DETEKSI KINERJA PROSESOR KOMPUTER CLIENT DENGAN CARA REMOTE UNTUK MENDUKUNG APLIKASI PEMROSESAN PARALEL

HERO WINTOLO
Jurusan Teknik Informatika
Sekolah Tinggi Teknologi Adisujtipto Yogyakarta
Jl. Janti, Blok R, Lanud Adisutjipto Yogyakarta
herowintolo@gmail.com

Abstract

Processing an instruction made by micro-processor in a personal computer using a queuing mechanism. More and more quantity and quality of instruction to be processed lead time process performed by micro-processors become increasingly longer. Decrease the processing time of this instruction is done by increasing the number of micro-processors used to process instructions together or known as parallel processing. Parallel processing can use a special computer to perform the process, examples of mainframe, or by using a personal computer connected to the computer network Local Area Network. The performance of this processor can be monitored at one point agreed upon as a server. Monitoring the performance of this by designing an application that uses sockets as the pathway to report the client computer processor performance to the applications installed on the server computer. Utilization of client computer processor to support parallel processing performance in a cost that is used in solving problems in parallel processing becomes less expensive than the use of the mainframe.

Keyword: Parallel processing, Monitoring, sockets

1. Latar Belakang Masalah

Jaringan komputer berbentuk Local Area Network (LAN) mengalami perkembang pada perangkat keras dan perangkat lunak. Perkembangan perangkat keras diukur dengan indikator penggunaan media transmisi yang dapat menghantarkan data dari komputer ke komputer yang lain dalam LAN sebesar lebih 1 Gb/ detik. Perkembangan perangkat keras media transmisi juga diikuti perkembangan peralatan jaringan antara lain *LAN Adapter*, *switch* dan *router*. Dukungan perangkat keras jaringan komputer tidak dapat optimal jika tidak didukung dengan perangkat lunak yang handal dan fungsional. Perangkat lunak jaringan komputer sangat terkait dengan sumber daya yang dapat diakses komputer *personal* arau *client* (baca : klien) melelaui perangkat keras jaringan komputer. Dengan perangkat lunak yang beredar, berbasis *open source* atau retail, sebuah komputer dapat digunakan untuk memproses *job* dan instruksi sesuai dengan perlakuan pengguna komputer. Proses, job dan instruksi, terjadi pada unit pembentuk komputer yang bernama *processor* atau dikenal dengan nama *microprocessor*.

Kinerja *processor* dalam mengerjakan *job* dan instruksi dapat dipantau secara *remote* dengan memanfaatkan media transmisi yang menghubungkan antar komputer dalam LAN. Pemantauan jarak jauh (*remote*) ini dilakukan dengan menanamkan program pada sebuah komputer yang difungsikan sebagai klien yang terhubung dengan komputer *server* yang memilki program pemantau melalui sebuah jaringan komputer LAN dengan *topology star*. Komputer *server* yang diberi program pemantau komputer klien secara jarak jauh ini dapat menerima informasi dari sejumlah komputer klien tentang kondisi prosesornya. Dengan mengetahui kondisi prosesor dan RAM(Random Access Memory) ini, diharapkan komputer

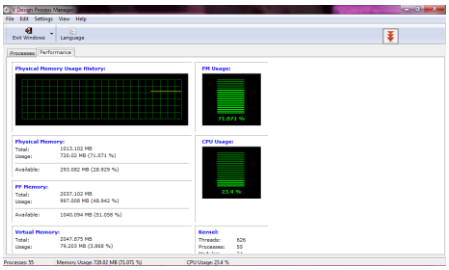
server dapat menerapkan sistem *load balancing* sehingga setiap komputer klien akan mendapatkan *job* dari *server* secara seimbang.

