IMPLEMENTASI *LOAD BALANCING* DENGAN METODE *BUBBLE SORT* STUDI KASUS PENGISIAN KRS DI STTA

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

Abstract

The increase of request brings large traffic to networks especially to data centers, large enterprises and portal websites. In addition, server provides more and more information by using applications. Most servers have to provide all-day services, and any service interruption or key data loss in communication will result in business loss. All these require high performance and high reliability on application services. However, the increase of server processing speed and memory access speed is greatly lower than that of the network bandwidth and applications. In addition, the increase of network bandwidth makes server resource consumption more serious. Therefore, the servers become the network bottleneck, and the traditional single device mode becomes the network failure point. Load balancing (LB) comes to provide a good solution by providing multiple servers form a server cluster, with each server providing the same or similar services. A load balancing application is managed at the front end of the server cluster to distribute user requests in the server cluster according to pre-configured load balancing rules, provide services, and maintain the servers. Load balancing bring good advantage such as bring faster application, low cost, the available resources will not be wasted, and no high-end devices are needed for the new resources. It is expandability. When services are increasing, the system can satisfy the needs by adding servers, without affecting the existing services and reducing service quality. Load balancing also has high reliability. When a server fails, the LB application or LB device redistributes user requests to other servers in the same cluster, ensuring uninterrupted services.

Keywords: ***load balancing, synchronization, server.***

1. Pendahuluan

Dari tahun ke tahun STTA (Sekolah Tinggi Teknologi Adisutjipto) mengalami perkembangan, terutama dalam kebutuhan akan layanan informasi. Kebutuhan akan sistem informasi penunjang kegiatan perkuliahan perlu ditingkatkan seiring pertambahan mahasiswa.

Sistem Akademik di STTA menggunakan *database* terpusat menggunakan aplikasi Postgres. Aplikasi Postgres sendiri memiliki banyak keterbatasan, salah satunya adalah kemampuan dalam menangani permintaan dari *client*. Oleh karena itu perlu dibuat sebuah sistem yang dapat meningkatkan kinerja *server* dalam menangani permintaan layanan dari *client*. Salah satu cara adalah dengan menerapkan sistem terdistribusi dengan membagi beban layanan *server database*.

Model yang dibuat adalah dengan menggunakan *database* terpisah yang berfungsi sebagai *temporary database* yang melayani permintaan *client*. Data hasil pemrosesan yang valid akan tersimpan pada *server* utama. Proses *load balancing* terjadi dengan mengalihkan *request* layanan dari *client* kepada *server* yang memiliki lebih sedikit proses yang sedang ditangani.

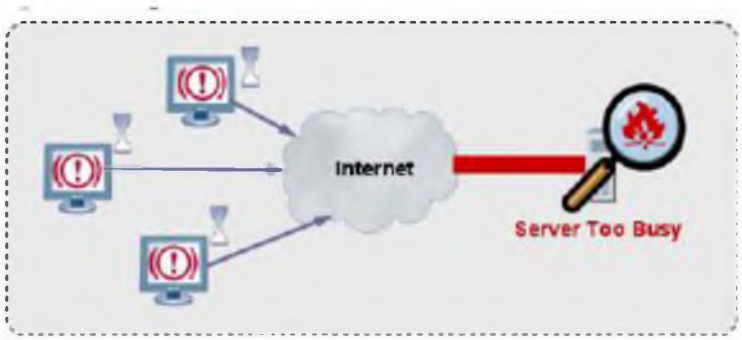
Efek yang dihasilkan adalah beban *server* utama menjadi lebih ringan karena beban juga terdistribusi ke *temporary server*. Sebagai implementasinya adalah dengan menerapkan model tersebut pada sistem pengisian KRS dimana *server database* mendapatkan *request* layanan dalam jumlah besar oleh banyaknya *client* yang mengakses dalam kurun waktu tertentu (masa pengisian krs).

Sistem informasi yang memanfaatkan jaringan komputer sering mengalami masalah dalam hal pemanfaatan *resource* yang merujuk pada performa layanan, waktu pemrosesan layanan, maupun ketersediaan layanan itu sendiri. *Load balancing* digunakan untuk menyelesaikan permasalahan tersebut dengan menyeimbangkan permintaan layanan antara beberapa *server*. *Load balancing* dilakukan dengan menggunakan pelbagai aturan tergantung jenis layanan, *hardware*, dan pengaturan oleh administrator sistem.

