Penerapan *Quick Sort* pada Pemrosesan Parallel Berbasis *Personal Komputer* (PC) dalam *Local Area Network* (LAN) Guna Mendukung *Load Balancing*

Hero Wintolo
Jurusan Teknik Informatika
Sekolah Tinggi Teknologi Adisutjipto Yogyakarta
herowintolo@stta.ac.id

Abstract

Aplikasi yang digunakan dalam pemrosesan parallel yang berbasis PC dalam LAN dapat berjalan dengan menggunakan *socket* yang saling dihubungkan antar PC tersebut. Pemrosesan ini terjadi ketika komputer PC yang bertindak sebagai *server*, mengirimkan sejumlah instruksi untuk diproses oleh komputer *client* yang tingkat kesibukan prosesornya dapat dibaca dari sisi *server*.

Data indicator proses setiap PC tersebut belum terurut sehingga dapat menimbulkan pembebanan kerja eksekusi instruksi menjadi tidak seimbang. Untuk menyeimbangkan dibutuhkan metode pengurutan data kondisi prosesor *client* dengan menggunakan metode *Quick Sort*. Metode ini berkerja dengan cara membandingkan setiap sisi kanan dan kiri dari rangkaian data yang akan diurutkan dengan menemukan *pivot* terlebih dahulu.

Hasil pengurutan menggunakan metode *Quick Sort* ini dapat digunakan untuk load balancing dari instruksi yang akan dikirimkan oleh *server* ke komputer *client* untuk dilakukan sebuah proses, sehingga keseimbangan kinerja dari aplikasi pemrosesan paralel ini dapat dicapai.

Kata kunci : *Quick Sort,* Pemrosesan Paralel, *Load Balancing*

Abstract

Applications that use parallel processing in a LAN-based PC can run by using a socket that is interconnected between PCs. Processing occurs when the PC acting as a server, send some instructions to be processed by the client computer processor that level of activity can be read from the server side.

Data indicator process every PC has not been sequenced so as to cause the work load becomes unbalanced instruction execution. To balance the needs sorting method client condition

data processor using the Quick Sort. This method works by comparing each of the right and left of the series of data to be sorted to find the first pivot.

The results of sequencing using Quick Sort method can be used for load balancing of the instructions that will be sent by the server to the client computer to do a process, so that the balance of the performance of parallel processing applications can be achieved.
Keywords: Quick Sort, Parallel Processing, Load Balancing

1. Latar Belakang Masalah

Penggunaan prosesor komputer untuk menyelesaikan berbagai macam proses dari sebuah perangkat lunak pada sebuah PC tanpa terhubung dan terhubung dengan LAN secara kuantitatif tidak banyak berubah sejak era system operasi D Operating System (DOS) yang berbasis teks hingga system operasi berbasis x windows. Penyebabnya adalah para konsumen cenderung menggunakan prosesor pada sebuah PC dengan perangkat lunak yang disesuaikan dengan kebutuhannya sehari-hari, misal penggunaan pengolah kata dan tabel yang cenderung tidak membutuhkan tambahan yang revolusioner. Contoh, dalam mengerjakan pengetikan dokumen, pengguna komputer lebih nyaman jika menggunakan perangkat lunak yang biasanya digunakan, jika pabrikan perangkat lunak mengeluarkan produk yang sama dengan berbagai macam tambahan yang tidak terlalu fungsional, maka konsumen akan tetap bertahan dengan perangkat lunak yang lama.

Secara kualitatif penggunaan prosesor komputer dalam menyelesaikan proses dari sebuah perangkat lunak mengalami banyak perubahan, bahkan ada beberapa kecurigaan adanya kerjasama dalam pengembangan kecepatan prosesor dengan peningkatan produk perangkat lunaknya. Bagi para pengguna komputer yang sangat terpengaruh perangkat lunak produksi terbaru akan sangat membutuhkan kinerja prosesor yang setara atau lebih tinggi dari persyaratan untuk menjalankan perangkat lunak tersebut.

Pemrosesan parallel berbasis PC dalam LAN atau yang lebih besar skalanya, dikenal juga sebagai *Grid Computing* yang perkembangannya tidak secepat system terdistribusi dan system terpusat. Hal ini disebabkan oleh persyaratan penggunaan *Grid Computing* yang lebih *exclusive* dibandingkan yang lainnya. Dengan sejumlah PC dalam LAN, seseorang dapat membuat aplikasi *client server* yang termasuk dalam lingkup system terdistribusi, dan membuat aplikasi WEB yang digunakan dalam system intranet untuk mendukung system informasi yang terpusat.

