# ANALISIS PERFORMANSI AUTENTIKASI *SINGLE SIGN ON* PADA WEB MENGGUNAKAN LDAP

Futuh Hilmi[1], R. Rumani M[2], Budhi Irawan[3].

Fakultas Elektro dan Komunikasi – Institut Teknologi Telkom
Jl Telekomunikasi No. 1, Dayeuhkolot, Bandung 40257, Indonesia
[1]futuh_fun@yahoo.com, [2]rrm@ittelkom.ac.id,
[2]r_rumani_m@yahoo.com, [3]bir@ittelkom.ac.id

**Abstrak**

Identifikasi yang sering ditemui adalah dalam bentuk *login* yang menggunakan *username* dan *password.* Disini akan timbul kesulitan untuk mengelola *login* tersebut jika seorang pengguna memiliki *login* yang berbeda-beda untuk setiap sistem aplikasi, karena dengan demikian seseorang harus dapat mengingat banyak *username* dan *password*. *Single Sign-On* ( SSO ) merupakan fasilitas yang memberikan kemudahan untuk *user* yang melakukan *login* ketika menjelajah di *internet.* Dengan menggunakan metode SSO ini, setiap *user* hanya perlu memiliki satu *username*, dan satu *password*. *User* hanya perlu *login* satu kali saja agar dapat menggunakan semua fasilitas yang ada di *web* utama. *User* tidak perlu mengingat banyak *account*, dan tidak perlu berulang kali melakukan *login*. Hal ini juga dapat mempermudah dalam pengorganisasian data *user* yang ada, karena menggunakan tempat penyimpanan data *user* yang terpusat. CAS digunakan untuk menangani masalah komunikasi antara *web*. LDAP digunakan sebagai sebuah protokol layanan direktori , dimana semua data *user* disimpan di dalam LDAP. Dari hasil analisa performansi *single sign on* pada *web,* diperoleh nilai *response time server* sebesar 0.90 *second*, dan dalam 1 detik *server* mampu melayani 110 *request* dari *user*. Fasilitas ini dapat melayani hingga 500 *user* yang melakukan *request* secara bersamaan dalam satu waktu kepada *server single sign on.* Hal ini dikarenakan respon *error server* ketika melayani 500 *user* bernilai nol. Waktu tercepat *server* dalam melayani *user* adalah 191 ms dan waktu terlama dalam melayani *user* adalah 11864 ms.

***Kata kunci**: autentikasi, single sign-on, web, LDAP*

## 1. Pendahuluan

Seiring dengan aplikasi yang menggunakan *web*, adanya identifikasi terhadap *user* sudah menjadi faktor yang sangat penting. Identifikasi yang sering ditemui adalah dalam bentuk *login* yang manggunakan *username* dan *password.* Disini akan timbul kesulitan untuk mengelola *login* tersebut jika seorang pengguna memiliki *login* yang berbeda-beda untuk setiap sistem aplikasi, dengan konsekwensinya seseorang harus selalu mengingat banyak *username* dan *password.*

Ada suatu sistem yang sangat mempermudah bagi *user*, *web server* atau aplikasi *web* yang menerapkan berbagai bentuk pengaturan autentikasi dan *session web*. *Single Sign-On* adalah salah satu dari autentikasi alternatif yang sangat praktis dan efisien.

Tujuan dari penelitian ini adalah untuk mendesain sistem otentikasi terpusat (*Single Sign On*) yang dapat diterapkan pada berbagai aplikasi berbasis *web*. Dengan mengimplementasikan sistem ini, maka *user* akan dipermudah karena hanya memiliki satu *account* untuk dapat mengakses beberapa layanan yang ada. Setelah sistem dirancang, dilakukan analisis performansi autentikasi *single sign-on* pada *web* menggunkanan LDAP.

Sistem yang dirancang menggunakan *openLDAP server* dengan membuat sistem *Single Sign-On* yang digunakan untuk *login* ke sebuah *web*. Sistem operasi yang digunakan adalah

Linux Ubuntu 10.04, sedangkan *Web* yang digunakan adalah sebuah prototype, dimana dalam *web* ini disediakan *web* lain yang bisa diakses menggunakan *Single Sign-On*.

Dalam penelitian ini, yang digunakan sebagai *database user* adalah LDAP*server*, sedangkan *web server* menggunakan MySQL dan untuk *Single Sign On* digunakan *Central Authetication Service*.