2. Landasan Teori

Load Balancing

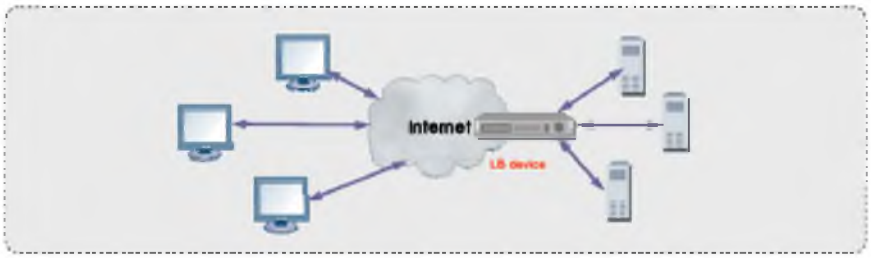
Server load balancing memiliki banyak pengertian. Dalam pustaka milik Microsoft *load balancing* adalah teknologi *clustering* yang memperbaiki skalabilitas dan ketersediaan layanan. Untuk menjamin ketersediaan layanan dilakukan pengecekan apakah ada kegagalan pada *host* tertentu, jika ditemukan maka akan mendistribusikan kembali *traffic* pada *host* yang masih berfungsi. Penjelasan lain didapat dari *paper* yang diterbitkan oleh Hewlett Packard dimana latar belakang penggunaan teknologi *load balancing* adalah peningkatan layanan yang mempengaruhi *traffic* jaringan terutama pada *data center*. Pada jaringan skala besar seperti pada perusahaan besar, layanan seperti HTTP, FTP, SMTP harus tersedia sepanjang waktu dan adanya interupsi pada servis atau putusnya komunikasi akan merugikan sisi bisnis, sehingga diperlukan layanan yang memiliki performa tinggi dan dapat tersedia setiap waktu. Saat itu peningkatan kecepatan *server* dan kecepatan akses *memory* lebih rendah dibanding peningkatan *network bandtwidth*, sehingga peningkatan *bandwidth* jaringan dapat menyebabkan jaringan *bottleneck*.



Gambar 1 *Server* Tidak Merespon

Gambar 1 menunjukan sistem dengan *single server* memungkinkan kegagalan dalam memberikan layanan. Untuk mengatasinya dibangun sistem *cluster server* untuk menyeimbangkan beban diantara *server* dengan teknologi *load balancing*. *Multiple server* membentuk *cluster server* yang menyediakan layanan yang sama. *Load balancing device* (LB

device) ditempatkan pada *front end cluster* untuk mendistribusikan *request user* pada *server* menurut aturan tertentu. Skema *load balancing* ditunjukan pada gambar 2.



Gambar 2 Skema *Load Balancing*

Pengertian lain *load balancing* adalah definisi dari proses dan teknologi yang mendistribusikan beban komputasi jaringan ke beberapa *server* dengan memanfaatkan peralatan jaringan. Lalu lintas jaringan akan diarahkan ke *server* yang ditentukan. Proses *load balancing* bersifat transparan terhadap *end user*.
Load balancing menerapkan beberapa fungsi diantaranya:

1. Melakukan *intercept* lalu lintas jaringan (misal lalulintas web).
2. Melakukan pembagian *traffic* jaringan tiap-tiap *request* dan memutuskan *server* mana yang akan memberi respon layanan.
3. Mengawasi kinerja *server* dan memastikan *server-server* tersebut memberikan tanggapan setiap *request*. Jika tidak maka *server* tersebut tidak dilibatkan dalam sistem.
4. Memberikan keunggulan dengan menggunakan lebih dari satu *server* serta menghindari kegagalan dalam memberikan layanan *server*.

