control and user management needed too. To collect the data, I use SNMP (Simple Network Management Protokol) and other supporting tools. Because this cluster is open for public, so the monitoring system that being develop is the system that can be access by public from web as the media.

# DAFTAR ISI

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Teknologi *cluster* yang dikembangkan oleh LIPI dibutuhkan untuk mengikat beberapa *server* agar menjadi suatu sistem tunggal sumber daya komputasi ilmu pengetahuan (*computational science*) yang melakukan pekerjaan besar. Dari sisi pengguna, ia merasa bahwa pekerjaan yang dia berikan telah dibagi ke mesin fisik yang berbeda karena pada dasarnya *cluster* dikategorikan sebagai tipe memori yang terbagi-bagi pada mesin paralel karena setiap *node* mempunyai memori dan prosesor tersendiri.

Biasanya pemakaian sebuah *cluster* cenderung hanya untuk jaringan yang tertutup (*closed private network*) dengan akses keluar yang terbatas[3] dan pengalokasian node yang *single* block dimana jika terjadi kerusakan pada beberapa node *cluster*nya dapat dialokasikan kepada node yang lain, Sehingga *user* yang menjalankan program paralelnya tidak mengetahui kondisi fisik dari *cluster* tersebut dan hanya akan memperoleh hasil akhir dari *jobs*nya saja tanpa mengetahui bagaimana kinerja dari paralel *cluster* tersebut. Karena *cluster* yang dibangun dapat diakses melalui jaringan *internet* (*open cluster*) yang dirancang dengan *multi block*, dimana setiap *user* yang menjalankan sistem ini mendapatkan alokasi *node* sendiri, maka sangat memungkinkan untuk memonitor aktifitas yang dilakukan *cluster* tersebut sehingga dapat diketahui informasi mengenai seberapa besar beban untuk masing-masing node ketika suatu *job* sedang *running*, bagaimana kinerja prosesor untuk masing-masing *node*, berapa suhu prosesor, berapa memori yang tersedia, berapa *bandwith* pada *network interface card (*NIC*)* selama proses pembagian kerja antara *node* satu dengan *node* yang lain.

Informasi diatas adalah informasi yang sangat signifikan dan diperlukan oleh pihak pengelola dan juga *user* sebagai pengguna sistem. *Open cluster* yang dikembangkan oleh LIPI belum memiliki semua komponen yang dibutuhkan oleh suatu sistem *cluste*r, termasuk sistem *monitoring*. Oleh sebab itu penulis mencoba

mengembangkan sistem *monitoring* tersebut, terutama yang berhubungan dengan sumber daya *hardware*. Karena sistem *cluster* ini dapat diakses melalui *internet*, tentu saja dukungan sistem *monitorin*g ini juga dapat diakses melalui *internet*.

## 1.2 Tujuan

Tugas Akhir ini disusun untuk mengembangkan suatu sistem *monitoring* dan sistem kontrol pada *open cluster* yang dapat memonitor aktivitas dan indikator dari sumberdaya *cluster* sehingga user dapat mengetahui sumber daya dari *cluster* tersebut. Semua *progres report* dapat dilihat oleh user yang menjalankan *job*nya pada paralel *cluster* tersebut melalui *web* secara *on demand* .

## 1.3 Perumusan Masalah

Pada Tugas Akhir ini dirumuskan masalah sebagai berikut :

1. Bagaimana membuat suatu sistem yang dapat memonitor aktivitas dan indikator pada paralel *cluste*r.
2. indikator yang akan dimonitor antara lain adalah sumber daya *hardware* seperti CPU *resources* (segala sesuatu yang berhubungan dengan sistem statistik), *memory resources*, *traffic input* ataupun *output* yang terjadi (kinerja *Ethernet*), *job progress*, *running* proses, temperatur, dan lain-lain yang dirasa cukup mempengaruhi kinerja komputasi paralel tersebut.
3. Tipe *monitoring* adalah *realtime* dan *on demand* melalui media web.
4. *Interface* yang digunakan untuk menampilkan indikator yang telah dibaca kepada user adalah CGI(*Commond Gateway Interface*)/HTML.
5. Perangkat lunak yang digunakan untuk membangun aplikasi *monitoring* ini adalah *python*, karena bahasa pemrograman *Python* yang bersifat *intepreter* sehingga mendukung akses secara *realtime* di dalam pemerolehan data *monitoring*. Alasan yang lain karena bahasa pemrograman *Python* menggunakan pendekatan secara *Object Oriented Programming*

(OOP) sehingga ideal untuk pemrograman *web* secara *Common Gateway Interface* (*CGI*).

## 1.4 Batasan Masalah

Ruang lingkup dalam pembahasan Tugas Akhir ini adalah sebagai berikut :

1. *Job* yang dijalankan oleh *user* pada paralel *cluste*r merupakan program komputasi paralel.
2. Tidak membahas secara mendalam mengenai bagaimana cara merancang dan membuat aplikasi dengan menggunakan berbagai macam *algoritma* dan fungsi pemrograman paralel dalam lingkungan *cluster* karena yang diutamakan adalah membahas mengenai sistem *monitoring*.
3. membaca semua indikator yang terjadi selama *job* dijalankan dari sistem *linux* di setiap *node* maupun di semua *node*.
4. Menampilkan semua *item* yang telah dimonitor melalui media *web*.
5. semua proses dalam pembuatan perangkat lunak hanya dilakukan pada OS (*Operating System*) *linux* saja.
6. Tidak dilakukan perancangan paralel *cluster* karena yang lebih difokuskan disini adalah pembuatan sistem *monitoring* untuk mengetahui kinerja dari paralel *cluster*.

## 1.5 Metode Penelitian

Metode penelitian yang digunakan pada Tugas Akhir ini adalah:

1. **Studi literatur**, yaitu dengan melakukan studi berdasarkan referensi dan berbagai diskusi pembahasan baik dengan dosen pembimbing maupun dengan orang yang berkompeten pada kasus ini.
2. **Pembuatan sistem monitoring,** yang meliputi tahapan terstruktur sebagai berikut:
   a. Perancangan perangkat lunak yang akan menjadi *interface* untuk menampilkan hasil dari sistem *monitoring*

b. Implementasi dan Uji Coba

3. **Studi Pengembangan Aplikasi** yang bertujuan untuk menentukan metodologi pengembangan Perangkat Lunak yang digunakan dengan pendekatan terstruktur.
4. **Analisa sistem,** dengan melakukan ujicoba
5. **Mengambil kesimpulan**

## 1.6 Sistematika Penulisan

Adapun sistematika penulisan Tugas Akhir ini adalah sebagai berikut :

**BAB I** **Pendahuluan**

Berisi latar belakang masalah, tujuan penulisan, perumusan masalah dan batasannya, metodologi penyelesaian masalah yang digunakan, serta sistematika penulisan yang memuat susunan penulisan Tugas Akhir ini.

**BAB II** **Landasan Teori**

Dalam bab ini menguraikan landasan teori yang mendukung dan mendasari penulisan Tugas Akhir ini, yaitu mengenai sistem *monitoring* serta penerapan aplikasinya.

**BAB III** **Perancangan sistem**

Bab ini berisi perancangan dari aplikasi sistem *monitoring* berbasis *web* untuk *open cluster*.

**BAB IV** **Implementasi dan Analisa Sistem**

Bab ini berisi implementasi dan analisa sistem. Implementasi program monitoring yang dibuat dan analisa sistem yang meliputi analisa fasilitas sistem dan performansinya terhadap sistem *cluster* itu sendiri.

**BAB V** **Kesimpulan dan Saran**

Berisi kesimpulan dari penulisan Tugas Akhir ini dan saran untuk pengembangan lebih lanjut.

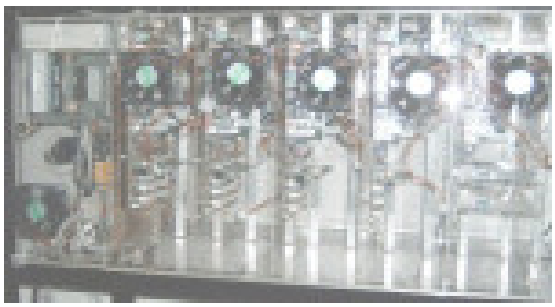
# BAB II
# DASAR TEORI

## 2.1 *Open cluster*

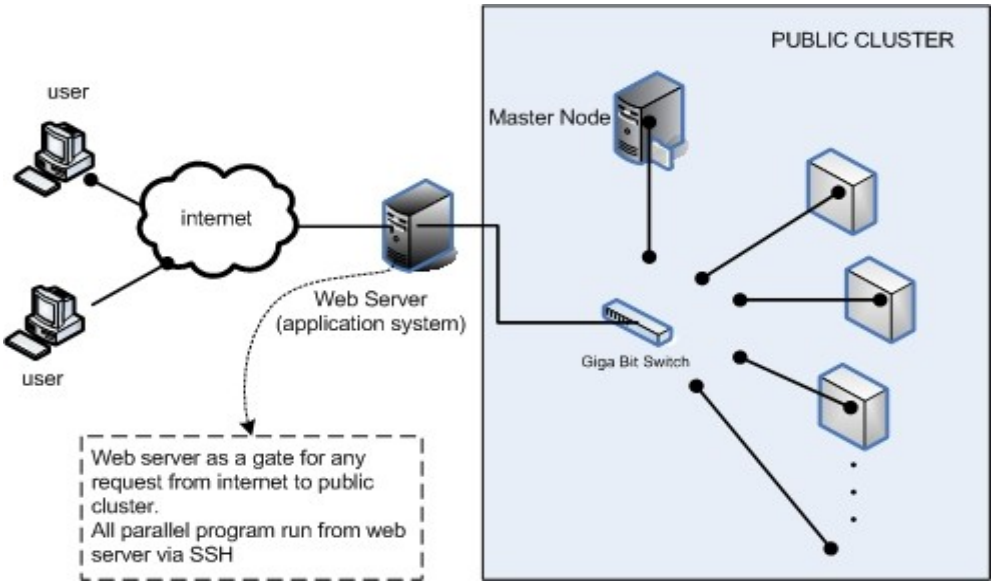
*Open cluster* (*public cluster*) merupakan *cluster* yang sedang dikembangkan di *Puslit Fisika*-LIPI (Lembaga Ilmu Pengetahuan Indonesia) yang dapat diakses melalui internet untuk kebutuhan komputasi paralel. *Cluster* yang dikembangkan diharapkan dapat mengakomodasi kebutuhan tersebut. Untuk itu spesifikasi yang digunakan adalah prosesor p4/2.4 GHz/1 Gbytes RAM/Gigabit *ethernet*[3]. Sebagai *operating sistem* (OS) digunakan *linux*.

Semua *node motherboard* diletakkan sejajar, dimana *master node* diletakkan paling akhir. Masing-masing *node* memiliki *power supply* yang secara langsung akan dikontrol oleh *master node*. Masing-masing *motherboard* memiliki *Gigabit ethernet* yang dihubungkan ke *switch*. Gambar fisik *open cluster* seperti gambar 2.1 dibawah.

*Master node* digunakan sebagai *gateway* untuk dunia luar seperti terlihat pada gambar 2.2, dimana *master node* bertindak sebagai penghubung *web server* yang memeberikan akses kepada *user* untuk memonitor kinerja dari *cluster* tanpa mengakses masing-masing *cllient node* dan *master node*. Itulah sebabnya mengapa sistem ini dikatakan *secure* karena semua akses hanya melalui *master node*.



Gambar 2.1 Tampilan fisik *cluster LIPI*




Gambar 2.2 Diagram *network cluster* dengan *master node* sebagai *gateway*

Saat ini *open cluster* yang ada terdiri dari 9 node dengan spesifikasi seperti tabel dibawah ini:

Tabel 2.1 spesifikasi *cluster* di LIPI[3]

| node | spesifikasi | kapasitas {Gflops) |
|---|---|---|
| 1 | Master, Processor 2xP3-933 MHz, RAM 1Gb, 40 Gb IDE HDD | 1 |
| 2 | Processor 486-66 Mhz, RAM 32 Mb, diskless | 0.1 |
| 3 | Processor 486-66 Mhz, RAM 32 Mb, diskless | 0.1 |
| 4 | Processor P4 2.4 Ghz, RAM 1 Gb, diskless | 5 |
| 5 | Processor P4 2.4 Ghz, RAM 1 Gb, diskless | 5 |
| 6 | Processor P4 2.4 Ghz, RAM 1 Gb, diskless | 5 |
| 7 | Processor P4 2.4 Ghz, RAM 1 Gb, diskless | 5 |
| 8 | Processor P4 3 Ghz, RAM 1 Gb, 40 Gb SATA HDD | 6 |
| 9 | Processor P4 3 Ghz, RAM 1 Gb, 40 Gb SATA HDD | 6 |

Mengurus *cluster* tidak sama seperti mengurus *single* PC. Hal ini disebabkan karena *resource* didistribusikan ke beberapa *node*. Untuk itu dibutuhkan manajemen yang baik untuk membantu *user* dan *administrator* dalam menggunakan *cluster* seperti [10] :

- *Shutting down* atau *booting up* masing-masing *node*.
- *Remote rlogin* dan *remote* eksekusi perintah ke masing-masing *node*

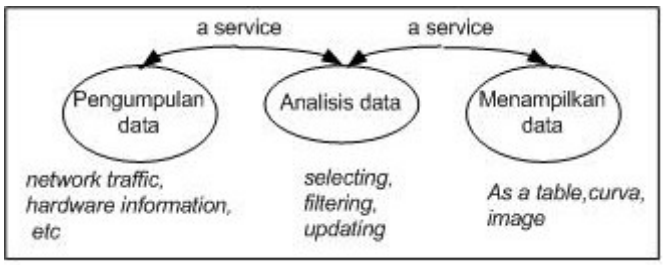
- Memonitor statistik yang penting seperti CPU, *Memory I/O*, dan penggunaan *network* dari masing-masing atau semua *node*.
- *Browse* konfigurasi sistem dari masing-masing *node* melalui *single point.*

*Cluster* ini belum mendukung sistem *monitoring*, untuk itu penulis akan mengembangkan aplikasi sistem *monitoring* pada *cluster*.

## 2.2 Sistem *monitoring*

Sistem *monitoring* merupakan suatu proses untuk mengumpulkan data dari berbagai sumber daya. Biasanya data yang dikumpulkan merupakan data yang *real time.* Secara garis besar tahapan dalam sebuah sistem *monitoring* terbagi ke dalam tiga proses besar seperti yang terlihat pada gambar 2.3 , yaitu:

1. proses di dalam pengumpulan data *monitoring*
2. proses di dalam analisis data *monitoring*
3. proses di dalam menampilkan data hasil *montoring*




Gambar 2.3 Proses dalam sistem *monitoring*

Aksi yang terjadi di antara proses-proses dalam sebuah sistem *monitoring* adalah berbentuk *service*, yaitu suatu proses yang terus-menerus berjalan pada interval waktu tertentu. Proses-proses yang terjadi pada suatu sistem monitoring dimulai dari pengumpulan data seperti data dari *network traffic*, *hardware information*, dan lain-lain yang kemudian data tersebut dianalisis pada proses analisis data dan pada akhirnya data tersebut akan ditampilkan.

Pada beberapa aplikasi sistem *monitoring*, akses benar-benar dibatasi dari *local host* terminal saja. Pertanyaannya apakah bisa dilakukan *monitoring* dari jarak jauh, dimana semua data yang dikumpulkan dari terminal komputer yang

berada di lokasi berbeda dengan instrumennya misalnya dengan menggunakan jaringan LAN (*Local Area Network*) atau bahkan *internet*. Untuk menjalankan sistem *monitoring* yang seperti ini sangat memungkinkan sekali dapat dilakukan dengan menggunakan *interface* program yang dapat menjembatani pengguna melalui *web browser* pada *remote* terminal. *Interface* program ini disebut CGI (*Common Gateway Interface*) yang biasanya tersedia pada *linux*.

## 2.3 SNMP (*Simple Network Management Protocol*)[6][7][9]

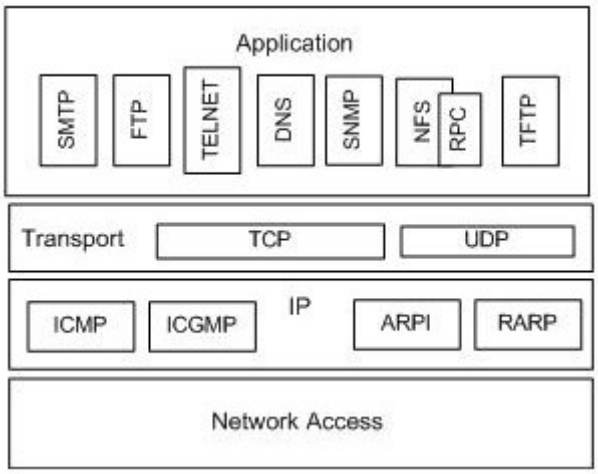
*Simple Network Management Protocol* (SNMP) adalah protokol yang digunakan untuk manajemen jaringan, seperti memonitor suatu peralatan *network* (misalnya *router*), peralatan komputer, dan *device* lain seperti *UPSs*.

Protokol ini dibutuhkan untuk membantu para *administrator* jaringan memonitor dan mengawasi jaringan. SNMP bukanlah perangkat lunak untuk melakukan menajemen jaringan, melainkan protokol ini menjadi basis pembuatan perangkat lunak manajemen jaringan. Tanpa SNMP, manajemen jaringan harus dilakukan dengan membuat aplikasi khusus untuk manajemen jaringan setiap jenis komponen jaringan dari setiap *vendor*.

SNMP memberikan kerangka manajemen standar untuk setiap *vendor* komponen jaringan dan pengembang aplikasi manajemen jaringan. Hasilnya adalah aplikasi manajemen jaringan yang mengimplementasikan SNMP dapat mengawasi dan mengontrol semua perangkat yang juga mengimplementasikan SNMP, meskipun perangkat-perangkat tersebut berasal dari vendor yang berbeda. Karena pada umumnya SNMP ini digunakan untuk memonitor *router* dan *host-host* di *internet*, maka protokol ini sangat sesuai sekali digunakan untuk aplikasi sistem *monitoring* yang dikerjakan dalam tugas akhir ini. SNMP sekarang ini terdiri dari 3 versi yaitu SNMP*v1*, SNMP *v2c* dan yang terakhir adalah SNMP *v3*. *agent* SNMP yang digunakan dalam tugas akhir ini adalah *Net*-SNMP *5.2.1* yang sudah mendungkung ketiga versi SNMP tersebut.

### 2.3.1 *Overview* TCP/IP [5],[9]

TCP/IP (*Transmision Control Protocol/Internet Protokol*) adalah sekumpulan protokol yang didesain untuk melakukan fungsi-fungsi komunikasi data pada jaringan WAN (*Wide Area Network*). TCP/IP terdiri dari sekumpulan protokol yang masing-masing bertanggungjawab atas bagian tertentu dalam komunikasi data. IP merupakan inti dari TCP/IP dan merupakan protokol terpenting dalam *internet layer*. IP menyediakan pelayanan pengiriman paket elementer dimana jaringan TCP/IP dibangun.




Gambar 2.4 Model TCP/IP[5]

Fungsi dari IP (*internet* protokol) adalah sebagai *routing* datagram ke *remote host*, dimana IP melewatkan data antara *network access layer* dan *host to host transport layer*. SNMP terletak pada *application layer*. Sebagai protokol *transport* SNMP biasanya menggunakan UDP (*User Datagram Protocol*) karena protokol ini relatif lebih efektif dalam pemakaian *bandwith*. Sebenarnya TCP juga dapat digunakan sebagai *transport layer* SNMP, akan tetapi karena karena protokol TCP cukup rumit dan memerlukan sejumlah memori dan sumber daya CPU maka lebih dianjurkan protokol UDP.

### 2.3.2 Komponen SNMP[1]

Komponen-komponen SNMP dapat dibagi menjadi dua kelompok, yaitu :

1. Manejer jaringan

   Manejer jaringan adalah sebuah komputer yang terhubung ke perangkat melalui jaringan komputer dan menjalankan perangkat lunak manajemen jaringan. Perangkat lunak tersebut mengawasi dan mengontrol jaringan dengan mengirimkan dan menerima pesan-pesan SNMP ke dan dari perangkat jaringan.

2. Perangkat jaringan

   Perangkat jaringan adalah perangkat yang dapat diawasi dan dikontrol dengan menggunakan SNMP. Agar perangkat tersebut dapat dimonitor dengan SNMP, maka peralatan tersebut harus memiliki aplikasi yang dapat menerima dan mengirim kembali pesan-pesan SNMP.

Sekelompok manejer dan perangkat jaringan disebut satu kelompok administratif jika terikat secara administratif atau fisik. Untuk membedakan satu kelompok administratif dengan yang lain, dibuat konsep ***community***. Sekelompok manajer jaringan dan perangkat jaringan yang termasuk dalam suatu kelompok administratif diberi nama yang dapat membedakannya dari kelompok administratif yang lain. Dalam istilah SNMP, kelompok administratif yang memiliki nama tersebut disebut suatu ***community*****.** Pengertian tentang konsep ***community*** ini dibutuhkan saat membahas pesan-pesan SNMP.

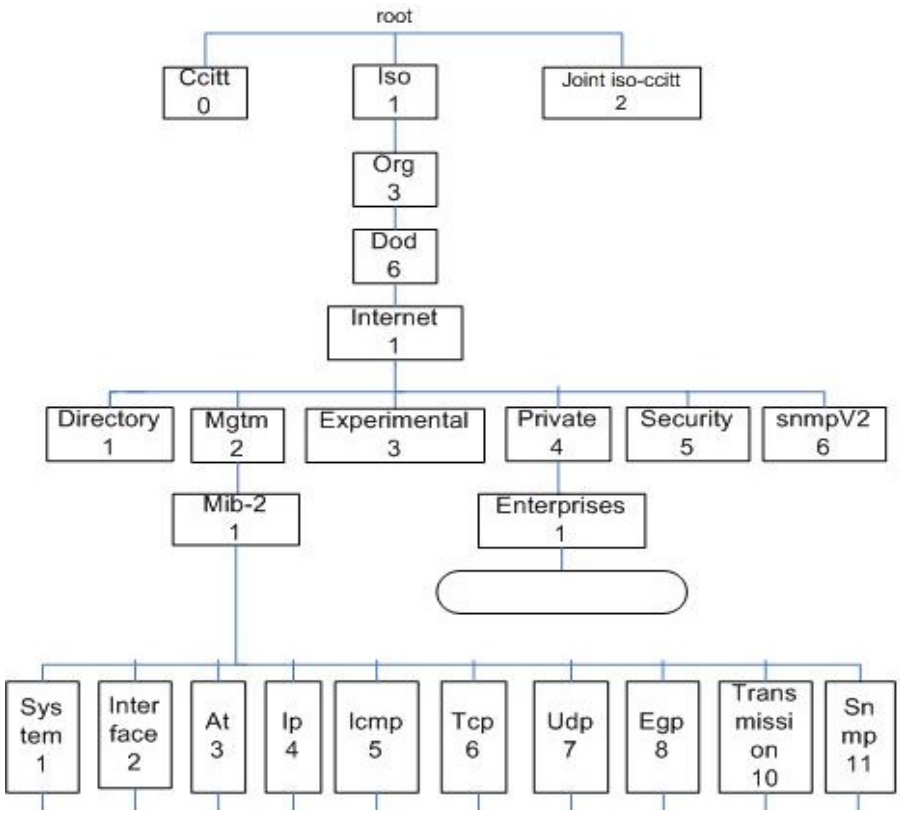
### 2.3.3 MIB ( *Management Information Base)*[9]

MIB adalah kumpulan data-data informasi manajemen yang diorganisasikan dalam sebuah struktur data. Setiap data memiliki data identitas dan nilai yang unik. Nilai setiap data harus sesuai dengan tipe dari data tersebut. Sebuah aplikasi manajemen jaringan melakukan *monitoring* dengan cara melihat dan mengubah nilai dari data-data tertentu.