2. Metodologi Penelitian

Penelitian ini menggunakan metode penelitian :

1. Metode Pengamatan (*Observasi*)

Pengamatan dan pencatatan untuk mendapatkan data yang dapat digunakan dalam identifikasi permasalahan. Pengamatan dilakukan terhadap kondisi prosesor dan RAM dari sebuh computer yang tercatat pada aplikasi *windows task manager*. Pengamatan ini dilakukan agar data yang tercatat dalam aplikasi tersebut dapat dipindahkan dalam sebuah program dan dikirimkan ke program yang terpasang di computer lainnya yang terhubung dalam jaringan computer menggunakan socket.



Gambar 1 Tampilan *windows task manager*

2. Metode Studi *Literature*

Studi *literature* dilakukan untuk mendukung data teori yang dapat digunankan dalam proses perancangan perangkat lunak. Sumber utama dari metode ini adalah buku perpustakaan dan jurnal bidang keilmuwan Teknik Informatika atau Ilmu Komputer. Seluruh sumber pustaka didapat dijumpai di perpustakaan STTA Yogyakarta dalam bentuk hard book ataupun soft book. Buku-buku yang digunakan untuk menyusun landasan teori merupakan buku yang kompetensi dalam keilmuwan computer. Sedangkan buku yang digunakan untuk merancang aplikasi menggunakan buku yang bersifat aplikaitf dalam bidang keilmuwan computer.

3. Metode Pengembangan (*Water Fall*)

Pengembangan perangkat lunak yang digunakan untuk sinkronisasi data dalam penelitian ini menggunakan metode *water fall*. Hasil identifikasi permasalahan dianalisa dan dibuat desain awalnya. Setelah selesai maka selanjutnya adalah proses implementasi yaitu pengujian, jika dalam pengujian masih terdapat permsalahan maka akan dilakukan proses ulang identifikasi dan seterusnya sehingga perangkat lunak hasil rancangan sudah dapat dibuktikan kehandalannya.

3. Landasan Teori

3.1 Pemrosesan Paralel

Pemrosesan parallel berbeda dengan pemrosesan tunggal pada sebuah personal komputer. Personal komputer (satu prosesor) yang banyak digunakan pada sebuah perusahaan atau rumahan merupakan peralatan komputer yang digunakan untuk keperluan pengolahan data dengan skala kecil. Pengolahan data dapat berupa pengolahan teks, gambar dan *data base* yang bersifat personal.

Pemrosesan parallel memiliki karakteristik antara lain penggunaan multi prosesor, baik dalam jaringan ataupun sebuah PC (Core 2 Duo), aplikasi pengolah data, dan data yang

memilki ukuran besar. Penggunaan prosesor multi ini untuk meningkatkan kecepatan proses eksekusi yang direpresentasikan dengan nilai *speedup*. Nilai ini merupakan rasio antara eksekusi dengan prosesor tunggal dibandingkan dengan eksekusi menggunakan prosesor sebanyak *n*.

$$\text{Speed up} = \frac{T_1}{T_n} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Dengan

T_1 = waktu yang dibutuhkan untuk mengeksekusi menggunakan satu buah prosesor

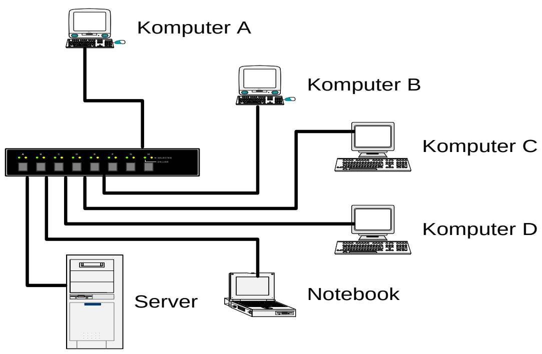
T_n = waktu yang dibutuhkan untuk mengeksekusi menggunakan N buah prosesor

Hasil perhitungan *speed up* ini dapat digunakan untuk menghitung efisiensi dari penggunaan pemrosesan parallel. Efisiensi merupakan perbandingan antara nilai *speed up* yang dihasilkan dari ukuran prosesor (N) yang digunakan untuk mencapai nilai tersebut. Dengan mengetahui efisiensi ini maka peningkatan kecepatan pemrosesan parallel dapat mengacu pada jumlah prosesor yang harus ditambahkan

$$\text{Efisiensi} = \frac{SpeedUp}{N} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

Dengan N = jumlah prosesor yang digunakan

Dari hasil efisiensi ini, selanjutnya dibandingkan dengan jumlah ukuran data yang diolah, yang akan menghasilkan nilai keefektifan sebuah permosesan paralel.