Prosesor pada setiap PC yang terhubung dalam LAN ini dapat digunakan untuk pemrosesan parallel yang mendukung penyelesaian sebuah permasalahan yang membutuhkan kinerja prosesor dan dilakukan secara serentak, sehingga efisiensi prosesor pada *Grid Computing* dapat dicapai lebih baik dibandingkan yang lainnya.

2. Pemrosesan Paralel Berbasis PC

Grid Computing

Pemrosesan parallel menghasilkan *speed up* dan efisiensi yang dapat digunakan sebagai acuan penambahan prosesor untuk mempercepat waktu proses. Waktu untuk memproses sebuah data tidak mutlak tergantung pada berapa prosesor yang ada dan digunakan dalam sebuah proses, tetapi masih ada celah untuk menutup kebutuhan prosesor yang banyak dengan penggunaan sumber daya perangkat lunak yang pada akhirnya akan membawa kesan berbiaya murah dibandingkan penambahan prosesor.

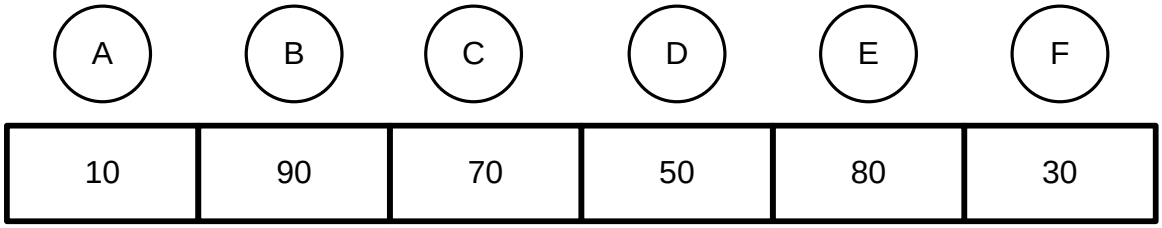
Perangkat lunak yang digunakan dalam perosesan parallel harus cocok dalam konfigurasi sistemnya, jika tidak maka pemrosesan parallel menjadi sangat mahal karena perangkat keras dan lunaknya mengharuskan bersifat homogen. Selain itu karakterisktik permasalahan yang akan didekati dengan penyelesaian komputasi parallel bersifat eksklusif. Dengan mengadakan sumber daya manusia yang memilki kemampuan dalam merancang perangkat lunak untuk pemrosesan parallel maka biaya yang harus dikeluarkan sebuah perusahaan hanya untuk pengadaan perangkat keras komputer dan jaringannya. Perangkat lunak cukup dibuat sendiri untuk mengatasi permasalahan perusahaan yang membutuhkan penyelesaian secara serentak dengan cabang yang banyak. Disinilah peran dari Komputasi Grid itu diperlukan.

Paradigma *grid computing* merupakan suatu paradigma komputasi tersebar yang memungkinkan pembagian, pengaturan, dan penggunaan sumber daya komputasi yang tersebar dari beragam posisi geografis secara efektif dan efisien. Sistem *grid computing* mengatur pembagian penggunaan sumber daya yang heterogen, dengan *platform* yang berbeda-beda, dan umumnya tersebar ditempat yang berbeda-beda serta dalam domain administratif yang berbeda pula. Sistem *grid* computing atau sistem komputasi *grid* telah banyak diterapkan dalam berbagai organisasi di dunia. Beberapa contoh organisasi yang telah menerapkan sistem komputasi *grid* adalah National Science Foundation's National Technology *Grid*, NASA's Information Power *Grid*, Pratt & Whitney, Bristol-Myers Squibb Co., dan American Express.

Quick Sort

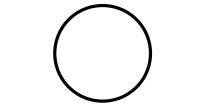
Quick sort merupakan salah satu metode yang digunakan dalam teknik pengurutan data yang dikenal sebagai *divide and conquer algorithm*, merupakan salah satu metode yang digunakan untuk pengurutan data. *Quick sort* dimulai dengan melakukan penyeleksian untuk menentukan nilai median. Nilai ini, yang disebut tumpuan (pivot), kemudian dipindahkan ke satu sisi pada bagian data yang terbagi oleh tumpuan, jika data memiliki besaran yang lebih besar dari tumpuan maka data akan di pindahkan ke sisi lain, dengan kata lain data yang bernilai kecil berada pada sisi kiri sedangkan data yang bernilai lebih besar berada pada sisi kanan, atau sebaliknya.