Dalam SNMP, data informasi manajemen disebut *object* dan tipe data disebut *sintaks*. Tipe data yang paling mendasar dalam SNMP adalah *integer* atau *octet string*.

### 2.3.3.1 Struktur *ISO* dan *CCITT object* MIB [1][5]

Untuk memberikan identitas yang unik untuk setiap MIB, maka *International Organization for Standardization* (ISO) dan *International Telegraph and Telephone Consultative Committee* (CCITT) membuat struktur informasi dalam sebuah *global tree*, dimana setiap *object* memiliki nama unik berupa sederet bilangan asli yang dipisahkan oleh titik. *Global naming tree* tersebut dapat dilihat pada gambar *2.5* dibawah ini.




Gambar *2.5 The tree of object identifier* [9]

### 2.3.3.2 MIB standar *internet* [8], [9]

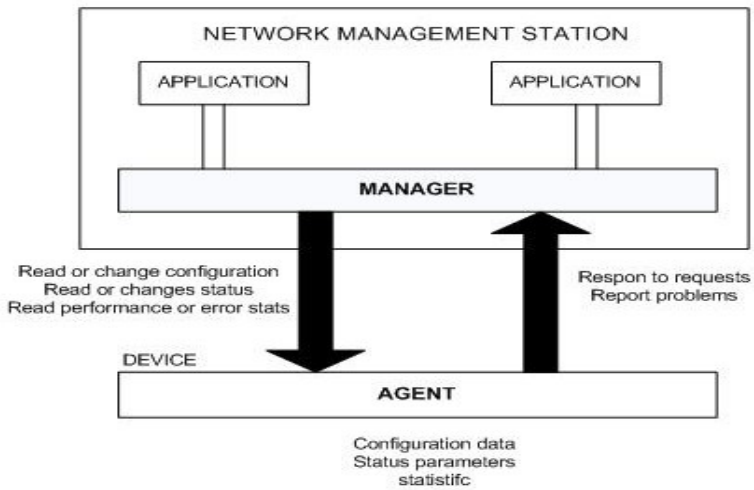
Sekarang ini MIB-MIB standar internet terletak dibawah identitas **1.3.6.1.** Dari gambar 2.4 dapat dilihat bahwa ada enam node yang didefinisikan dibawah internet yaitu: *director* ***(1.3.6.1.1)***, *mgmt* ***(1.3.6.1.2)*** , *experimental****(1.3.6.1.3)***, *private****(1.3.6.1.4)***, *security****(1.3.6.1.5)***, *dan* SNMP*V2****(1.3.6.1.6)***

*Object-object* tersebut adalah akar untuk *Object-object* dibawahnya. *Object-object* manajemen jaringan adalah **mgmt**, tepatnya dibawah *object* **MIB-2** yang posisinya tepat dibawah **mgmt** dengan identitas **1.3.6.1.2.1**[4]. Dari gambar *2.5* terdapat sepuluh *object* yang sering disebut sebagai MIB-II (MIB versi 2) dan dapat digunakan untuk memonitor dan mengontrol perangkat jaringan pada umumnya.

*Object* ***private*** yang berada dibawah *object internet* merupakan *object* yang menjadi akar pohon identitas untuk berbagai organisasi atau perusahaan selain organisasi standar yang ingin memiliki identitasnya sendiri. Identitas tersebut kemudian dapat menjadi akar dari *object-object* yang akan dibuat sendiri oleh organisasi atau perusahaan itu. Semua organisasi dan perusahaan yang mendaftar akan diberikan sebuah nomor identitas dibawah *object* ***enterprises*** (**1.3.1.4.1**) yang letaknya dibawah *object* ***private***.

### 2.3.4 *Agent* SNMP

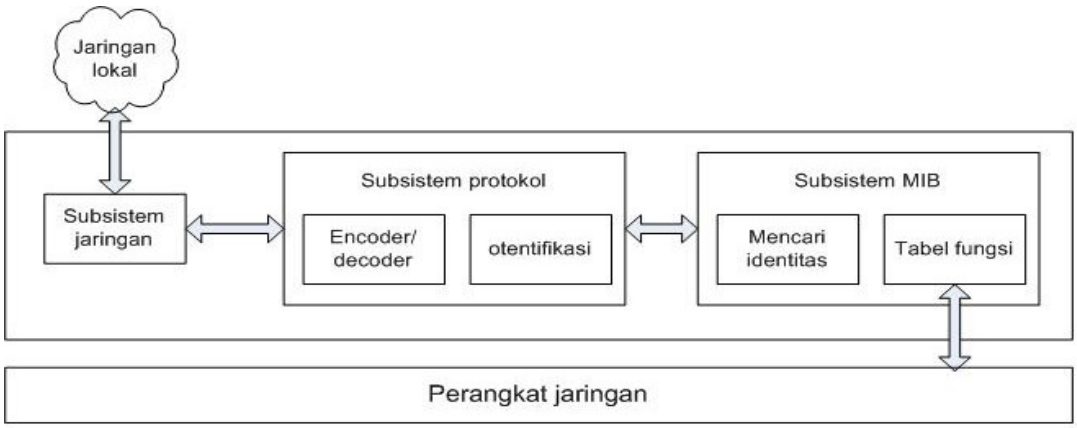
Untuk membuat suatu perangkat yang dapat dimonitor dengan SNMP, harus dibuat aplikasi yang disebut *agent* SNMP. *Agent* SNMP adalah sebuah aplikasi yang berjalan di perangkat jaringan dan bertugas menjawab pesan-pesan SNMP dan mengirimkan pesan SNMP tentang suatu kejadian di perangkat tersebut.




Gambar 2.6 Interaksi antara *manager* dan *agent*[9]

*Agent* menerima masukan pesan dari *manager*. Pesan ini meminta *request* untuk membaca atau menulis data pada *device*. Kemudian *agent* membawa *request* tersebut dan mengirimkan kembali respon. *Agent* tidak selalu harus menunggu untuk dimintai informasi. Ketika suatu masalah yang signifikan terjadi, *agent* mengirim suatu *notification message* yang disebut *trap* kepada satu *manager* atau lebih.

Untuk membuat suatu *agent* SNMP, perlu diketahui subsistem-subsistem yang ada pada *agent* SNMP. Gambar 2.7 memperlihatkan subsistem-subsistem yang ada pada *agent* SNMP[7].




Gambar 2.7 subsistem dalam *agent* SNMP

dari gambar 2.7 dapat didefinisikan subsistem-subsistem yang ada dalam *agent* SNMP, yaitu:

- Subsistem jaringan

  Subsistem ini berfungsi untuk menghubungkan *agent* SNMP dengan jaringan komputer. Jika subsistem ini meerima pesan SNMP, maka pesan tersebut akan diberikan kepada subsistem protokol . setelah diproses, subsistem protokol akan memberikan pesan SNMP yang harus dikirimksn oleh subsistem jaringan.

- Subsistem protokol

  Subsistem ini melakukan dua hal, yaitu: *encoding/decoding* dan otentifikasi. E*ncoding/decoding* mengubah pesan SNMP yang diterima sesuai aturan pengkodean *BER* (*Basic Encoding Rule*).

Sedangkan otentifikasi mengecek apakah pesan SNMP yang diterima tersebut otentik. Pada SNMPv1, otentifikasi dilakukan hanya dengan mengecek nama *community* yang ada dalam pesan SNMP.

- Subsistem MIB

  Subsisten ini melakukan dua hal, yaitu: mencari identitas *object* yang diminta, kemudian memanggil fungsi tersebut. Pencarian identitas *object* dilakukan sesuai jenis pesannya. Sedangkan fungsi yang dipanggil adalah fungsi yang mengakses parameter-parameter sistem yang berhubungan dengan *object* yang diminta.

## 2.4 RRDTOOL(*Round Robin Database*) [3]

RRDTools adalah suatu database dengan struktur yang sudah jelas di dalam melakukan analisis data *monitoring*. Pada tool ini analisis data berupa rataan (*AVERAGE*), nilai maksimum (*MAX*), nilai minimum (*MIN*), dan nilai terakhir (*LAST*) pada suatu interval tertentu yang telah ditetapkan sebelumnya.

Proses pengumpulan data yang terjadi dengan memanfaatkan salah satu metode di dalam RRDTools, yaitu update database. Metode tersebut dapat berguna apabila digabungkan dengan service di dalam sistem *monitoring* sehingga data yang tersimpan dalam database merupakan sample data yang lengkap dalam kurun waktu tertentu.

Proses pembentukan gambar grafik sebagai hasil dari *monitoring* adalah dengan memanfaatkan satu dari dua metode yang mungkin, yaitu *graph* atau *fetch*. Dengan menggunakan *graph* maka akan terbentuk gambar grafik, sedangkan menggunakan *fetch* akan diperoleh data mentah. Namun, kedua metode tersebut juga harus digabungkan dengan *service* yang ada di dalam sebuah sistem *monitoring* karena proses pemanggilan metode *graph* maupun metode *fetch* berdasarkan kurun waktu tertentu dan dengan interval waktu yang dimiliki oleh sistem *monitoring* maka akan di peroleh hasil *monitoring* yang terbaru untuk data yang terbaru juga.

# BAB III
# PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai perancangan dari aplikasi sistem *monitoring* untuk analisa sistem kerja dan performansinya apakah mengganggu sistem *cluster* itu sendiri atau tidak.

## 3.1 Identifikasi Kebutuhan Sistem

Sebelum membangun sistem, aktifitas untuk mengidentifikasi kebutuhan dari sistem yang akan dikembangkan harus dilakukan. Hal ini bertujuan agar sistem yang dikembangkan dapat bekerja sesuai dengan tujuan dan kebutuhan pemakainya.

### 3.1.1 Kebutuhan Fungsional (*Functional Requirement*)

- Sistem dijalankan pada *application server*
- Fungsi Produk

  Fungsi sistem yang dikembangkan adalah untuk mem-visualkan proses *monitoring* dan menampilkan informasi sumberdaya *hardware*, jaringan dari *cluster* berupa grafik. Fasilitas yang diberikan pada sistem ini adalah:

  1. *user*
     - *Monitoring cluster*

       Berisi informasi sumber daya *hardware cluster*
  2. administrator
     - *monitoring cluster*

       Berisi informasi sumber daya *hardware cluster* dan penampilan grafik secara *default*
     - manajemen *cluster*

       Berisi informasi mengenai pengaturan *node*, item *monitoring*, sistem kontrol, dan *service* untuk mengaktifkan agen SNMP
     - manajemen *user*

berisi tentang pengaturan *user* yang menggunakan fasilitas sistem *monitoring*.

- Karakteristik Pemakai

  Pengguna sistem ini adalah *user* biasa dan administrator, dimana pengguna biasa memiliki kemampuan untuk mengakses *internet* dan administraror memiliki kemampuan menggunakan sistem yang sedang dikembangkan.
- Administrator dapat melakukan *restore database*

### 3.1.2 Kebutuhan Tambahan (*Nonfunctional Requirement*)

Tools yang dibutuhkan dalam pengembangan sistem adalah sebagai berikut:

1. Sistem Operasi

   Sistem Operasi yang digunakan sebagai *platform* dari sistem *monitoring* untuk *open cluster* ini adalah linux. Pemilihan sistem operasi ini didasarkan berapa pertimbangan sebagai brikut:
   - Sistem *cluster* yang dikembangkan menggunakan sistem operasi linux
   - Stabilitas
   - *Multitasking* dan *multiuser*
   - *Open Source*
   - Dukungan yang baik untuk aplikasi-aplikasi yang berbasiskan TCP/IP
   - Gratis
2. *Web server*

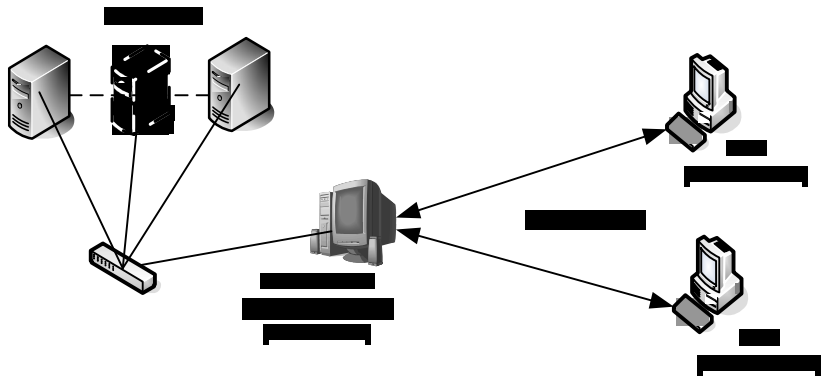
   *Web server* digunakan untuk melayani permintaan halaman yang digunakan pada aplikasi-aplikasi berbasis *Web*. Perangkat lunak *web server* yang digunakan adalah *Apache2* dengan dukungan *Python*/CGI/HTML untuk kebutuhan pemrograman *scripting web*-nya.
3. Jaringan TCP/IP

4. *python* 2.3, scripting untuk membuat modul dan *interface* web
5. *net-snmp*-5.2.1.tar.gz ,sebagai *agent* SNMP
6. *rrdtool*-1[1].0.49.tar.gz, sebagai database untuk menyimpan data *monitoring* yang dibutuhkan.
7. *parport_ctrl software,* sebagai sarana untuk sistem kontrol.

## 3.2 Arsitektur Sistem

### 3.2.1 Arsitektur Fungsional

Secara fungsional, akan dijelaskan mengenai entitas-entitas yang akan terlibat dengan pembangunan sistem *monitoring*.



Gambar 3.1 Arsitektur fungsional

Entitas-entitas yang terlibat dalam sistem adalah:

1. *Master node/Monitoring server* (*webserver*)

   Entitas ini merupakan entitas yang akan dikembangkan dalam tugas akhir ini. *Monitoring server* akan mencatat data-data pada saat *user* melakukan pengaksesan dan menjalankan *job*nya dan *master node* sebagai *server* dari *node client.*

2. *User* (*web browser*)

   *User* merupakan entitas yang memiliki hak akses terhadap data-data yang terdapat pada *monitoring server*. *User* dapat membaca data-data tersebut melalui *web browser*. *User* dalam hal ini terdiri dari *user* biasa dan

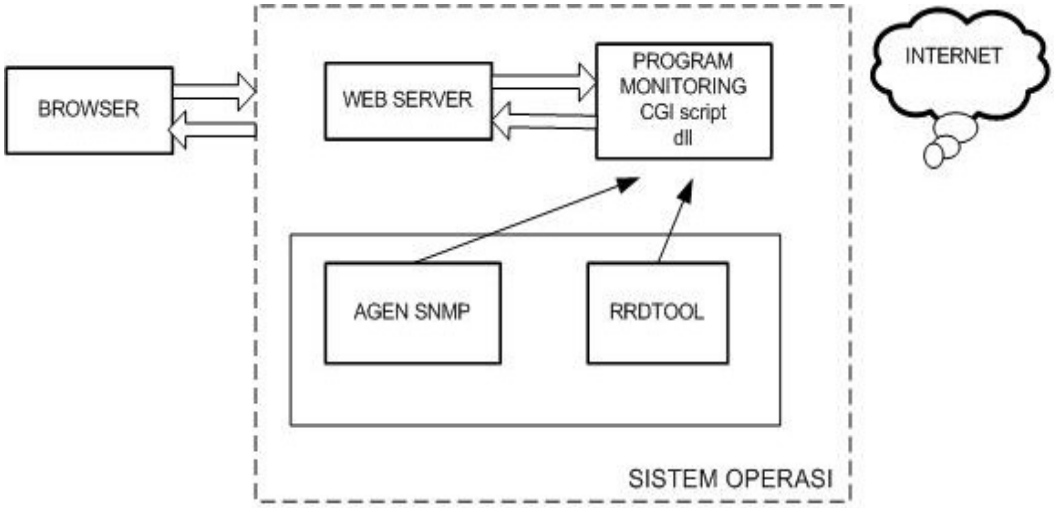
administrator yang masing-masing memiliki perbedaan fasilitas dalam pengaksesan. Hal ini akan dibahas pada bab IV tentang implementasi dan analisa sistem.

3. *Client node*

   Merupakan sistem yang yang akan dimonitor sumber daya *hardware* maupun *software*nya.

### 3.2.2 Arsitektur Komunikasi Perangkat Lunak

Untuk lebih menjelaskan mengenai sistem yang akan dikembangkan, maka selanjutnya akan dijelaskan spesifikasi sistem dipandang dari perangkat lunak. Gambar berikut menjelaskan bagaimana perangkat lunak yang berada di komputer dapat berkomunikasi sehingga membentuk suatu sistem.




Gambar 3.2 Arsitektur komunikasi perangkat lunak

## 3.3 Perancangan sistem

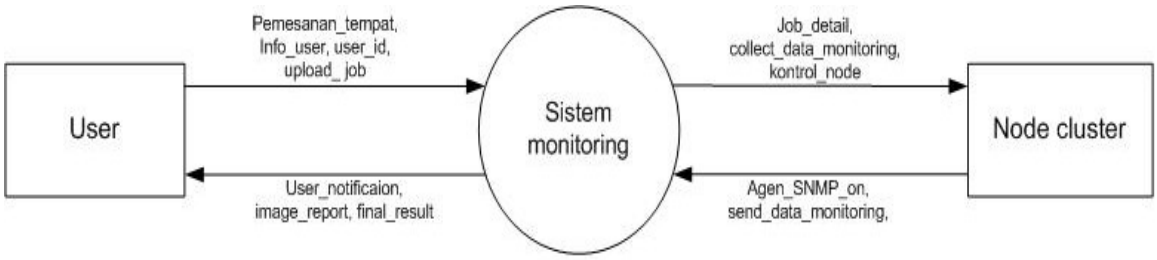
Perancangan sistem meliputi :

2. perancangan proses yang dibutuhkan untuk membangun aplikasi sistem *monitoring* ini.
3. perancangan tampilan layar untuk mempresentasikan *interface* (antarmuka) yang akan dijumpai oleh pengguna (*user*) dalam menggunakan layanan ini.

### 3.3.1 Perancangan proses

#### 3.3.1.1 *Context* diagram

Diagram konteks sistem adalah sebagai berikut:

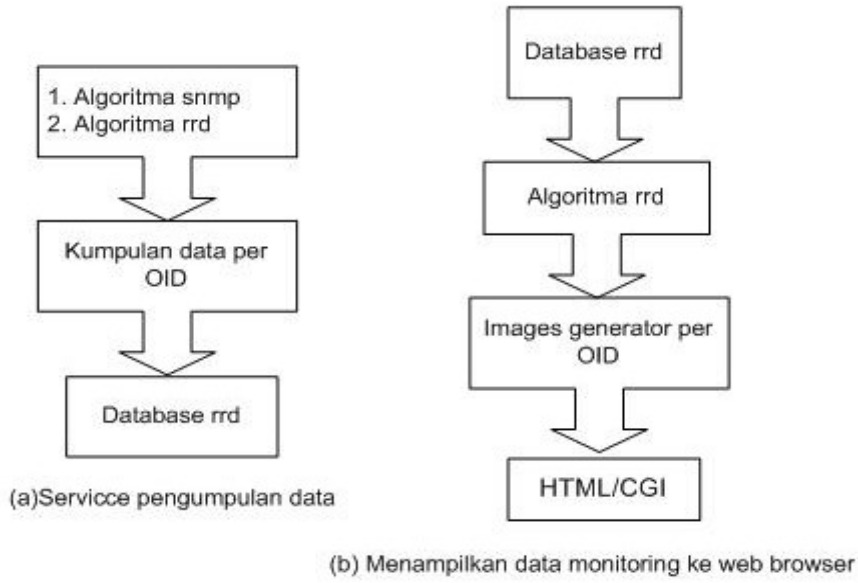



Gambar 3.3 *Context* diagram

#### 3.3.1.2 Proses pembentukan dan menampilkan data *monitoring* ke *web* browser.

*Monitoring* sistem ini dikembangkan dengan menggunakan SNMP (*Simple Network Management Protocol*). Untuk itu pada perancangan sistem dibutuhkan agen SNMP untuk memperoleh data yang dibutuhkan kemudian data tersebut akan dikumpulkan ke dalam suatu database dan dibuat grafiknya dengan menggunakan Rrdtool (*Round Robin Database*), dimana perangkat lunak ini yang akan bertindak sebagai database untuk mengumpulkan data *monitoring*.

Hubungan antara pengambilan data melalui SNMP, pengumpulan database dengan RRDtool hingga ditampilkan dalam bentuk grafik pada *web*browser dapat dilihat pada gambar 3.4 dibawah ini :




Gambar 3.4 Perancangan SNMP dan rrdtool

Keterangan :

(a) Service pengumpulan data

- Algoritma snmp

  *"Pesan SNMP –v 2c" + self.__host + "–c" + self.__password + OID"*

  Pesan SNMP yang digunakan adalah *snmpwalk* untuk menampilkan semua data yang dapat dibaca oleh SNMP pada jaringan komputer, *snmpget* menampilkan nilai dari satu OID, *snmpgetnext* menampilkan nilai selanjutnya setelah OID sebelumnya dan *snmpstatus* untuk melihat status dari OID.

  -v 2c menunjukkan versi SNMP yang digunakan yaitu versi 2c

  *self.__host* adalah nama atau nomor ip dari *host*, dan *self.__password* adalah nama dari *community* pada *host* tersebut.

  OID adalah obyek *identifier* yang digunakan untuk membedakan satu obyek dengan obyek lainnya. Identifikasi obyek-obyek manajemen jaringan sebagai obyek *monitoring* dapat dilihat pada lampiran B.