Gambar 2 LAN dengan n buah klien dan 1 server

3.2 Konfigurasi Jaringan Komputer

Penelitian membutuhkan konfigurasi jaringan komputer untuk perancangan aplikasi dan ujicobanya. Jaringan komputer merupakan kumpulan dari komputer yang heterogen dan dihubungkan dengan menggunakan media transmisi serta memiliki standard dalam menghubungkannya. Penggunaan jaringan komputer dalam berbagai bidang telah membuat batas-batas negara menjadi samar. Pertukaran informasi dapat terjadi dalam hitungan detik dari satu titik ke titik yang lain, dari satu negara ke negara lain.

Jaringan komputer dapat dibedakan berdasarkan jarak antar processor komputer yang terhubung dengan media transmisi, antara lain, LAN (*Local Area Network*), MAN (*Metropolitan Area Network*), WAN (*Wide Area Network*), Internet dan Intranet. Penggunaan media transmisi dapat berupa media transmisi terkendali maupun yang tidak terkendali. Media transmisi terkendali antara lain kabel UTP (*Universal Twisted Pair*), kabel *Coaxial*

dan Kabel *Optic*. Adapun media transmisi yang tidak terkendali antara lain sinar *infrared* dan laser, serta gelombang radio.

Jaringan komputer LAN dengan media tarnsmisi kabel UTP dengan *topology star*. Topology merupakan cara yang digunakan dalam menghubungkan komputer membentuk jaringan komputer. Ada 3 topology yaitu : bus, ring dan star. Dalam penelitian ini topology yang star, seperti terlihat pada gambar 2, dengan menambahkan peralatan jaringan komputer yang bernama *switch*. *Switch* jaringan (atau switch untuk singkatnya) adalah sebuah alat jaringan yang melakukan bridging transparan (penghubung segementasi banyak jaringan dengan forwarding berdasarkan alamat MAC). Switch juga merupakan penghubung beberapa alat untuk membentuk suatu *Local Area Network* (LAN). Fungsi dan cara kerja *Switch* :

- Bekerja di lapisan Data Link
- Setiap port didalam *switch* memiliki domain collision sendiri-sendiri
- Memiliki tabel penterjemah pusat yang memiliki daftar penterjemah untuk semua port
- Memungkinkan transmisi secara full duflex (dua arah)

3.3 Socket Programming

Pemrograman komputer merupakan cabang dari keilmuan bidang komputer science, dengan menggunakan program yang dibuat oleh seseorang yang dikenal dengan nama programmer, komputer yang merupakan susunan perangkat keras yang terorganisir secara rapi, dapat digunakan sesuai dengan kebutuhan manusia dalam menyelesaikan permasalahan. Karena program yang dibuat oleh seorang programmer akan menjadi aplikasi yang *user friendly* dan dapat dikembangkan sesuai dengan spesifikasi perangkat keras komputer.

Pemrograman socket termasuk dalam pemrograman komputer tingkat rendah. Pemrograman tingkat rendah disini diartikan sebagai pemrograman bahasa komputer yang mudah dipahami oleh komputer, tetapi sulit dipahami oleh manusia. Hal ini mengakibatkan bahasa komputer yang mendukung pemrograman tingkat rendah tidak membutuhkan compiler yang baik dibandingkan dengan bahasa pemrograman tingkat tinggi.

Program yang dituliskan dengan bahasa pemrograman untuk membuat socket yang berguna untuk komunikasi antara satu komputer dengan komputer yang lainnya yang terhubung dalam jaringan komputer. Socket-socket ini akan sangat berguna dalam pengiriman dan penerimaan pesan yang menjadi tulang punggung pada aplikasi untuk pemrosesan parallel.