Metode Bubble Sort

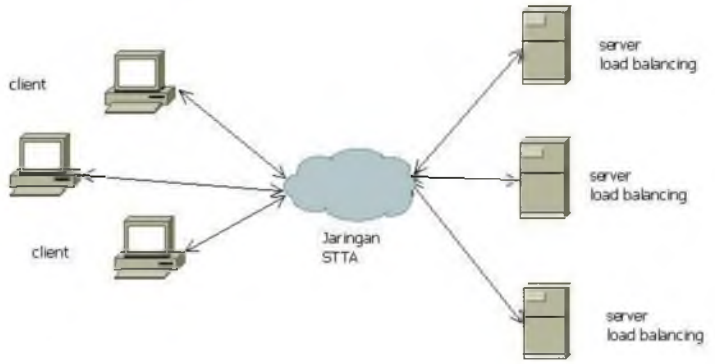
Metode *bubble sort* adalah salah satu metode yang digunakan untuk mengurutkan data. Untuk mengurutkan data dilakukan proses pembandingan semua elemen data. Jika terdapat N data dan data terkoleksi dari urutan 0 sampai dengan N-1 maka algoritma pengurutan dengan metode *bubble sort* adalah sebagai berikut:

1. Bandingkan posisi data i = 0 dan j = 1.
2. Jika data di posisi i lebih besar daripada data di posisi j, maka data di posisi i di tukar dengan data diposisi j (*swap*).
3. Kemudian, lakukan perbandingan data di posisi i = 1 dan data di posisi j = 2. Lakukan langkah 2, begitu juga untuk data berikutnya hingga i = N-2 dan j = N-1.
4. Ulangi langkah 1, 2 dan 3 untuk data diposisi 0 sampai dengan data di posisi N-2. Untuk tahap selanjutnya data yang dibandingkan akan semakin berkurang sebab data dengan nilai yang lebih besar akan terposisi di bagian sebelah kanan data.

3. Perancangan Sistem

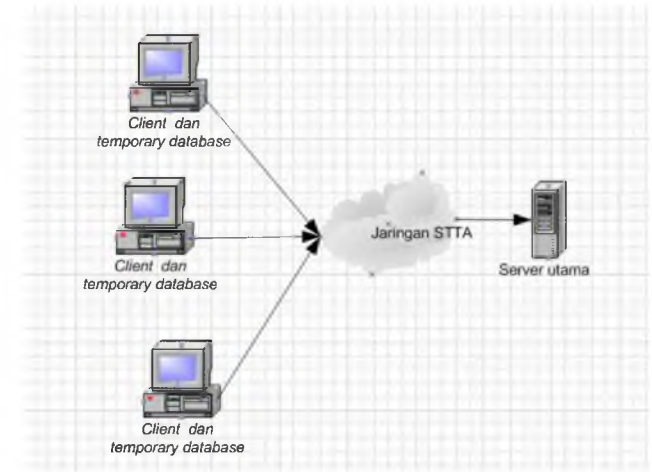
Perancangana Konsep Jaringan

Salah satu persiapan dalam membangun sistem ini adalah menyiapkan konsep arsitektur jaringan. Penjelasan konfigurasi jaringan terdapat pada gambar 3.