## 2. Kajian Pustaka

### 2.1 Autentikasi

Autentikasi adalah proses usaha pengecekan identitas seorang pengguna sistem komunikasi pada proses *login* ke dalam sebuah sistem[1]. Pengguna yang telah lolos pengecekan identitas adalah pengguna resmi pada sistem, orang yang memiliki otoritas atas sistem, atau mungkin aplikasi yang berjalan pada sistem. Penggunaan sistem otentikasi diharapkan dapat membentuk sebuah sistem khusus, yang hanya dapat dipergunakan oleh orang-orang yang memiliki hak guna. Ada perbedaan antara otentikasi dengan otorisasi; istilah otentikasi digunakan untuk pembuktian sebagai proses pengecekan identitas seorang pengguna, sedangkan otorisasi adalah proses pengecekan bahwa pengguna yang dikenal memiliki kekuasaan untuk melakukan tindakan tertentu[2].

### 2.2 *Web Service*

*Web service* merupakan suatu antarmuka yang menyediakan berbagai kumpulan operasi yang dapat diakses pada jaringan melalui pengiriman pesan *Extensible Markup Language* (XML). Standar industri untuk pengiriman pesan XML pada *web service* sekarang ini adalah dengan menggunakan *Simple Object Access Protocol* (SOAP). SOAP merupakan protokol komunikasi berbasis XML sederhana yang digunakan untuk pertukaran data terstruktur antar aplikasi dalam suatu jaringan[9].

### 2.3 *Single Sign-On* (SSO)

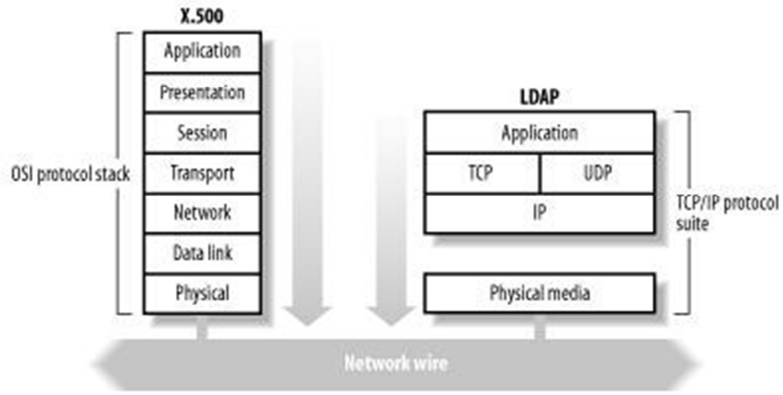
Teknologi *Single Sign-On* (SSO) adalah teknologi yang mengizinkan pengguna jaringan atau *user* agar dapat mengakses sumber daya dalam jaringan hanya dengan menggunakan satu akun pengguna saja[7]. Dengan menggunakan SSO, seorang pengguna hanya cukup melakukan proses otentikasi sekali saja untuk mendapatkan izin akses terhadap semua layanan yang terdapat di dalam jaringan.

### 2.4 *Lightweigth Directory Access Protocol*

#### 2.4.1 *Lightweight*

LDAP dikatakan ringan jika dibandingkan dengan X.500 servis direktori[5]. Hal ini dikarenakan pada mulanya *root* LDAP sangat dekat terkait dengan X.500. LDAP pada mulanya dibuat sebagai protokol *desktop* yang lebih muda yang digunakan sebagai *gateway* untuk X.500 *server*. X.500 merupakan sebuah standar. X.500 bersifat "*Heavyweight"* karena membutuhkan *client* dan *server* untuk berkomunikasi menggunakan *Open System Interface (OSI)* protokol.

LDAP dikatakan ringan ("*lightweight"*) karena menggunakan relatif sedikit pesan yang dipetakan secara langsung pada TCP *layer* (biasanya *port 389*) dari protokol TCP/IP[6]. Karena X.500 merupakan *layer* aplikasi protokol (dalam OSI *layer*) maka ini akan membawa lebih banyak bawaan, seperti *network header* yang dipasang pada paket di setiap *layer* sebelum akhirnya dikirimkan ke jaringan.