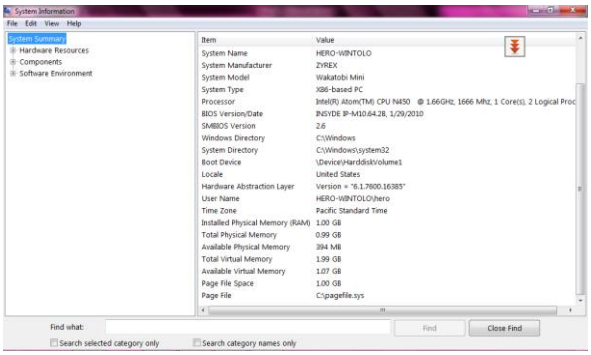
4. Perancangan Perangkat Lunak

Perancangan perangkat lunak dilakukan dengan menginisialisasi kebutuhan perangkat keras dan perangkat lunak serta sumber daya jaringan komputer yang tersedia, sehingga hasil dari perancangan dapat langsung diujicobakan, jika masih ada permasalahan akan segera diperbaiki.

4.1 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras komputer untuk mendukung perancangan perangkat lunak ada 7 unit, satu unit digunakan untuk *Server* dan komputer yang lainnya digunakan untuk penempatan perangkat lunak *client*. Spesifkasi komputer *client* tersebut antara lain : Komputer dengan *processor core2duo*, memori (ram) 512 mb dan hd 40 gb, monitor, *keyboard*, *mouse*, *port usb* dan rj 45

Sedangkan spesifikasi komputer *server* yang digunakan untuk merancang aplikasi yang akan digunakan di *server* atau *client*, dapat dilihat pada gambar 3.



Gambar 3 Spesifikasi komputer *Server*

4.2 Kebutuhan Perangkat Lunak

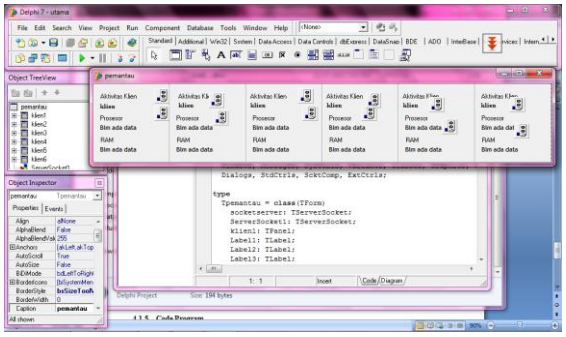
Perancangan aplikasi yang akan digunakan dalam pada penelitian ini membutuhkan perangkat lunak delphi dan komponen *socket* dan sistem operasi ms windows 7

4.3. Peralatan Jaringan Komputer

Penelitian ini membutuhkan peralatan jaringan komputer yang akan digunakan dalam proses uji coba hasil perancangan perangkat yang menghubungkan antar komputer dalam sebuah jaringan komputer, peralatan tersebut adalah *Switch*.

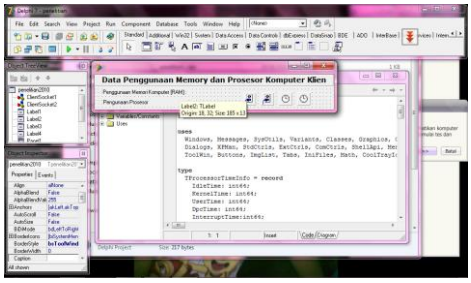
4.4 Perancangan Aplikasi

Aplikasi ini dirancang dalam dua bentuk, bentuk pertama untuk *server* dan bentuk kedua untuk klien. Tampilan aplikasi untuk *server* dapat dilihat pada gambar 4 dan tampilan aplikasi untuk klien dapat dilihat pada gambar 5.