Gambar 3 Rancangan Jaringan *Load Balancing*

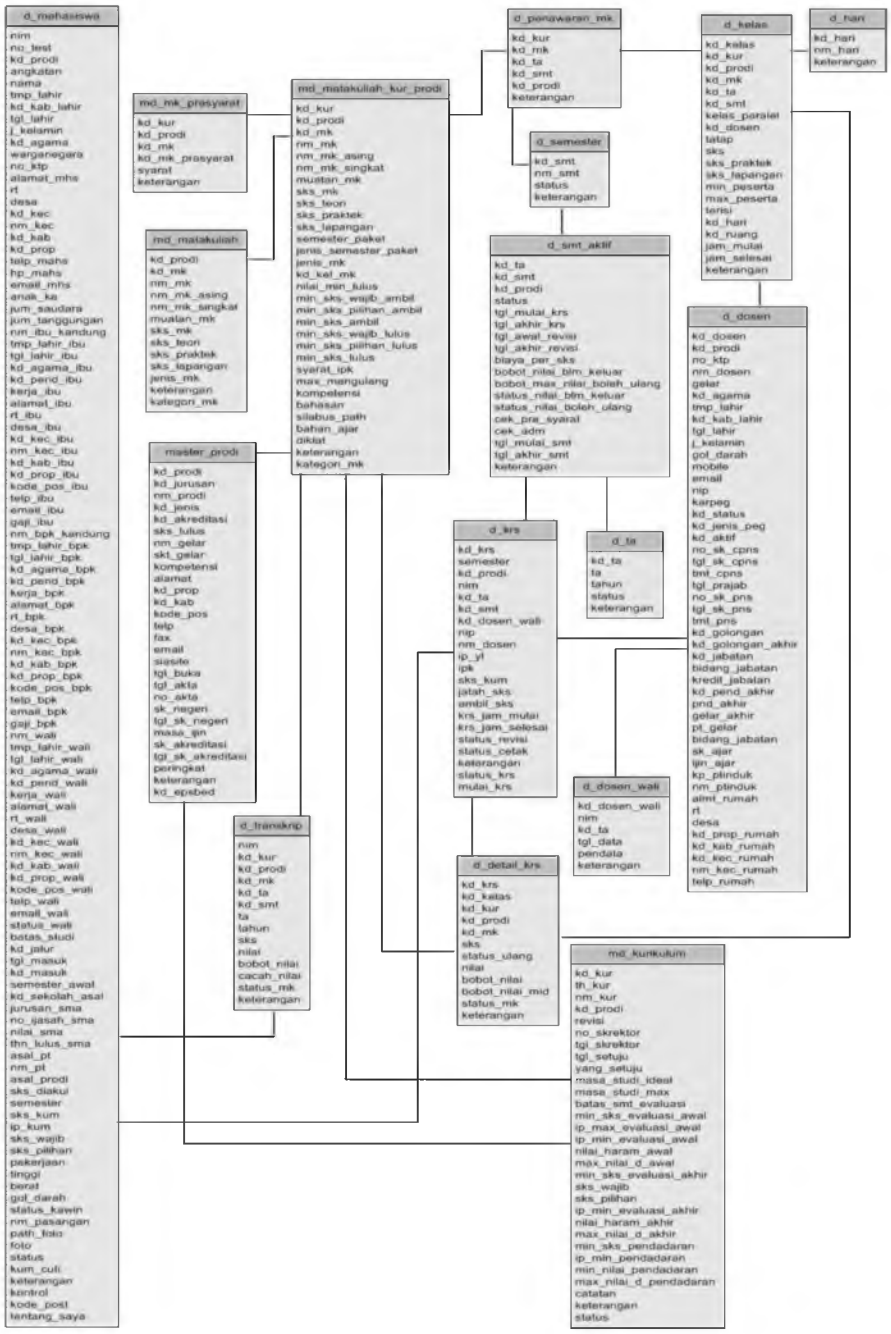
Gambar 3 merupakan konfigurasi jaringan dengan *temporary database* terpisah dari *client*. Sedangkan untuk model jaringan dengan *temporary database* terletak di sisi *user* seperti pada gambar 4