**Gambar 2.1 X.500 dengan OSI, LDAP dengan TCP/IP[5]**

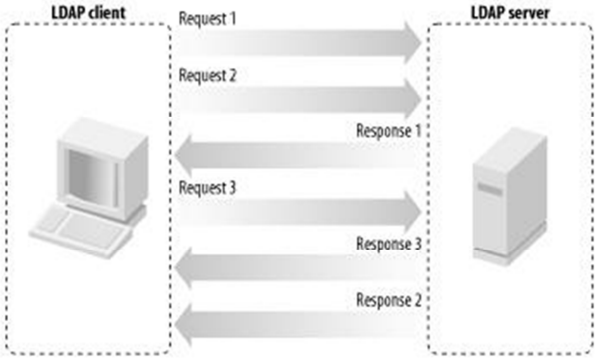
LDAP juga dikatakan ringan ("*lightweight*"), karena menghilangkan banyak operasi X.500 yang jarang digunakan. LDAPv3 hanya memiliki sembilan operasi utama dan menyediakan model sederhana untuk *programmer* dan *administrator*. Menyediakan satu set operasi yang lebih kecil dan sederhana yang mengijinkan pengembang untuk fokus pada seluk-beluk dari program tanpa harus mengerti keunggulan dari protokol yang jarang digunakan. Dengan jalan ini pembuat LDAP berharap untuk meningkatkan penggunaan dengan menyediakan pengembangan aplikasi yang lebih mudah.

### 2.4.2 *Directory*

Jaringan servis direktori bukanlah suatu hal yang baru, khususnya untuk *Domain Name System* (DNS)[3]. Servis direktori dan *database* memiliki karakteristik masing-masing, seperti pencarian cepat dengan skema yang dapat diperluas. Perbedaannya adalah direktori dibuat untuk "dibaca" lebih banyak dari pada "ditulis". Sedangkan untuk *database* diasumsikan untuk operasi "baca" dan "tulis" memiliki frekuensi yang sama. Asumsi bahwa direktori "dibaca" lebih sering dari pada "ditulis" merupakan suatu keunggulan yang spesifik untuk sebuah *database*.[4]

### 2.4.3 *Access Protocol*

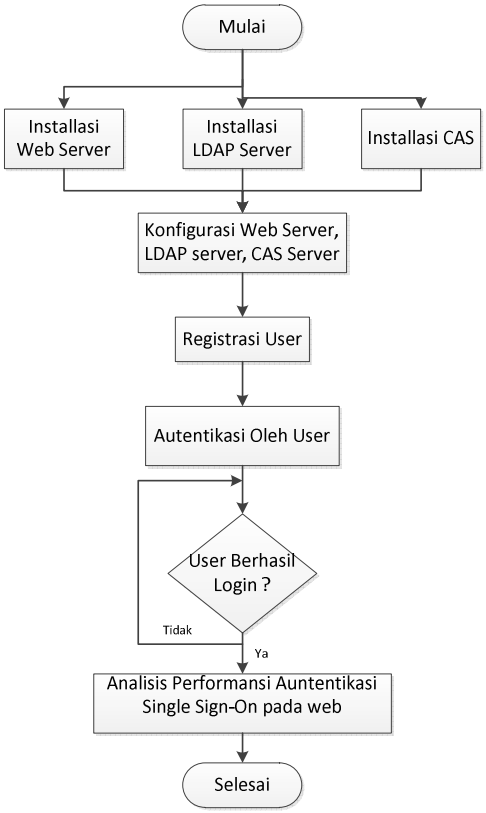
LDAP merupakan *message-based, client/server protocol* yang didefinisikan dalam RFC 2251[5]. LDAP bersifat *asynchronous* (meskipun banyak peralatan menyediakan baik *blocking* dan *nonblocking* API), ini berarti bahwa sebuah *client* dapat menimbulkan banyaknya permintaan dan *response*-nya mungkin datang dalam urutan yang berbeda ketika permintaan itu dimunculkan.