- Algoritma rrd

Pada tahapan ini algoritma rrd yang digunakan adalah create dan update.

-Tahap 1 : create rrdtool (rrd_name, item_type, interval)

*Command* create = *"rrdtool create" + self.__RRD_DIRS_DATA + rrd_name + ".rrd" + "–step" + interval *

*DS : item_name :item_type : interval*2 :u:u*
*RRA:AVERAGE:0.5:1*

.
.

*Command* diatas akan menghasilkan database dengan nama rrd_name.rrd, dimana *data source* (DS) yang merupakan tipe dari suatu nilai data dinamakan item_name dengan item_type bergantung pada data yang dihasilkan, interval waktunya adalah 300 detik (*default). Data source* yang ada antara lain :

- GAUGE, biasanya digunakan untuk suhu.
- COUNTER, biasanya digunakan untuk data yang terus naik, seperti *counter Inoctets* pada *router*. *Counter* tidak akan pernah menurun kecuali terjadi *overflow*.
- DERIVE, dipergunakan untuk menyimpan nilai turunan dari nilai terakhir yang telah dimasukkan dengan nilai yang baru dari *data-source*. Cara kerjanya sama dengan *counter*, tetapi tanpa pengecekan *overflow*.
- ABSOLUTE, dipergunakan untuk *counter* yang cenderung me-*reset* nilai yang diperoleh ketika membaca input..

RRD menyimpan data kedalam RRA (*Round Robin Archieves*), dimana masing-masing RRA menyimpan sejumlah data dari semua *data- source* yang telah didefinisikan.

`Self.__RRD_DIRS DATA` adalah direktori yang akan menyimpan database yang dihasilkan. Dalam hal ini disimpan pada /data/system/rrd.

-Tahap 2 : update rrdtool (rrd_name)

*Command* update = *"rrdtool update " + self.__RRD_DIRS_DATA + rrd_name + ".rrd N"*

Pada tahap ini data pada database rrd di-*update* terus secara *real time* sehingga akan didapat kumpulan data per OID yang dimonitor. Semua data mentah yang terdapat pada database rrd ini dapat dilihat dengan menggunakan perintah *fetch*. Data-data ini yang kemudian dibuat grafiknya.

(b) Menampilkan data *monitoring* ke *web browser*

Pada proses menampilkan data *monitoring* ke *web browser* dibuat dalam bentuk grafik.

Algoritma rrd pada tahapan ini : rrdtool graph (name_rrd)

*Command* rrdtool graph : *"rrdtool graph" + self.__RRD_DIRS_IMAGE + rrd_name + ".png" + *

Masukan : data dari database rrd

Keluaran : images generator per OID yan diberi nama *rrd_name.png*

Pada tahap ini data yang telah disimpan dalam database rrd akan diperlihatkan dalam bentuk grafik dan akan disimpan pada `self.__RRD_DIRS_IMAGE` yaitu direktori /images/rrd.

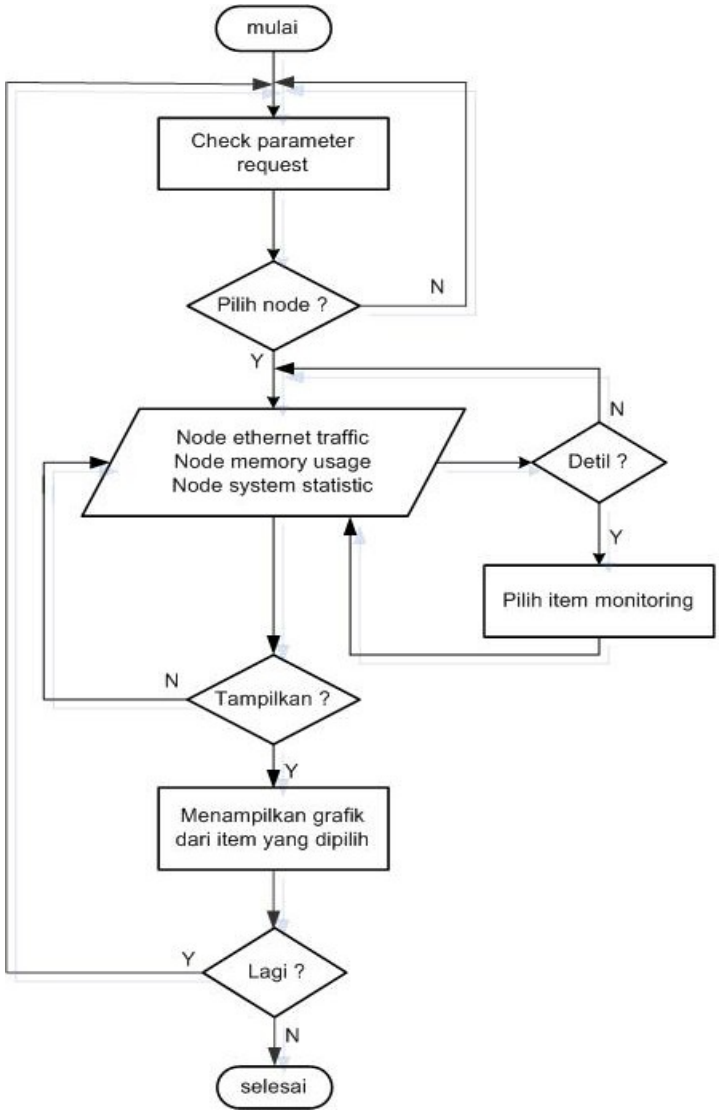
Dalam satu gafik ditampilkan data maksimum (MAX), minimum(MIN) dan rata-rata (AVERAGE).

Keluaran *images generator* per OID ini yang kemudian ditampilkan pada *web browser* dengan menggunakan HTML/CGI.

### 3.3.1.3 Diagram alir proses

**A. proses *monitoring* sistem**

Perancangan proses sistem *monitoring* yang dapat diakses *user* melalui media *web*dapat dilihat pada diagram alir dibawah ini :




Gambar 3.5 Diagram alir proses *monitoring cluster*

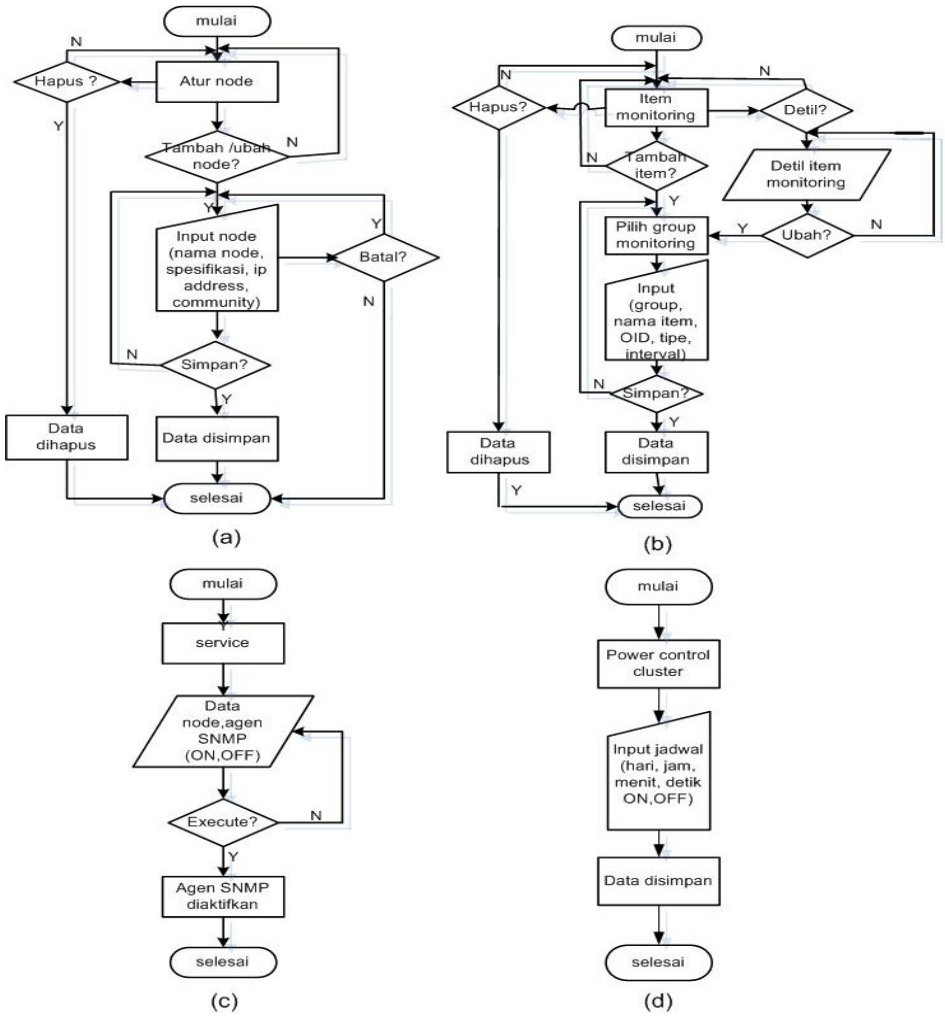
Proses *monitoring cluster* secara umum adalah sebagai berikut:

- Informasi yang hendak dimonitor terlebih dulu di cek parameter *request*nya kemudian dipilih *node* yang ingin dimonitor.
- Untuk setiap *node* terdapat 3 group monitor yaitu *ethernet traffic*, *memory usage* dan sistem *statistic*, dimana unuk setiap group monitor terdapat item yang dapat dipilih sendiri oleh *user*. Secara detil item yang dapat dimonitor pada sistem ini dapat dilihat pada lampiran C.
- Item *monitoring* yang dipilih akan ditampilkan grafiknya dan akan tersimpan ke dalam file dengan nama sesuai dengan nama *node_group monitor_item*

*monitoring*, sehingga apabila *user* ingin menampilkannya secara *default*, maka tinggal memilih kembali file yang ada.

### B. Proses manajemen *cluster*

Proses manajemen *cluster* terdiri dari proses pengaturan *node*, item *monitoring*, power kontrol dan service. Untuk lebih jelasnya dapat dilihat pada flowchart dibawah ini :




Gambar 3.6 Diagram alir sistem manajemen *cluster*

(a) Proses atur *node*

(b) Proses *item monitoring*

(c) Proses *service*

(d) Power kontrol

(a) Proses atur *node*

proses atur *node* dibuat untuk mempermudah apabila ada penambahan atau perubahan *node* pada *cluster*. Proses atur *node* secara umum adalah sebagai berikut:

- Untuk menambah atau mengubah *node* diperlukan data dari *node* yang ingin ditambah ataupun diubah. Inputan yang dibutuhkan adalah nama *node*, spesifikasi, *ip address* dan *community*.
- *Ip address* dan *community* adalah inputan yang mutlak harus diisi. Dimana *Ip address* merupakan alamat ip dari *node* yang bersangkutan (misal: 10.10.10.1) dan *community* adalah password yang dibuat pada agen SNMP untuk *Ip address/network* pada *node*.
- Apabila semua data yang dimasukkan benar maka data akan disimpan pada `self.__DIRS_DATA_SYSTEM + self.__NODES_FILE` (/data/system/*node*s.dat).

(b) Proses atur item *monitoring*

Seperti halnya pada pengaturan *node*, pengaturan pada item *monitoring* juga dibuat untuk mempermudah penambahan atau pengubahan item *monitoring*. Prosesnya adalah sebagai berikut:

- Data yang dibutuhkan untuk menambah atau mengubah item *monitoring* adalah group monitor, nama group, nama item, tipe, OID dan interval.
- Untuk tipe,OID dan interval harus sesuai dengan ketentuan yang ada pada SNMP. Hal ini dapat dilihat pada lampiran C.
- Semua data akan disimpan dalam `self.__DIRS_DATA_SYSTEM + self.__MONITORING_ITEM_FILE` (/data/system/*monitoring*_item.dat).

(c) proses service

Proses service dibuat untuk eksekusi agen SNMP, dimana setiap *node* memiliki agen sendiri. Proses instalasi agen SNMP dapat dilihat pada lampiran A.

- Eksekusi pada service ini terdiri dari `startservice, stopservice, process,` dan `isServiceRunning`
- `Startservice` untuk mengaktifkan agen SNMP pada *node* yang dipilih dan `stopservice` untuk mematikan agen SNMP.
- Service akan aktif jika data pada *node* yang bersangkutan sesuai dan dapat dijalankan prosesnya.

(d) power kontrol

power kontol dibuat untuk mematikan atau menyalakan *power supply node cluster* secara langsung. Proses yang terjadi pada bagian ini adalah :

- *Schedule* atau pengaturan jadwal *on-off node*
- Penyimpanan jadwal *on-off node*

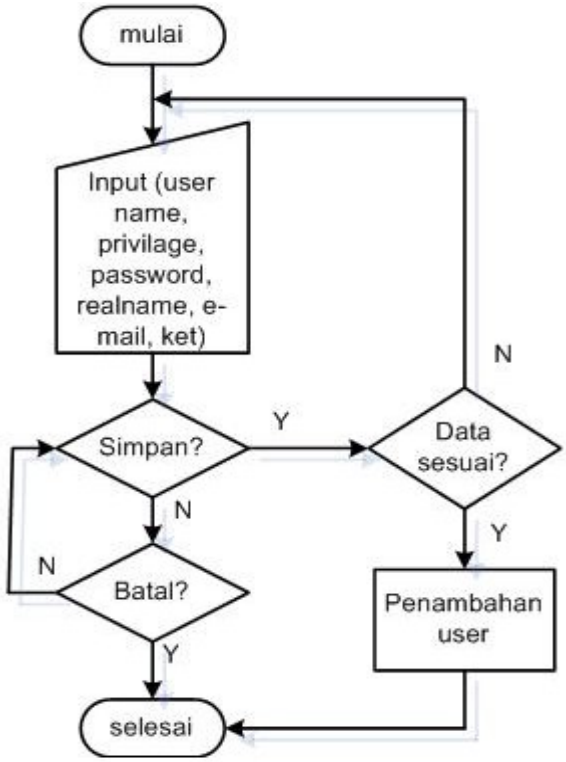
**C. Proses manajemen *user***

Manajemen *user* meliputi proses penambahan *user*, dimana *user* yang telah mendaftarkan diri dan telah memenuhi persyaratan akan dimasukkan sehingga *user* memperoleh account untuk menggunakan fasilitas ini.

Proses manajemen *user* seperti yang terlihat pada gambar 3.7 digunakan untuk penambahan atau pembatalan *user*. Untuk penambahan usr langkah-langkah yang harus dilakukan adalah :

- Untuk menambah *user* dibutuhkan inputan nama *user*, privilage, password, real name, e-mail,dan keterangan (optional)
- Privilage digunakan untuk membedakan *user* biasa dengan administrator, dimana sebagai *user* biasa privilagenya adalah *user*, sedangkan untuk administrator digunakan admin.

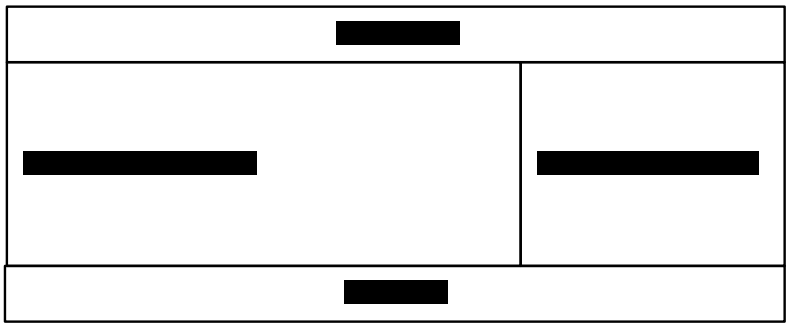
  Apabila data yang dimasukkan sesuai maka diproses penambahan *user*




Gambar 3.7 Diagram alir proses manajemen *user*

### 3.3.2 Perancangan tampilan layar

Perancangan tampilan layar dilakukan agar pengguna mendapatkan kemudahan dalam memmfaatkan fasilitas-fasilitas yang disediakan oleh sistem. Untuk memudahkan navigasi antar halaman, maka dirancang suatu desain global untuk menjaga konsistensi tampilan layar. Tugas akhir ini menggunakan media *webbrowser* sebagai terminal bagi pengguna. Untuk itu halaman *web*dari sistem ini terdiri dari bagian-bagian sebagai berikut:



Gambar 3.8 Tampilan layar

1. Bagian atas (header), terdiri dari:
    - Frame logo yang berisi logo dari layanan
2. Bagian utama (main), terdiri dari:
    - Frame utama, yang berisi keluaran atau layanan utama dari program (sistem) yang dibuat. Frame ini akan menampilkan proses yang dipilih oleh *user* dari frame menu.
    - Frame menu, yang berisi menu-menu yang dapat diakses oleh *user*. Frame ini berubah sesuai dengan *user* yang menggunakan, dimana terdapat perbedaan menu untuk *user* biasa dan administrator.
3. Bagian bawah (footer), terdiri dari:
    - Frame hak cipta, yang berisi publikasi dan versi dari program.

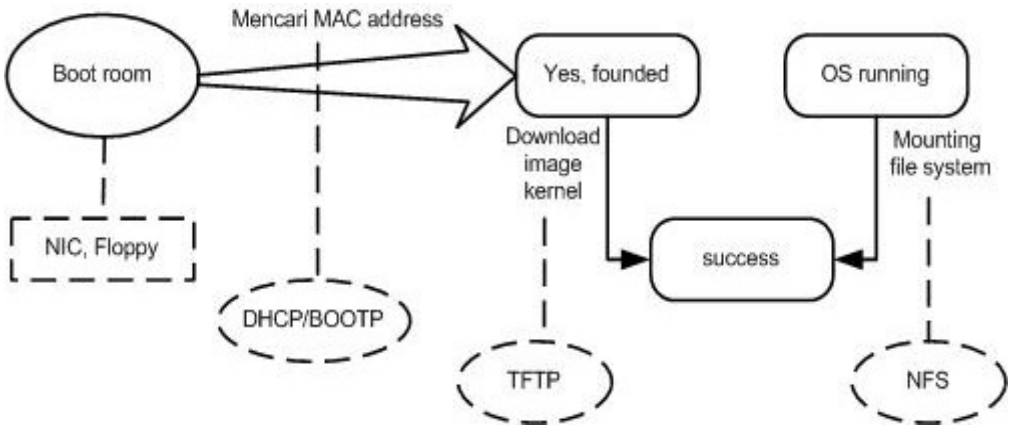
# BAB IV
# IMPLEMENTASI DAN ANALISA SISTEM

## 4.1 Implementasi sistem

Sebelum sistem diimplementasikan, dipastikan dulu semua *tools* yang dibutuhkan untuk menjalankan sistem *monitoring* ini telah *ter*in*stall* pada komputer *web server*. Komputer yang dipergunakan sebagai *node cluster* terdiri dari dua tipe, yaitu *node diskless workstation* dan *node non diskless workstation*.

Untuk *node* yang non *diskless* perlu di *set*ting pada masing-masing *node* secara manual agar *agent* SNMP dapat berjalan. Sedangkan untuk *node* yang *diskless*, pada dasarnya sama dengan *node* yang *non diskless* dimana masing-masing *node* harus mendukung *agent* SNMP. Akan tetapi karena *node diskless* tidak memiliki *hardisk* maka sistem operasi yang dijalankan dengan menggunakan FAI (*Fully Automatic* In*stallation*), atau LTSP (*Linux Termina*l *Server Pages*).

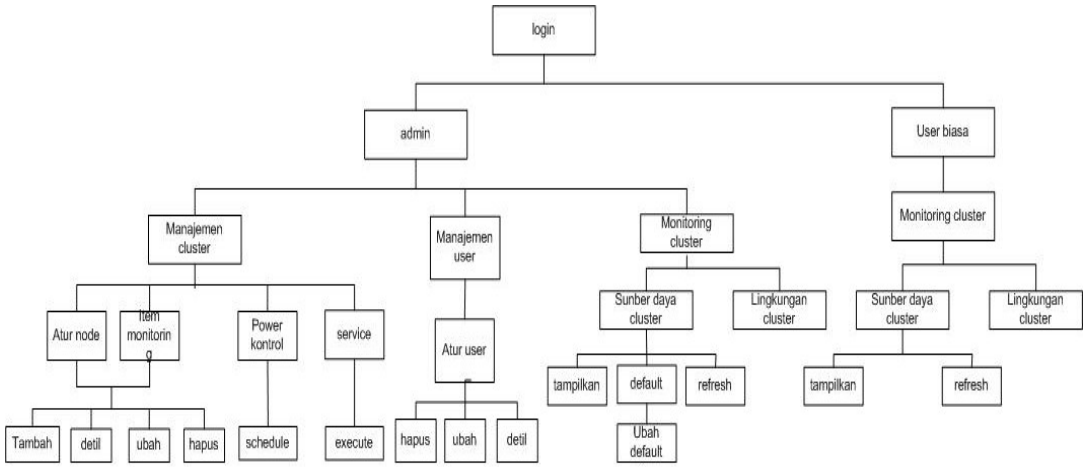
*Node diskless* yang menggunakan FAI dilakukan dengan cara menginstall *agent* SNMP yang dibutuhkan melalui *master node* sedangkan LTSP digunakan LTSP4.0 yang sudah mendukung *agent* SNMP untuk masing-masing *node clientnya*. Implementasi *diskless* sistem seperti terlihat pada gambar 4.1.




Gambar 4.1 Implementasi *diskless system*

### 4.1.1 Implementasi in*terface* sistem *monitoring*

Aplikasi ini disajikan melalui media *web browser*. Secara umum struktur menu dapat dilihat pada gambar 4.2 dibawah ini.




Gambar 4.2 Implementasi struktur menu

Karena sistem ini merupakan perkembangan dari sistem sebelumnya yang telah menangani masalah *book*in*g user* (proses pendaftaran dan pemesanan tempat), maka penulis lebih menekankan implementasi pada sistem *monitoring*nya mulai dari proses *log*in.

