LAPORAN PENELITIAN

PERANGKAT LUNAK UJI PERFORMANSI DAN KAPASITAS SITUS WEB TEROTOMASI MULTI AGEN

Oleh: Gede Karya, S.T., M.T., CISA

Jurusan Teknik Informatika

Fakultas Teknologi Informasi dan Sains

Universitas Katolik Parahyangan

Juni 2011

ABSTRAK
PERANGKAT LUNAK UJI PERFORMANSI DAN KAPASITAS SITUS WEB TEROTOMASI MULTI AGEN

Gede Karya, S.T., M.T.
Jurusan Teknik Informatika, Fakultas Teknologi Informasi dan Sains,
Universitas Katolik Parahyangan

Salah satu rangkaian penting dalam pengembangan perangkat lunak adalah fase uji coba. Agar perangkat lunak yang dikembangkan berkualitas dan handal, maka selain uji fungsional dan terintegrasi, diperlukan juga uji performa dan kapasitas. Pada penelitian ini dikembangkan perangkat lunak yang dapat digunakan untuk menguji performa dan kapasitas perangkat lunak berbasis web (situs web). Protokol yang disupport adalah HTTP dan HTTPS dengan operasi GET. Untuk memenuhi kriteria kapasitas koneksi digunakan pendekatan multi agen terdistribusi dengan teknologi multi threading. Agar koordinasi antar agen dapat sinergi, maka dilakukan proses otomasi kontrol dan monitoring. Output yang dapat dihasilkan oleh perangkat lunak ini berupa: halaman web uji, performa (waktu) dan kapasitas (koneksi). Perangkat lunak ini diimplementasikan dengan bahasa Java dengan teknologi J2SE sehingga dapat berjalan lintas platform (sistem operasi). Dari hasil pengujian, banyaknya agen uji yang dapat diemulasi tergantung pada prosesor, memori dan kartu jaringan.

Kata kunci: perangkat lunak uji, kapasitas, performansi, multi agen

Kata Pengantar

Laporan penelitian ini merupakan rekapitulasi dari sejumlah aktivitas untuk mendisain dan mengimplementasikan perangkat lunak uji kapasitas dan performa aplikasi berbasis web. Penelitian ini diperlukan untuk melengkapi uji fungsional yang selama ini dilakukan dalam pengembangan sistem informasi berbasis web di Unpar. Penelitian ini didanai oleh Unpar melalui LPPM.

Terima kasih kepada Ketua Jurusan Teknik Informatika, Dekan Fakultas Teknologi Informasi dan Sains atas dukungan dan arahan serta persetujuan atas usulan penelitian ini. Terima kasih juga kepada Ketua LPPM atas pendanaan yang diberikan. Selain itu juga Penulis sampaikan terima kasih kepada Sdr. Anoki Kiyosi atas keikutsertaannya dalam mengimplementasikan program hasil rancangan dalam bahasa Java, dan rekan-rekan staf Biro Teknologi Informasi atas kesertaannya dalam proses pengujian dalam rangka mempersiapkan perangkat lunak aplikasi berbasis web untuk FRS sebagai sample uji.

Akhir kata semoga laporan ini dapat memberikan gambaran motivasi, metodologi dan hasil dari penelitian yang telah dilakukan.

Peneliti,

Gede Karya, S.T., M.T., CISA

Daftar Isi

Daftar Gambar

BAB 1 PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang, rumusan masalah, tujuan dan hasil serta sistematika pembahasan.

1.1. Latar Belakang

Salah satu hal terpenting dalam sebuah produk perangkat lunak adalah kualitas. Kualitas didefinisikan sebagai kemampuan suatu produk untuk memenuhi/ memuaskan kebutuhan penggunanya. Kebutuhan perangkat lunak dapat dibagi menjadi 2, yaitu: kebutuhan fungsional dan kebutuhan non fungsional. Kebutuhan fungsional berkaitan dengan proses bisnis atau fungsi yang diotomasi oleh perangkat lunak. Fungsi ini terkait dengan aktifitas dan tata cara/ perhitungan (alur logika) yang ada di dalamnya beserta data-data yang dioleh oleh perangkat lunak tersebut. Kebutuhan non fungsional mencakup kebutuhan akan reliabilitas, kapasitas, performansi dan sejenisnya.