**Gambar 2.3 LDAP request dan response[6]**

## 3. Design dan Pengujian Sistem

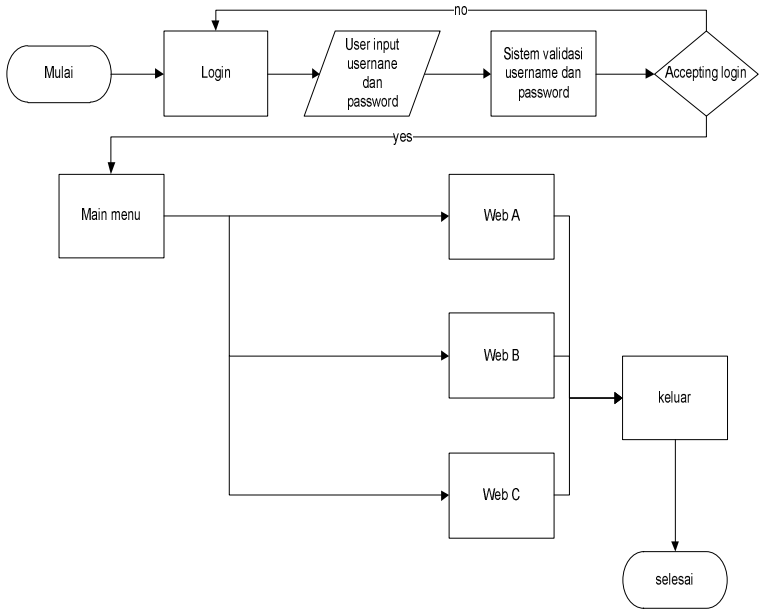
### 3.1 Perancangan Sistem




**Gambar 3.1 Diagram Alir Perancangan Sistem[8]**

Langkah pertama yang akan dilakukan adalah instalasi *web server*, instalasi *LDAP server*, dan instalasi *Central Authentication Service* (CAS)[8]. Istilah *Central Authentication Service* (CAS) berfungsi untuk melakukan otentikasi dan otorisasi *user* serta memberikan *ticket* elektronik atau *Services Ticket* (ST) ke *client*[10]. Dengan tiket ini *user* dapat melakukan akses ke beberapa aplikasi. Manajemen data *user* yang digunakan dalam sistem ini menggunakan struktur internal dari LDAP *directory* yang ditangani oleh sebuah *server* LDAP. Berikutnya akan dilakukan konfigurasi pada *web*, *LDAP server*, dan *CAS* agar ketiganya saling terhubung. Setelah semua sistem terintegrasi dengan baik, maka langkah berikutnya adalah melakukan proses otentikasi pada *web.* Pada tahap ini *user* harus *login* untuk dapat mengakses web yang telah terhubung ke *LDAP server.* Namun, jika terjadi kesalahan atau *error* pada saat *login* maka *user* harus mencoba lagi untuk *login* kembali. Untuk dapat *login* ke sistem, *user* harus terlebih dahulu melakukan registrasi. Proses ini adalah proses memasukkan data *user* kedalam *database* agar *user* terdaftar sehingga bisa melakukan *login*. *Database* yang digunakan untuk menyimpan data *user* ini adalah *database* dari LDAP *server.* Ketika *user* berhasil *login* maka akan mulai dilakukan analisa performansi *single sign-on* pada web yang berbasis LDAP.

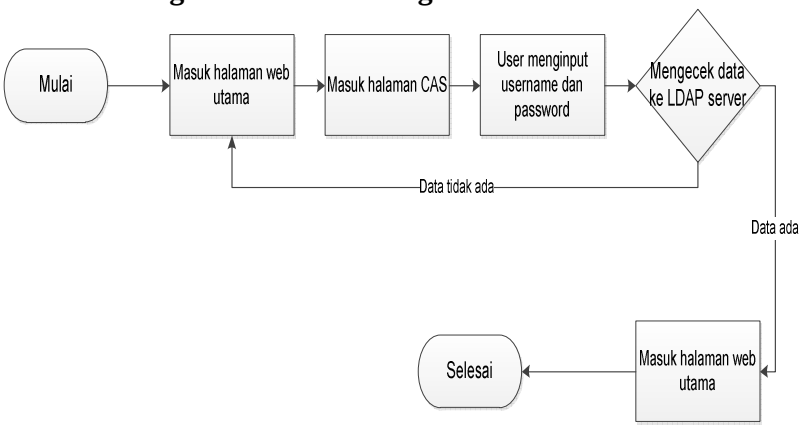
### 3.2 Perancangan Proses Sistem




**Gambar 3.2 Diagram alir proses keseluruhan**

*User* harus melakukan login dahulu. Setelah *user* berhasil melewati tahapan *login*, *user* baru dapat melakukan pilihan layanan yang terdapat pada menu utama. *Username* dan *password* yang diinput oleh *user* adalah *username* dan *password* yang terdaftar pada LDAP *server*, bukan pada *web server*.

### 3.3 Perancangan Proses User Login




**Gambar 3.3 Diagram alir untuk login**

Pertama *user* harus mengakses halaman utama web. Jika *user* tidak *login*, maka *user* tidak bisa menikmati layanan yang disediakan *web* utama. Ketika *user* melakukan *login*, maka akan tampil halaman CAS. Dihalaman itu *user* harus memasukkan *username* dan *password* yang telah tersimpan di LDAP *server*. Ketika *user* berhasil *login* maka *user* dapat memanfaatkan layanan yang ada.