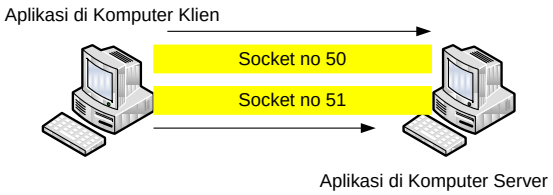
Gambar 4 Rancangan Aplikasi *Server*

Pada tampilan aplikasi untuk *server*, terlihat ada 6 kotak yang didalamnya memiliki kesamaan, hal ini dimasudkan agar data yang dikirimkan dari 6 komputer klien yang akan terhubung ke aplikasi ini akan diterima masing-masing kotak tersebut untuk memudahkan dalam pengamatannya.



Gambar 5 Rancangan Aplikasi Klien

Rancangan aplikasi untuk klien dibuat dengan menampilkan kondisi prosesor dan RAM komputer klien yang akan dikirimkan datanya menggunakan *socket* dengan nomor yang sama pada sockter yang ada di *server* secara esklusif, hal ini dimasudkan agar pada uji cobanya nanti terlihat aliran data yang jelas dan sama antara yang tertampil di klien dan *server* .



Gambar 6 Ilustrasi hubungan antara aplikasi klien dan *server*

Pada aplikasi klien dipasang 2 buah *socket* dengan nomor berbeda dan sama dengan *socket* yang dipasang di *server* , agar *socket-socket* ini dapat terhubung dan mengirimkan data dari klien ke *server* . Pada gambar 6, *socket* nomor 50 pada klien terhubung dengan *socket* nomor 50 pada *server* , kemudian *socket* nomor 51 pada klien terhubung dengan *socket* nomor 51 pada *server* .

Pada landasan teori, sub bab pemrograman *socket*, dijelaskan bahwa *socket* merupakan bahasa tingkat rendah, sehingga komputer dapat saling berkomunikasi dengan komputer yang lainnya menggunakan perangkat keras jaringan komputer yang mendukung *socket* tersebut. *Socket* harus memiliki nomor yang unik dan tidak boleh sama dengan nomor yang sudah ada jika *socket* tersebut akan berkomunikasi dengan *socket* yang terpasang pada komputer yang lainnya.

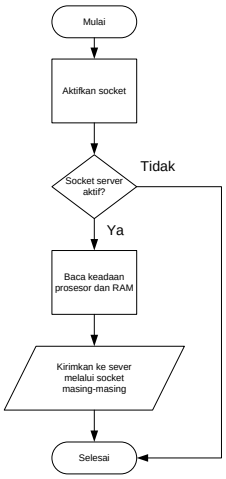
5. Algoritma dan Code Program

Algoritma dari aplikasi klien diujudkan dalam bentuk flowchart seperti terlihat pada gambar 6. Aplikasi yang di komputer klien akan dapat berkerja mengirimkan data ke komputer *server* jika aplikasi yang ada di komputer *server* sudah dijalankan.

5.1 Code Program Aplikasi Klien

Prosedur membaca keadaan prosesor

```
1.    procedure Tpenelitian2010.Timer2Timer(Sender: TObject);
2.    var
3.    CPUUsage : extended;
4.    Buffer : Array[0..299999] Of Char;
5.    TimeInfo : ^TProcessorTimeInfo;
6.    begin
7.    NtQuerySistemInformation(8,@Buffer,300000,0);
8.    TimeInfo := @Buffer;
9.    S2 := TimeInfo^.KernelTime + TimeInfo^.UserTime - TimeInfo^.IdleTime;
10.   CPUUsage := ((S2-S1) / 100000);
11.   S1 := S2;
12.   CUsage := Min(CPUUsage,100);
13.   if (CUsage >= 1) then
14.   label4.caption := CPUUsageN + ' ' + FormatFloat('####.###', CUsage) + ' %'
15.   else
16.   if (CUsage < 1) and (CUsage > 0) then
17.   label4.caption := CPUUsageN + ' 0' + FormatFloat('####.###', CUsage) + ' %'
18.   else
19.   if (CUsage = 0) then
20.   label4.caption := CPUUsageN + ' 0.000 %';
21.   client*socket*1.*Socket*.SendText(label4.Caption);
22.   end;
```