Bagaimana cara memastikan bahwa suatu perangkat lunak berkualitas? Caranya adalah dengan melakukan uji kualitas (pengujian). Karena komponen kualitas adalah fungsional dan non fungsional, maka pengujiannyapun demikian. Pada pengujian fungsional, dapat dilakukan uji terima yaitu dengan mengecek semua fungsi yang disyaratkan apakah sudah dapat berjalan/ diakomodasi oleh perangkat lunak. Jadi fungsi perangkat lunak dibandingkan dengan fungsi-fungsi dalam proses bisnis yang diotomasi. Dengan demikian dapat dilakukan secara simulasi pada lingkungan pengembangan.

Khusus untuk uji non fungsional, ada beberapa yang tidak dapat dilakukan dengan mudah, misalnya: uji performansi pada saat *peak season*. Ini memerlukan lingkungan ekstrem sesuai dengan definisi kapasitas, berupa *sample user* dan kondisi transaksi maksimal. Demikian juga dengan uji reliabilitas, memerlukan sejumlah kapasitas dan rentang waktu yang cukup panjang untuk menjamin bahwa dalam jangka waktu yang disyaratkan perangkat lunak tidak akan bermasalah.

Untuk suatu kelayakan operasi, baik uji fungsional maupun uji non fungsional harus dilakukan, sehingga menjamin bahwa perangkat lunak sudah siap. Hal ini menjadi suatu resiko yang signifikan khususnya pada sistem enterprais, dimana akan melayani pengguna dan transaksi dalam jumlah besar dengan intensitas dan reliabilitas yang memadai.

Khusus untuk uji non fungsional, diperlukan suatu alat dan lingkungan uji yang dapat mewakili kondisi yang ingin diujikan. Oleh karena itu, pada penelitian ini dikembangkan sebuah perangkat lunak uji yang dapat digunakan untuk melakukan uji kapasitas dan performansi.

1.2. Rumusan Masalah dan Batasan

Berdasarkan latar belakang di atas, maka pada penelitian ini dirumuskan masalah yang akan ditangani adalah:

Bagaimana mengembangkan perangkat lunak uji coba yang dapat mensimulasikan kondisi trafik transaksi (kapasitas) tertentu dan mengemulasi user agent yang terlibat dalam operasional?

Agar lebih fokus, dalam penelitian ini dibatasi pada hal-hal berikut:

1. Aplikasi yang diuji dibatasi pada aplikasi berbasis web, dengan protokol *Hyper Text Transfer Protocol* (HTTP) dan HTTPS (*secure*).
2. Metode yang digunakan dalam operasi HTTP adalah GET.
3. Bahasa pemrograman yang digunakan adalah Java dengan teknologi *Java 2 Standar Edition* (J2SE), sehingga dapat berjalan pada lingkungan *multi platform*.

1.3. Tujuan dan Hasil yang Diharapkan

Berdasarkan rumusan dan latar belakang di atas, maka tujuan yang ingin dicapai pada penelitian ini untuk mengembangkan sebuah perangkat lunak uji kapasitas dan performansi sistem perangkat lunak aplikasi berbasis web.

Oleh karena itu, hasil akhir yang diharapkan adalah perangkat lunak uji yang dapat menghasilkan output berupa:

1. Halaman web hasil uji, yang menyatakan apakah *request* berhasil atau tidak.

2. Indikator performa dalam satuan *mili second* (ms)

3. Indikator kapasitas dalam satuan koneksi konkuren.

1.4. Metodologi Penelitian

Penelitian ini dilakukan dengan metode rekayasa produk, khususnya produk perangkat lunak. Tahapan yang dilalui antara lain:

1. Studi dan eksplorasi tentang implementasi protokol HTTP dan HTTPS, multi threading dan disain protokol aplikasi server telematika.