### 3.4 Skenario Pengujian Sistem

Pada penelitian dilakukan pengujian terhadap *single sign on* pada *web* dengan beberapa macam skenario, diantaranya:

1. Pengujian terhadap CAS ( *Central Authentication Service* ).

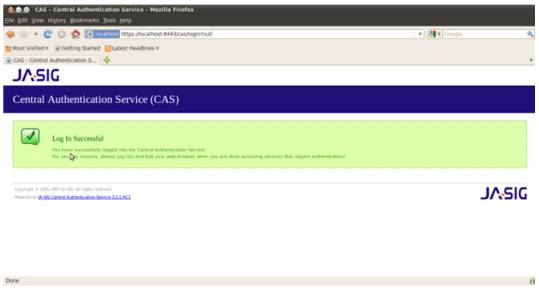
2. Pengujian terhadap *single sign on* pada *web.*
3. Analisa performansi *Single Sign On.* Pada tahap ini dilakukan analisa dengan dua skenario. Kedua skenario tersebut dilakukan untuk mengetahui kecepatan akses *user* dan respon *error* dari *Single Sign On.* Skenario pertama adalah dengan mengirimkan *thread request* kepada *server* CAS sejumlah 1, 10, 50, 100, 200, 300, 500 buah secara bersamaan dalam satu waktu, sedangkan skenario kedua adalah dengan megirimkan *thread request* kepada *server* CAS sejumlah 1, 10, 50, 100, 200, 300, 500 buah dengan adanya *delay* 0.5 detik untuk setiap kali *user* melakukan *request.*

4 Analisa Kinerja Sistem Keseluruhan

Uji coba dilakukan dengan mengakses aplikasi CAS, aplikasi ini mempunyai URL : https://localhost:8443/cas/login?null.

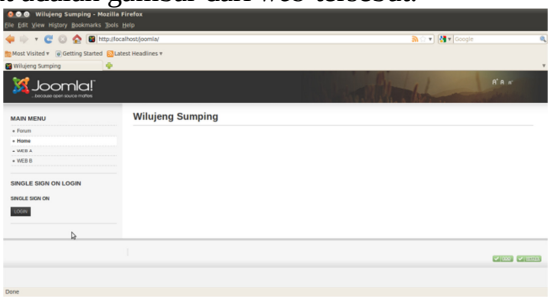

**Gambar 4.1 Tampilan halaman CAS**

Proses selanjutnya adalah *user* melakukan proses otentikasi dengan memasukkan *username* dan *password* di halaman *login* CAS. *User* akan berhasil *login*, jika CAS telah terhubung ke LDAP *server* dan *user* tersebut telah terdaftar di LDAP *server.*



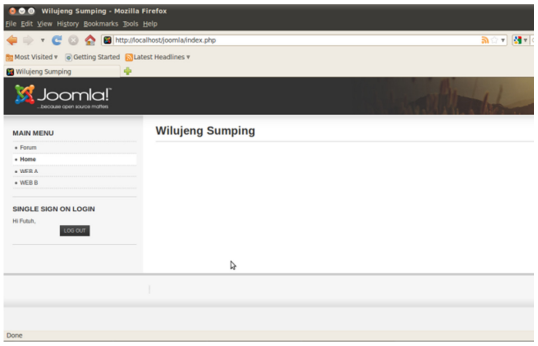

**Gambar 4.2 Halaman CAS ketika *user* berhasil *login***

Pada tahap ini dilakukan pengujian terhadap proses *single sign on* pada *web*. Langkah pertama adalah membuka *web* utama yaitu pada *browser*, dengan alamat web : localhost/joomla/. Berikut adalah gambar dari *web* tersebut.