Pada tulisan ini, metode pengurutan *quick sort* dari 6 buah data kondisi prosesor yang digunakan untuk melakukan pemrosesan secara parallel. Kondisi yang dimaksud disini adalah prosesor dari setiap PC yang terhubung pada LAN akan diurutkan rankingnya berdasarkan keadaan prosesor dalam atau sedang melakukan sebuah proses atau tidak dengan cara remote[Hero Wintolo, 2010]. Ilustrasi kecepatan 6 buah prosesor ini dapat dilihat pada gambar 1.



Gambar 1 Posisi awal kondisi 6 buah prosesor

Keterangan gambar



= Prosesor

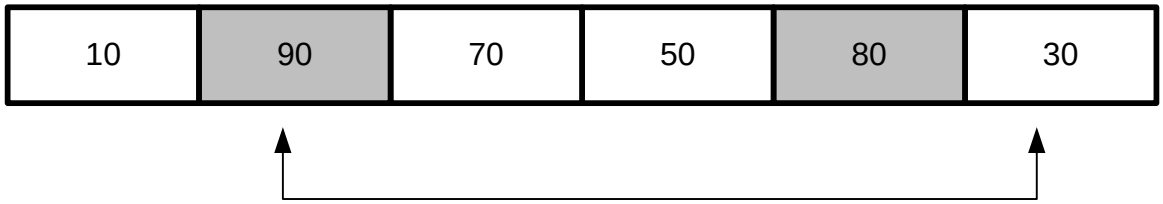
= Kondisi Proses dari prosesor

Dari keadaan yang didapat berdasarkan gambar1, maka dibutuhkan langkah-langkah yang dilakukan pada metode quick sort yaitu :

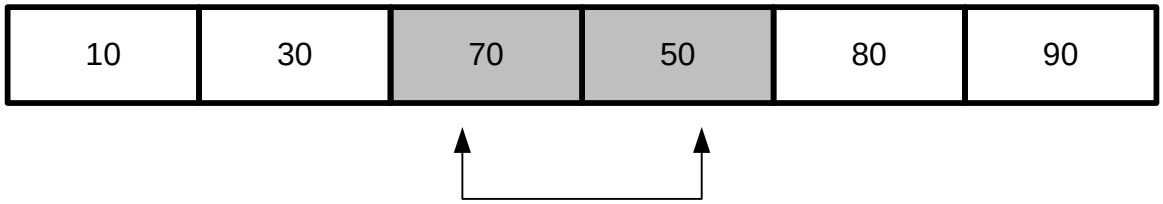
1. Seleksi

   Seleksi dari gambar 1, digunakan untuk mencari pivot dengan menggeser elemen dimulai dari elemen yang paling kiri ke arah kanan dan elemen yang paling kanan ke arah kiri dengan cara membandingkan nilain elemen terlebih dahulu jika elemen kiri lebih besar

dari yang kanan maka tukarkan, jika tidak, maka tidak perlu ditukarkan. Ilustrasi dari proses seleksi ini dapat dilihat pada gambar 2.



Dari kiri begeser ke kanan dan yang kanan bergeser ke kiri, setelah itu dibandingkan untuk yang lebih kecil diletakkan di kiri dan yang besar di kanan



Dari kiri begeser ke kanan dan yang kanan bergeser ke kiri, setelah itu dibandingkan untuk yang lebih kecil diletakkan di kiri dan yang besar di kanan

PIVOT

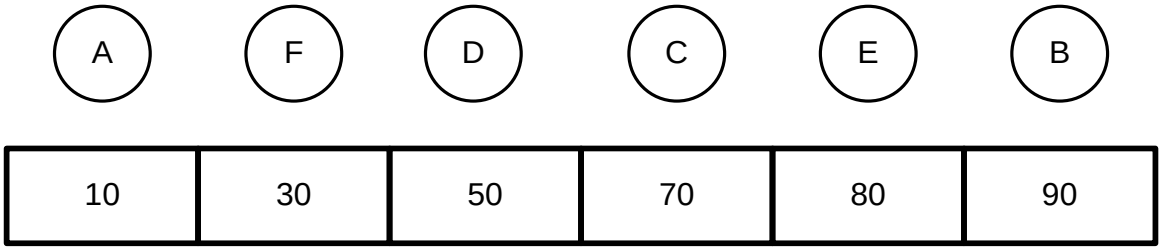
Gambar 2 Ilustrasi proses seleksi hingga didapatkan pivot

2. Devide

Proses ini sebetulnya adalah akhir dari proses seleksi yang menghasilkan elemen yang disebut sebagai pivot yang berada diantara dua bagian elemen atau titik tengah dari sekumpulan elemen. Lihat gambar 2.