2. Rekayasa perangka lunak, dengan tahapan sebagai berikut:

    a. Analisis kebutuhan perangkat lunak, baik fungsional maupun non fungsional.

    b. Disain perangkat lunak, mulai dari disain global dan disain detail (basis data, disain protokol dan aplikasi).

    c. Implementasi, berupa pengkodean dalam bahasa Java.

    d. Pengujian, dilakukan untuk kasus sederhana dan kasus kompleks.

3. Uji coba lapangan. Hal ini akan dilaksanakan dengan mengambil sample aplikasi web yang ada di Unpar.

1.5. Sistematika Pembahasan

Laporan penelitian ini disajikan dengan sistematika sebagai berikut:

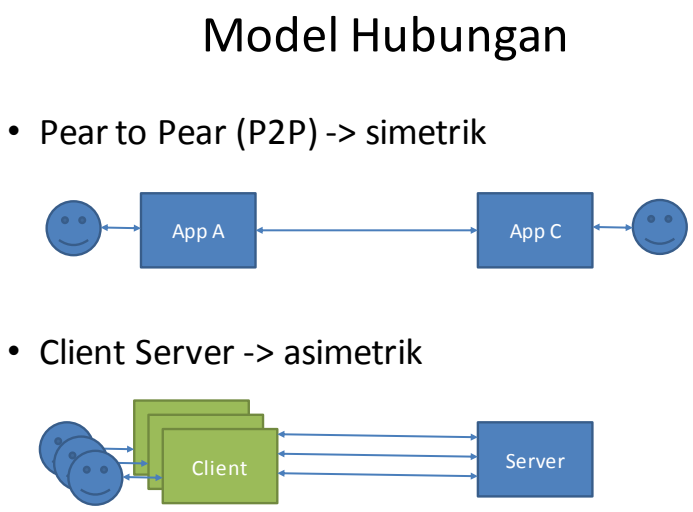
1. Pada bab 1 Pendahuluan dijelaskan tentang latar belakang, rumusan masalah, tujuan, hasil, metodologi dan sistematika pembahasan.
2. Bab 2 Kajian Pustaka, berisi kajian tentang aplikasi telematika, multi threading dan sistem aplikasi berbasis web.
3. Bab 3 Analisis Masalah dan Disain Solusi, berisi pembahasan masalah dan gambaran solusi serta disain dari solusi yang diajukan.
4. Bab 4 Implementasi dan Pengujian, membahas hasil implementasi dan pengujian pernagkat lunak yang dilakukan.
5. Bab 5 Kesimpulan dan Potensi Pengembangan, berisi kesimpulan yang diambil berdasarkan hasil penelitian dan potensi pengembangan yang mungkin untuk penelitian selanjutnya.

BAB 2 STUDI PUSTAKA

Pada bagian ini dibahas tentang hasil studi pustaka yang berhubungan dengan peneltian ini.

2.1. Konsep dan Disain Protokol Aplikasi Telematika

Aplikasi telematika adalah sekumpulan perangkat lunak aplikasi (selanjutnya disebut entitas) yang bekerjasama pada lingkungan jaringan komunikasi data untuk menjalankan suatu fungsi atau layanan tertentu [1]. Dalam konteks ini, setiap entitas dapat berkomunikasi dengan entitas lain memanfaatkan jalur komunikasi yang ada. Pada jaringan internet, protokol yang digunakan adalah TCP/IP. Komunikasi data antar entitas dilakukan pada layer 4 (*Application Layer*) .