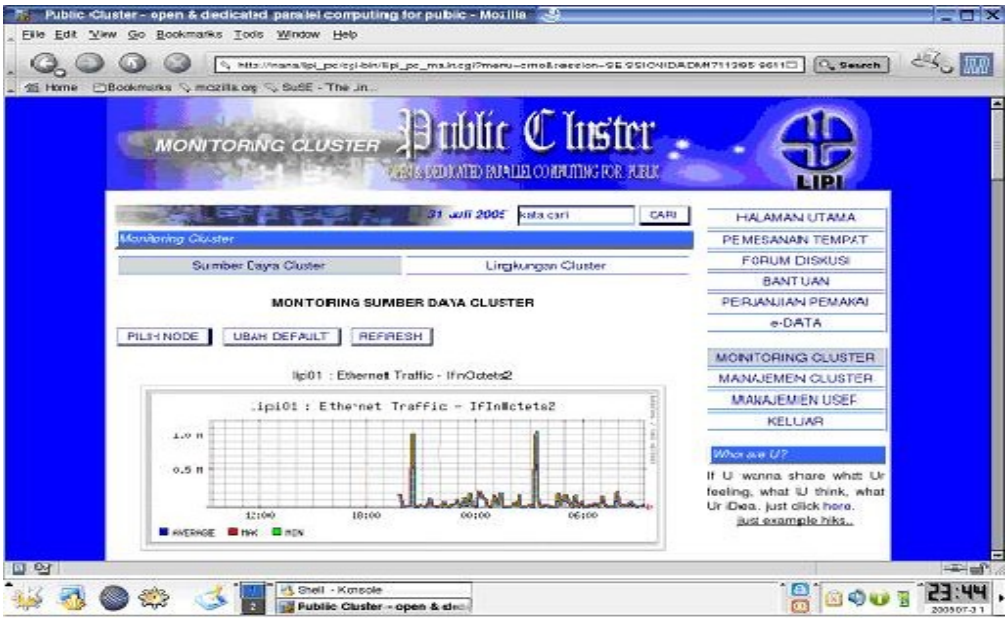
Proses *log*in pada struktur menu implementasi sistem *monitoring open cluster* terdiri atas dua *user* yaitu login sebagai *user* biasa (*common user*) dan *log*in sebagai administrator (*adm*in *user*). *User* biasa adalah *user* yang memamfaatkan *open cluster* untuk menjalankan program paralelnya, sedangkan administrator adalah pihak pengelola sistem. Menu in*terface* untuk masing-masing *user* ini berbeda seperti dijelaskan berikut ini.

In*terface* untuk sistem *monitoring open cluster* adalah sebagai berikut:

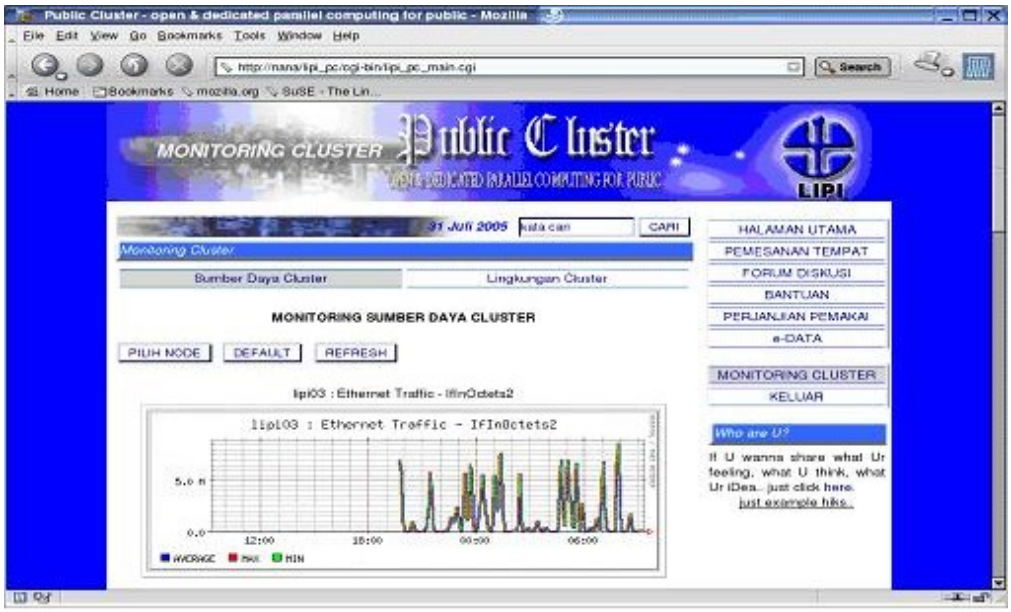
1. In*terface* untuk halaman *monitoring cluster*

   Halaman ini menampilkan hasil *monitoring* sumber daya *hardware* masing-masing *node* pada *cluster*. Terdapat dua in*terface* untuk halaman *monitoring cluster* sesuai dengan *privilage user* yang menggunakan sistem seperti terlihat pada gambar 4.3(a) in*terface* halaman *adm*in dan

4.3(b) in*terface* halaman *user* biasa. Perbedaannya hanya terletak pada *default* sistem, dimana *user* biasa tidak bisa mengubah grafik yang akan ditampilkan pada menu *default*.




Gambar 4.3(a) In*terface* halaman admin



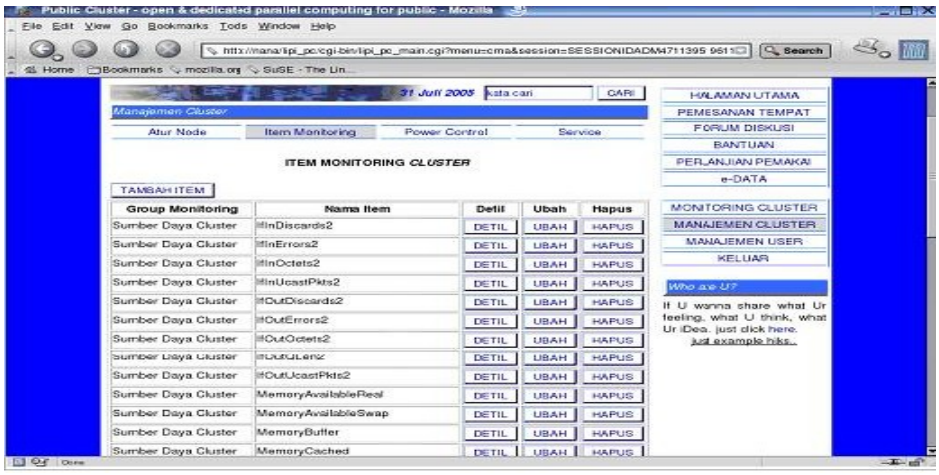

Gambar 4.3(b) In*terface* halaman *user* biasa

2. In*terface* halaman manajemen *cluster*

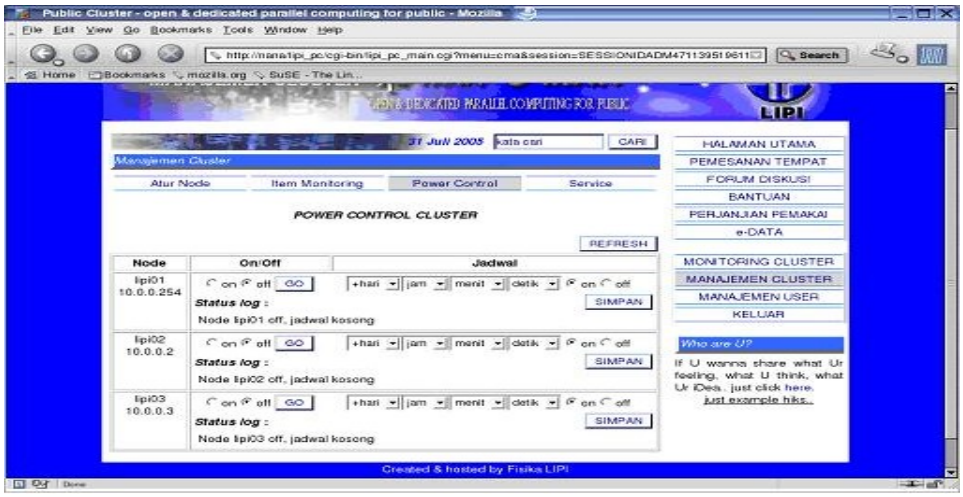
   Merupakan in*terface* untuk *adm*in *user* dimana pada halaman ini terdapat pengaturan *node* (gambar 4.4(a)), *item monitoring* (gambar 4.4(b)), *power* kontrol (gambar 4.4(c)), dan *service* (gambar 4.4(d)).

   - In*terface* pengaturan *node* ditujukana agar *adm*in *user* dapat mengubah, menghapus dan menambah *node-node* yang ada pada *cluster*.
   - in*terface item monitoring* menampilkan semua *item* sumber daya *hardware* yang dapat dimonitor. Pada halaman ini *adm*in *user* juga dapat menambah, mengubah dan menghapus *item monitoring*.
   - In*terface power* kontrol ditujukan agar *adm*in *user* dapat mematikan atau menghidupkan *node* secara langsung atau dijadwalkan. Sistem kontrol ini menggunakan *relay* yang dihubungkan ke *port parallel*.
   - In*terface service* ditujukan agar pengumpulan data *monitoring* dapat dilakukan, dimana *adm*in *user* dapat mematikan atau menghidupkan *service* suatu *node*.

Gambar 4.4(a) In*terface* pengaturan *node*




Gambar 4.4(b) In*terface item monitoring*

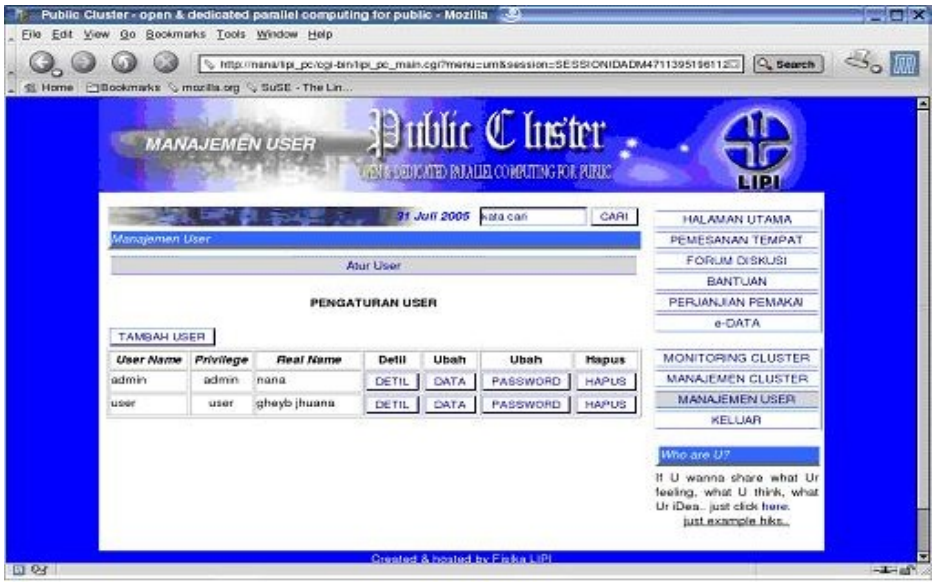



Gambar 4.4(c) In*terface power* kontrol


Gambar 4.4(d) In*terface power* kontrol

3. In*terface* halaman manajemen *user*

   In*terface* halaman manajemen *user* ini juga merupakan in*terface* untuk *adm*in *user* agar dapat mengatur *user* yang menjalankan sistem *monitoring*. Dalam hal ini *adm*in dapat menambah *user* baru, menghapus *user* yang ada dan mengubah data *user*. In*terface* halaman manajemen *user* dapat dilihat pada gambar 4.5.




Gambar 4.5 In*terface* halaman manajemen *user*

### 4.1.2 implementasi pada perangkat lunak

Dalam implementasi perangkat lunak sistem *monitoring* untuk *open cluster* terdapat beberapa *module* (*class*) yang merupakan kontrol untuk menjalankan prosedur apabila *user* mengakses in*terface* sistem. *Module-module* tersebut adalah *module lipi_pc_snmp, module lipi_pc_rr*d, *module lipi_pc_services.* Dan untuk *power* kontrol menggunakan *parport_ctrl* dengan module *lipi_pc_schedule* dan *lipi_pc_parallel_port*

- **Fungsi-fungsi yang dapat diakses dari modul**

Tabel 4.1 Fungsi –fungsi pada module

| Module (class) | Function |
|---|---|
| lipi_pc_snmp.py | ➢ isAlive<br>➢ isValidOID4*monitoring*(OID)<br>➢ value(OID) |
| lipi_pc_rrd.py | ➢ crate(rrd_name, *item*_type, interval)<br>➢ update(rrd_name)<br>➢ graph(rrd_name) |
| lipi_pc_services.py | ➢ startService<br>➢ stopService<br>➢ isServiceRunning |
| lipi_pc_parallel_port.py | ➢ send(*node*_name, port_id, isON)<br>➢ sendScherdule(*param) |
| lipi_pc_schedule.py | ➢ startSchedule(*node*_name, port_id, isON)<br>➢ stopSchedule(*node*_name) |

- **kegunaan**

Tabel 4.2 Kegunaan dari fungsi –fungsi pada *module*

| Class/Function | Keterangan |
|---|---|
| Lipi_pc_snmp.py | In*terface* internal sistem *monitoring* ke *agent* SNMP |
| ➢ isAlive | Untuk melakukan pengecekan apakah *agent* SNMP sedang *runn*i*ng* atau tidak. Fungsi ini me-*return boolean*. |
| ➢ isValidOID4*monitoring*(OID) | Untuk melakukan pengecekan apakah suatu obyek *identifier* (OID) ada atau tidak (*valid*). Fungsi ini me-*return boolean* dan memamfaatkan pesan *get parameter* dari SNMP. |
| ➢ value(OID) | Untuk memperoleh nilai dari suatu OID dengan memamfaatkan *get parameter* SNMP. Fungsi ini me-*return tuple* yaitu pasangan status dan nilai yang didapat. |
| lipi_pc_rrd.py | In*terface* internal sistem dengan RRD*tool* |
| ➢ create(name_rrd, *item*_type, interval) | Untuk membuat database RRD dengan nama name_RRd, tipe dari obyek *monitoring item*_type dan interval pengumpulan data *monitoring*. Menggunakan algoritma RRd*create* seperti yang sudah dijelaskan pada sub-bab 3.3.1.2 |
| ➢ update (name_rrd) | Untuk menambahkan nilai baru pada suatu database RRD dengan nama name_RRd. Menggunakan algoritma RRd*update* seperti yang sudah dijelaskan pada sub-bab 3.3.1.2 |
| ➢ graph(name_rrd) | Untuk mengenerate grafik dari database RRD dengan nama name_RRd. Menggunakan algoritma RRd*graph* seperti yang sudah dijelaskan pada sub-bab 3.3.1.2 |
| lipi_pc_services.py | Untuk menjalankan servis pada masing-masing *node cluster*. |

| | |
|---|---|
| ➢ startService<br><br>➢ stopService<br><br>➢ is ServiceRunning | Untuk menghidupkan suatu servis dengan membuat *fork*ing dan menjalankan PID ke sebuah file *temporary*<br>Untuk mematikan suatu servis dengan cara meng-*kiIl* PID dari servis yang bersesuaian<br>Untuk mengecek apakah servis pada suatu *node* sedang berjalan atau tidak. |
| lipi_pc_parallel_port.py<br>➢ send(*node*_name, port_id, isON)<br><br>➢ sendSchedule(*param) | In*terface* in*ternal* sistem dengan *parport_ctrl*<br>Untuk mematikan dan menyalakan *node cluster* dengan nama *node_name* dan nomor *port port_id*<br><br>Untuk mematikan atau menyalakan suatu *node* sesuai dengan jadwal yang ditentukan. |
| lipi_pc_schedule.py<br><br>➢ startchedule(*node*_name, port_id, isON)<br><br>➢ stopSchedule (*node*_name) | In*terface* untuk mematikan atau menyalakan *node* berdasarkan jadwal.<br><br>Untuk memulai menyalakan atau mematikan *node* dengan menggunakan fungsi IsendSchedule(*param)<br><br>Untuk menghapus jadwal suatu *node* |

#### 4.1.2.1 Security dalam penyimpanan data sistem *monitoring*

Seperti yang sudah dijelaskan pada sub-bab 3.3.1.2, penyimpanan data pada sistem *monitoring* ini adalah dengan menggunakan *file*. Hal ini disebabkan karena data yang dibutuhkan tidak terlalu besar. Semua *file* diletakkan pada satu *directory* yang tidak dapat diakses dari luar oleh orang lain. Hal ini dilakukan dengan menambahkan *.htaccess* pada *apache web server*.

*Fil-file* yang digunakan untuk menyimpan data pada sistem *monitoring* ini adalah sebagai berikut:

- untuk menyimpan semua data *user* yang dapat menggunakan fasilitas ini disimpan pada *file .secret*. data yang disimpan meliputi *user_log*in, *user_privilage, user_password, user_name, email_address* dan keterangan.
- Data *node* disimpan pada *file nodes.dat* yang meliputi *node_name, node_spesification, IP_address, community_name, port_id, power_status, schedule_st*ing, *service_status* dan *agentt_status.*
- Data *item monitoring* disimpan pada *file monitoring_item.dat.* meliputi *group_monitoring, object_name, object_*OID, *object_type,* in*terval_value.*
- Database pengumpulan data *item monitoring* disimpan pada satu *folder* yaitu RRd yang meliputi semua data *node* yang dimonitor. Semua file pada folder ini memiliki *ekstension.*RRd untuk data dan *ekstension.png* untuk grafik hasil *monitoring* yang menunjukkan data dan grafik dari masing-masing *item* yang *dimonitoring*.

### 4.1.2 Ujicoba sistem

Ujicoba sistem *monitoring* ini tidak dilakukan dengan menggunakan *open cluster* LIPI secara langsung karena belum stabilnya sistem tersebut. Ujicoba yang dilakukan dengan menggunakan suatu sistem yang memiliki spesifikasi yang hampir mirip dengan sistem *cluster* yang sebenarnya.

Sistem ini dibangun dengan menggunakan tiga buah komputer dimana satu komputer sebagai *web server* dan *master node* dan dua komputer lainnya sebagai *client node*. Ketiga komputer ini dihubungkan dengan *switch* sehinga membentuk jaringan LAN (*local Area Network*). Kedua komputer *client* adalah *node diskless* karena untuk implementasi *node non diskless* dianggap sama seperti *master node*. Spesifikasi dari masing-masing *node* adalah sebagai berikut:

Tabel 4.3 Spesifikasi sistem ujicoba

| *node* | Spesifikasi | Ip address |
|---|---|---|

| | | |
|---|---|---|
| Lipi01,*master node* | AMD Athlon 1Ghz, 256 SDRAM PC 133, 100 Mbps fast *ethernet* adapter RTL 8139 | 10.0.0.254 |
| Lipi02,*diskless* | Intel Pentium III 933 Mhz, 100 Mbps fast *ethernet* adapter RTL 8139 | 10.0.0.2 |
| Lipi03,*diskless* | Intel (r) x86, 130,96 KB RAM , 100 Mbps fast *ethernet* adapter RTL 8139 | 10.0.0.3 |

Sistem ini dijalankan dengan menggunakan LTSP, dimana komputer lipi01 dan lipi02 *boot*ing dengan menggunakan *floppy* dan *mendownload kernel* LTSP dari lipi01. proses *boot*ing ini seperti terlihat pada gambar 4.1. selanjutnya dilakukan ujicoba sehingga menghasilkan grafik *monitoring* untuk masing-masing *node* (lampiran D) dan log sistem bahwa telah diperoleh data *monitoring* untuk masing-masing *node* (lampiran E). Ujicoba dilakukan selama kurang lebih sepuluh jam mulai dari pukul 19:41:5 hingga pukul 9:17:8.

## 4.2 Analisa sistem

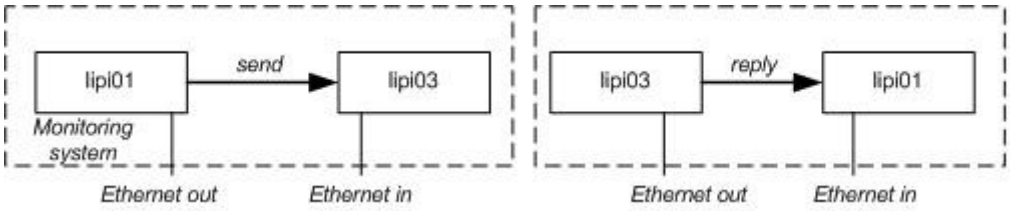
### 4.2.1 Analisis performance sistem *monitoring*

Analisis *performance* yang dilakukan adalah dengan menunjukkan bahwa selama sistem *monitoring* sedang *runn*ing tanpa menggunakan sebagian besar dari sumber daya pada komputer *web server*, khususnya ketika *service* pengumpul data sedang *runn*ing, sistem *monitoring* ini tidak mengganggu kinerja dari *server*. Untuk melihat *performance* tersebut dilakukan ujicoba dengan menjalankan sistem *monitoring*, sekaligus untuk melihat apakah sistem *monitoring* ini sesuai atau tidak.

- **Analisis keakuratan data sistem monitoring**

Untuk melihat *performance* dari *traffic* masukan/keluaran *master node* (lipi01) bila dibandingkan dengan *node client* (lipi03), dimana kedua komputer dalam keadaan *idle* dan pada lipi01 hanya aplikasi sistem *monitoring* yang bekerja. Ilustrasinya, seperti gambar 4.6 dibawah. Pada saat kedua komputer

tersebut dalam keadaan *idle*, lipi01 mengirimkan *request* untuk meminta data *monitoring* dari lipi03 sehingga *ethernet out* bekerja. Kemudian lipi03 akan menerima *request* tersebut sehingga *ethernet in* bekerja. Pada proses *reply* hal yang sama juga terjadi, dimana lipi03 mengirim kembali *request* yang diminta oleh lipi01, sehingga ethernet out pada lipi03 dan thernet in pada lipi01 bekerja. Seharusnya *ethernet out* lipi01 sama dengan *ethernet in* lipi03, demikian juga sebaliknya.



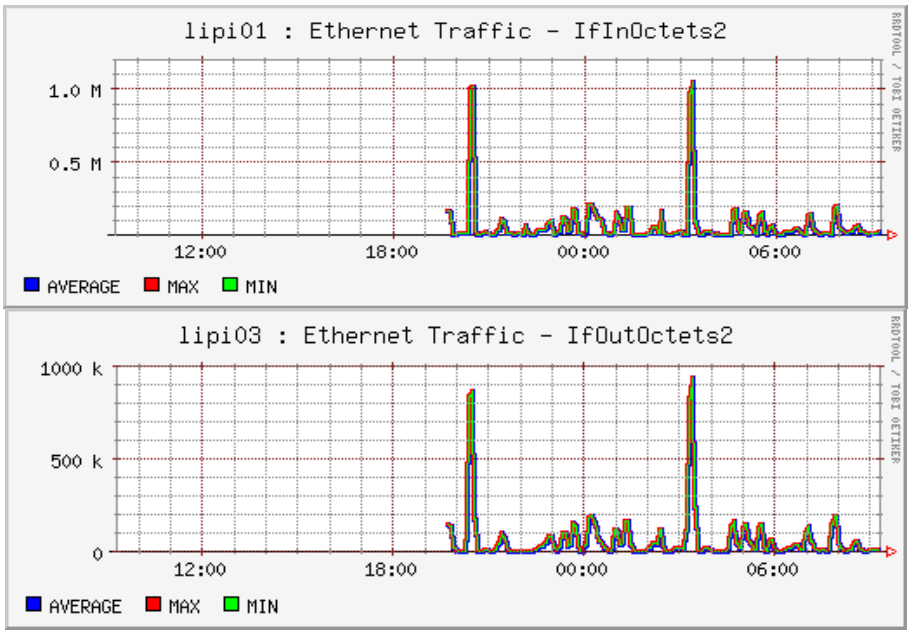

Gambar 4.6 Ilustrasi kerja *ethernet*

Pada gambar 4.6 dan 4.7 dibawah memperlihatkan bahwa *traffic* masukan yang diterima oleh lipi01 sesuai dengan *traffic* keluaran dari lipi03. begitu juga sebaliknya, hanya ada perbedaan nilai yang relatif kecil jika dilihat dari nilai pada tabel 4.4 yang merupakan data *monitoring* antara lipi01 dan lipi03. Sehingga dapat disimpulkan bahwa pada proses *monitoring* ini terdapat pengurangan jumlah paket yang terkirim. Hal ini bisa saja disebabkan karena adanya *loss* jaringan saat pengiriman data atau tidak semua data dapat dikirim kembali.