Gambar 4 Rancangan Dengan *Temporary Database* Di Client

Perancangan Database

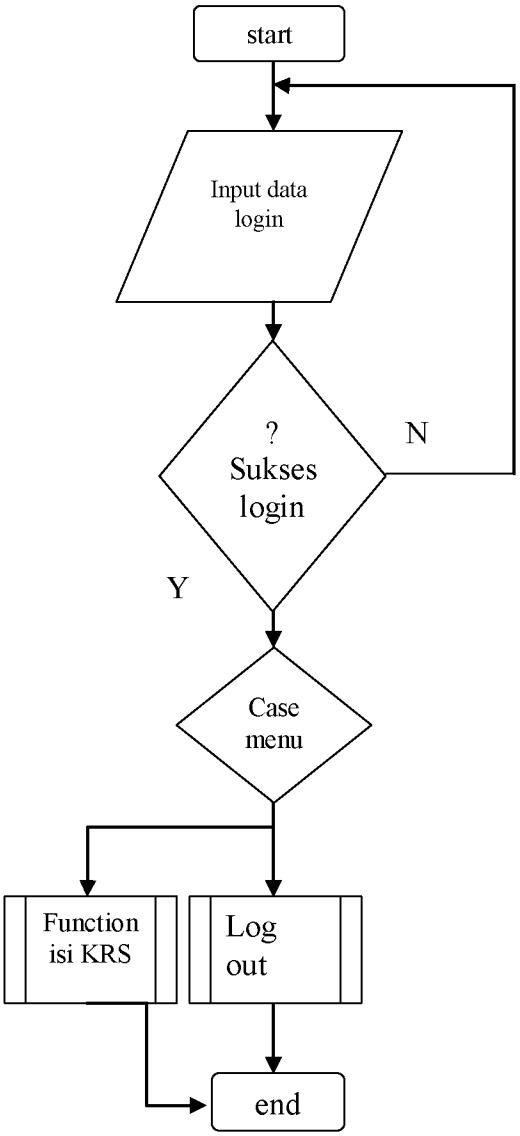
Sistem pengisian KRS ini menggunakan *database* dari sistem pengisian KRS yang lama. Penjelasan tabel dan relasinya terdapat pada gambar 5. Pada sistem pengisian KRS terdapat dua hal yang utama yaitu pengaturan koneksi *database*, dan pengisian KRS itu sendiri. Pengaturan koneksi menggunakan dua koneksi, yaitu koneksi ke *server* utama, dan koneksi ke *temporary server*. Pada sistem ini terdapat beberapa proses *query* yang ditangani oleh *temporary server* dan sebagian lagi ditangani oleh *server* utama. Beberapa *query* yang ditangani oleh *server* utama adalah pengambilan data matakuliah, dan pengambilan data KRS semester yang lalu. Sementara *query* yang ditangani oleh *temporary server* adalah pengisian KRS dan detail matakuliahnya.



Gambar 5 Rancangan Tabel Sistem Pengisian KRS

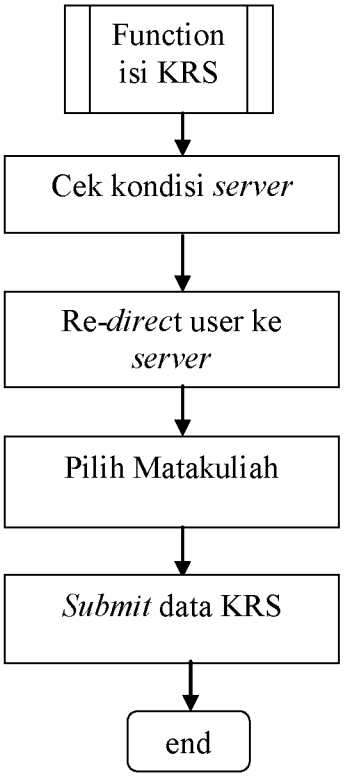
Perancangan Aplikasi Pengisian KRS

Proses pengisian KRS dimulai saat mahasiswa *login*, kemudian memilih menu pengisian KRS. Proses re-*direct* terjadi saat user memilih menu pengisian KRS. User akan diarahkan ke *server* yang sudah ditentukan oleh program, kemudian daftar matakuliah akan ditampilkan. User kemudian memilih dan men-*submit* data matakuliah yang dipilih. Data KRS tersimpan di *server* bersangkutan selama masa KRS. Data KRS akan dikirim ke *server* utama setelah masa pengisian KRS berakhir. *Flowcart* dari menu utama sistem pengisian KRS adalah pada gambar 6.



Gambar 6 Flowcart Menu Utama Pengisian KRS

Di dalam menu utama pengisian KRS terdapat submenu isi KRS. Gambar 7 menjelaskan *function* isi KRS yang digunakan apabila menggunakan model *temporary database* pada *server* terpisah. Sedangkan untuk model *temporary database* terletak pada sisi *client* tidak membutuhkan proses pengecekan kondisi server, daftar matakuliah langsung ditampilkan, dan data KRS langsung disimpan pada *database* di *client* masing-masing.