3. Recur dan Conquer

Recur dan conquer merupakan proses pengurutan elemen-elemen yang berada pada sisi kanan dan kiri dari pivot dengan menggunakan mekanisme *recursive*. Sehigga akan didapat hasil yang dapat dilihat pada gambar 3.

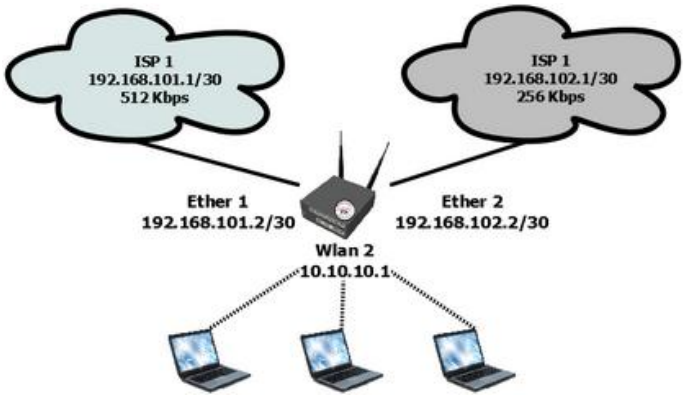


Gambar 3 Hasil pengurutan data dengan menggunakan *quick sort*

Dari hasil pengurtan terhadap kondisi 6 buah prosesor ini, didapatkan hasil bahwa proses A, F, D, C, E dan B merupakan urutan kondisi dan keadaan prosesor yang akan diberikan instruksi untuk mengerjakan proses selanjutnya dari komputer *server*, dengan mekanisme ini diharapkan terjadinya proses *load balancing*.

Load Balancing

Load balancing merupakan mekanisme penyeimbangan beban infrastrukur teknologi informasi yang terkait dengan jaringan komputer agar tercapainya optimaliasasi dan pemanfaatan sumber daya secara maksimal. Kita ambil contoh mekanisme *load balancing* dalam pembagian koneksi internet yang memanfaatkan 2 atau lebih provider yang akan digunakan secara bersama-sama oleh semua PC yang terhubung dalam LAN, lihat gambar 4.



.

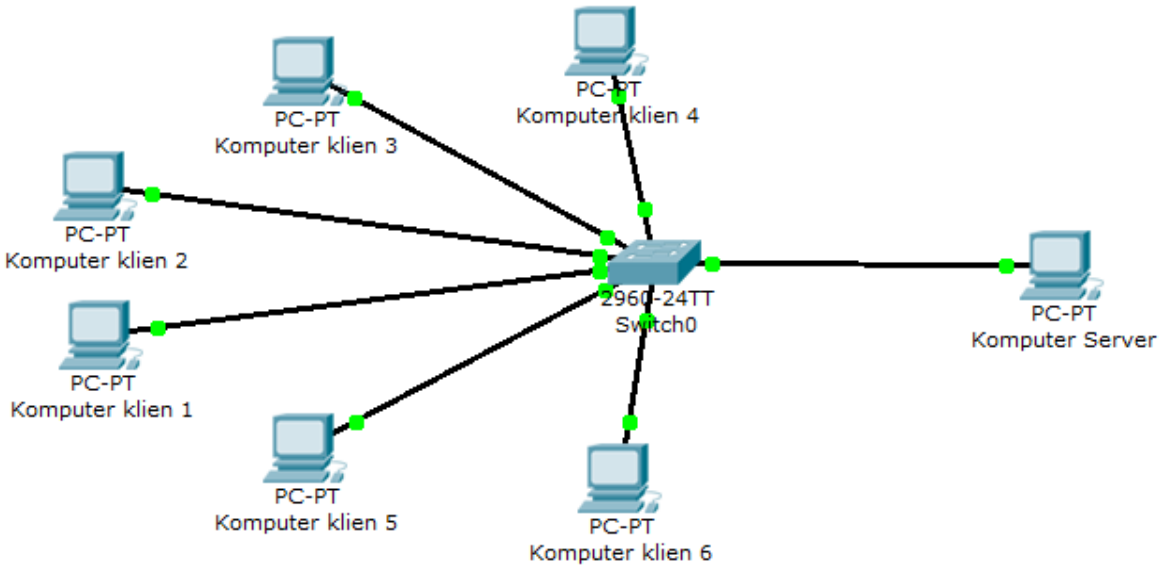
Gambar 4 *Load balancing* untuk menjumlahkan 512+256 dan membagikannya pada sejumlah komputer

Bandwidth yang didapat dari 2 buah provider masing masing masing-masing bernilai 512 Kbps dan 256 Kpbs akan dijumlahkan, tetap bukan berarti 512+256= 768 Kbps, tetapi 512+256 = 256+256+256 Kbps. Dengan kondisi seperti ini, maka setiap komputer *client* yang terhubung ke

jaringan internet akan mendapatkan jatah *bandwidth* yang sama, dengan cara membagi sejumlah komputer yang terhubung.