Gambar 6 Flowchart sistem pada aplikasi klien

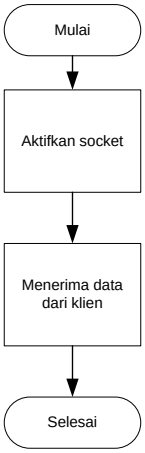
Lima belas baris code program pada prosedur membaca keadaan prosesor termasuk bagian utama dari program yang dijalankan pada komputer klien. Keadaan prossor yang dibaca terlihat mulai baris 7 hingga baris 12. Keadaan prosesor yang ditampilkan dalam aplikasi ini, jika dilihat dari code program merupakan pemanfaatan dari sistem informasi yang dimiliki oleh sistem operasi. Hasil pembacaan keadaan prosesor dimasukan pada variable `label4.caption` dan dikirimkan ke aplikasi *server* menggunakan perintah `client`*`socket`*`1.`*`Socket`*`.SendText(label4.Caption).`

Prosedur membaca keadaan RAM

```
1.  procedure Tpenelitian2010.Timer1Timer(Sender: TObject);
2.  var
3.  Memory: TMemoryStatus;
4.  R, P: Extended;
5.  begin
6.  Memory.dwLength := SizeOf(Memory);
7.  GlobalMemoryStatus(Memory);                    { XXXX(in bytes) / (1024.0 * 1024) =
    XXXX(in Megabytes) }
8.  R := (Memory.dwTotalPhys / (1024.0 * 1024)) - Memory.dwAvailPhys / (1024.0 * 1024));
9.  P := ((Memory.dwAvailPhys / (1024.0 * 1024)) / (Memory.dwTotalPhys / (1024.0 *
    1024))) * 100;
10. P := 100 - P;
11. label1.Caption:= MemUsage + ' ' + FormatFloat('####.###', R) + ' MB (' +
    FormatFloat('###.###', P) + ' %)';
12. client socket2.Socket.SendText(label1.Caption);
13. end;
```

Prosedur membaca keadaan RAM yang terdiri dari 13 baris code program, program yang menunjukan pembacaan RAM dimulai dari baris 6 hingga baris ke 10. Setelah keadaan RAM dapat dibaca, selanjutnya melalui client*socket*2, data tersebut dimasukan dalam variable label1.caption dan dikirm ke aplikasi *server* dengan perintah `client`*`socket`*`2.`*`Socket`* `.SendText(label1.Caption).`

Pada gambar 7, terlihat bahwa aplikasi *server* hanya bekerja dengan cara menerima data yang dikirimkan dari setiap kompter klien yang terhubung ke aplikasi *server* . Data yang diterima sama dengan data yang ada pada komputer klien secara *real time*. Akibatnya aplikasi serever harus dijalankan pertama kali dibandingkan dengan aplikasi klien dan ditutup paling akhir setelah aplikasi klien ditutup.



Gambar 7 *Flowchart* sistem pada aplikasi *server*

5.2 Code Program Aplikasi *Server*

Prosedure membaca *socket* untuk menerima data prosesor dari aplikasi klien

```
1.  procedure Tpemantau.socketserver ClientRead(Sender: TObject;
2.  Socket: TCustomWinSocket);
3.  begin
4.  label3.Caption:=(socket.ReceiveText);
5.  end;
```

Prosedure membaca *socket* untuk menerima data RAM dari aplikasi klien

```
1.  procedure Tpemantau.Server Socket1ClientRead(Sender: TObject;
2.  Socket: TCustomWinSocket);
3.  begin
4.  label4.Caption:=(socket.ReceiveText);
5.  end;
```

Prosedure membaca *socket* untuk menerima data nama *host* dari aplikasi klien

```
1.  procedure Tpemantau.socketserver ClientConnect(Sender: TObject;
2.  Socket: TCustomWinSocket);
3.  begin
4.  label6.Caption:=socket.RemoteHost;
5.  end;
```