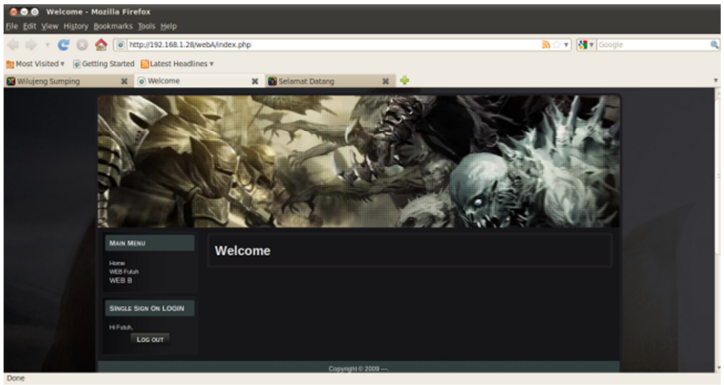

**Gambar 4.3 Tampilan halaman *web* utama**

Melakukan *Login* pada fasilitas *single sign on*, berakibat *user* langsung diarahkan ke halaman CAS. Dihalaman CAS *user* diharuskan memasukkan *username* dan *password*, dimana *username* dan *password* sebelumnya sudah harus terdaftar di *databse* LDAP *server*. *Login* dengan *username* ”futuh” dan *password* ”123456”. Setelah berhasil *login*, kembali ke halaman *web* utama sebagai *user* yang sudah terotentifikasi. Berikut gambar *web* ketika *login* sudah berhasil.

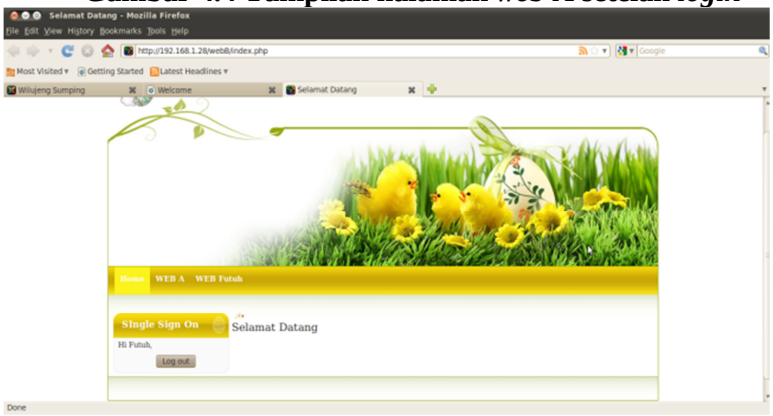



**Gambar 4.3 Tampilan halaman *web* utama setelah *login***

Dari gambar diatas dapat dilihat ketika *user* sudah *login* maka akan muncul nama *user* sebagai *user* yang sudah terotentifikasi. Proses berikutnya adalah menguji keberhasilan dari *single sign on*. Pada *web* utama sudah terdapat menu yang menyediakan dua *web* lain yang sudah dilengkapi fasilitas *single sign on*. Berikut gambar dari WEB A dan WEB B setalah *login single sign on*.




**Gambar 4.4 Tampilan halaman *web* A setelah *login***



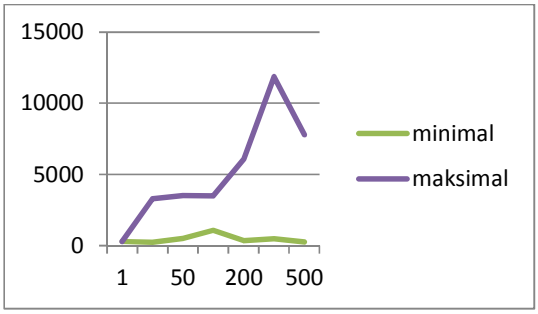

**Gambar 4.5 Tampilan halaman *web* B setelah *login***

Pada tahap ini selanjutnya dilakukan analisa performansi *server* CAS, dengan menggunakan *software Web Stress Tools 7*. Skenario pertama pada proses analisa ini adalah

dengan mengirimkan *thread request* kepada *server* CAS sejumlah 1, 10, 50, 100, 200, 300, 500 buah secara bersamaan dalam satu waktu, mensimulasikan *user* melakukan akses ke aplikasi CAS secara bersamaan. Berikut adalah tabel waktu respon untuk skenario pertama.

**Tabel 4.1 Tabel waktu respon *server* CAS skenario pertama**

| User | Waktu respons (ms) | | respon error (%) |
|---|---|---|---|
| | minimal | maksimal | |
| 1 | 290 | 290 | 0 |
| 10 | 229 | 3299 | 0 |
| 50 | 501 | 3516 | 0 |
| 100 | 1078 | 3496 | 0 |
| 200 | 358 | 6064 | 0 |
| 300 | 474 | 11864 | 0 |
| 500 | 254 | 7783 | 0 |