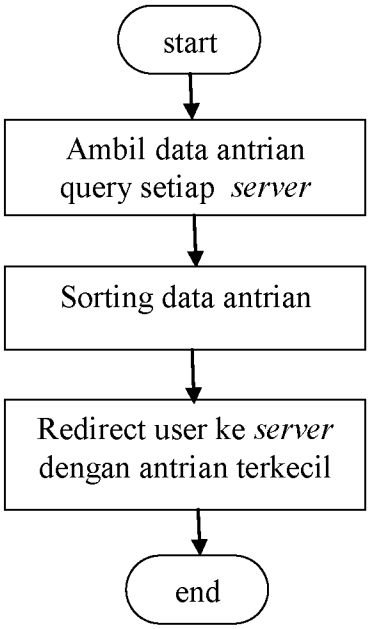
Gambar 7 Flowcart Function Isi KRS

Perancangan Sinkronisasi Database

Dalam perancangan sistem ini dibutuhkan alur diagram alir atau flow chart yang menjelaskan bagaimana alur logika program bekerja dari awal sampai akhir. Antara lain yaitu flow chart untuk proses *load balancing,* proses transfer data dari *temporary server* ke *server* utama, penyimpanan informasi *server* yang terlibat dalama proses *load balancing* dan pengisian KRS. Keseluruhan proses tersebut terdapat pada menu Admin.

Perancangan Load Balancing

Program yang digunakan untuk menerapkan konsep load *load balancing* menggunakan tabel untuk menyimpan konfigurasi *server* yang terlibat dalam proses *load balancing*. Data yang disimpan adalah data untuk koneksi ke *database server* temporary yaitu alamat IP, username, password, port. Flowcart proses penentuan *server* untuk pengisian KRS dijelaskan pada gambar 8. Algoritma ini digunakan apabila *temporary database* terletak pada *server* terpisah.



Gambar 8 Proses Pemilihan *Server* KRS

4. Implementasi Dan Pembahasan

Program pengisian KRS dengan *load balancing* ini ditujukan untuk meningkatkan performa sistem yang memanfaatkan layanan *database*. Penurunan performa layanan suatu *server* terjadi jika *server* tidak mampu menangani *request* dalam jumlah besar. Efek yang terjadi adalah terjadi antrian dari *user*, mengakibatkan proses menjadi lama.

Setiap *server* memiliki batasan dalam jumlah *request* yang bisa ditangani. Hal tersebut dipengaruhi dari kemampuan aplikasi *database* itu sendiri dan juga *hardware* yang digunakan. Aplikasi *database* hanya bisa memproses satu *query* dalam satu waktu. Sebelum melakukan proses *query,* aplikasi *client* terlebih dahulu membangun koneksi dengan *server database*. Data koneksi tersimpan dalam tabel pg_stat_activity. Jika tidak terjadi *request* berupa *query,* koneksi akan memiliki status *idle*. Jika beberapa koneksi melakukan proses *query,* maka akan dilakukan proses antrian menurut urutan waktu. Permintaan *query* yang lebih dulu masuk akan diproses lebih awal.

Program dengan *load balancing* ini menggunakan parameter jumlah koneksi untuk menentukan *server* mana yang akan menangani *request* dari *client*. Data hasil pengecekan tiap *server* akan ditampung dalam sebuah variable *array* untuk kemudian diurutkan.

Tes Pengisian KRS

Pengujian ini digunakan untuk mengetahui apakah program pengisian KRS dapat bekerja dengan baik. Hal lain yang dipertimbangkan dalam pengujian ini adalah kecepatan dalam proses pengisian KRS, serta untuk mengetahui apakah ada tidaknya kendala dalam menjalankan sistem. Ujicoba terdiri dari dilakukan dengan *temporary database* terpisah. Selanjutnya akan diambil kesimpulan dari kedua pengujian tersebut.

Tes pengisian KRS dengan LAN dilakukan pada jaringan lokal di lab *inherent* STTA. Dari hasil pengujian KRS dengan sistem *temporary database* terpisah diperoleh data berikut:

Tabel 1 Hasil uji coba KRS dengan *load balancing*

No	Jumlah *User*	Waktu rata-rata (detik)
1	1 *user*	2
2	4 *user*	3
3	8 *user*	5
4	10 *user*	5

Pengujian berikutnya adalah pada sistem pengisian KRS lama berbasis WEB. Dari pengujian diperoleh data berikut.