Code program yang dituliskan pada aplikasi yang digunakan komputer *server* yang terdiri dari 3 item seperti diatas diulang sebanyak jumlah komputer klien yang akan terhubung dengan aplikasi *server* tersebut, jika jumlah klien yang terhubung 6 maka ketiga prosedur tersebut ditulis sebanyak 6 kali. Prosedur yang dituliskan menggunakan code program diatas pada prinsipnya hanya menerima data dari aplikasi klien yang mengirimkan datanya melalui sebuah *socket* yang telah diberi nomor yang unik. Penerimaaan data dari aplikasi klien ditulis dengan code program *socket*.RemoteHost, sebagai contoh untuk menrima data dari aplikasi klien tentang nama host dari klien yang terhubung ke aplikasi *server* .

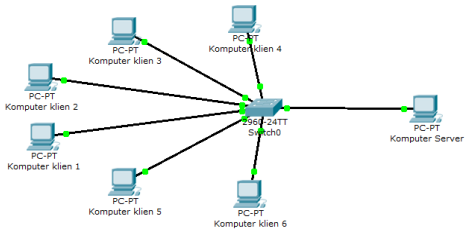
5.3 Cara Uji Coba

Pengujian aplikasi hasil rancangan ini dilakukan dalam sebuah jaringan komputer LAN, dengan enam buah komputer diposisikan sebagai klien dan laptop yang diposisikan sebagai *server* yang akan menerima data kondisi prosesor dan RAM komputer klien. Aplikasi yang dipasangkan pada setiap komputer klien akan terhubung ke aplikasi yang dipasangkan pada komputer *server* menggunakan *socket* yang dikhususkan untuk menerima data kondisi prosesor dan RAM. Setelah itu, aplikasi *server* harus dijalankan terlebih dahulu, karena *socket* yang ada pada aplikasi ini bersifat pasif, sedangkan aplikasi pada komputer

klien dijalankan setelah aplikasi *server* berjalan, karena *socket* pada aplikasi klien aktif dalam mengirimkan data. Jika hal ini dibalik kondisinya akan menimbulkan pesan kesalahan pada komputer klien yang gagal mengirimkan data ke *server* .

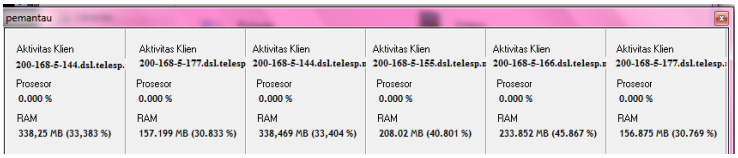
6. Uji Coba

Uji coba dilakukan pada jaringan komputer LAN, dengan enam buah komputer berposisi sebagai klien dengan konfigurasi seperti gambar 8. Seluruh komputer yang digunakan dalam uji coba, baik klien atau *server* , diberi alamat (IP address versi 4) dengan alamat jaringan kelas C yaitu 200.168.5.xxx dan alamat host 1 hingga 7, tetapi nama grup tidak sama.



Gambar 8 Konfigurasi LAN dalam ujicoba

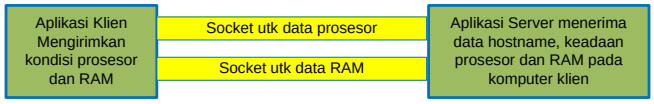
Setelah IP diberikan pada setiap komputer, selanjutnya aplikasi sisi *server* diaktifkan terlebih dahulu, lihat gambar 9. Setelah itu aktif, selanjutnya aplikasi sisi klien diaktifkan, hal ini dilakukan supaya tidak muncul pesan kesalahan pada aplikasi klien karena mencari *socket* yang terhubung ke aplikasi *server* . Karena penggunaan *socket* yang khusus, satu *socket* melayani satu data, maka kecepatan aliran data dari komputer klien ke komputer *server* dalam hitungan kurang atau sama dengan satu detik. Selain itu, data kegiatan dari prosesor dan RAM yang selalu berubah-ubah, dikarenakan dalam posisi idle atau sedang memproses sesuatu, maka data yang dikirimkan akan menyusuaikan. Artinya data yang tampil pada aplikasi *server* selalu berubah secara real time.