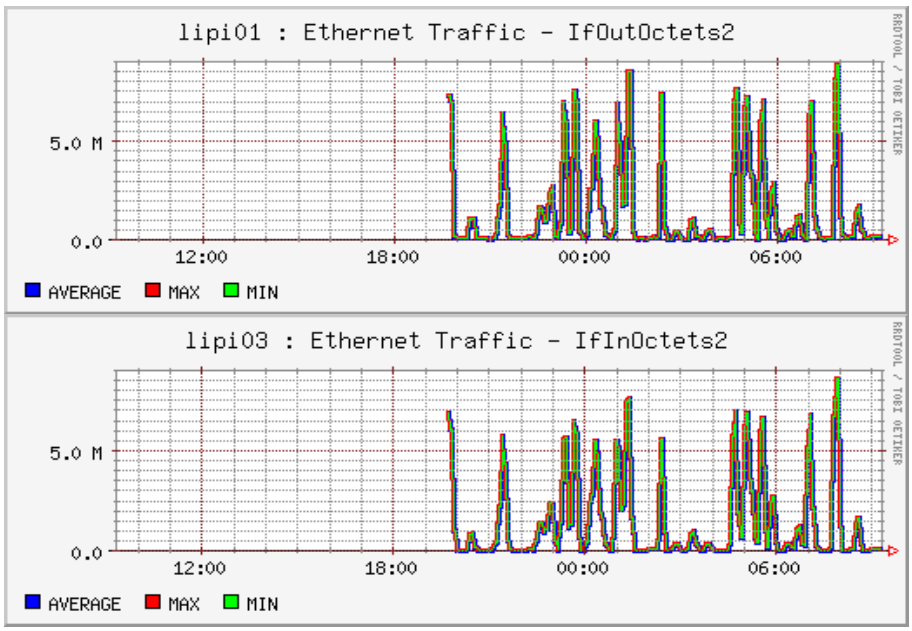
Tabel 4.4 Data hasil *monitoring ethernet traffic*

| lipi01 *Ethernet*InOctets (byte) | lipi03 *EthernetOut*Octets (byte) | lipi01 *EthernetOut*Octets (byte) | lipi03 *Ethernet*InOctets (byte) |
|---|---|---|---|
| 583652203 | 494906082 | 2796430520 | 2819437228 |
| 633269192 | 540454116 | 683900879 | 590410139 |
| 683900879 | 565659011 | 2237306440 | 1694069149 |
| 671541312 | 565659011 | 2268363932 | 1698967825 |
| 674295519 | 566538669 | 2301056321 | 1704139008 |
| 677774266 | 568328875 | 2332957212 | 1710543038 |
| 683159606 | 570051545 | 2370589935 | 1716870065 |

| | | | |
|---|---|---|---|
| 687502870 | 571173663 | 2406058465 | 1722111588 |
| 691125948 | 615422803 | 2438586802 | 2331974694 |
| 2771912476 | 2078099750 | 994388906 | 918539854 |
| 1298421822 | 1166545959 | 3098188133 | 2336883409 |

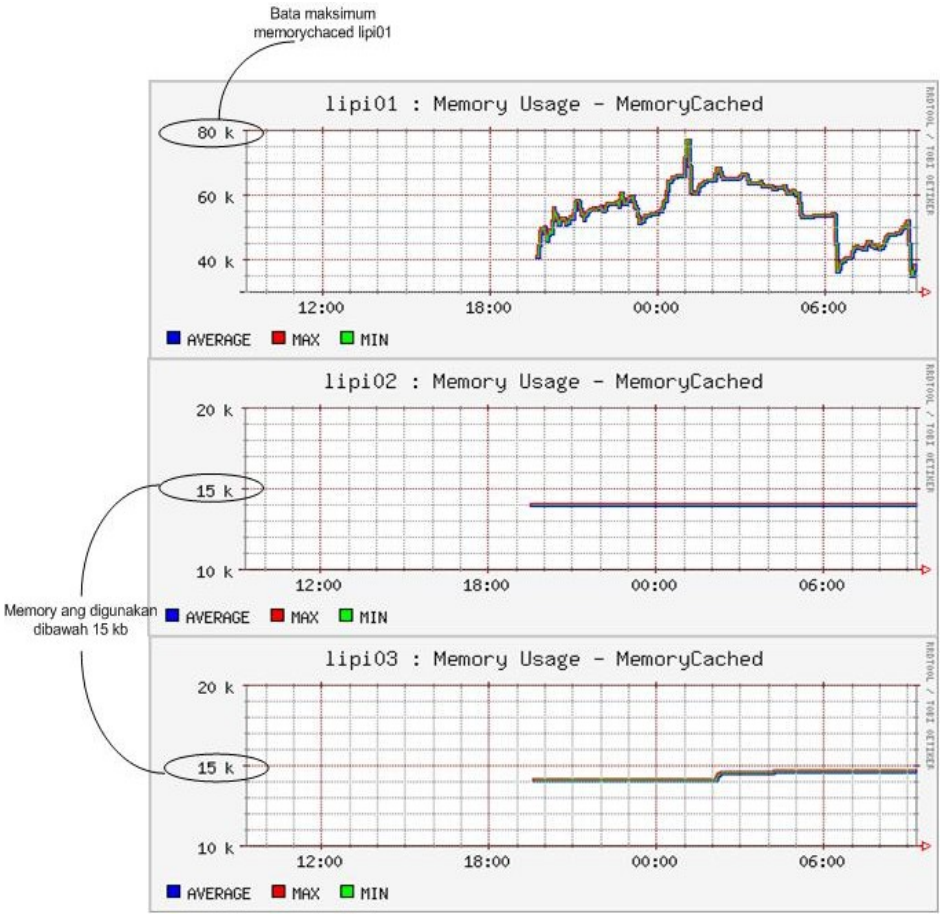



Gambar 4.7 *Ethernet traffic inoctets* lipi01dan *Ethernet trafficOutOctets* lipi03



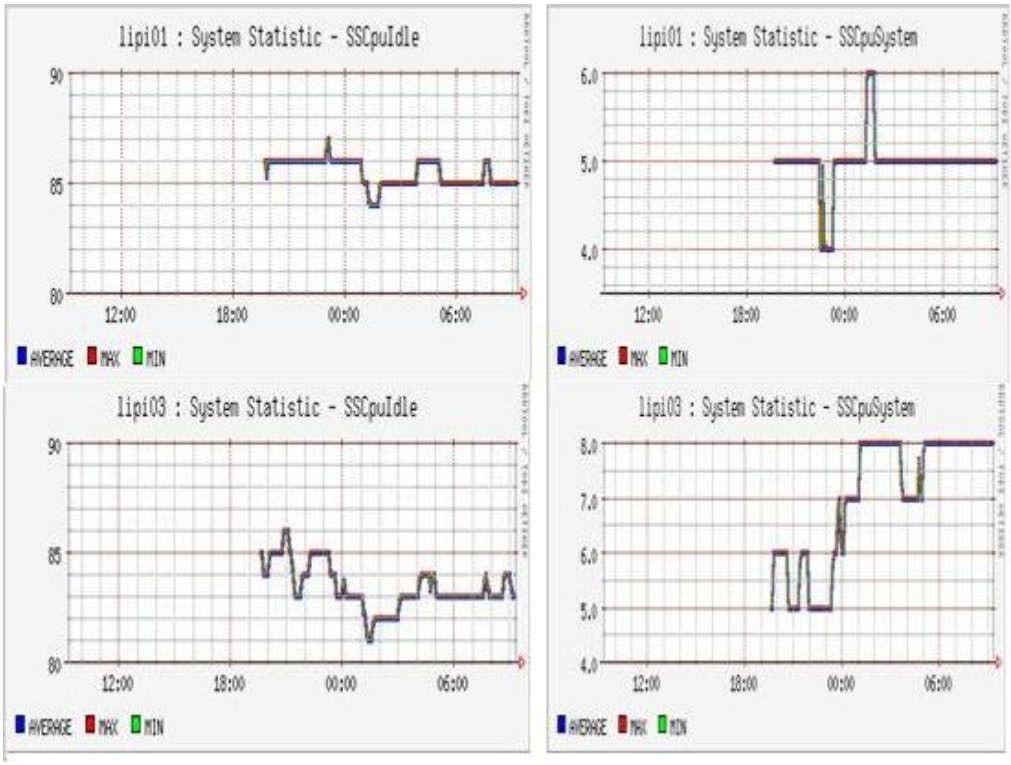

Gambar 4.8 *Ethernet trafficOutOctets* lipi01 dan *Ethernet traffic inoctets* lipi03

*Memory chaced* merupakan satu memory dalam prosesor untuk menjalankan semua perintah sesuai dengan batas maksimumnya. Pada ujicoba sistem *monitoring* ini terlihat bahwa *memory chaced* yang bekerja pada lipi01 adalah 80 KB sedangkan pada lipi02 dan lipi03 hanya digunakan dibawah 15 KB (gambar 4.9). *Memory chaced* yang bekerja pada *server* lebih besar karena pekerjaan yang dilakukan oleh *server* lebih banyak. Untuk melakukan suatu pekerjaan yang berat lebih cenderung digunakan *memory chaced* ini karena letaknya yang dekat dengan prosesor sehingga delay untuk pelaksanaan perintah cenderung tidak ada.




Gambar 4.9 *Memory chaced* lip01, lipi02, dan lipi03

Gambar 4.10 memperlihatkan keadaan *cpuidle dan cpusystem*, dimana lipi01 lebih stabil jika dibandingkan dengan *node client*nya. Hal ini bisa saja dikarenakan perbedaan *hardware* dari masing-masing komputer, dimana lipi01 memiliki spesifikasi memori dan prosesor dua kali lebih baik dibading lipi03. Komputer dengan spesifikasi prosesor lebih tinggi kerja memorinya akan lebih stabil dalam kondisi lingkungan yang sama (sistem operasi yang sama). Untuk itu komputer yang berperan sebagai *server* sangat membutuhkan spesifikasi yang tinggi karena semua proses dititik beratkan pada server.




Gambar 4.10 *System statistic* lip01 dan lipi03

Dari sisi perangkat lunak, Skema percobaan dilakukan hanya untuk menentukan seberapa cepat interval yang sebaiknya diberikan pada *service* pengumpul data yang sedang *runnin*g agar sistem ini tidak mengganggu kinerja dari *master node.* Dengan pengumpulan data setiap 5 menit, sistem *monitoring* ini dirasa sudah bisa mengumpulkan data *monitoring* yang lengkap.

*Dari hasil log* sistem (lampiran E) ada data *unknown.* Hal ini disebabkan karena dalam *data acquisition* sangat dimungkinkan bahwa tidak diperoleh suatu data untuk disimpan ke dalam RRD. Interval waktu yang diberikan secara *default* adalah 300 detik sehingga jika dalam interval waktu *2 (600 detik) tidak d*item*ukan data baru maka RRD*Tool* secara otomatis memasukkan nilai *UNKNOWN* ke dalam *database*.

- **Analisis akses sistem *monitoring* malalui *internet***

Dari sisi pengguna perlu dianalisa seberapa besar *bandwith* atau besarnya data yang dibutuhkan untuk mengakses sistem monitoring ini. Besar data yang dibutuhkan untuk melihat tampilan utama adalah 96.7 Kb. Untuk layar *monitoring* seperti yang terlihat pada gambar 4.3(a) dan 4.3(b), besar data tergantung dari jumlah *item* yang d*imonitoring*, dimana untuk setiap *item*nya bervariasi antara 2.6 Kb sampai 4.9 Kb. Misalnya untuk 12 *item monitoring* dibutuhkan 126.7 Kb untuk melihat semua tampilan layar, sedangkan untuk 6 *item* dibutuhkan sekitar 100 Kb.

Dari data tersebut dapat disimpulkan bahwa *interface* yang digunakan untuk sistem *monitoring* ini tidak terlalu besar. Misalkan saja akses *internet* yang memiliki kecepatan 5 kbps. Jadi untuk membuka halaman utama sistem ini hanya dibutuhkan waktu sekitar 20 detik, dan untuk melihat halaman *monitoring* yang memonitor sebanyak 12 *item* hanya dibutuhkan waktu sekitar 26 detik.

## 4.3 Kelemahan dan keunggulan sistem

Sistem *monitoring* untuk *open cluster* ini tentu saja memiliki kelebihan dan kekurangan yang lumrah dimiliki oleh sebuah sistem. Pada sub-bab ini penulis akan menguraikan keungulan dan kelemahan sistem *monitoring.*

### 4.3.1 Keunggulan

Sistem *monitoring* yang dikembangkan memiliki keunggulan dimana sistem ini dapat diakses melalui internet tanpa harus melihat langsung kondisi fisik dari sistem *cluster* itu sendiri. Beberapa keunggulan sistem adalah :

1. Apikasi sistem *monitoring* ini dapat diakses dengan menggunakan *web browser* baik itu *platform w*in*dows* atau *l*in*ux.*
2. Aplikasi ini memungkinkan *user* dengan mudah untuk mengamati permasalahan atau kondisi beban yang terlalu berat yang dialami masing-masing *node*.
3. Kemudahan untuk mengakses sistem *monitoring. User* yang menggunakan aplikasi ini dapat memilih sendiri sumber daya *cluster* yang ingin dimonitor.
4. Kemudahan untuk menambah, mengubah atau mengurangi *item monitoring*, *node*, *user*, yang semuanya diatur dalam manajemen *cluster* sehingga jika ada penambahan tidak perlu mengubah *source code* (perangkat lunak) sistem *monitoring*
5. Adanya *power kontrol* untuk mematikan dan menyalakan *power supply secara langsung* atau dengan penjadwalan.
6. Sistem *monitoring* ini dikembangkan dengan beberapa tools sehingga kinerja masing-masing tools juga sangat dibutuhkan.
   - Dengan menggunakan SNMP memungkinkan kita untuk memperoleh data *monitoring* mengenai jaringan pada komputer.
   - RRD*Tool* memiliki kelebihan sebagai berikut:
     1. *Data Acquisition*, di dalam *monitoring* suatu sistem diperlukan ketersediaan data pada interval waktu yang konstan. Namun, sayangnya kita tidak mungkin selalu mampu untuk mengambil data pada interval waktu yang tepat. Oleh karena itu, RRD*Tool* memberikan kemudahan di dalam melakukan *log* data dengan tidak terikat pada interval waktu tersebut. RRD*Tool* secara otomatis akan melakukan interpolasi nilai dari sumber data tersebut pada slot waktu terakhir *(latest official time-slot).*
     2. *Consolidation*, dengan menggunakan fungsi konsolidasi RRD*Tool* secaraotomatis akan melakukan analisis data ketika suatu data baru dimasukkan ke dalam RRD. Hal ini memberikan keuntungan bagi kita, seperti misalnya apabila kita menyimpan data dengan interval

waktu 1 menit. Maka akan memerlukan tempat di dalam *disk* yang tidak kecil apabila kita menginginkan suatu grafik yang merupakan hasil analisis data dalam kurun waktu 1 tahun.

3. *Round Rob*in *Archives (RRA),* memberikan jaminan bahwa ukuran dari RRD tidak akan mengalami pertambahan dan data yang lama secara otomatis akan dibuang. Data dengan *consolidation* yang sama akan disimpan ke dalam sebuah *RRA*.

### 4.3.2 Kelemahan

Informasi yang diperoleh memungkinkan *user* untuk mengurangi kesalah pahaman tentang kinerja *cluster*, namun aplikasi ini masih jauh dari sempurna, dimana masih banyak isu mengenai *monitoring* dan manajemen *cluster* yang belum dimiliki oleh sistem ini. Kelemahan yang dimiliki sistem adalah:

1. sistem tidak secara otomatis menampilkan grafik hasil *monitoring* yang terbaru karena akan membebani kinerja *web server*. Namun, untuk melihat grafik hasil *monitoring* yang terbaru maka *user* dapat meng-*klik* tombol *refresh* yang telah disediakan.
2. Karena database sistem *monitoring* hanya menggunakan *file* maka data yang dapat disimpan juga terbatas
3. *Loading* untuk *service* pengaktifan *agent* snmp memerlukan waktu yang sedikit lama diawal inisialisasi. Karena diperlukan koneksi ke masing-masing *node* untuk mengetahui apakah terdapat *agent* snmp atau tidak.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Adapun kesimpulan yang dapat dibuat dari penulisan tugas akhir ini adalah sebagai berikut :

1. Sistem monitoring berbasis web untuk open cluster ini telah berhasil diimplementasikan dengan menggunakan SNMP sebagai protokol pengumpul data monitoring dan database Round Robim (RRDtool) untuk analisis data monitoring dan menampilkan data hasil monitoring dalam bentuk grafik.
2. Dari hasil ujicoba, data yang cukup akurat dapat dilihat dari *Performance traffic* masukan/keluaran *master node* (lipi01) bila dibandingkan dengan *node client* (lipi03) pada proses *monitoring* ini sesuai dengan ilustrasi. Akan tetapi terdapat pengurangan jumlah paket yang terkirim. Hal ini bisa saja disebabkan karena adanya *loss* jaringan saat pengiriman data atau tidak semua data dapat dikirim kembali.
3. Dari segi *performance* memori dan cpu, terlihat jelas bahwa spesifikasi *hardware* sangat berpengaruh dalam kestabilan kerja prosesornya. *Server* yang memiliki spesifikasi *hardware* dua kali lebih bagus dari *node client* lebih stabil dan lebih optimal dalam pemakaian *memory chaced*nya.
4. User yang menggunakan aplikasi sistem *monitoring* ini tidak membutuhkan waktu lama untuk dapat melihat hasil monitoring, dimana untuk melihat 12 item yang dimonitor hanya dibutuhkan sekitar 26 detik dengan besar data 126.7 Kb (dengan asumsi kecepatan user untuk mengakses internet sebesar 5 Kbps).

## 5.2 Saran

1. Aplikasi *monitoring real time* untuk analisis selanjutnya perlu dikembangkan lagi seperti penggunaan statistik yang berasal dari ekstenal maupun internal sistem seperti sensor suhu. Untuk itu perlu dibuat sub *agent* baru untuk pendefinisian *mib* baru yang mendukung kinerja agen *snmp* yang utama.

2. Untuk aplikasi sistem *cluster* dengan *node diskless* perlu dicari solusi lain bagaimana meletakkan *tools* yang dibutuhkan pada *node client* selain menggunakan LTSP. Penulis menyarankan menggunakan FAI dengan menggunakan *tools* untuk debian (*.deb*).

# DAFTAR PUSTAKA


[1] Case, J., M.Fedor, M.Schoffstall, and J.davin, *Simple Network Management Protocol*. *RFC 1157. SNMP Research, Performance system Intenational, MIT Laboratory for Computer science*, may, 1990

[2] Downey, Allen, Mayer Chris, and Green Tea Pres, *How to think like computer scientis*t, April 2002

[3] Hadiyanto dan Handoko, L.T., "*The Development of mini Open Cluster for the Computation Society in Indonesia*", Indonesian Computation Society, Pusat Penelitian Fisika, Kompleks Puspiptek Serpong Tangerang, 30 Agustus 2004

[4] McClogrie, K., and M.Rose, *Management information base for network Management of TCP/IP-based Internets.RFC 1213. Hughes LAN System and Performance System International*, marc 1991

[5] Modul praktikum komdat, lab cnc, 2003

[6] Net-SNMP.sourceforge.net

[7] Perkins, D., and, E.McGinnis, *Undestanding SNMP MIBs*, New Jersey:prentice-hall,Inc.,1997

[8] Rose, M, and K. Mc Cloghrie, *Structure and identification of Management for TCP/IP-based Internet*, RFC 1155

[9] Sidnie Feit, Dr, McGRAW-HILL-, *A Guide to Network Managemen*t, International Edition

[10] Uthayopas, Surachai Phaisithbenchapol, Krisana Chongbarirux, *Building a Resources Monitoring System for SMILE Beowulf Cluste**r**Putchong*, Parallel Research Group Computer and Network System Research Laboratory Department of Computer Engineering Faculty of Engineering, Kasetsart University Bangkok, Thailand

# LAMPIRAN A

## Proses instalasi tools yang digunakan untuk sistem monitoring berbasis web

Sistem monitoring yang dikembangkan merupakan sebuah sistem yang terintegrasi dengan *software* atau program lain. Sistem ini berjalan di atas *Linux Operating System*, oleh karena itu sebuah distro *Linux* sudah pasti diperlukan. Dalam tahap uji coba, penulis mempergunakan distro *Debian GNU/Linux Sarge 3.1*. Sistem *monitoring* dikembangkan dengan menggunakan bahasa pemrograman *Python* sehingga lingkungan pemrograman *Python* juga diperlukan. *Software* atau program pendukung yang lain yang dibutuhkan adalah *Linux Terminal Software Project (LTSP)* versi 4, *RRDTool, Agen SNMP*.

1. **Instalasi Debian Sarge 3.1**

   Mungkin proses instalasi sistem operasi *Debian* menurut sebagian orang adalah hal yang cukup sulit untuk dilakukan karena seluruh konfigurasi yang otomatis tidak disediakan. Namun, telah banyak tutorial dan buku-buku yang menjelaskan proses detil instalasi dari sistem operasi tersebut. Di sini penulis hanya ingin menekankan bahwa dalam pemilihan kartu jaringan atau *NIC* sebaiknya mencari kartu jaringan yang telah memperoleh dukungan sistem operasi *Linux*. Karena kartu jaringan yang berfungsi dengan baik merupakan bagian yang paling mendasar dan paling penting di dalam sistem *monitoring* ini agar semua proses berikutnya dapat dilakukan. Kartu jaringan yang dapat dipergunakan seperti misalnya: *NIC* dengan driver RTL 8139 sangat familiar dengan *Linux*.