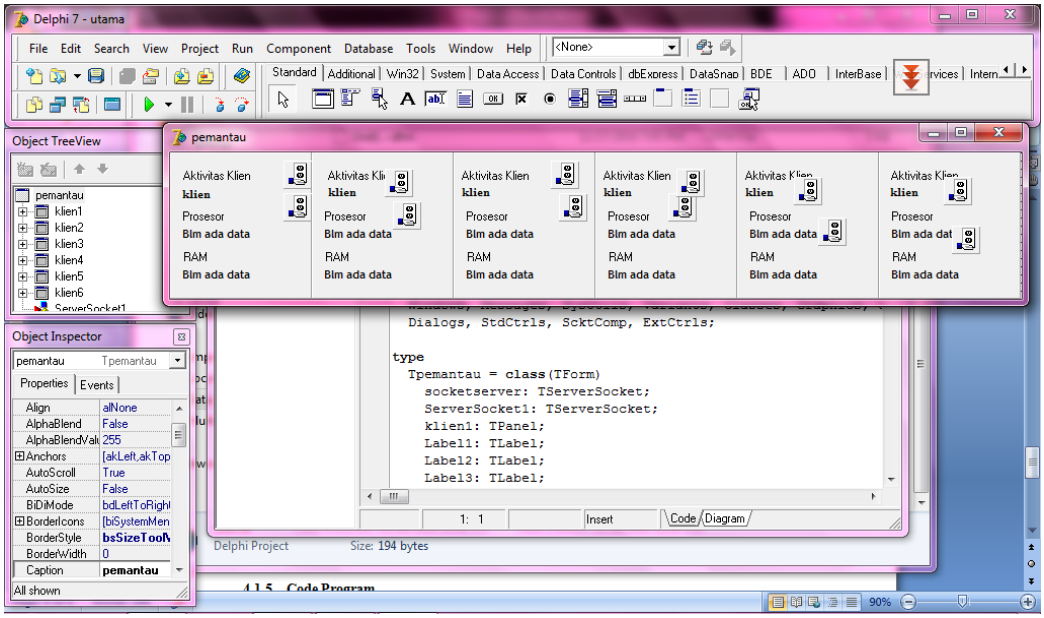
Berdasarkan penjelasan ini, maka dalam penelitian ini proses pengurutan kondisi prosesor yang akan dilakukan oleh komputer *server* menjadi sangat penting sekali, karena setelah terjadinya pengurutan maka *server* dapat memberikan perintah kepada komputer *client* yang kondisi prosesornya paling kecil prosesnya untuk melakukan sebuah proses yang dikirimkan dari *server*, sehingga *load balancing* dapat tercapai.

