LOAD BALANCING MENGGUNAKAN METODE BUBBLE SORT PADA SISTEM DATABASE DENGAN MULTI SERVER

Prayitno, Agus Basukesti, Yuliani Indrianingsih
Jurusan Teknik Informatika
Sekolah Tinggi Teknologi Adisutjipto Yogyakarta
informatika@stta.ac.id

ABSTRACT

The increasing demand for the utilize services computer networks has increased the traffic on the network. In general, the server provides service round the clock without a break, so that problems such as breakdown of communication in the network is lost that must be avoided. This cases need for a better system with high performance and reliable every time. Current server technology development will be fast but the memory access speed is slower than the increase in the consumption of bandwidth. This is one of the factors causing the bottleneck experienced of server. With the load balancing can be provide a better solution. The concept used to provide a server has the same service. Then, the load balancing will check the CPU usage on each server. The server will perform selection based on the value of the smallest CPU Usage of all existing servers. The application load balancing is placed before the user accessing of server with the object of distributing requests from the user to any of the servers that have been selected by the load balancer. Load balancing provides many benefits to systems that use it. The first, low cost, existing resources do not need to be replaced or discarded. Second, don't always need high end equipment. Third, easily developed when needed arose and a drastic decline in the quality of service. Load balancing is also more secure, when a failure occurs on the server, load balancing application will redirect the user to the server is still available.

Keywords: load balancing, CPU Usage, server

1. Latar Belakang

Seiring waktu, perkembangan sistem informasi berbasis *client-server* semakin pesat, semakin banyaknya jumlah *client* membuat *server* kesulitan dalam menangani kebutuhan *client* secara bersamaan. Spesifikasi *server* yang sangat terbatas, akan menyebabkan berkurangnya efisiensi dan kecepatan *server* dalam melayani *client*. Solusi yang bisa ditawarkan adalah dengan membangun sistem berbasis *multi-server*, dimana ada sebuah aplikasi yang berada diantara *server* dan *client* yang bertujuan untuk menentukan tingkat kesibukan dan pemakaian *CPU Usage* dari masing-masing *server*.

2. Landasan Teori

Load Balancing

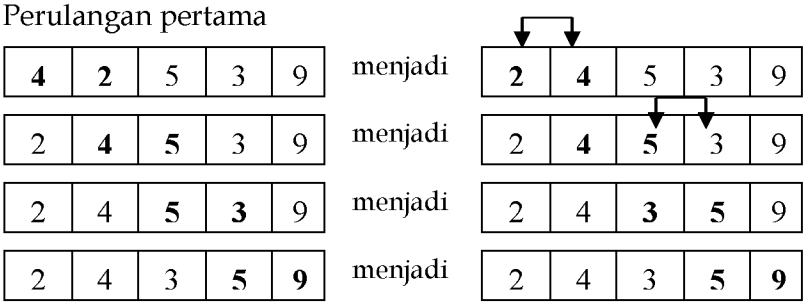
Banyak pendapat mengenai *load balancing*. Dalam *paper* yang ditulis oleh Irfan Darmawan, "*Load balancing* adalah suatu teknik mendistribusikan beban kerja (*workload*) secara merata antara dua atau lebih komputer, *link* jaringan, CPU (*Central Processing Unit*), *hard-drive*, atau sumber daya lain. Dalam rangka untuk mendapatkan pemanfaatan sumber

daya yang optimal, memaksimalkan *throughput*, meminimalkan waktu tanggap, dan menghindari *overload*." (Darmawan, 2009: 145).

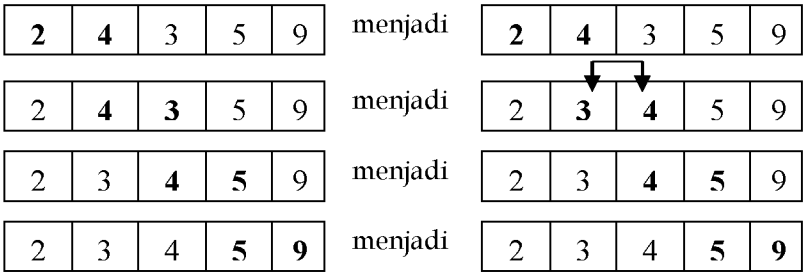
Bubble Sort

Bubble sort merupakan cara pengurutan yang sederhana. Cara kerjanya adalah mengulang proses pembandingan antara tiap-tiap elemen *array* dan menukarnya apabila urutannya salah. Pembandingan elemen-elemen ini akan terus diulang hingga tidak perlu dilakukan penukaran lagi. Contoh proses pengurutan dalam metode *bubble sort*, misal terdapat sebuah *array* dengan nilai elemen-elemen "4 2 5 3 9", maka proses yang terjadi adalah sebagai berikut:

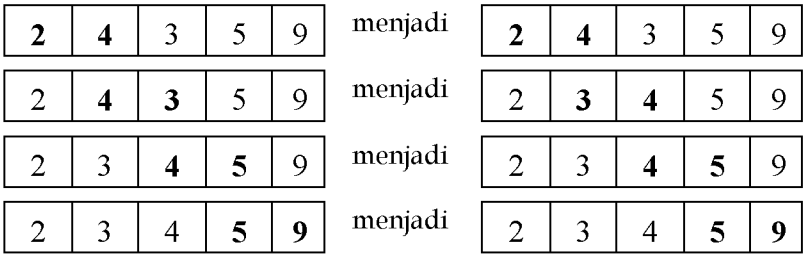
Perulangan pertama



Perulangan kedua



Perulangan ketiga



Dari contoh proses diatas, *array* telah terurut pada perulangan kedua langkah kedua, namun algoritma tetap dilanjutkan sampai perulangan kedua selesai. Perulangan ketiga dilakukan karena definisi terurut dalam algoritma *bubble sort* adalah tidak ada satupun penukaran pada suatu perulangan, sehingga perulangan ketiga dibutuhkan untuk memastikan keurutan *array* tersebut.

Socket Programing

Socket adalah sebuah cara untuk berkomunikasi dengan program atau *node* lain menggunakan *file descriptor*. *Socket* memungkinkan adanya pertukaran informasi antara proses yang pada sebuah mesin atau seluruh jaringan. Mendistribusikan pekerjaan ke mesin yang paling efisien dan mempermudah akses pada data terpusat. Pemrograman *socket* sendiri

adalah pemrograman yang menggunakan *socket* untuk dapat berkomunikasi dengan program lain.

3. Perancangan dan Analisa Sistem

Spesifikasi Hardware

Spesifikasi *hardware* untuk *server* adalah:

1. Komputer dengan *processor Intel® Core™ 2 Quad* Q8200 @2.33GHz (4 CPUs).
2. *Harddisk* 320 GB.
3. Memori (RAM) 4Gb
4. *Monitor*.
5. *Keyboard* dan *Mouse*

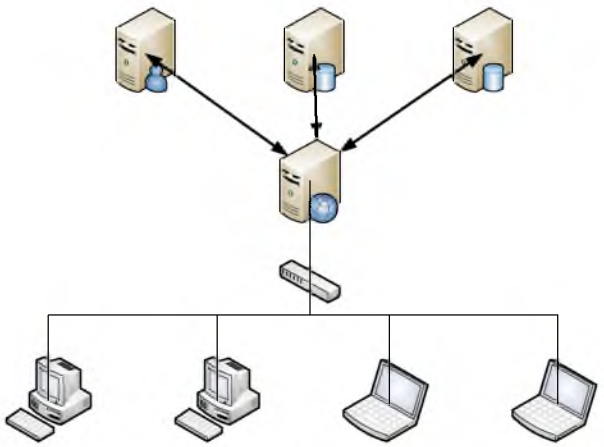
Spesifikasi Software

Dalam pembuatan sistem ini perangkat lunak yang digunakan adalah sebagai berikut:

1. Sistem Operasi Windows 7 Ultimate 32-bit untuk *server*, dan Windows XP Profesional SP2 32-bit untuk *user*.
2. PostgreSQL 9.1.
3. Delphi 7
4. Komponen Zeos *database connector*

Skema Jaringan

Salah satu persiapan dalam membangun sistem ini adalah menyiapkan konsep arsitektur jaringan. *Server load balancer* akan bertugas sebagai pembagi beban pada setiap *server* data. Beban pada *server* data akan berkurang dan hal ini akan mempercepat proses komunikasi *user* dengan *server*. *Server load balancer* dan *server* data berada pada *network address* yang sama. Penjelasan konfigurasi jaringan tampak pada gambar 1.

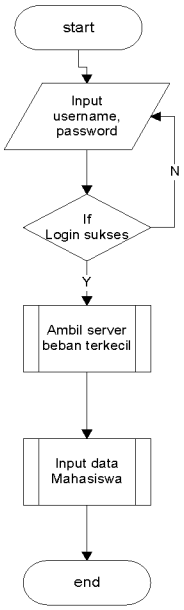


Gambar 1. Rancangan Jaringan *Load Balancing*

Flowchart Sistem

Rancangan ini digunakan untuk mendesain dan merepresentasikan suatu program. Sebelum pembuatan program, fungsinya adalah mempermudah dalam menentukan alur logika program yang akan dibuat. Sesudah pembuatan program fungsinya adalah untuk menjelaskan alur program kepada orang lain atau *user*.