2. **Instalasi web server dan python**

   Web server yang digunakan adalah apache. dukungan sofware ini sudah ada pada linux source demikian juga python. Untuk apache, p

ada httpd.conf perlu diedit agar cgi enable yaitu menambahkan *ExecCGI*. Selain itu pada addhandler ditambahkan *addhandler.cgi* agar semua program .cgi bisa dibaca oleh web server.

### 3. Instalasi agen SNMP server

1. Resource : net-snmp.5.2.1.tar.gz
2. untar  dan unzip resource dengan menggunakan:

```
$gunzip net-snmp-5.2.1.tar.gz
$tar xvf net-snmp-5.2.1.tar.gz
```

3. compile package

```
$./configure --with-mib-modules="agentx"
------------------------------------------------------------
            Net-SNMP configuration summary:
------------------------------------------------------------

  SNMP Versions Supported:    1 2c 3
  Net-SNMP Version:           5.2.1
  Building for:               linux
  Network transport support:  Callback Unix TCP UDP
  SNMPv3 Security Modules:    usm
  Agent MIB code:             mibII ucd_snmp snmpv3mibs
notification target agent_mibs agentx utilities agentx
  SNMP Perl modules:          disabled
  Embedded perl support:      disabled
  Authentication support:     MD5
  Encryption support:

------------------------------------------------------------
$make
$make install
$cd local; make install; cd ..
$cd mibs; make install; cd ..
```

4. copi file EXAMPLE

```
$cp/usr/local/src/netsnmp5.2.1/EXAMPLE.conf
/usr/local/share/snmp/snmpd.conf
```

5. modifikasi /usr/local/share/snmp/snmpd.conf  sebagai berikut
    - ganti community dengan password yang diinginkan, biasanya untuk host/ip address digunakan “public” dan untuk localhost digunakan “private”
    - pada baris baru di akhir file tambahkan “master agentx”
6. fix beberapa library yang dibutuhkan

```
• $ln -s /usr/local/lib/libnetsnmp-0.5.0.0.2.so
  /lib/libnetsnmp-0.5.0.0.2.so
• $ln -s /usr/local/lib/libnetsnmpagent-0.5.0.0.2.so
  /lib/libnetsnmpagent-0.5.0.0.2.so
• $ln -s /usr/local/lib/libnetsnmphelpers-0.5.0.0.2.so
  /lib/libnetsnmphelpers-0.5.0.0.2.so
• $ln -s /usr/local/lib/libnetsnmpmibs-0.5.0.0.2.so
  /lib/libnetsnmpmibs-0.5.0.0.2.so
```

7. cek apakah snmp jalan atau tidak

```
$ ps awwux | grep snmp
```

8. jalankan snmp

```
$ cd ${HOME}/net-snmp-5.0.pre2/agent
$ ./snmp –f -L
```

**4. Instalasi agen SNMP pada client**

- **Dengan FAI (F*ullyAutomatic Installation)***

Resource : net-snmp.5.1.3.rpm

Libnet-devel 5.1.3.rpm

1. karena rpm (redhat package manager) merupakan paket redhat, maka harus diubah ke dalam pakket debian (deb)

```
$ alien –to-deb |grep *.rpm
```

2. install dengan menggunakan perintah :

```
$ dpkg –i |grep *.deb
```

- **Dengan LTSP**

*1.* Mempersiapkan *source binary LTSP-4.0*

*2.* Melakukan *extract* seluruh *source binary* yang ada (karena *source*-nyasungguh banyak), dan akan diperoleh *file system LTSP* yang berada di dalam *directory i386* dan kernel *LTSP* dengan nama *vmLinuz-x.x.x*

*3.* Pindahkan *directory i386* ke *directory* /opt/ltsp/ dan pindahkan kernel *LTSP* ke *directory* /tftpboot/lts

*4.* Lakukan perubahan pada *file /etc/dhcp.conf* untuk mendaftarkan *MACaddress NIC* dari *node client* ke *diskless system LTSP*.

*5.* Lakukan perubahan pada *file /etc/default/tftp* untuk me-*refer* atau menunjuk ke *directory /tftpboot*

6. Lakukan perubahan pada *file /etc/hosts* untuk melakukan *export file system LTSP*. Pastikan bahwa baris */opt/ltsp/i386* telah ada dalam *file* tersebut.
7. Pastikan bahwa beberapa *diamon* berikut telah *running* di *server LTSP*, yaitu: *dhcpd, tftpd,* dan *nfs*.

8. Nyalakan *node diskless* dengan menggunakan *floppy*. Apabila tidak terdapat kesalahan maka *diskless system* telah *running*.
9. Untuk melakukan konfigurasi yang spesifik pada *node LTSP* maka lakukan perubahan pada *file /opt/ltsp/i386/etc/lts.conf*.

- **Instalasi RRDtool**

Resource : rrdtool-1[1].0.49.tar.gz

1. untar  dan unzip resource dengan menggunakan :

   ```
   $ tar –zxvf rrdtool-1[1].0.49.tar.gz
   ```

2. install rrdtool dengan cara :

   ```
   rrdtool-1.0.49$ sh configure
   rrdtool-1.0.49$ make
   rrdtool-1.0.49$ make install
   ```

- **Instalasi Sistem monitoring Public Cluster LIPI**

Letakkan source sistem *monitoring public cluster* LIPI pada *directory* yang merupakan bagian *file system* dari *apache server* sehingga dapat diakses apabila kita mengetikkan alamat dari *directory* tersebut di *web browser*. Misal: di bawah *directory /var/www/htdocs*.

## LAMPIRAN B

## IDENTIFIKASI OBJEK-OBJEK MANAJEMEN JARINGAN SEBAGAI OBJEK MONITORING

- Menunjukkan sistem (host) yang sedang berjalan sesuai dengan ip addressnya.

```
lipi00:/# snmpget -v 2c 10.0.0.254 -c public .1.3.6.1.2.1.1.1.0
SNMPv2-MIB::sysDescr.0 = STRING: Linux lipi00 2.4.27-speakup #1 Thu
Aug 19 21:46 :14 CEST 2004 i686
lipi00:/# snmpget -v 2c 10.0.0.2 -c public .1.3.6.1.2.1.1.1.0
SNMPv2-MIB::sysDescr.0 = STRING: Linux lipi02 2.4.26-ltsp-3 #1 Tue
Apr 19 04:08: 58 UTC 2005 i686
lipi00:/# snmpget -v 2c 10.0.0.3 -c public .1.3.6.1.2.1.1.1.0
SNMPv2-MIB::sysDescr.0 = STRING: Linux lipi03 2.4.26-ltsp-3 #1 Tue
Apr 19 04:08: 58 UTC 2005 i686
lipi00:/# snmpstatus -v 2c 10.0.0.254 -c public .
[UDP: [10.0.0.254]:161]=>[Linux lipi00 2.4.27-speakup #1 Thu Aug 19
21:46:14 CEST 2004 i686] Up: 7:20:29.85
Interfaces: 0, Recv/Trans packets: 140738/140738 | IP:
13413812/26770139
lipi00:/# snmpstatus -v 2c 10.0.0.2 -c public .
[UDP: [10.0.0.2]:161]=>[Linux lipi02 2.4.26-ltsp-3 #1 Tue Apr 19
04:08:58 UTC 2005 i686] Up: 7:46:01.76
Interfaces: 0, Recv/Trans packets: 0/0 | IP: 2494564/1905429
lipi00:/# snmpstatus -v 2c 10.0.0.3 -c public .
[UDP: [10.0.0.3]:161]=>[Linux lipi03 2.4.26-ltsp-3 #1 Tue Apr 19
04:08:58 UTC 2005 i686] Up: 7:32:41.73
Interfaces: 0, Recv/Trans packets: 0/0 | IP: 24095331/11315697
```

- Hasil identifikasi obyek-obyek manajemen jaringan yang sesuai untuk sumber daya *hardware ethernet traffic* yang terdapat pada mib dengan group *interface*

```
lipi00:/# snmpwalk -v 2c 10.0.0.254 -c public interface
(.1.3.6.1.2.1.2)
IF-MIB::ifNumber.0 = INTEGER: 3
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.3 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifMtu.1 = INTEGER: 16436
IF-MIB::ifMtu.2 = INTEGER: 1500
IF-MIB::ifMtu.3 = INTEGER: 1500
IF-MIB::ifSpeed.1 = Gauge32: 10000000
IF-MIB::ifSpeed.2 = Gauge32: 100000000
IF-MIB::ifSpeed.3 = Gauge32: 10000000
IF-MIB::ifPhysAddress.1 = STRING:
```

```
IF-MIB::ifPhysAddress.2 = STRING: 0:8:54:7:d3:33
IF-MIB::ifPhysAddress.3 = STRING: 0:8:54:5:10:43
IF-MIB::ifAdminStatus.1 = INTEGER: up(1)
IF-MIB::ifAdminStatus.2 = INTEGER: up(1)
IF-MIB::ifAdminStatus.3 = INTEGER: down(2)
IF-MIB::ifOperStatus.1 = INTEGER: up(1)
IF-MIB::ifOperStatus.2 = INTEGER: up(1)
IF-MIB::ifOperStatus.3 = INTEGER: down(2)
IF-MIB::ifInOctets.1 = Counter32: 45234749
IF-MIB::ifInOctets.2 = Counter32: 1641414367
IF-MIB::ifInOctets.3 = Counter32: 0
IF-MIB::ifInUcastPkts.1 = Counter32: 131624
IF-MIB::ifInUcastPkts.2 = Counter32: 11220979
IF-MIB::ifInUcastPkts.3 = Counter32: 0
IF-MIB::ifInDiscards.1 = Counter32: 0
IF-MIB::ifInDiscards.2 = Counter32: 0
IF-MIB::ifInDiscards.3 = Counter32: 0
IF-MIB::ifInErrors.1 = Counter32: 0
IF-MIB::ifInErrors.2 = Counter32: 2
IF-MIB::ifInErrors.3 = Counter32: 0
IF-MIB::ifOutOctets.1 = Counter32: 45236524
IF-MIB::ifOutOctets.2 = Counter32: 1496864753
IF-MIB::ifOutOctets.3 = Counter32: 0
IF-MIB::ifOutUcastPkts.1 = Counter32: 131648
IF-MIB::ifOutUcastPkts.2 = Counter32: 22634169
IF-MIB::ifOutUcastPkts.3 = Counter32: 0
IF-MIB::ifOutDiscards.1 = Counter32: 0
IF-MIB::ifOutDiscards.2 = Counter32: 0
IF-MIB::ifOutDiscards.3 = Counter32: 0
IF-MIB::ifOutErrors.1 = Counter32: 0
IF-MIB::ifOutErrors.2 = Counter32: 0
IF-MIB::ifOutErrors.3 = Counter32: 0
IF-MIB::ifOutQLen.1 = Gauge32: 0
IF-MIB::ifOutQLen.2 = Gauge32: 0
IF-MIB::ifOutQLen.3 = Gauge32: 0
IF-MIB::ifSpecific.1 = OID: SNMPv2-SMI::zeroDotZero
IF-MIB::ifSpecific.2 = OID: SNMPv2-SMI::zeroDotZero
IF-MIB::ifSpecific.3 = OID: SNMPv2-SMI::zeroDotZero
```

- Hasil identifikasi obyek-obyek manajemen jaringan yang sesuai untuk sumber daya *hardware memory usage* yang terdapat pada mib dengan group *memory*

```
lipi00:/# snmpwalk -v 2c 10.0.0.3 -c public memory
(.1.3.6.1.4.1.2021.4)
UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 0
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 0
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 127308
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 99032
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 99028
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 16000
UCD-SNMP-MIB::memShared.0 = INTEGER: 0
```

```
UCD-SNMP-MIB::memBuffer.0 = INTEGER: 68
UCD-SNMP-MIB::memCached.0 = INTEGER: 14076
UCD-SNMP-MIB::memSwapError.0 = INTEGER: 1
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING: Running out of swap space
(0)
```

- Hasil identifikasi obyek-obyek manajemen jaringan yang sesuai untuk sumber daya *hardware system statistic* yang terdapat pada mib dengan group *systemstats*

```
lipi00:/# snmpwalk -v 2c 10.0.0.3 -c public systemstats
(1.3.6.1.4.1.2021.11)
UCD-SNMP-MIB::ssIndex.0 = INTEGER: 1
UCD-SNMP-MIB::ssErrorName.0 = STRING: systemStats
UCD-SNMP-MIB::ssSwapIn.0 = INTEGER: 0
UCD-SNMP-MIB::ssSwapOut.0 = INTEGER: 0
UCD-SNMP-MIB::ssIOSent.0 = INTEGER: 0
UCD-SNMP-MIB::ssIOReceive.0 = INTEGER: 0
UCD-SNMP-MIB::ssSysInterrupts.0 = INTEGER: 1007
UCD-SNMP-MIB::ssSysContext.0 = INTEGER: 589
UCD-SNMP-MIB::ssCpuUser.0 = INTEGER: 9
UCD-SNMP-MIB::ssCpuSystem.0 = INTEGER: 6
UCD-SNMP-MIB::ssCpuIdle.0 = INTEGER: 84
UCD-SNMP-MIB::ssCpuRawUser.0 = Counter32: 247273
UCD-SNMP-MIB::ssCpuRawNice.0 = Counter32: 0
UCD-SNMP-MIB::ssCpuRawSystem.0 = Counter32: 161725
UCD-SNMP-MIB::ssCpuRawIdle.0 = Counter32: 2195916
```

## LAMPIRAN C

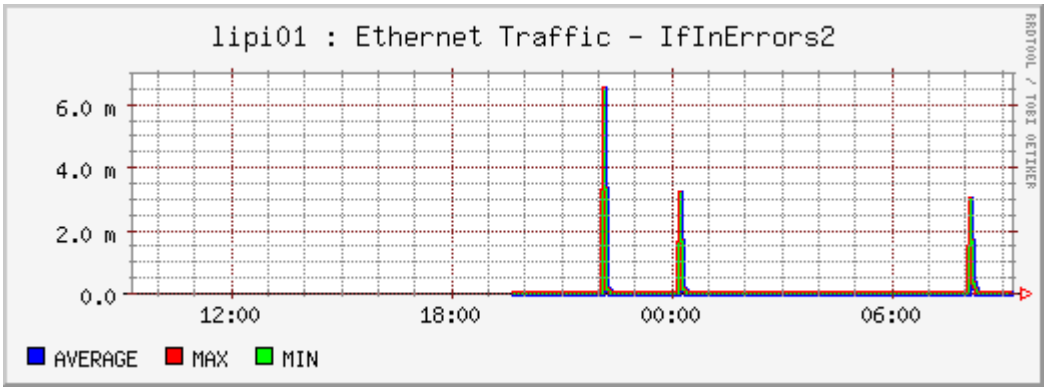
### Sumber daya cluster (item yang dimonitor)

| | | | | |
|---|---|---|---|---|
| sistem statistic | Name | OID | Type | counter (sec) |
| | SSSwapIn | UCD-SNMP-MIB::ssSwapIn.0 | DERIVE | 60 |
| | SSSwapOut | UCD-SNMP-MIB::ssSwapOut.0 | DERIVE | 60 |
| | SSIOSent | UCD-SNMP-MIB::ssIOSent.0 | DERIVE | 60 |
| | SSIOReceive | UCD-SNMP-MIB::ssIOReceive.0 | DERIVE | 60 |
| | SSSystemInterrupts | UCD-SNMP-MIB::ssSysInterrupts.0 | DERIVE | 60 |
| | SSSystemContext | UCD-SNMP-MIB::ssSysContext.0 | DERIVE | 60 |
| | SSCpuUser | UCD-SNMP-MIB::ssCpuUser.0 | DERIVE | 60 |
| | SSCpuSystem | UCD-SNMP-MIB::ssCpuSystem.0 | DERIVE | 60 |
| | SSCpuIdle | UCD-SNMP-MIB::ssCpuIdle.0 | DERIVE | 60 |
| | SSCpuRawUser | UCD-SNMP-MIB::ssCpuRawUser.0 | COUNTER | 60 |
| | SSCpuRawNice | UCD-SNMP-MIB::ssCpuRawNice.0 | COUNTER | 60 |
| | SSCpuRawSystem | UCD-SNMP-MIB::ssCpuRawSystem.0 | COUNTER | 60 |
| | SSCpuRawIdle | UCD-SNMP-MIB::ssCpuRawIdle.0 | COUNTER | 60 |
| | SSCpuRawKernel | UCD-SNMP-MIB::ssCpuRawKernel.0 | COUNTER | 60 |
| | SSCpuRawSent | UCD-SNMP-MIB::ssIORawSent.0 | COUNTER | 60 |
| | SSIORawReceived | UCD-SNMP-MIB::ssIORawReceived.0 | COUNTER | 60 |
| | SSRawInterrupts | UCD-SNMP-MIB::ssRawInterrupts.0 | COUNTER | 60 |
| | SSRawContexts | UCD-SNMP-MIB::ssRawContexts.0 | COUNTER | 60 |
| | SSRawSwapIn | UCD-SNMP-MIB::ssRawSwapIn.0 | COUNTER | 60 |
| | SSRawSwapOut | UCD-SNMP-MIB::ssRawSwapOut.0 | COUNTER | 60 |
| | | | | |
| Memory usage | Name | OID | Type | counter (sec) |
| | TotalSwap | UCD-SNMP-MIB::memTotalSwap.0 | DERIVE | 60 |
| | AvailableSwap | UCD-SNMP-MIB::memAvailSwap.0 | DERIVE | 60 |
| | TotalReal | UCD-SNMP-MIB::memTotalReal.0 | DERIVE | 60 |
| | AvailableReal | UCD-SNMP-MIB::memAvailReal.0 | DERIVE | 60 |
| | TotalFree | UCD-SNMP-MIB::memTotalFree.0 | DERIVE | 60 |
| | MinimumSwap | UCD-SNMP-MIB::memMinimumSwap.0 | DERIVE | 60 |
| | Shared | UCD-SNMP-MIB::memShared.0 | DERIVE | 60 |
| | Buffer | UCD-SNMP-MIB::memBuffer.0 | DERIVE | 60 |
| | Cached | UCD-SNMP-MIB::memCached.0 | DERIVE | 60 |
| | SwapError | UCD-SNMP-MIB::memSwapError.0 | DERIVE | 60 |
| | | | | |
| ethernet traffic | Name | OID | Type | counter (sec) |
| | IfInOctets | IF-MIB::ifInOctets.2 | COUNTER | 300 |
| | IfInUcastPkts | IF-MIB::ifInUcastPkts.2 | COUNTER | 300 |
| | IfInErrors | IF-MIB::ifInErrors.2 | COUNTER | 300 |
| | IfOutOctets | IF-MIB::ifOutOctets.2 | COUNTER | 300 |
| | IfOutUcastPkts | IF-MIB::ifOutUcastPkts.2 | COUNTER | 300 |

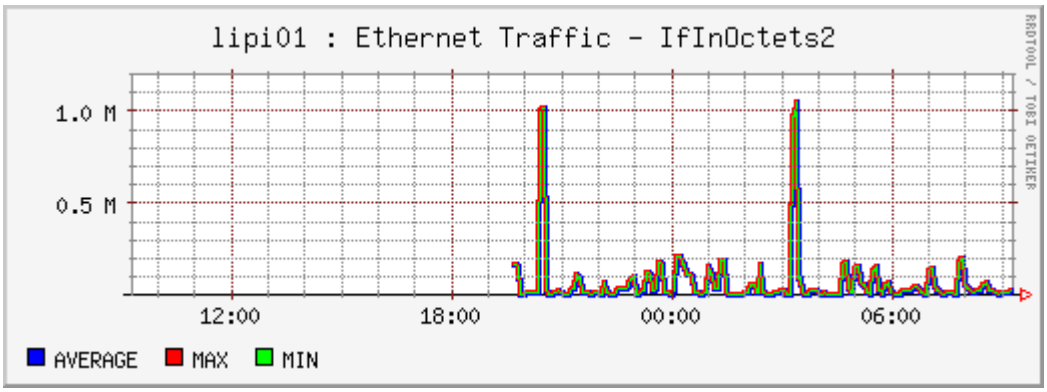
| | | | | |
|---|---|---|---|---|
| | IfOutDiscards | IF-MIB::ifOutDiscards.2 | COUNTER | 300 |
| | IfOutErrors | IF-MIB::ifOutErrors.2 | COUNTER | 300 |
| | IfInDiscards | IF-MIB::ifInDiscards.2 | COUNTER | 300 |
| | IfOutQLen | IF-MIB::ifOutQLen.2 | COUNTER | 300 |

# LAMPIRAN D

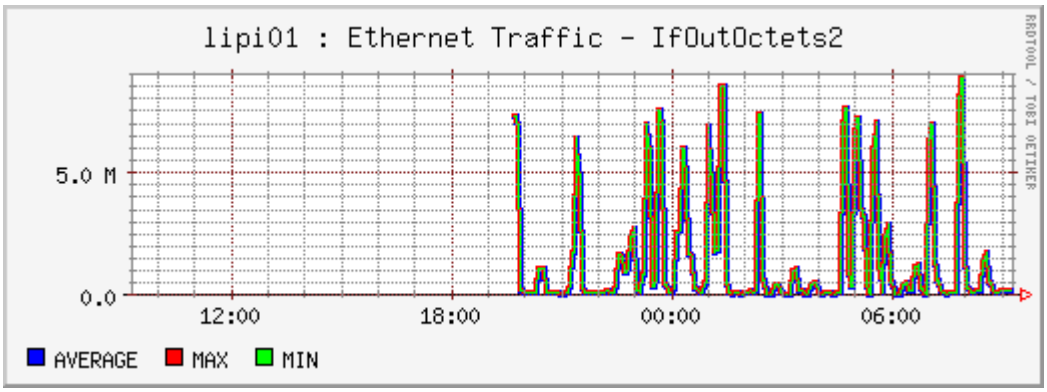
## Grafik monitoring sistem untuk masing-masing node