**Gambar 4.6 Grafik unjuk kerja CAS *Server* skenario pertama**

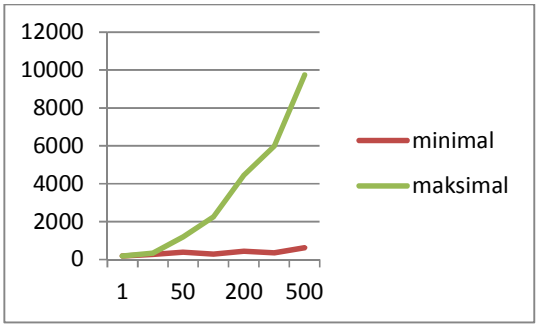
Waktu minimal adalah waktu terkecil yang dibutuhkan oleh *user* untuk mengakses *server*. Dari tabel diatas, ketika *server* melayani 1 *user,* kecepatan akses *user* adalah 290ms. Akan tetapi ternyata itu bukanlah waktu tercepat, waktu kecepatan akses *user* tertinggi adalah sebesar 229ms dan terjadi ketika *server* sedang menerima *request* dari 10 *user* secara bersamaan. Waktu kecepatan akses dari *user* yang bernilai paling tinggi pun ternyata tidak terjadi ketika *server* melanayani *request* dari 500 *user* secara bersamaan. Ketika *user* melayani *request* dari 500 *user* secara bersamaan, kecepatan akses *user* hanya bernilai 254ms. Waktu kecepatan akses tertinggi adalah sebesar 1078ms dan terjadi ketika *server* melayani 100 *user* secara bersamaan. Hal ini mungkin saja terjadi dikarenakan beberapa faktor, diantaranya adalah *traffic* pada jaringan, ketersediaan *bandwith* dari *sever* dan kemampuan *hardware* terutama *prosesor*. Waktu maksimal adalah waktu terlama dari *server* untuk merespon *request* dari *user*. Dapat juga dikatakan sebagai waktu kecepatan akses terlama *user*. Ketika 1 *user* mengakses *server* waktu akses terlama *user* bernilai 290ms. Besarnya waktu terus meningkat seiring dengan meningkatnya jumlah *user* yang *request* ke *server*. Ketika 500 *user* melakukan *request* secara bersamaan ke *server* waktu terlama adalah sebesar 7783ms. Semakin besarnya nilai waktu kecepatan akses *user* disebabkan oleh meningkatnya *request* ke server dari *user,* menyebabkan menurunnya kemampuan *server* dalam melayani *user*. Respon *error* adalah prosentase dari perbandingan antara jumlah *user* yang *request* ke *server* dengan jumlah *user* yang dapat dilayani oleh *server*. Ketika hanya ada 1 *user,* respon *server* bernilai 0, artinya *server* dapat melayani *user*. Ketika 10, 50, 100, 200, 300 hingga 500 *user* melakukan *request* secara bersamaan pun, respon *error* tetap bernilai nol. Hal ini akan menunjukkan server dapat melayani 500 *request* dari *user* dalam waktu yang bersamaan.

Skenario kedua dilakukan dengan memberikan beban *request* ke *server* CAS yaitu dengan diberikannya *delay* 0.5 detik untuk setiap kali *user* melakukan *request.* Pada skenario kedua

ini jumlah *user* yang akan *request* ke *server* sama dengan jumlah *user* pada skenario pertama yaitu 1, 10, 50, 100, 200, 300, 500. Berikut tabel waktu respon sebagai hasil yang diperoleh dalam skenario kedua.

**Tabel 4.2 Tabel waktu respon *server* CAS skenario kedua**

| User | Waktu respon(ms) | | respon error (%) |
|---|---|---|---|
| | minimal | maksimal | |
| 1 | 191 | 191 | 0 |
| 10 | 267 | 335 | 0 |
| 50 | 376 | 1182 | 0 |
| 100 | 284 | 2249 | 0 |
| 200 | 444 | 4438 | 0 |
| 300 | 352 | 5972 | 0 |
| 500 | 610 | 9737 | 0 |