Pada rancangan flowchart aplikasi load balancing dengan menggunakan metode bubble sort pada sistem multi database terdiri dari 3 bagian, yaitu proses login, proses pemilihan beban server terkecil, dan proses input data mahasiswa. Rancangan ini dapat dilihat pada gambar 2.



Gambar 2. Perancangan *Flowchart* Aplikasi *User Interface*

4. Pengujian dan Analisa Hasil

Uji Coba Pemilihan *Server* Data

Uji coba ini berada dalam jaringan lokal dengan memanfaatkan media transmisi kabel LAN dan *Switch*. Pengujian ini dilakukan untuk melihat bekerja atau tidaknya aplikasi *Server Load Balancing* dalam melakukan pengecekan kesibukan *server* data melalui nilai *CPU Usage server* data itu sendiri.

Pada uji coba ini, akan diambil semua nilai *CPU Usage* dari *server* data dan akan dibandingkan nilai tersebut satu sama lain agar dapat menilai kinerja dari aplikasi *server load balancing*. Dari hasil pengujian pemilihan *server* data diperoleh data seperti pada tabel 1

Tabel 1. Hasil uji coba pemilihan *server* data

No	Uji coba ke-	Alamat IP Client	Nilai CPU Usage			Server yang di pilih	Kecepatan koneksi dengan server (ms)	Kecepatan menyimpan data (ms)
			Server 1 (%)	Server 2 (%)	Server 3 (%)			
1	1	192.168.20.10	0	49	5	Server 1	313	78
2	2	192.168.20.10	0	1	2.25	Server 1	313	0
		192.168.20.14	0	1	2.25	Server 1	328	31
3	3	192.168.20.10	0	3	2.5	Server 1	313	0
		192.168.20.14	0	3	2.5	Server 1	313	14
		192.168.20.54	0	3	2.5	Server 1	639	15
4	4	192.168.20.10	24	5.25	2.75	Server 3	422	0
		192.168.20.14	24	5.25	2.75	Server 3	375	31
		192.168.20.54	24	5.25	2.75	Server 3	390	0
		192.168.20.101	24	5.25	2.75	Server 3	764	0

Analisa Pengurutan Bubble Sort

Pada proses *bubble sort* aplikasi *load balancer* mengirim permintaan data CPU *usage* kepada setiap *server*. Data CPU *usage* yang diterima oleh aplikasi *load balancer* dan ditampung oleh variabel bertipe *array*. Proses *sorting* dilakukan dengan membuat perulangan sebanyak N-1, dimana N adalah jumlah *server*. Proses pengurutan dalam aplikasi server *load balancing* terlihat seperti pada gambar 3.

Gambar 3. Proses pengurutan *Cpu Usage* dengan *bubble sort*

Proses pengurutan terlihat seperti pada penjelasan berikut ini:

Perulangan pertama

1. Nilai-nilai elemen ditaruh kedalam *array* dan kemudian dilakukan perbandingan nilai data pertama dengan kedua

2. Kemudian dilakukan perbandingan antara nilai elemen kedua dengan nilai elemen ketiga. Pada tahap ini proses perulangan pertama selesai.

Perulangan kedua

1. Pada proses ini terlihat bahwa data telah terurut. Tetapi pada proses *bubble sort* pengurutan dilakukan sampai tidak terdapat perpindahan data pada proses perulangan tersebut, sehingga dilakukan proses pengurutan sekali lagi untuk memastikan tidak terdapat pertukaran nilai data.

2. Proses selanjutnya adalah membandingkan nilai pada elemen kedua dan ketiga. Pada tahap ini didapat bahwa urutan pada *array* adalah benar, dimana nilai elemen kedua lebih kecil dari pada nilai elemen ketiga.

Analisa Pengambilan Nilai CPU Usage

Pada proses pengambilan nilai *CPU usage*, proses dimulai ketika sebuah *request* dari *client* untuk mendapatkan alamat IP *server database*. Pada saat tombol *login* dari sebuah aplikasi *client* dieksekusi, aplikasi akan mengirimkan sebuah pesan berupa kata "kirim" kepada aplikasi *server load balancing*. Saat pesan telah berhasil diterima oleh aplikasi *server load balancing*, maka akan dilanjutkan untuk memerintahkan *socket* mengeksekusi sebuah tombol yang berfungsi untuk mengirim pesan yang sama kepada aplikasi *client load balancing* yang ada pada setiap *server* data yang terkoneksi oleh aplikasi *server load balancing*.