3. Rancangan Aplikasi

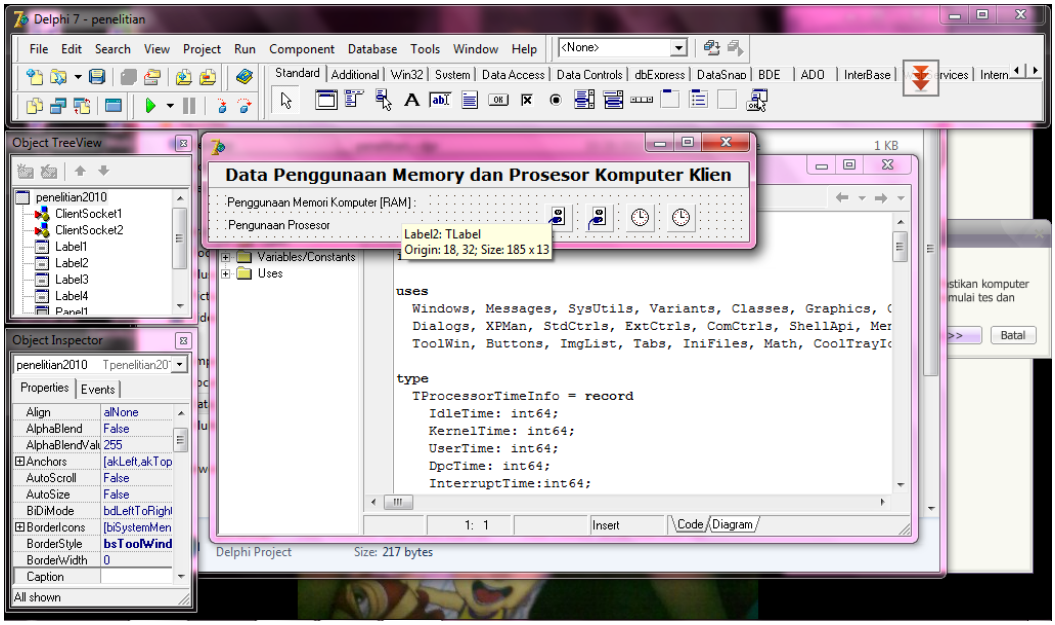


Gambar 5 Konfigurasi LAN dalam ujicoba

Aplikasi yang dirancang dalam tulisan ini merupakan pengembangan dari penelitian sebelumnya dengan menambahkan prosedur pengurutan menggunakan metode *quick sort*. Pada aplikasi sisi *server* rancangan aplikasi dapat dilihat pada gambar 6, sedangkan untuk sisi *client* dapat dilhat pada gambar 7.



Gambar 8 Rancangan Aplikasi *Server*



Gambar 7 Rancangan Aplikasi *Client*

Pada aplikasi sisi *server* ditambahkan sebuah prosedur untuk mengurutkan data kondisi setiap prosesor *client* yang sedang mengerjakan proses yang dikirim dari komputer *server*. Prosedur pengurutan ini menggunakan algoritma pengurutan data *quick sort*. Prosedur tersebut adalah sebagai berikut

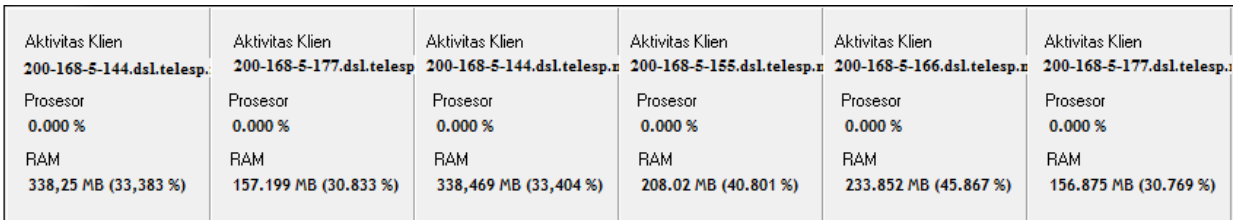
```
1.    procedure QS(var Q: array of Integer; rendah, tinggi: Integer) ;
2.    var
```

```
3.    nilairendah, nilaitinggi, Pivot, a: Integer;
4.    begin
5.    nilairendah := rendah;
6.    nilaitinggi := tinggi;
7.    Pivot := Q[(nilairendah +nilaitinggi) div 2];
8.    repeat
9.    while Q[nilairendah] < Pivot do Inc(nilairendah) ;
10.   while Q[nilaitinggi] > Pivot do Dec(nilaitinggi) ;
11.   if nilairendah <= nilaitinggi then
12.   begin
13.   a := Q[nilairendah];
14.   Q[nilairendah] := Q[nilaitinggi];
15.   Q[nilaitinggi] := a;
16.   Inc(nilairendah) ;
17.   Dec(nilaitinggi) ;
18.   end;
19.   until nilairendah > nilaitinggi;
20.   if nilaitinggi > rendah then QuickSort(Q, rendah, nilaitinggi) ;
21.   if nilairendah < tinggi then QuickSort(Q, nilairendah, tinggi) ;
22.   end;
```

Prosedur ini akan mengurutkan kondisi prosesor setiap komputer *client* menjadi terurut sehingga proses pengiriman berkas ke komputer dapat dipilh dari kondisi komputer dengan nilai proses terkecil.

4. Analisa

Pengujian yang dilakukan terhadap komputer yang terhubung dalam LAN seperti pada gambar 5, menghasilkan data pada aplikasi *server* seperti pada gambar 8. Kondisi prosesor pada setiap komputer *client* terbaca 0% atau tidak sedang memproses *job*, sedangkan kondisi RAM memiliki perbedaan antara satu komputer *client* dengan komputer *client* yang lainnya.