Gambar 2.1. Model Hubungan Komunikasi Aplikasi Telematika

Dilihat dari entitas mana yang menginisiasi, komunikasi data antar entitas dalam aplikasi telematika (Gambar 2.1) dapat dilakukan dalam 2 mode, yaitu *Pear-To-Pear* (P2P) atau *Client Server* (CS). Pada model P2P inisiasi dapat dilakukan oleh entitas mana saja. Dalam hal ini semua entitas setingkat, dengan demikian sifatnya simetris. Pada model CS inisiasi

dilakukan oleh aplikasi Client, sedangkan Server hanya akan merespon untuk memberikan jawaban atas apa yang diminta oleh Client. Dengan demikian sifatnya asimetris.

Agar komunikasi dapat berjalan dengan baik, diperlukan adanya aturan berkomunikasi. Aturan ini disebut protokol. Dalam disain protokol, pada dasarnya ditentukan 3 hal yaitu:

1. Format data yang dikomunikasikan
2. Siklus komunikasi dan *flow of control*
3. Definisi perintah, data dan aksi

Pada beberapa protokol aplikasi, seperti HTTP, SMTP, POP format data yang digunakan adalah text. Setiap perintah didefinisikan sebagai sebuah token, dan disertai dengan data yang akan diproses oleh tujuan. Siklus komunikasi diinisiasi oleh Client dan direspon oleh Server sebagai jawaban. Respon dari Server termasuk untuk penanganan kesalahan, seperti: kesalahan sintaks, token tidak dikenali, atau kesalahan urutan (*flow of control*).

Dalam sistem CS, Server dapat melayani lebih dari 1 Client. Oleh karena itu, implementasinya menggunakan multi proses atau multi threading. Konsep multi threading dan implementasinya pada Java dapat dilihat pada bagian 2.2. Pada bagian 2.3 dapat dilihat bagaimana imlementasi Client sebagai User Agent pada sistem aplikasi berbasis protokol HTTP.

2.2. Konsep Multi Threading dan Implementasinya pada Lingkungan Java

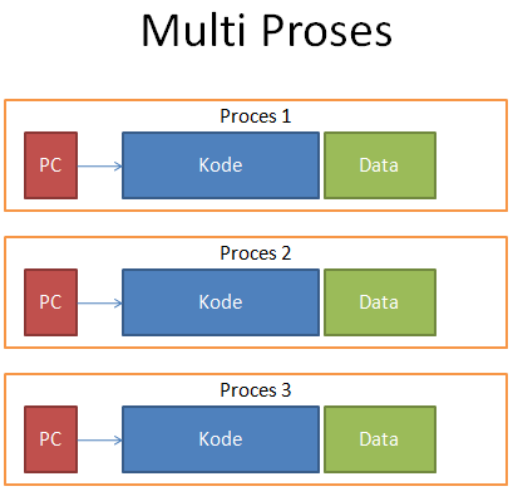
Multi threading merupakan suatu konsep multi proses di mana ada resource sharing terhadap kode proses. Dengan adanya sharing ini, maka kode proses tidak perlu diduplikasi penuh. Untuk n thread, diperlukan 1 instance kode proses, n program counter dan n instance data referensi yang menjadi ruang kerja bagi proses tersebut. Dengan mekanisme multi threading, memungkinkan pemrosesan berjalan paralel, baik yang sebenarnya, maupun pseudo-parallelism [1]. Dengan demikian kapasistas sistem aplikasi meningkat dengan kebutuhan resource yang minimal. Misalkan kode proses sebesar P dan data referensi sebesar D serta overhead proses sebesar O, maka untuk menghasilkan kapasitas n proses diperlukan resource R seperti pada Rumus 2.1.