Tabel 2 Hasil uji coba KRS lama berbasis *web*

No	Jumlah *User*	Waktu rata-rata (detik)
1	1 *user*	2
2	4 *user*	3
3	8 *user*	5
4	10 *user*	6

Sebagai perbandingan lagi dilakukan pengujian pengisian KRS dengan menggunakan sistem yang telah diterapkan berbasis *desktop*. Waktu yang diperoleh adalah waktu untuk pengisian KRS untuk setiap satu matakuliah. Hasil dari pengujian sistem lama terdapat pada tabel 3.

Tabel 3 Hasil uji coba KRS lama berbasis desktop

No	Jumlah *User*	Waktu rata-rata (detik) tiap satu matakuliah
1	1 *user*	6
2	4 *user*	8
3	8 *user*	15
4	10 *user*	17

Uji Coba Sinkronisasi Pada LAN

Pada pengujian ini proses *sinkronisasi* dilakukan pada jaringan LAN. Dari hasil pengujian diperoleh data yang dijelaskan pada tabel 4.

Tabel 4 Hasil Uji Coba sinkronisasi Data pada LAN

No	*Record temporary server*	Waktu (detik)
1	10	1
2	50	1
3	100	2
4	150	3

Uji Coba Sikronisasi Pada Jaringan Internet

Pada pengujian ini proses *sinkronisasi* dilakukan pada jaringan Internet. Tes pengujian ini digunakan untuk mengetahui tingkat keberhasilan dan keakuratan dalam mentrasfer data dari *temporary server* ke *server* utama, serta untuk mengetahui tingkat kecepatan dalam mentrasfer data. Sebagai uji coba dilakukan proses sinkronisasi data dengan memanfaatkan jasa *web hosting* menggunakan modem dengan kecepatan 512kbps. Kecepatan dalam transfer data tergantung dari jumlah *record* di *server* utama, dan jumlah data yang akan di transfer. Dari hasil pengujian diperoleh pada tabel 5.

Tabel 5 Hasil Uji Coba Transfer Data pada Jaringan Internet

No	*Record temporary server*	Waktu (detik)
1	10	6
2	50	9
3	100	13
4	150	15

Pembahasan Sistem Pengisian KRS

Dari hasil uji coba diperoleh data kisaran waktu yang dibutuhkan untuk proses pengisian KRS dengan *load balancing* diperoleh waktu rata-rata 4.52 detik. Rata-rata waktu tersebut lebih cepat dibanding waktu yang diperlukan oleh sistem lama berbasis WEB yang membutuhkan waktu rata-rata 4.96 detik, dan 14.26 detik untuk pengisian KRS berbasis *deskto*. Data hasil uji coba menunjukan sistem pengisian KRS dengan *load balancing* lebih cepat baik dalam jumlah sedikit akses, maupun akses yang lebih banyak.

Perbandingan sistem pengisian KRS *load balancing* dengan sistem pengisian KRS (portal akademik) berbasis *desktop* adalah :

1. Dalam hal kecepatan sistem pengisian KRS dengan *load balancing* memiliki kecepatan yang lebih baik daripada pengisian KRS oleh portal akademik.
2. Sistem pengisian KRS berbasis *load balancing* belum memiliki fitur yang sebaik portal akademik seperti cek hasil studi, cek jadwal dan absensi.
3. Sistem pengisian KRS dengan *load balancing* dapat diakses pada jaringan lebih luas melalui browser karena berbasis WEB, sedangkan portal akademik hanya dapat diakses pada PC yang sudah ter-*install* aplikasi portal akademik.
4. Portal akademik memberikan keamanan yang lebih baik dibandingkan sistem pengisian KRS dengan *load balancing* yang berbasis WEB.

Perbandingan sistem pengisian KRS dengan *load balancing* dengan sistem pengisian KRS lama berbasis WEB adalah :

1. Kecepatan akses cepat pada jumlah user sedikit, sedangkan pada jumlah akses banyak sistem pengisia KRS dengan *load balancing* memberikan kecepatan lebih baik.
2. Sistem lama memiliki fitur yang lebih baik seperti cek transkip nilai, cetak KRS, ubah *password login*.