Pengambilan nilai *CPU Usage* pada aplikasi *client load balancing* dilakukan dengan memanfaatkan sebuah *module* yang telah diciptakan oleh Alexey A. Dynnikov. Saat pesan diterima oleh aplikasi *client load balancing*. Aplikasi akan mengirimkan nilai *CPU Usage* terbaru dan alamat IP dari *server* data kepada aplikasi *server load balancing*.

Analisa Pemilihan *Server*

Uji coba dilakukan dengan mengkondisikan proses *login* aplikasi *User Interface* secara bersamaan. Dari hasil uji coba diperoleh data pemilihan *server* yang dilakukan oleh aplikasi *load balancing*. Data-data tersebut didapat kesimpulan bahwa aplikasi *load balancing* berhasil melakukan pemilihan *server* data dengan nilai *CPU Usage* yang terkecil. Kelebihan dari penggunaan aplikasi *load balancing* ini adalah dapat membantu mengurangi beban *server* data dalam melayani *request* aplikasi *client*.

Dari hasil uji coba tersebut diperoleh data waktu koneksi yang dibutuhkan client untuk melakukan autentifikasi pada *server* data. Waktu rata-rata yang diperoleh oleh 4 user dengan *single database* adalah 33 ms. Sedangkan rata-rata waktu pada sistem *multi database* dengan *load balancing* oleh 4 user adalah 488 ms.

Rata-rata waktu tersebut lebih lambat dibanding dengan waktu yang diperlukan oleh sistem *single database*. Hal ini karena pada sistem *multi database* dengan *load balancing* terdapat proses pemilihan *server* data terlebih dahulu, sehingga sistem dengan *single database* memiliki waktu koneksi yang lebih cepat ketika menangani *request* dari *user* dengan jumlah yang sedikit. Sebaliknya, sistem *multi database* dengan *load balancing* ini akan berfungsi secara optimal ketika menangani *request* dari *client* yang berjumlah banyak.

Hasil uji coba pengukuran kecepatan penyimpanan data diperoleh rata-rata waktu tidak begitu berbeda jauh namun ada kelebihan kecepatan peyimpanan pada aplikasi *multi database* dengan *load balancing*. Rata-rata waktu yang didapat pada sistem *single database* untuk 4 *user* yang melakukan penyimpanan data secara bersamaan adalah 22 ms. Sedangkan waktu rata-rata yang diperoleh pada sistem *multi database* dengan *load balancing* untuk 4 *user* yang mengakses secara bersamaan adalah 7,5 ms. Rata-rata waktu tersebut lebih cepat dibanding dengan waktu yang diperoleh oleh sistem dengan *single database*.

Didapat juga hasil data uji coba lain dari perbandingan antara penggunaan *single database* dengan *multi database* menggunakan *load balancing*. Yaitu, penggunaan *multi database* dengan *load balancing* mempercepat proses *query* penyimpanan database dikarenakan *server* data yang terpilih tidak dalam keadaan melakukan pekerjaan yang lebih berat dari pada *server* data lainnnya.

5. Kesimpulan

1. Aplikasi yang dirancang dalam tugas akhir ini dapat beroperasi dengan baik untuk melakukan pemilihan alamat jaringan *server* data secara otomatis ketika *user* melakukan proses *login*.
2. Proses pengurutan pada aplikasi berhasil dilakukan setelah dibandingkan dengan teori metode *bubble sort*.
3. Dari data yang di peroleh, waktu yang dibutuhkan untuk terkoneksi dengan *database* pada proses *load balancing* lebih lama dibandingkan dengan sistem yang menggunakan *single database*, sedangkan kecepatan penyimpanan data lebih cepat dibandingkan dengan sistem yang menggunakan *single database*.

Daftar Pustaka

Darmawan, I, Kuspriyanto, Yoga Priyana. 2009. " *Perancangan Algoritma Load Balancing pada Topologi Dynamic Tree Jaringan Grid Computing*."

http://journal.uii.ac.id/index.php/Snati/article/viewFile/1264/1024

(diakses pada tanggal 2 Januari 2013)

Dynnikov, A. A. 2000. AdCpuUsage

http://www.aldyn.ru/products/cpu_usage/index.html

(diakses pada tanggal 12 November 2012)

Pardosi, Mico. 2004. *Bahasa Pemrograman Turbo Pascal 7.0 for MS-DOS dan Windows*. Surabaya: Indah.

Sismoro, Heri S.Kom., dan Kusrini Iskandar S.Kom. 2004. *Struktur Data dan Pemrograman dengan Pascal*. Yogyakarta: Penerbit Andi..

Wilkinson, Barry and Michael Allen. 2010. *Parallel Programing Teknik dan Aplikasi Menggunakan Jaringan Workstation & Komputer Paralel*. Yogyakarta: Penerbit Andi, Edisi Kedua.