$$R = P + n*(D+O) \quad \text{..... (2.1)}$$

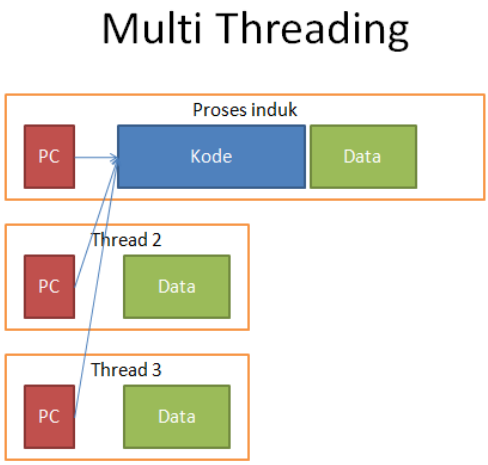
Bandingkan jika menggunakan proses normal dengan kebutuhan resource seperti pada rumus 2.2. Dengan demikian untuk kapasitas besar, misalnya: 10.000 proses, maka multi threading akan jauh lebih efisien dari sisi resource.

$$R = n * (P + D + O) \quad \text{..... (2.2)}$$

Ilustrasi dari perbandingan multi proses dan multi *threading* dapat dilihat pada gambar 2.2 dan 2.3.



Gambar 2.2 Penggunaan Resource Multi Proses



Gambar 2.3 Penggunaan Resource Multi Threading

Java mendukung dengan baik konsep dan implementasi dari multi threading [2]. Pada Java thread diturunkan dari class Thread atau Runnable. Pada gambar 2.4.a dan 2.4.b dapat dilihat contoh penggunaan thread.

Kode yang dieksekusi oleh Thread terdapat pada method run(), artinya semua thread akan menjelankan method ini sekali, jika sudah keluar dari method run() maka thread akan terminate. Dengan demikian jika ingin mengimplementasikan thread sebagai service, maka diperlukan proses pengulangan di method ini. Contohnya dapat dilihat pada Gambar 2.4.c.

```
1   package webtest;
2
3   import java.io.FileWriter;
4   import java.io.IOException;
5   import java.io.PrintWriter;
6   import java.util.Date;
7   import java.util.logging.Level;
8   import java.util.logging.Logger;
9
10  public class TestThread extends Thread
11  {
12      private static int threadcount=0;
13      private static int threadactive=0;
```

(a) Penggunaan extents dari Thread

```
1   package Gateway;
2
3   import java.io.*;
4   import java.net.*;
5   import java.nio.channels.*;
6   import java.nio.channels.spi.SelectorProvider;
7   import java.util.*;
8
9   public class Listener implements Runnable{
10      private int clientCount = 0;
11      private Selector _Selector;
12      private ServerSocketChannel _ServerSocketChannel;
13      private int serverPort;
```

(b) Penggunaan implemen dari Runnable

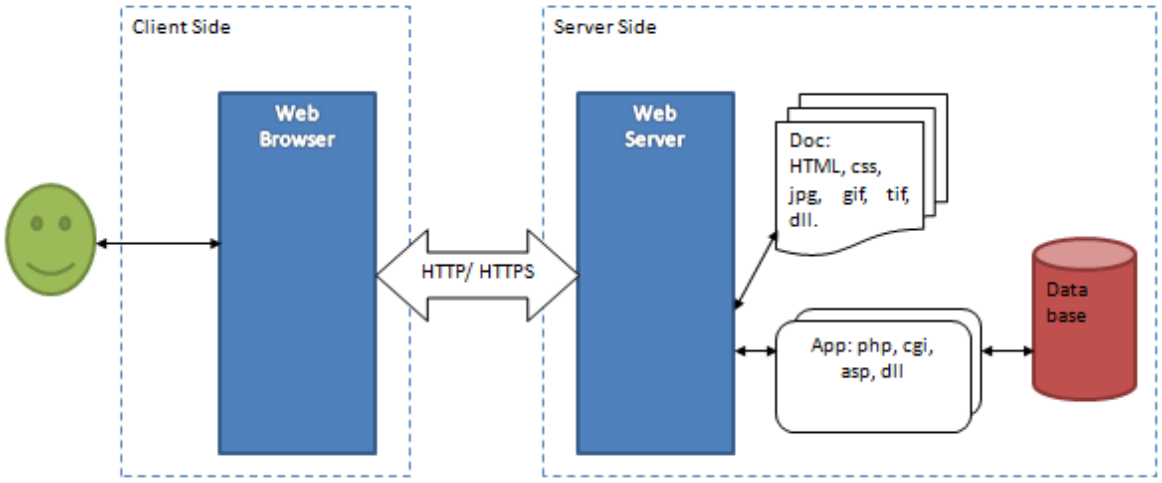
```
37      public void run(){
38          try {
39              SelectionKey acceptKey = _ServerSocketChannel.register(_
40              System.out.println("Acceptor loop...");
41              while (acceptKey.selector().select() > 0) {
42                  Set readyKeys = _Selector.selectedKeys();
43                  Iterator i = readyKeys.iterator();
```