Gambar 9 Tampilan aplikasi sisi *server*

7. Analisa Hasil Uji Coba

Dari hasil ujicoba yang dilakukan dan diilustrasikan pada gambar 10, aliran data dari aplikasi yang dipasangkan pada komputer klien akan mengalir ke aplikasi yang dipasangkan pada komputer *server* melalui *socket*. *Socket* sebanyak 2 pasang ini melayani perjalanan data dari komputer klien yaitu data hostname, data kinerja prosesor dalam prosentase dan data kinerja RAM dalam bentuk satuan MB dan prosentase, ke aplikasi yang dipasangkang di komputer *server* . Dari enam buah komputer klien yang terhubung ke komputer *server* , semuanya berjalan sesuai dengan rumusan permasalahan dari penelitian ini, hasil selengkapnya dapat dibaca pada table 1.



Gambar 10 Ilustrasi hubungan antara aplikasi yang berada pada klien dengan *server*

Tabel 1 Hasil Ujicoba

No	Nama Komputer Klien	*Socket* P	*Socket* RAM	Data di *Server*
1	Komputer 1	50	51	Ada
2	Komputer 2	52	53	Ada
3	Komputer 3	54	55	Ada
4	Komputer 4	56	57	Ada
5	Komputer 5	58	59	Ada
6	Komputer 6	60	61	Ada

Komputer 1 menggunakan *socket* nomor 50 akan memberikan data nama host kepada aplikasi *server* , selain itu dengan *socket* yang sama, data keadaaan prosesor dikirmkan ke aplikasi *server* . Sedangkan data kondisi RAM dikirimkan menggunakan *socket* nomer 51 ke aplikasi *server* . Pada komputer *server* , *socket* nomer 50 dibuka/ diaktifkan untuk menrima data dari komputer klien yang menggunakan *socket* nomor ini, begitu pula pada *socket* nomer 51, dibuka/diaktifkan untuk menerima data dari komputer klien yang menggunakan *socket* nomor ini. Hal ini terulang juga pada komputer klien nomor 2 hingga 6 dengan *socket* yang terurut. Pengurutan nomor *socket* untuk setiap komputer klien dengan memasukan nomor *socket* yang digunakan untuk memantau prosesor saat aplikasi klien dijalankan, sedangkan *socket* yang digunakan untuk memantau RAM, dimasukan dalam program dengan algoritma menambahkan satu data bertipe integer ke data nomor *socket* yang diisikan, sehingga secara otamatis akan muncul angka dengan selisih satu.

8. Kesimpulan Dan Saran

8.1 Kesimpulan

Dari hasil ujicoba yang dilakukan pada bab empat dari penelitian ini, maka dapat disimpulkan :

1. Aplikasi yang dibuat dan diujicobakan, baik pada komputer klien dan server, tidak ada kesalahan dalam bentuk *syntax* atau *semantic*.
2. Aplikasi yang dipasang pada komputer klien dapat mengirimkan data nama komputer, kondisi prosesor dan RAM ke aplikasi yang dipasangkan pada komputer server menggunakan socket yang diberi nomor unik.

8.2 Saran

Penelitian ini dapat dilanjutkan untuk mendapatkan data keadaan prosesor setiap komputer klien yang sedang melakukana proses secara terurut guna mendukung basis ilmu dalam bidang pemrosesan parallel.

9. DAFTAR PUSTAKA

Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar, *Introduction to Parallel Computing*, Second Edition, Addison Wesley,2003

Hesham El-Rewini, Mostafa Abd-El-Barr, *Advanced Computer Architecture And Parallel Processing*, A John Wiley & Sons, Inc Publication, 2005

Ibaroudene, Djaffer, "*Parallel Processing, EG6370G: Chapter 1, Motivation and History*." St Mary's University, San Antonio, TX. Spring 2008

Quinn, Michael J, *Parallel Programming in C with MPI and Open MP*, Boston: McGraw Hill, 2004