**Gambar 4.7 Grafik unjuk kerja CAS *Server* skenario kedua**

Waktu minimal adalah waktu terkecil yang dibutuhkan oleh seorang *user* untuk mengakses *server*. Dari tabel diatas, ketika *server* melayani 1 *user*, kecepatan akses *user* adalah 191ms. Waktu kecepatan akses tertinggi adalah sebesar 610ms dan terjadi ketika *server* melayani 500 *user*. Waktu maksimal adalah waktu terlama dari *server* untuk merespon *request* dari *user*, yang merupakan waktu kecepatan akses terlama *user*. Ketika 1 *user* mengakses *server* waktu akses terlama diperoleh 191ms. Besarnya waktu terus meningkat seiring dengan meningkatnya jumlah *user* yang *request* ke *server*. Ketika 500 *user* melakukan *request* ke *server* waktu terlama adalah sebesar 9739ms. Semakin besarnya nilai waktu kecepatan akses *user* disebabkan oleh meningkatnya *request* ke server dari *user*, berakibat menurunnya kemampuan *server* dalam melayani *user*. Semakin banyak jumlah *user* yang melakukan *request* ke *server*, maka semakin besar waktu yang dibutuhkan oleh *server* untuk merespon *request* dari *user*. Tetapi besarnya waktu respon berbeda pada skenario pertama dan skenario kedua. *Server* lebih cepat merespon ketika *user* diberi *delay* sebanyak 0.5s daripada ketika *user* melakukan *request* secara bersamaan terhadap *server*. Respon *error* adalah prosentase dari perbandingan antara jumlah *user* yang *request* ke *server* dengan jumlah *user* yang dapat dilayani oleh *server*. Ketika 1, 10, 50, 100, 200, 300 hingga 500 *user* melakukan *request*, respon *error* tetap bernilai nol.

## 5.1. Kesimpulan

Kesimpulan yang dapat diambil dari hasil penelitian yang telah dilakukan, adalah sebagai berikut :

1. *Response time server* bernilai 0.90 *second* dengan nilai *concurrency* sebesar 99.46. *Server* dapat melayani 110 *request* dalam 1 detik.
2. Kecepatan akses bagi *user* bergantung pada banyaknya *request* yang diterima *server*. Jika *server* sedang melayani banyak *request* maka kecepatan akses pun akan menurun. Ketika tidak ada *delay* kecepatan akses *user* minimal bernilai 229ms dan kecepatan akses *user* maksimal bernilai 11864 ms. Ketika diberi *delay* sebesar 0.5s kecepatan akses *user* minimal bernilai 191ms dan kecepatan akses *user* maksimal bernilai 9739 ms.
3. *Server* dapat melayani hingga 500 *user* secara bersamaan. Hal ini dapat dilihat ketika *server* melayani 500 *user* secara bersamaan, nilai *error rate* adalah nol (0).

**5.2. Saran**

Beberapa saran yang dapat diberikan untuk pengembangan sistem *single sign on* ini adalah sebagai berikut :

1. *Web* yang digunakan bukan lagi sebuah *prototype*, akan tetapi merupakan sebuah *web* utuh dengan dilengkapi berbagai fitur.
2. Untuk pengembangan selanjutnya dapat dicoba untuk mengintegrasikan aplikasi *desktop* sehingga dapat menggunakan metode *Single Sign On* ini, khususnya untuk aplikasi *Messenger*.
3. Untuk selanjutnya dapat dianalisa keamanan dan cara menanggulangi serangan-serangan dari luar sistem, agar aplikasi *Single Sign On* ini mempunyai tingkat keamanan yang memadai dan dapat diterapkan dilingkungan dalam kondisi nyata di lapangan, bukan di dalam skala laboratorium.

**Referensi**

[1]. Arkills, Brian., 2003, *LDAP Directories Explained : An Introduction and Analysis*, Addison Wesley, Boston, Massachusetts, USA.

[2]. Carter, Gerald., 2003, *LDAP System Administration*, O'Reilly, Inc, California, USA.

[3]. Donley, Clayton., 2000, *LDAP Programming : Directory Management and Integration*, Manning Publications, Co., USA.

[4]. Howes, Timothy A., Mark S. Smith, and Gordon S. Good., 1999, *Understanding and Deploying LDAP Directory Services*, Macmillan Technical Publishing, USA.

[5]. Howes, Tim., Mark S. Smith, 1997, *LDAP ; Programming Directory – Enabled Applications with Lightweight Directory Access Protocol*, Macmillan Technical Publishing, USA.

[6]. Loshin, Peter., 2000, *Bigbook of Lightweight Directory Access Protocol (LDAP) Rfcs*, Morgan Kauffman Publishers, USA.

[7]. Mularien, Peter., 2010, *Spring Security 3*, Packt Publishing, USA.

[8]. Siswoutomo, Wiwit., 2008, *Step by Step Joomla!*, Penerbit : ElexMedia Komputindo, Gramedia, Jakarta.

[9]. Sofana, Iwan., 2007, *Mudah Membangun Server dengan Fedora Core*, Penerbit Informatika, Jakarta.

[10]. Wilcox, Mark., 1999, *Implementing LDAP*, Wrox Press, Inc., USA.