(c) Kode yang dieksekusi Thread pada method run()

Gambar 2.4. Penggunaan Thread pada Java

2.3. Implementasi User Agent HTTP pada Lingkungan Java

Perangkat lunak berbasis web menggunakan protokol *Hyper Text Transfer Protocol (HTTP)* dan *Secure Hyper Text Transfer Protocol (HTTPS)*. Komponen utama dari sistem aplikasi berbasis web dapat dilihat pada Gambar 2.5.



Gambar 2.5. Arsitektur Peranglat Lunak Berbasis Web

Arsitektur komunikasi dasar yang digunakan adalah *client-server*. Di sisi client (*client* side) pengguna dapat mengakses layanan web menggunakan web browser. Di sisi server (*server side)* harus tersedia web server dan aplikasi lain yang dapat diakses melalui web server oleh web browser.

Web browser memiliki kemampuan untuk menginterpretasi dokumen dalam format Hyper Text Markup Language (HTML) berikut kelengkapannya seperti: Cascading Style Sheet (CSS), gambar dengan berbagai format seperti: JPEG, GIF, PNG dan lainya, serta dapat mengeksekusi script tambahan seperti Java Script dan Visual Basic Script (VB Script). Web browser dapat mengirimkan permintaan dalam bentuk Uniform Resources Locator (URL) menggunakan protokol HTTP atau HTTPS (*HTTP request*) ke web server. Selanjutnya web server akan memproses URL ini dengan mengasosiasikannya ke permintaan dokumen/ file (seperti HTML atau gambar). Jika yang diminta tersebut adalah berupa file script yang didukung oleh web server, seperti: php, asp dan Common Gateway Interface (CGI), maka file tersebut akan dieksekusi sesuai dengan mapping eksekutor yang terdaftar di konfigurasi web server. Hasil pemrosesan dalam format HTML disampaikan kembali ke web server untuk diteruskan ke web browser sebagai hasil atau *HTTP respon*.

Protokol HTTP menggunakan clear text dengan operasi GET, POST dan PUT. Pada mayoritas web server, operasi yang diimplementasikan adalah GET dan POST. Pada operasi GET semua parameter dari request ada pada URL. Dengan demikian, perintah keseluruhan terbatas pada panjang maksimal URL yang diijinkan (255 karakter). Pada operasi POST jalur antara perintah dengan data/ parameter dipisahkan. Jalur data dalam bentuk stream, sehingga panjangnya tidak terbatas. Oleh karena itu operasi POST ini cocok untuk pengiriman form dan data dalam bentuk file.