Pembahasan Sinkronisasi Database

Hasil Uji coba sinkronisasi data antara *server* utama dan *temporary server* dengan jumlah *record temporary server* sebanyak 150 *record* pada LAN membutuhkan waktu 6 detik, sedangkan pada jaringan internet membutuhkan waktu 15 detik. Waktu yang dibutuhkan untuk proses sinkronisasi dipengaruhi jumlah *record,* hal tersebut ditunjukan dengan

peningkatan waktu proses yang berbanding lurus dengan jumlah *record*. Semakin banyak jumlah *record database* semakin banyak waktu yang diperlukan. Kecepatan eksekusi program pada jaringan internet juga dipengaruhi oleh kecepatan akses internet.

5. Kesimpulan

1. Sistem pengisian KRS dengan *load balancing* dapat meningkatkan kecepatan dalam menangani setiap *request*. Hal tersebut berdasarkan hasil percobaan dimana diperoleh rata-rata 4.52 detik. Rata-rata waktu tersebut lebih cepat dibanding sistem lama berbasis WEB yang membutuhkan waktu rata-rata 4.96 detik, dan 14.26 detik untuk pengisian KRS berbasis *desktop*.
2. Waktu yang dibutuhkan untuk proses sinkronisasi dipengaruhi oleh jumlah *record* database dan kecepatan transfer pada jaringan. Hal tersebut berdasarkan hasil percobaan dimana diperoleh peningkatan waktu eksekusi rata-rata 0.24 detik setiap penambahan 50 *record*.
3. Sistem ini menggunakan sinkronisasi yang diproses manual oleh seorang admin. Sinkronisasi dilakukan setelah masa pengisian KRS berakhir dan setelah masa revisi KRS berakhir.

Saran

1. Proses sinkronisasi dapat dikembangkan menjadi sinkronisasi otomatis. Pemanfaatan *crontab* pada sistem operasi *linux* bisa menjadi solusinya.
2. Pengembangan lebih lanjut adalah *server* utama tidak menggunakan *temporary server* melainkan server lain yang memiliki *service* yang sepenuhnya sama dengan *server* utama dengan teknik sinkronisasi yang lebih cepat sehingga dapat memberikan layanan tidak hanya pengisian KRS.

6. Daftar Pustaka

Bourke, Tony, 2011."*Server Load Balancing*", O'Reilly & Associates, Inc., Sebastopol.

Ema, Utami, 2006."*RDBMS dengan PostgreSQL di GNU/Linux*", Penerbit Andi, Yogyakarta.

Hewlett-Packard Development Company, L.P, 2012." *Load Balancing Technology White Paper*".

Indrajit, Eko, 2002."*Buku Pintar Linux: Database Server PostgreSQL*", PT. Elex Media Komputindo, Jakarta.

Suprianto, Dodit, 2009."*Buku Pintar Pemrograman PHP*", OASE Media, Bandung.

Anonim, 2009, Pengertian *Load Balancing http.//allnetedu.blogspot.com/2009/03/pengertian-load-balancing.html,* pada tanggal 23 April 2012, pukul 21.03 wib

Anonim, 2011, *Load Balancing*

http://imamnet.files.wordpress.com/2011/01/makalah-load-balancing.pdf pada tanggal 23 April 2012, pukul 21.03 wib

Anonim, 2011, *Load Balancing*

http://sisteminformasi.files.wordpress.com/2007/02/load-balancing.doc, pada tanggal 23 april 2012, pukul 21.03 wib

Anonim, 2011, *Virtual Server*

http://sisteminformasi.files.wordpress.com/2007/02/virtual-server.doc, pada tanggal 23 april 2012, pukul 21.03 wib

Anonim, 2011, Sinkronisasi

http://puancek.blogspot.com/2011/04/metode-transmisi-asinkron-dan-sinkron.html ,pada tanggal 23 april 2012, pukul 21.03 wib

Microsoft, 2012, *Network Load Balancing Technical Overview*

http://technet.microsoft.com/en-us/library/bb742455.aspx, pada tanggal 1 agustus 2012, pukul 20.00 wib

keywords: Server, load balancing, sinkronisasi, postgre