Gambar 8 Tampilan program *server* yang mencatat kondisi prosesor dan RAM komputer *client*

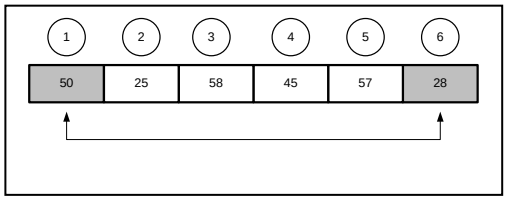
Kondisi RAM yang berbeda kondisi terpakainya ini tidak dapat dijadikan indicator nilai kesibukan komputer *client*, karena dalam pemrosesan parallel basis indikatornya adalah kondisi setiap prosesor sehingga data dari gambar 8 tersebut akan kami simulasikan dalam bentuk table 1, untuk menunjukan pengaruh pengurutan data dengan metode quick sort pada kondisi prosesor.

Tabel 1 Simulasi kondisi prosesor

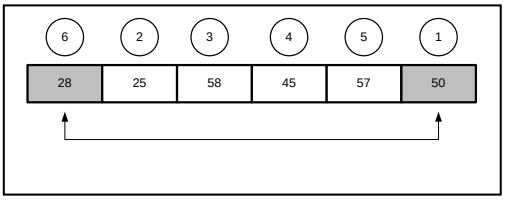
No	Nama/ Alamat IP	Indikator	
		Prosesor(%)	RAM(MB)
1	*Client*1/ 200.168.5.144	50	338,25
2	*Client*2/ 200.168.5.145	25	157,19
3	*Client*3/ 200.168.5.146	58	338,46
4	*Client*4/ 200.168.5.147	45	208,02
5	*Client*5/ 200.168.5.148	57	233,85
6	*Client*6/ 200.168.5.149	28	156,87

Proses pengurutan dengan menggunakan metode quick sort untuk mengurutkan kondisi prosesor komputer *client* adalah sebagai berikut :

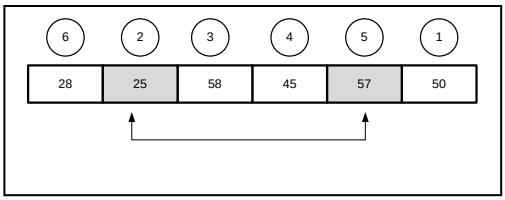
1. Kondisi awal dari 6 prosesor komputer *client* sebagai berikut :



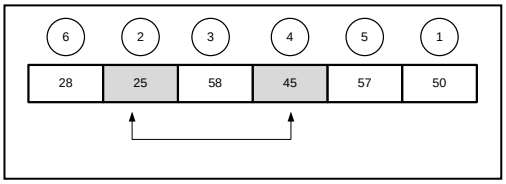
2. Kemudian dilakukan perbandingan antara sisi paling kiri dengan sisi paling kanan, lihat gambar kotak berwarna abu-abu, jika nilai sisi kiri lebih besar dari sisi kanan, maka kotak tersebut ditukar letaknya seperti pada gambar berikut :



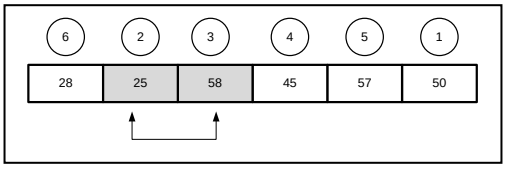
3. Setelah terjadi pertukaran tempat maka kondisi berikutnya adalah menggeser sisi kiri kearah kanan dan sisi kanan ke arah kiri, dan kemudian dibandingkan, seperti gambar berikut ini :



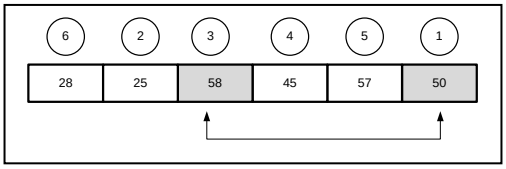
4. Jika nilai sisi kiri lebih kecil dari sisi kanan maka tidak terjadi pertukaran kotak, dan kemudian diteruskan dengan membandingkan sisi kanan berikutnya seperti terlihat pada gambar berikut ini :



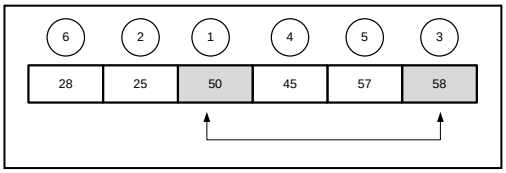
5. Kemudian sisi kanan digeser ke kiri lagi untuk membandingkannya dengan sisi kiri seperti terlihat pada gambar berikut ini :



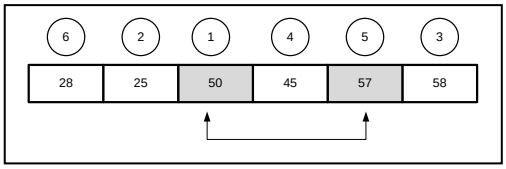
6. Posisi selanjutnya yaitu menggeser sisi kiri ke arah kanan dan membandingkan dengan sisi paling kanan seperti terlihat pada gambar berikut :