Pada konteks protokol HTTP/HTTPS, web browser disebut sebagai User Agent. Pada lingkungan Java, pengembangan User Agent dapat dilakukan dengan menggunakan library java.net. Class URL dapat digunakan untuk memproses suatu URL input dengan protocol HTTP dan HTTPS. Yang ditangani termasuk pengecekan terhadap format URL yang menghasilkan eksepsi MalformedURLException. Pada Gambar 2.6 dapat dilihat contoh penggunaanya.

```
6       // get url
7       try {
8           URL page = new URL(url); // Process the URL far enough to find the right handler
9           URLConnection urlc = page.openConnection();
10          urlc.setUseCaches(false); // Don't look at possibly cached data
11
12          header = "<br>Content-type = " + urlc.getContentType()
13          + "<br>\n\rContent-length = " + urlc.getContentLength()+"<br>\n\r"; // See how much of
14          // Read it all and print it out
15
16          String buf="";
17          BufferedReader br = new BufferedReader(new InputStreamReader(urlc.getInputStream()));
18          while (buf != null) {
19              try {
20                  // System.out.println(buffer);
21                  buffer = buffer + buf;
22                  buf = br.readLine();
23                  if (buf != null)
24                      length = length + buf.length();
25                  //Main.log(buf);
26              }
27              catch (IOException ioe) {
28                  // ioe.printStackTrace();
29                  buffer = buffer + ioe.getMessage();
30                  break;
31              }
32          }
33      }
34      catch (MalformedURLException mue) {
35              // System.out.println(url + " is not a URL that can be resolved");
36          buffer = buffer + url + " is not a URL that can be resolved";
37      }
```

Gambar 2.6. Contoh Penggunaan class URL

Khusus untuk HTTPS selain pemrosesan URL diperlukan juga penanganan terhadap sertifikat. Penanganan ini dapat dilakukan menggunakan class TrustManager. Dengan class ini juga dapat dilakukan mode Trush dengan menset trushAllCert pada saat inisiasi SSL contex sehingga tidak perlu melakukan pengecekan atau konfirmasi terhadap validitas dari

sertifikat. Hal ini terutama terkait dengan perlunya konfirmasi ke *certificate authority* (CA) pada *public key infrastructure* (PKI). Mekanisme trust ini sangat diperlukan dalam melakukan uji coba, karena situs yang diakses dalam mode uji coba belum didaftarkan pada CA, hanya menggunakan *self signed*. Pada Gambar 2.7 dapat dilihat penggunaan class TrustManager.

```
10      // Create a trust manager that does not validate certificate chains
11      TrustManager[] trustAllCerts = new TrustManager[]{
12          new X509TrustManager() {
13              public java.security.cert.X509Certificate[] getAcceptedIssuers() {
14                  return null;
15              }
16              public void checkClientTrusted(
17                  java.security.cert.X509Certificate[] certs, String authType) {
18              }
19              public void checkServerTrusted(
20                  java.security.cert.X509Certificate[] certs, String authType) {
21              }
22          }
23      };
24
25      // Install the all-trusting trust manager
26      try {
27          SSLContext sc = SSLContext.getInstance("SSL");
28          sc.init(null, trustAllCerts, new java.security.SecureRandom());
29          HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
30      } catch (Exception e) {
31      }
32
33      HttpsURLConnection.setDefaultHostnameVerifier( new HostnameVerifier(){
34          public boolean verify(String string,SSLSession ssls) {
35              return true;
36          }
37      });
```

Gambar 2.7. Contoh Penggunaan Trush Manager

BAB 3 ANALISIS KEBUTUHAN DAN DISAIN SOLUSI

Pada bagian ini dijelaskan tentang analisis kebutuhan perangkat lunak uji performansi dan kapasitas sistem berbasis web. Pada bagian ini juga dijelaskan bagaimana disain dan implementasi dari perangkat lunak tersebut pada lingkungan Java.