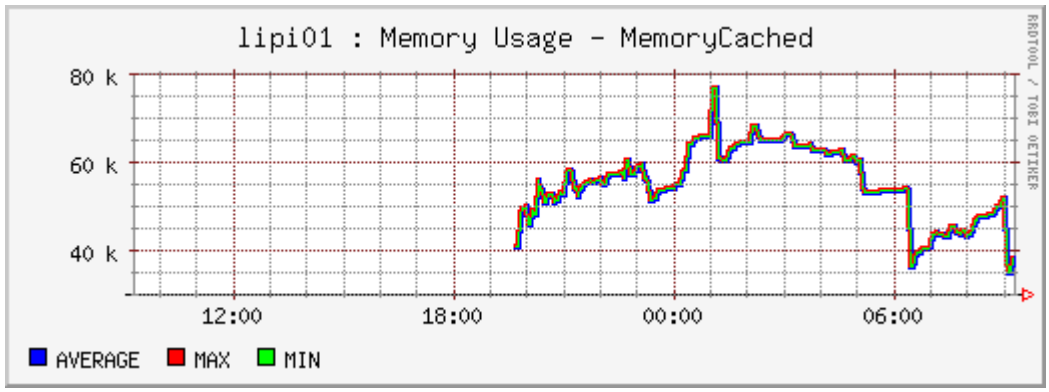

Gambar D.1 Grafik hasil monitoring lipi01- ethernet traffic-IfInErrors



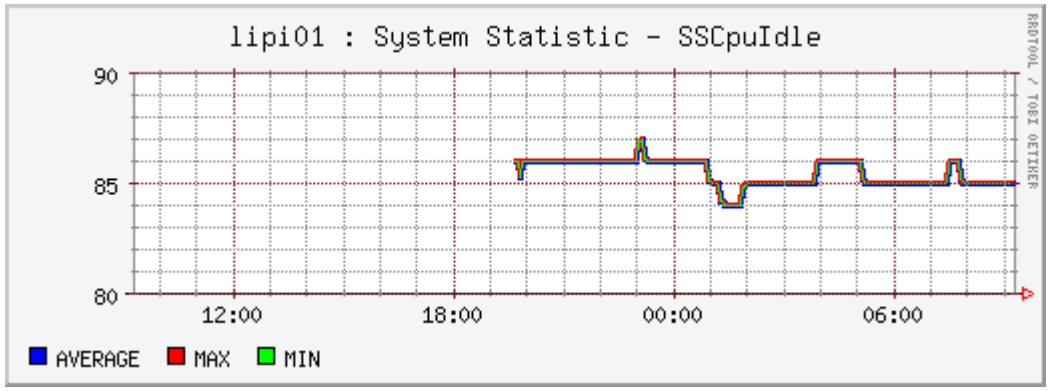

Gambar D.2 Grafik hasil monitoring lipi01- ethernet - traffic-IfInOctets



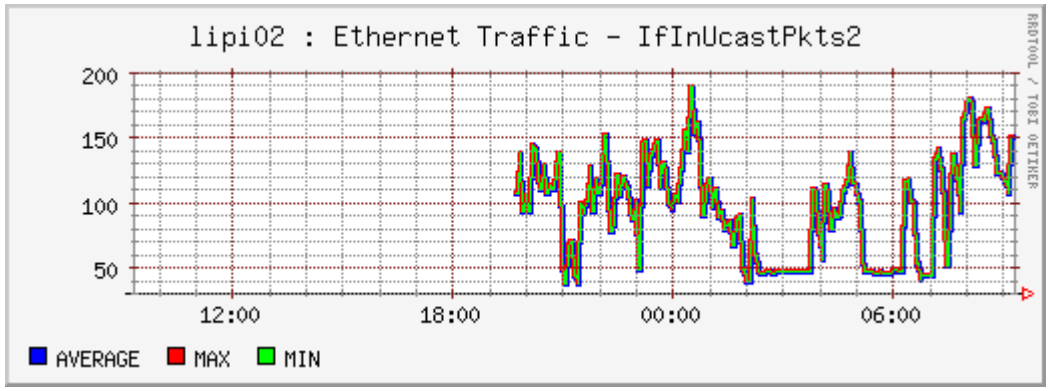

Gambar D.3 Grafik hasil monitoring lipi01- ethernet - traffic-IfOutOctets



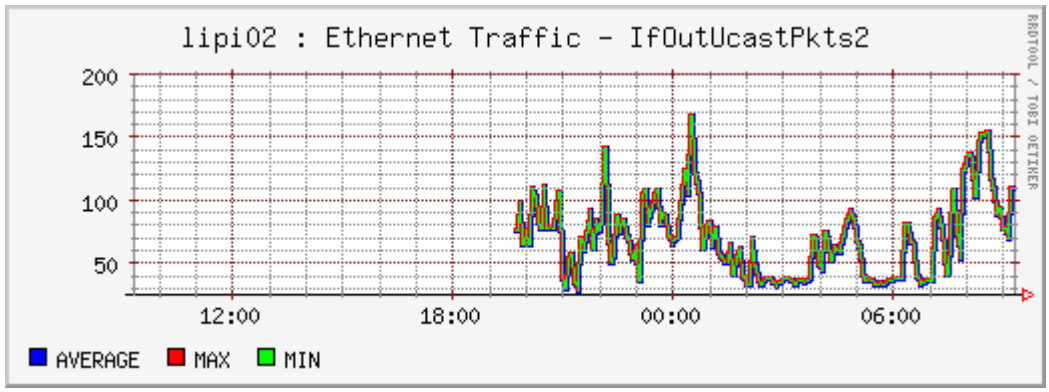

Gambar D.4 Grafik hasil monitoring lipi01 – memry usage-MemoryChached



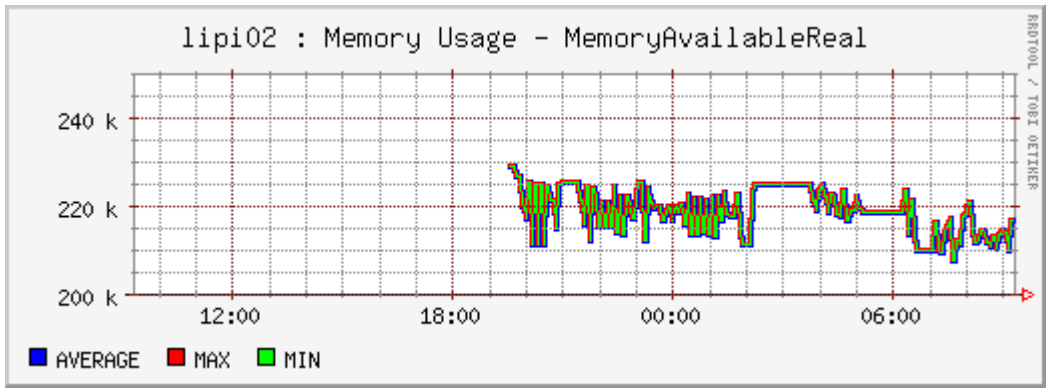

Gambar D.5 Grafik hasil monitoring lipi01 – System Statistic-SSCpuIdle



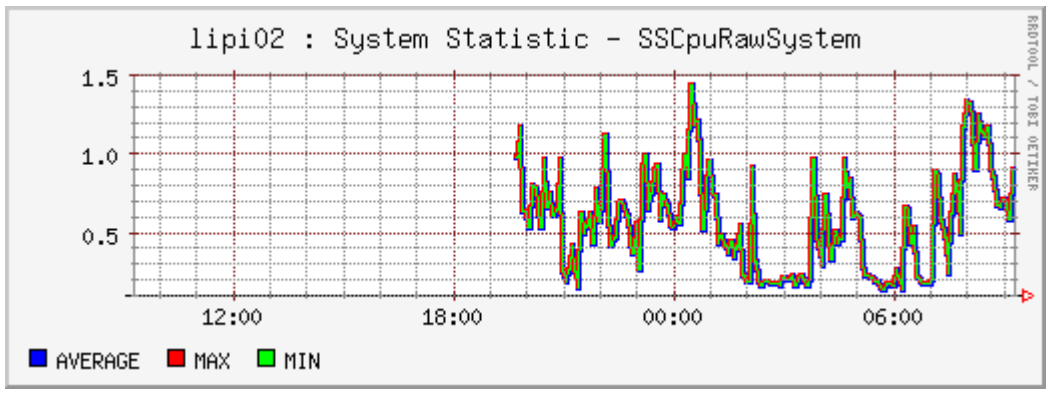

Gambar D.6 Ggrafik hasil monitoring lipi02- Ethernet Traffic-IfInUcastPkts



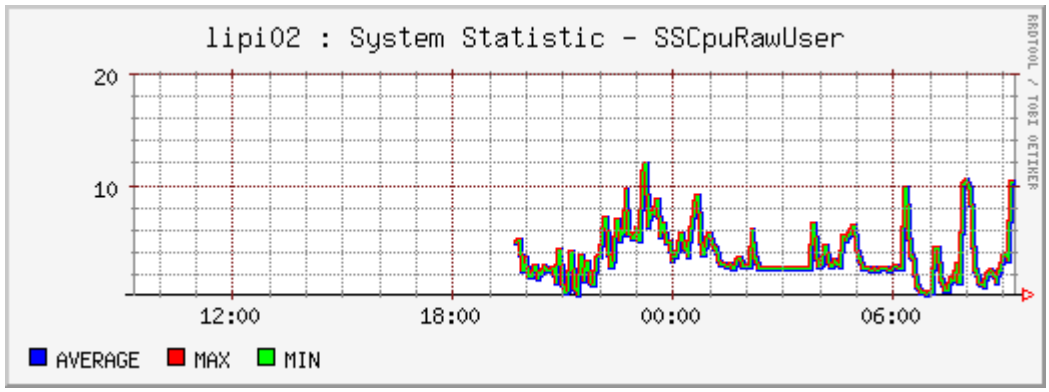

Gambar D.7 Grafik hasil monitoring lipi02- Ethernet Traffic-IfOutUcastPkts



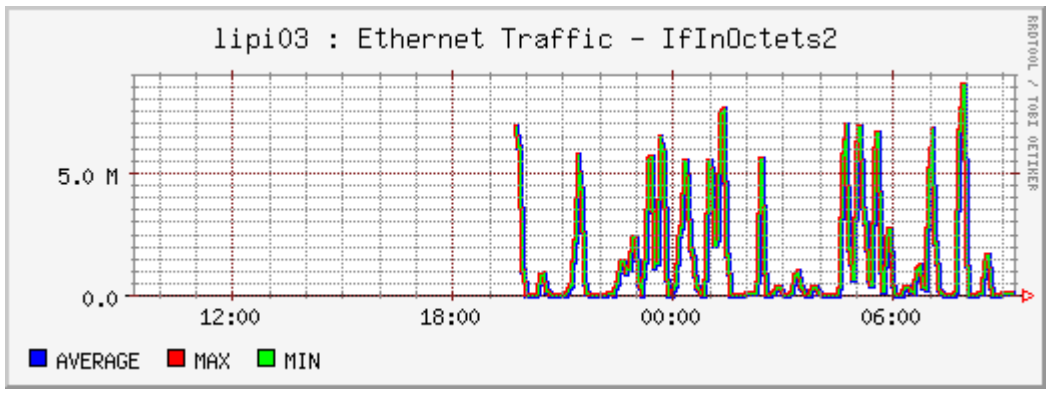

Gambar D.8 Grafik hasil monitoring lipi02-Memory Usage-MemoryAvailableReal



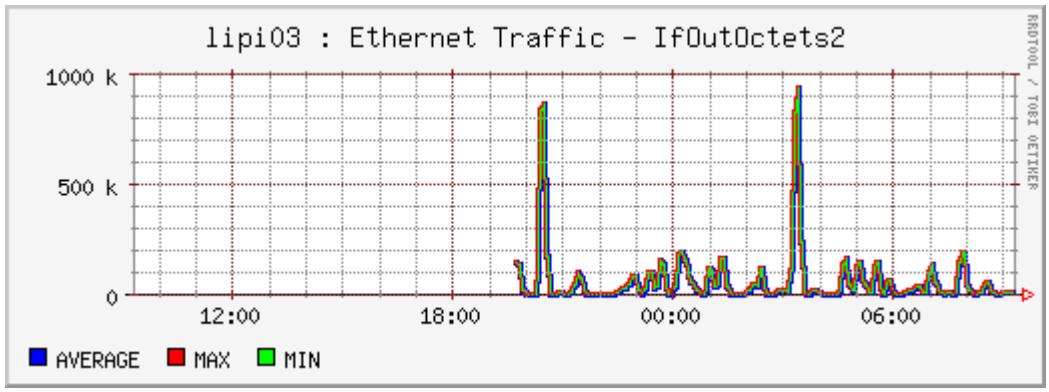

Gambar D.9 Grafik hasil monitoring lipi02- System Statistic-SSCpuRawSystem



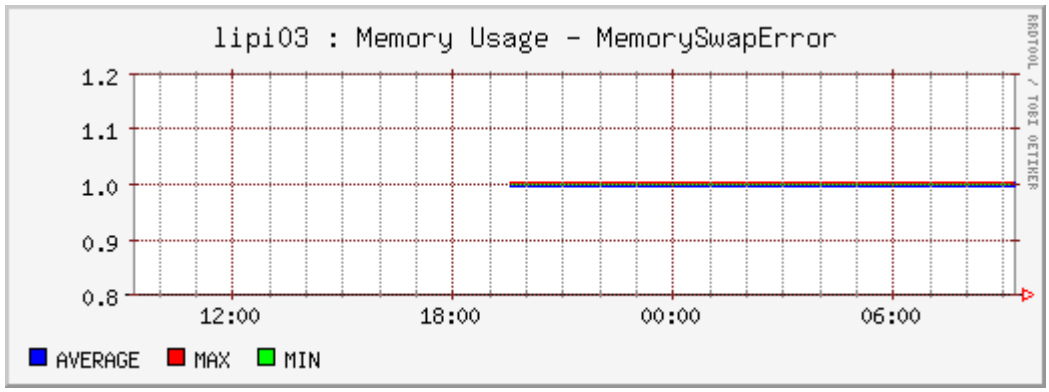

Gambar D.10 Grafik hasil monitoring lipi02 –System Statistic-SSCpuRawUser



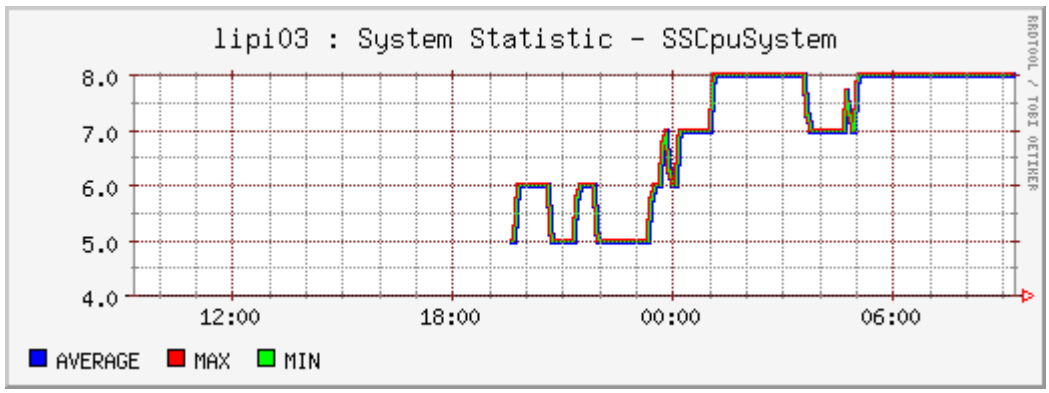

Gambar D.11 Grafik hasil monitoring lipi03- Ethernet Traffic-IfInOctects



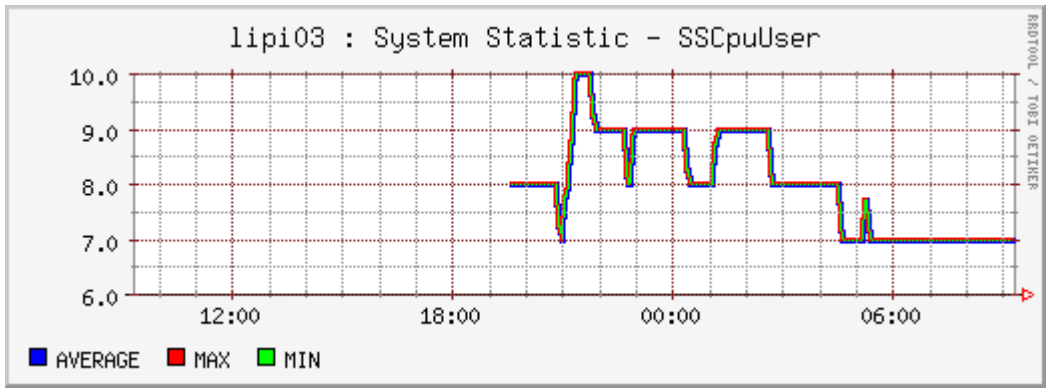

Gambar D.12 Grafik hasil monitoring lipi03- Ethernet Traffic-IfOutOctects




Gambar D.13 Grafik hasil monitoring lipi03- Memory Usage-MemorySwapError




Gambar D.14 Grafik hasil monitoring lipi03- System Statistic-SSCpuSystem




Gambar D.15 Grafik hasil monitoring lipi03- System Statistic-SSCpuUser

# LAMPIRAN E

## Log hasil ujicoba sistem

```
27 Juli 2005, 19:41:5    lipi01 : Ethernet Traffic - IfInErrors2 = 0
27 Juli 2005, 19:41:5    lipi01 : Ethernet Traffic - IfInOctets2 = 583652203
27 Juli 2005, 19:41:5    lipi01 : Ethernet Traffic - IfInDiscards2 = 0
27 Juli 2005, 19:41:5    lipi01 : Ethernet Traffic - IfInUcastPkts2 = 4815298
27 Juli 2005, 19:41:6    lipi01 : Ethernet Traffic - IfOutErrors2 = 0
27 Juli 2005, 19:41:6    lipi01 : Ethernet Traffic - IfOutDiscards2 = 0
27 Juli 2005, 19:41:6    lipi01 : Ethernet Traffic - IfOutQLen2 = 0
27 Juli 2005, 19:41:6    lipi01 : Ethernet Traffic - IfOutUcastPkts2 = 9952793
27 Juli 2005, 19:41:6    lipi01 : Ethernet Traffic - IfOutOctets2 = 2796430520
27 Juli 2005, 19:41:6    lipi01 : Memory Usage - MemoryAvailableSwap = 476056
27 Juli 2005, 19:41:7    lipi01 : Memory Usage - MemoryAvailableReal = 4280
27 Juli 2005, 19:41:7    lipi01 : Memory Usage - MemoryMinimumSwap = 16000
27 Juli 2005, 19:41:7    lipi01 : Memory Usage - MemoryBuffer = 49700
27 Juli 2005, 19:41:7    lipi01 : Memory Usage - MemoryCached = 86544
27 Juli 2005, 19:41:7    lipi01 : Memory Usage - MemoryShared = 0
27 Juli 2005, 19:41:7    lipi01 : Memory Usage - MemorySwapError = 0
27 Juli 2005, 19:41:8    lipi01 : Memory Usage - MemoryTotalReal = 256992
27 Juli 2005, 19:41:8    lipi01 : Memory Usage - MemoryTotalFree = 480092
27 Juli 2005, 19:41:8    lipi01 : System Statistic - SSCpuRawIdle = 1174290
27 Juli 2005, 19:41:8    lipi01 : Memory Usage - MemoryTotalSwap = 570268
27 Juli 2005, 19:41:9    lipi01 : System Statistic - SSCpuIdle = 87
27 Juli 2005, 19:41:9    lipi01 : System Statistic - SSCpuRawKernel = 64421
27 Juli 2005, 19:41:9    lipi01 : System Statistic - SSCpuRawSent = 967650
27 Juli 2005, 19:41:9    lipi01 : System Statistic - SSCpuRawSystem = 64421
27 Juli 2005, 19:41:9    lipi01 : System Statistic - SSCpuRawNice = 6225
27 Juli 2005, 19:41:10    lipi01 : System Statistic - SSCpuSystem = 4
27 Juli 2005, 19:41:10    lipi01 : System Statistic - SSCpuRawUser = 90967
27 Juli 2005, 19:41:10    lipi01 : System Statistic - SSCpuUser = 7
27 Juli 2005, 19:41:10    lipi01 : System Statistic - SSIOReceive = 49
27 Juli 2005, 19:41:10    lipi01 : System Statistic - SSIOSent = 36
27 Juli 2005, 19:41:10    lipi01 : System Statistic - SSIORawReceived = 1305878
27 Juli 2005, 19:41:10    lipi01 : System Statistic - SSRawContexts = 5764501
27 Juli 2005, 19:41:10      lipi01 : System Statistic - SSRawInterrupts =
13561225
27 Juli 2005, 19:41:11    lipi01 : System Statistic - SSSwapIn = 10
27 Juli 2005, 19:41:11    lipi01 : System Statistic - SSRawSwapOut = 43084
27 Juli 2005, 19:41:11    lipi01 : System Statistic - SSRawSwapIn = 33929
27 Juli 2005, 19:41:11    lipi01 : System Statistic - SSSwapOut = 13
27 Juli 2005, 19:41:11    lipi01 : System Statistic - SSSystemContext = 432
27 Juli 2005, 19:41:11     lipi01 : System Statistic - lmFanSensorsValue2 =
UNKNOWN
27 Juli 2005, 19:41:11     lipi01 : System Statistic - lmTempSensorsValue1 =
UNKNOWN
27 Juli 2005, 19:41:11    lipi01 : System Statistic - SSSystemInterrupts = 1015
27 Juli 2005, 19:41:12     lipi01 : System Statistic - lmVoltSensorsValue1 =
UNKNOWN
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfInDiscards2 = 0
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfInOctets2 = 463667151
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfInErrors2 = 0
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfOutDiscards2 = 0
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfInUcastPkts2 = 925898
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfOutOctets2 = 92989882
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfOutErrors2 = 0
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfOutUcastPkts2 = 778152
27 Juli 2005, 19:41:44    lipi02 : Ethernet Traffic - IfOutQLen2 = 0
27 Juli 2005, 19:41:45    lipi02 : Memory Usage - MemoryAvailableSwap = 0
27 Juli 2005, 19:41:45    lipi02 : Memory Usage - MemoryAvailableReal = 228880
```

```
27 Juli 2005, 19:41:46    lipi02 : Memory Usage - MemoryCached = 14036
27 Juli 2005, 19:41:46    lipi02 : Memory Usage - MemoryBuffer = 68
27 Juli 2005, 19:41:46    lipi02 : Memory Usage - MemoryMinimumSwap = 16000
27 Juli 2005, 19:41:47    lipi02 : Memory Usage - MemorySwapError = 1
27 Juli 2005, 19:41:47    lipi02 : Memory Usage - MemoryShared = 0
27 Juli 2005, 19:41:47    lipi02 : Memory Usage - MemoryTotalReal = 256916
27 Juli 2005, 19:41:47    lipi02 : Memory Usage - MemoryTotalFree = 228876
27 Juli 2005, 19:41:50    lipi02 : System Statistic - SSCpuIdle = 94
27 Juli 2005, 19:41:50    lipi02 : Memory Usage - MemoryTotalSwap = 0
27 Juli 2005, 19:41:51    lipi02 : System Statistic - SSCpuRawKernel = UNKNOWN
27 Juli 2005, 19:41:51    lipi02 : System Statistic - SSCpuRawSystem = 6881
27 Juli 2005, 19:41:51    lipi02 : System Statistic - SSCpuRawIdle = 1252566
27 Juli 2005, 19:41:51    lipi02 : System Statistic - SSCpuRawNice = 0
27 Juli 2005, 19:41:52    lipi02 : System Statistic - SSCpuUser = 5
27 Juli 2005, 19:41:52    lipi02 : System Statistic - SSCpuRawSent = UNKNOWN
27 Juli 2005, 19:41:52    lipi02 : System Statistic - SSCpuSystem = 0
27 Juli 2005, 19:41:52    lipi02 : System Statistic - SSCpuRawUser = 72413
27 Juli 2005, 19:41:52    lipi02 : System Statistic - SSIORawReceived = UNKNOWN
27 Juli 2005, 19:41:52    lipi02 : System Statistic - SSIOReceive = 0
27 Juli 2005, 19:41:52    lipi02 : System Statistic - SSRawContexts = UNKNOWN
27 Juli 2005, 19:41:52    lipi02 : System Statistic - SSIOSent = 0
27 Juli 2005, 19:41:53    lipi02 : System Statistic - SSRawSwapIn = UNKNOWN
27 Juli 2005, 19:41:54    lipi02 : System Statistic - SSRawInterrupts = UNKNOWN
27 Juli 2005, 19:41:54    lipi02 : System Statistic - SSRawSwapOut = UNKNOWN
27 Juli 2005, 19:41:54    lipi02 : System Statistic - SSSystemContext = 128
27 Juli 2005, 19:41:54    lipi02 : System Statistic - SSSwapIn = 0
27 Juli 2005, 19:41:54    lipi02 : System Statistic - SSSwapOut = 0
27 Juli 2005, 19:41:54    lipi02 : System Statistic - SSSystemInterrupts = 246
27 Juli 2005, 19:41:54      lipi02 : System Statistic - lmFanSensorsValue2 =
UNKNOWN
27 Juli 2005, 19:41:55      lipi02 : System Statistic - lmTempSensorsValue1 =
UNKNOWN
27 Juli 2005, 19:41:55      lipi02 : System Statistic - lmVoltSensorsValue1 =
UNKNOWN
27 Juli 2005, 19:42:26    lipi03 : Ethernet Traffic - IfInDiscards2 = 0
27 Juli 2005, 19:42:26    lipi03 : Ethernet Traffic - IfInUcastPkts2 = 9308336
27 Juli 2005, 19:42:26    lipi03 : Ethernet Traffic - IfInErrors2 = 0
27 Juli 2005, 19:42:27    lipi03 : Ethernet Traffic - IfInOctets2 = 2819437228
27 Juli 2005, 19:42:27    lipi03 : Ethernet Traffic - IfOutDiscards2 = 0
27 Juli 2005, 19:42:27    lipi03 : Ethernet Traffic - IfOutErrors2 = 0
27 Juli 2005, 19:42:27    lipi03 : Ethernet Traffic - IfOutOctets2 = 494906082
27 Juli 2005, 19:42:27    lipi03 : Ethernet Traffic - IfOutQLen2 = 0
27 Juli 2005, 19:42:27    lipi03 : Ethernet Traffic - IfOutUcastPkts2 = 4154168
27 Juli 2005, 19:42:28    lipi03 : Memory Usage - MemoryAvailableReal = 99148
27 Juli 2005, 19:42:28    lipi03 : Memory Usage - MemoryAvailableSwap = 0
27 Juli 2005, 19:42:28    lipi03 : Memory Usage - MemoryBuffer = 68
27 Juli 2005, 19:42:29    lipi03 : Memory Usage - MemoryCached = 14072
27 Juli 2005, 19:42:29    lipi03 : Memory Usage - MemoryShared = 0
27 Juli 2005, 19:42:29    lipi03 : Memory Usage - MemoryMinimumSwap = 16000
27 Juli 2005, 19:42:29    lipi03 : Memory Usage - MemorySwapError = 1
27 Juli 2005, 19:42:29    lipi03 : Memory Usage - MemoryTotalFree = 99236
27 Juli 2005, 19:42:30    lipi03 : System Statistic - SSCpuIdle = 85
27 Juli 2005, 19:42:30    lipi03 : Memory Usage - MemoryTotalReal = 127308
27 Juli 2005, 19:42:30    lipi03 : System Statistic - SSCpuRawIdle = 1074098
27 Juli 2005, 19:42:30    lipi03 : Memory Usage - MemoryTotalSwap = 0
27 Juli 2005, 19:42:30    lipi03 : System Statistic - SSCpuRawKernel = UNKNOWN
27 Juli 2005, 19:42:31    lipi03 : System Statistic - SSCpuRawNice = 0
27 Juli 2005, 19:42:32    lipi03 : System Statistic - SSCpuRawSent = UNKNOWN
27 Juli 2005, 19:42:32    lipi03 : System Statistic - SSCpuRawUser = 111637
27 Juli 2005, 19:42:32    lipi03 : System Statistic - SSCpuRawSystem = 68747
27 Juli 2005, 19:42:32    lipi03 : System Statistic - SSCpuSystem = 5
```

```
27 Juli 2005, 19:42:32    lipi03 : System Statistic - SSIORawReceived = UNKNOWN
27 Juli 2005, 19:42:34    lipi03 : System Statistic - SSCpuUser = 8
27 Juli 2005, 19:42:35    lipi03 : System Statistic - SSRawContexts = UNKNOWN
27 Juli 2005, 19:42:35    lipi03 : System Statistic - SSRawInterrupts = UNKNOWN
27 Juli 2005, 19:42:35    lipi03 : System Statistic - SSIOReceive = 0
27 Juli 2005, 19:42:35    lipi03 : System Statistic - SSIOSent = 0
27 Juli 2005, 19:42:36    lipi03 : System Statistic - SSRawSwapIn = UNKNOWN
27 Juli 2005, 19:42:36    lipi03 : System Statistic - SSSwapIn = 0
27 Juli 2005, 19:42:36    lipi03 : System Statistic - SSRawSwapOut = UNKNOWN
27 Juli 2005, 19:42:36    lipi03 : System Statistic - SSSystemInterrupts = 953
27 Juli 2005, 19:42:36    lipi03 : System Statistic - SSSwapOut = 0
27 Juli 2005, 19:42:36      lipi03 : System Statistic - lmFanSensorsValue2 =
UNKNOWN
27 Juli 2005, 19:42:36    lipi03 : System Statistic - SSSystemContext = 523
27 Juli 2005, 19:42:37      lipi03 : System Statistic - lmTempSensorsValue1 =
UNKNOWN
27 Juli 2005, 19:42:37      lipi03 : System Statistic - lmVoltSensorsValue1 =
UNKNOWN
27 Juli 2005, 19:46:5    lipi01 : Ethernet Traffic - IfInErrors2 = 0
27 Juli 2005, 19:46:5    lipi01 : Ethernet Traffic - IfInOctets2 = 633269192
27 Juli 2005, 19:46:6    lipi01 : Ethernet Traffic - IfInDiscards2 = 0
27 Juli 2005, 19:46:6    lipi01 : Ethernet Traffic - IfInUcastPkts2 = 5552269
27 Juli 2005, 19:46:6    lipi01 : Ethernet Traffic - IfOutDiscards2 = 0
27 Juli 2005, 19:46:6    lipi01 : Ethernet Traffic - IfOutErrors2 = 0
27 Juli 2005, 19:46:6    lipi01 : Ethernet Traffic - IfOutQLen2 = 0
27 Juli 2005, 19:46:6    lipi01 : Ethernet Traffic - IfOutUcastPkts2 = 11419221
27 Juli 2005, 19:46:6    lipi01 : Ethernet Traffic - IfOutOctets2 = 683900879
27 Juli 2005, 19:46:7    lipi01 : Memory Usage - MemoryAvailableSwap = 461544
27 Juli 2005, 19:46:8    lipi01 : Memory Usage - MemoryBuffer = 15468
27 Juli 2005, 19:46:8    lipi01 : Memory Usage - MemoryAvailableReal = 4200
27 Juli 2005, 19:46:8    lipi01 : Memory Usage - MemoryShared = 0
27 Juli 2005, 19:46:8    lipi01 : Memory Usage - MemoryCached = 40936
27 Juli 2005, 19:46:8    lipi01 : Memory Usage - MemoryMinimumSwap = 16000
27 Juli 2005, 19:46:8    lipi01 : Memory Usage - MemorySwapError = 0
27 Juli 2005, 19:46:8    lipi01 : Memory Usage - MemoryTotalFree = 468000
27 Juli 2005, 19:46:8    lipi01 : System Statistic - SSCpuRawIdle = 1186294
27 Juli 2005, 19:46:8    lipi01 : Memory Usage - MemoryTotalReal = 256992
27 Juli 2005, 19:46:9    lipi01 : Memory Usage - MemoryTotalSwap = 570268
27 Juli 2005, 19:46:9    lipi01 : System Statistic - SSCpuRawKernel = 72163
27 Juli 2005, 19:46:9    lipi01 : System Statistic - SSCpuIdle = 86
27 Juli 2005, 19:46:9    lipi01 : System Statistic - SSCpuRawSystem = 72163
27 Juli 2005, 19:46:10    lipi01 : System Statistic - SSCpuRawNice = 13102