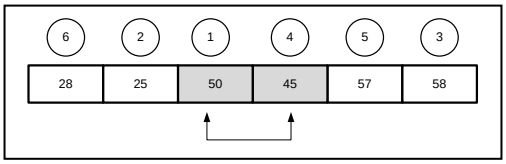
7. Kemudian sisi kanan di geser ke arah kiri untuk dibandingkan seperti gambar berikut ini :



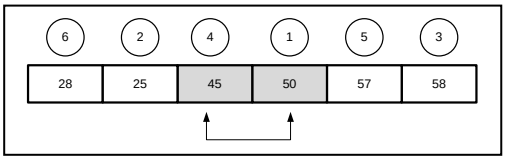
8. Selanjutnya jika tidak terjadi perubahan sisi kanan di geser ke arah kiri seperti pada gambar berikut ini :



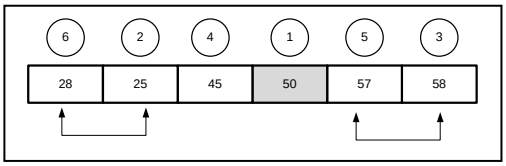
9. Dan kemudian sisi kanan digeser lagi ke arah seperti gambar berikut ini :



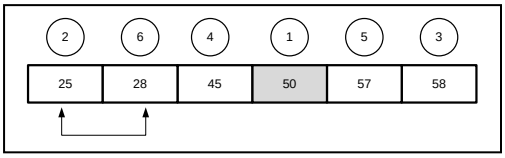
10. Kedua nilai di dalam kotak tersebut dibandingk jika ada perbedaan maka yang kecil diletakkan di sisi kiri dan yang lebih besar di sisi kanan seperti gambar berikut ini :



11. Proses ini menjadi proses terakhir dalam menentukan pivot dari data kondisi prosesor Komputer *client*, dan kemudian pivot akan dijadikan titik tengah untuk perbandingan selanjutnya yang melibatkan sisi kiri dengan sisi kiri serta sisi kanan dengan sisi kanan seperti gambar berikut ini :



12. Hasil perbandingan sisi-sisi tersebut akan menghasilkan hasil akhir dari proses *quick sort* seperti terlihat pada gambar berikut ini :



Kondisi pengurutan ini terjadi sangat cepat karena pembacaan data prosesor komputer *client* akan selalu berubah-ubah sesuai dengan keadaan komputer tersebut dalam keadaan melakukan eksekusi instruksi atau tidak, sehingga dari hasil analisa dan jalannya aplikasi didapatkan urutan pada table 1 menjadi urutan pada table 2.

Tabel 2 Urutan komputer berdasarkan kondisi prosesor

No	Nama/ Alamat IP	Indikator	
		Prosesor(%)	RAM(MB)
1	*Client*2/ 200.168.5.145	25	157,19
2	*Client*6/ 200.168.5.149	28	156,87
3	*Client*4/ 200.168.5.147	45	208,02
4	*Client*1/ 200.168.5.144	50	338,25
5	*Client*5/ 200.168.5.148	57	233,85
6	*Client*3/ 200.168.5.146	58	338,46

Sehingga dalam pemrosesan parallel yang instruksi yang dikirim oleh komputer *server* akan memilih komputer *client* yang kondisi prosesor dengan nilai terkecil, dan proses ini akan terus menerus dilakukan sehingga *load balancing* dapat tercapai, dimana prosesor yang tidak sibuk akan diberikan instruksi untuk melakukan proses.

5. Kesimpulan

1. Metode pengurtan data *Quick Sort* dapat diterapkan pada aplikasi pemrosesan parallel.
2. Hasil akhir pengurutan dapat digunakan untuk mengirimkan instruksi yang akan dip roses oleh prosesor komputer *client* sesuai dengan pengurtan yang telah dilakakukan.
3. *Load balancing* pada sekumpulan komputer *clinet* dalam pemrosesan parallel dapat tercapai.

6. Daftar Pustaka

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

Wintolo, Hero, 2010, Deteksi Kinerja Prosesor Komputer Dengan Cara Remote Untuk Mendukung Aplikasi Pemrosesan Paralel, Jurnal Ilmiah Angkasa, Vol.2 No. 2 November 2010

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

http://www.mikrotik.co.id/artikel_lihat.php?id=34 [diakses tanggal 3 Mei 2012]