27 Juli 2005, 19:46:10    lipi01 : System Statistic - SSCpuRawSent = 980434
27 Juli 2005, 19:46:10    lipi01 : System Statistic - SSCpuSystem = 5
27 Juli 2005, 19:46:10    lipi01 : System Statistic - SSCpuRawUser = 94392
27 Juli 2005, 19:46:10    lipi01 : System Statistic - SSCpuUser = 7
27 Juli 2005, 19:46:11    lipi01 : System Statistic - SSIOSent = 36
27 Juli 2005, 19:46:11    lipi01 : System Statistic - SSIOReceive = 50
27 Juli 2005, 19:46:11    lipi01 : System Statistic - SSIORawReceived = 1359590
27 Juli 2005, 19:46:11    lipi01 : System Statistic - SSRawContexts = 5952177
27 Juli 2005, 19:46:14      lipi01 : System Statistic - SSRawInterrupts =
15081143
27 Juli 2005, 19:46:14    lipi01 : System Statistic - SSRawSwapOut = 46837
27 Juli 2005, 19:46:14    lipi01 : System Statistic - SSSystemInterrupts = 1104
27 Juli 2005, 19:46:14      lipi01 : System Statistic - lmFanSensorsValue2 =
UNKNOWN
27 Juli 2005, 19:46:14    lipi01 : System Statistic - SSSwapIn = 10
27 Juli 2005, 19:46:15      lipi01 : System Statistic - lmVoltSensorsValue1 =
UNKNOWN
27 Juli 2005, 19:46:15    lipi01 : System Statistic - SSSystemContext = 436
27 Juli 2005, 19:46:15    lipi01 : System Statistic - SSSwapOut =1
```

# LAMPIRAN F

## Source code sistem monitoring

Perangkat lunak sistem *monitoring* terdieri dari beberapa *module*. Secara lengkap seperti dibawah ini :

```
Lipi_pc
    .htaccess
    index.html
    cgi_bin
        lipi_pc_main.cgi
        lipi_pc_analisis_performance.py
        lipi_pc
            __init__.py
            lipi_pc_html
                __init__.py
                lipi_pc_html.py
                lipi_pc_content
                    __init__.py
                    lipi_pc_monitoring_cluster.py
                    lipi_pc_manajemen_cluster.py
                    lipi_pc_manajemen_user.py
                    lipi_pc_error_page.py
                    lipi_pc_content_util
                        __init__.py
                        lipi_pc_content_util.py
            lipi_pc_site
                __init__.py
                lipi_pc_content_menu.py
                lipi_pc_error.py
                lipi_pc_site.py
            lipi_pc_util
                __init__.py
                lipi_pc_auth.py
                lipi_pc_config.py
                lipi_pc_paralle_port.py
                lipi_pc_process.pylipi_pc_rrd.py
                lipi_pc_schedule.py
                lipi_pc_services.py
                lipi_pc_snmp.py
                ipi_pc_util.py
    data
        .htaccess
        .secret
```

system
default.dat
monitoring_item.dat
nodes.dat
log
monitoring_data.dat
rrd
lipi01__Ethernet_Traffic__IfInDiscards2__Sumber_Daya_Cluster.rrd
lipi01__Ethernet_Traffic__IfInErrors2__Sumber_Daya_Cluster.rrd
service
images
rrd
lipi01__Ethernet_Traffic__IfInDiscards2__Sumber_Daya_Cluster.png
lipi01__Ethernet_Traffic__IfInErrors2__Sumber_Daya_Cluster.png

```
#!/usr/bin/python


#########################################################################
#########
# Import Modules
import cgi,cgitb;
cgitb.enable()
from lipi_pc.lipi_pc_html import lipi_pc_html
from lipi_pc.lipi_pc_site import lipi_pc_site,lipi_pc_error
from lipi_pc.lipi_pc_util import
lipi_pc_util,lipi_pc_auth,lipi_pc_config
#########################################################################
#########


#########################################################################
#########
# Declare Global Variable
lipiPCConfig = lipi_pc_config.LipiPCConfig()
config = lipiPCConfig.getConfig("../data/lipi_pc.conf")
config_system = lipiPCConfig.getConfig("../data/lipi_pc_system.conf")
lipiPCAuth =
lipi_pc_auth.LipiPCAuth("..",config_system,config["user_name"],config["
user_password"])
lipiPCHtml = lipi_pc_html.LipiPCHtml("..",config_system)
lipiPCSite = lipi_pc_site.LipiPCSite("..",config_system)
lipiPCError = lipi_pc_error.LipiPCError("..",config_system)
lipiPCUtil = lipi_pc_util.LipiPCUtil("..",config_system)
request = cgi.FieldStorage()
menu = ""
menu_link = ""
session = ""
```

```
privilege = ""
subtitle = ""
error_message = ""
privilege = "no_privilege"
#########################################################################
#########

#########################################################################
#########
# Checking Request
# Authentication
status = lipiPCSite.doAuthentication(request,lipiPCAuth)
if status == 1:
    privilege = lipiPCAuth.getUserPrivilege(request["username"].value)
    session = lipiPCAuth.newSession(privilege)
    subtitle = lipiPCSite.AKSES_BERHASIL
elif status == -2:
    subtitle = lipiPCError.INVALID_USER_PASSWORD[0]
    error_message = lipiPCError.INVALID_USER_PASSWORD[1]
elif status == -1:
    subtitle = lipiPCError.FIELD_EMPTY_WARNING[0]
    error_message = lipiPCError.FIELD_EMPTY_WARNING[1]
# Checking parameter from request
elif (request.__len__() == 1 and not request.has_key("menu")) or \
    (request.__len__() == 2 and (not request.has_key("menu") or not
request.has_key("session"))):
    subtitle = lipiPCError.UNKNOWN_PAGE_REQUEST[0]
    error_message = lipiPCError.UNKNOWN_PAGE_REQUEST[1]
# Logout
elif request.has_key("session") and request.has_key("menu") and \
    (request["menu"].value == lipiPCSite.MENU_ADMIN[3][3] or \
    request["menu"].value == lipiPCSite.MENU_USER[1][3]):
    if lipiPCAuth.deleteSession(request["session"].value):
        subtitle = lipiPCSite.KELUAR_BERHASIL
        session = ""
        menu = ""
        menu_link = ""
    else:
        pass
# Session
elif request.has_key("session") and request.has_key("menu") and \
    request["menu"].value != lipiPCSite.MENU_ADMIN[3][3] and \
    request["menu"].value != lipiPCSite.MENU_USER[1][3]:
    if lipiPCAuth.checkSession(request["session"].value):
        session = lipiPCAuth.updateSession(request["session"].value)
        if session != None:
            privilege = lipiPCAuth.getPrivilegeFromSession(session)
    else:
        subtitle = lipiPCError.EXPIRED_SESSION[0]
        error_message = lipiPCError.EXPIRED_SESSION[1]
# Menu
menu = lipiPCSite.getMenuItemFromRequest(request,session)
if menu != "":
    subtitle = menu
    menu_link = request["menu"].value
```

```
# Checking no parameter needed
if request.__len__() == 0 and menu != "":
    subtitle = lipiPCError.UNKNOWN_PAGE_REQUEST[0]
    error_message = lipiPCError.UNKNOWN_PAGE_REQUEST[1]
# Internal Error
elif session == None:
    # subtitle = lipiPCError.INTERNAL_SYSTEM_ERROR[0]
    # error_message = lipiPCError.INTERNAL_SYSTEM_ERROR[1]
    pass
# Checking Menu Utama - no need title
elif menu == lipiPCSite.MENU_UTAMA[0][1]:
    subtitle = ""
##########################################################################
#########

##########################################################################
#########
# Update Link on Menu per Menu Item
lipiPCSite.addSessionPerMenuItem(session)
##########################################################################
#########

##########################################################################
#########
# Content Page or Content Left Part
# Generate time and create form for searching
date_search =
lipiPCHtml.getDateSearch(session,lipiPCUtil.getLocalTime())
# Generating content
if error_message != "":
    from lipi_pc.lipi_pc_html.lipi_pc_content import lipi_pc_error_page
    lipiPCErrorPage =
lipi_pc_error_page.LipiPCErrorPage("..",config_system,lipiPCSite,date_s
earch,request)
    contentPage = lipiPCErrorPage.getErrorPage(error_message)
else:
    if menu == "" or menu_link ==
lipiPCSite.MENU_UTAMA[0][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_halaman_utama
        lipiPCHalamanUtama =
lipi_pc_halaman_utama.LipiPCHalamanUtama("..",config_system,lipiPCSite,
date_search,request)
        if subtitle == lipiPCSite.AKSES_BERHASIL and
request.has_key("username"):
            contentPage =
lipiPCHalamanUtama.getHalamanUtama(request["username"].value)
        else:
            contentPage = lipiPCHalamanUtama.getHalamanUtama()
    elif menu_link == lipiPCSite.MENU_UTAMA[1][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_pemesanan_tempat
        lipiPCPemesananTempat =
lipi_pc_pemesanan_tempat.LipiPCPemesananTempat("..",config_system,lipiP
CSite,date_search,request)
```

```
        contentPage = lipiPCPemesananTempat.getPemesananTempat()
    elif menu_link == lipiPCSite.MENU_UTAMA[2][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_forum_diskusi
        lipiPCForumDiskusi =
lipi_pc_forum_diskusi.LipiPCForumDiskusi("..",config_system,lipiPCSite,
date_search,request)
        if request.has_key("katacari"):
            contentPage =
lipiPCForumDiskusi.getForumDiskusi(request["katacari"].value)
        else:
            contentPage = lipiPCForumDiskusi.getForumDiskusi()
    elif menu_link == lipiPCSite.MENU_UTAMA[3][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_bantuan
        lipiPCBantuan =
lipi_pc_bantuan.LipiPCBantuan("..",config_system,lipiPCSite,date_search
,request)
        contentPage = lipiPCBantuan.getBantuan()
    elif menu_link == lipiPCSite.MENU_UTAMA[4][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_perjanjian_pemakai
        lipiPCPerjanjianPemakai =
lipi_pc_perjanjian_pemakai.LipiPCPerjanjianPemakai("..",config_system,l
ipiPCSite,date_search,request)
        contentPage = lipiPCPerjanjianPemakai.getPerjanjianPemakai()
    elif menu_link == lipiPCSite.MENU_UTAMA[5][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import lipi_pc_e_data
        lipiPCEData =
lipi_pc_e_data.LipiPCEData("..",config_system,lipiPCSite,date_search,re
quest)
        contentPage = lipiPCEData.getEData()
    elif menu_link == lipiPCSite.MENU_ADMIN[0][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_monitoring_cluster
        lipiPCMonitoringCluster =
lipi_pc_monitoring_cluster.LipiPCMonitoringCluster("..",config_system,l
ipiPCSite,date_search,request)
      if privilege == "admin":
            lipiPCMonitoringCluster =
lipi_pc_monitoring_cluster.LipiPCMonitoringCluster("..",config_system,l
ipiPCSite,date_search,request,True)
        contentPage = lipiPCMonitoringCluster.getMonitoringCluster()
    elif menu_link == lipiPCSite.MENU_ADMIN[1][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_manajemen_cluster
        lipiPCManajemenCluster =
lipi_pc_manajemen_cluster.LipiPCManajemenCluster("..",config_system,lip
iPCSite,date_search,request)
        contentPage = lipiPCManajemenCluster.getManajemenCluster()
    elif menu_link == lipiPCSite.MENU_ADMIN[2][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_manajemen_user
```

```
        lipiPCManajemenUser =
lipi_pc_manajemen_user.LipiPCManajemenUser("..",config_system,lipiPCAut
h,lipiPCSite,date_search,request)
        contentPage = lipiPCManajemenUser.getManajemenUser()
    elif menu_link == lipiPCSite.MENU_USER[0][3].split("&")[0]:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_monitoring_cluster
        lipiPCMonitoringCluster =
lipi_pc_monitoring_cluster.LipiPCMonitoringCluster("..",config_system,l
ipiPCSite,date_search,request)
        contentPage = lipiPCMonitoringCluster.getMonitoringCluster()
    else:
        from lipi_pc.lipi_pc_html.lipi_pc_content import
lipi_pc_error_page
        lipiPCErrorPage =
lipi_pc_error_page.LipiPCErrorPage("..",config_system,lipiPCSite,date_s
earch,request)
        subtitle = lipiPCError.UNKNOWN_PAGE_REQUEST[0]
        error_message = lipiPCError.UNKNOWN_PAGE_REQUEST[1]
        contentPage = lipiPCErrorPage.getErrorPage(error_message)
##########################################################################
#########

##########################################################################
#########
# Content Menu of Content Right Part
from lipi_pc.lipi_pc_site import lipi_pc_content_menu
lipiPCContentMenu =
lipi_pc_content_menu.LipiPCContentMenu("..",config_system,lipiPCSite,se
ssion,privilege,menu_link)
contentMenu = lipiPCContentMenu.getContentMenu()
##########################################################################
#########

##########################################################################
#########
# Content of the Page
headTag = lipiPCHtml.getHeadTag();
if config["automatic_refresh"] == "true" and
request.has_key("refresh"):
    headTag = lipiPCHtml.getHeadTag(True);
if contentPage.__len__() == 1:
    bodyTag =
lipiPCHtml.getBodyTag(contentPage[0],contentMenu,config["version"],subt
itle);
else:
    bodyTag =
lipiPCHtml.getBodyTag(contentPage[0],contentMenu,config["version"],cont
entPage[1]);
htmlTag = lipiPCHtml.getHtmlTag(headTag,bodyTag)
##########################################################################
#########

##########################################################################
#########
```

```
# Display to Web
print "Content-Type: text/html\n\n"
print htmlTag
##############################################################################
#########
```