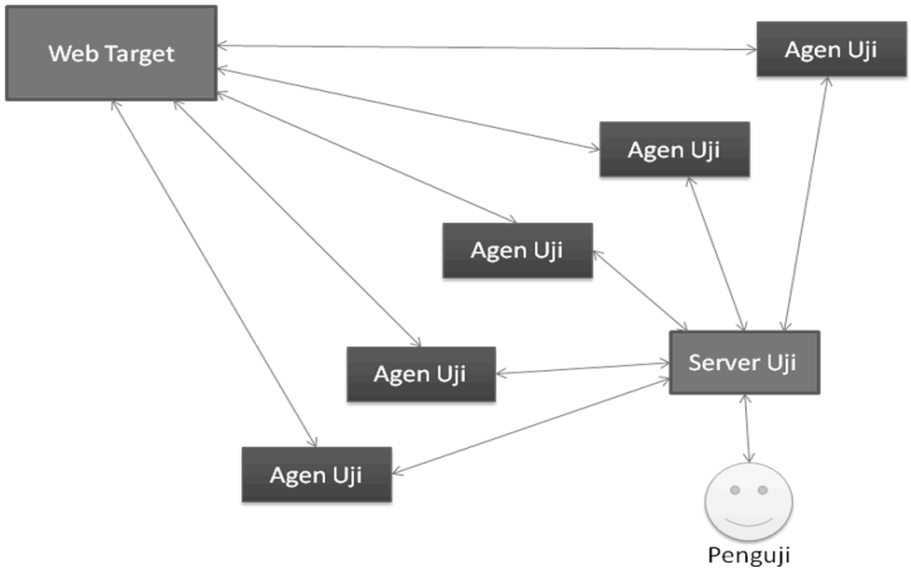
3.1. Analisis Kebutuhan

Berdasarkan latar belakang, rumusan masalah dan tujuan dapat didefinisikan bahwa yang dibutuhkan untuk melakukan uji coba performansi dan kapasitas berbasis web adalah sebuah perangkat lunak yang dapat memenuhi kriteria berikut:

1. Menerima data uji berupa URL.
2. Menerima masukan banyaknya emulasi User Agent (UA).
3. Menerima frequensi dari request URL dari setiap UA.
4. Mengirimkan request sesuai dengan URL, banyak UA secara simultan sebanyak frequensi masukan ke web server sasaran.
5. Menerima respon dari web server atas request yang disampaikan, baik respon berhasil maupun tidak berhasil.
6. Mengukur selisih waktu dari request sampai mendapat respon.
7. Menghasilkan data statistik rata-rata waktu akses dan kapasitas (UA yang dapat dilayani).

3.2. Gambaran Solusi

Berdasarkan definisi kebutuhan tersebut dapat diusulkan solusi dengan arsitektur logika yang dapat dilihat pada gambar 3.1.



Gambar 3.1. Arsitektur Logika Uji Kapasistas dan Performansi

Web Target (WT) merupakan web server yang menjadi host dari situs web yang akan diuji. Agen Uji (AG) merupakan emulasi dari 1 User Agent (UA), dengan demikian akan ada banyak AG untuk menghasilkan trafik request ke situs web yang diuji sesuai dengan persyaratan kapasitas (koneksi konkuren). Semua AG mendapat perintah uji berupa URL dan frekuensi uji untuk menghasilkan trafik secara simultan oleh Server Uji (SU). SU akan memberikan instruksi secara bersama-sama, sehingga semua AG juga melakukan proses request ke WT secara bersama-sama juga. Penguji (tester) dapat memasukkan perintah ke SU. Perintah yang dimasukkan berupa URL situs web yang akan diuji, banyaknya UA dan frekuensi request.

Bagaimana mengetahui indikator-indikator hasil uji? Sesuai dengan tujuan pada Bab 1 bahwa indikator hasil yang harus dihasilkan adalah:

1. Halaman web hasil uji, yang menyatakan apakah *request* berhasil atau tidak.
2. Indikator performa dalam satuan *mili second* (ms)
3. Indikator kapasitas dalam satuan koneksi konkuren.

Halaman web hasil uji, dapat dihasilkan oleh AU. Pada AU dapat dilihat file dengan format HTML yang berisi respon dari situs web yang diuji. Jika responnya adalah error, maka halaman error tersebut juga harus terbangkitkan.

Indikator performa, berupa selisih waktu antara request dan response dicatat oleh AU untuk setiap siklus request-response. Informasi ini dikumpulkan, kemudian dikirimkan ke SU setelah semua kewajiban AU (perintah dari SU) dijalankan. Dengan demikian diakhir uji, pada SU sudah terkumpul data statistik hasil uji dari semua AU. Data ini selanjutnya diolah menjadi statistik hasil uji performa.

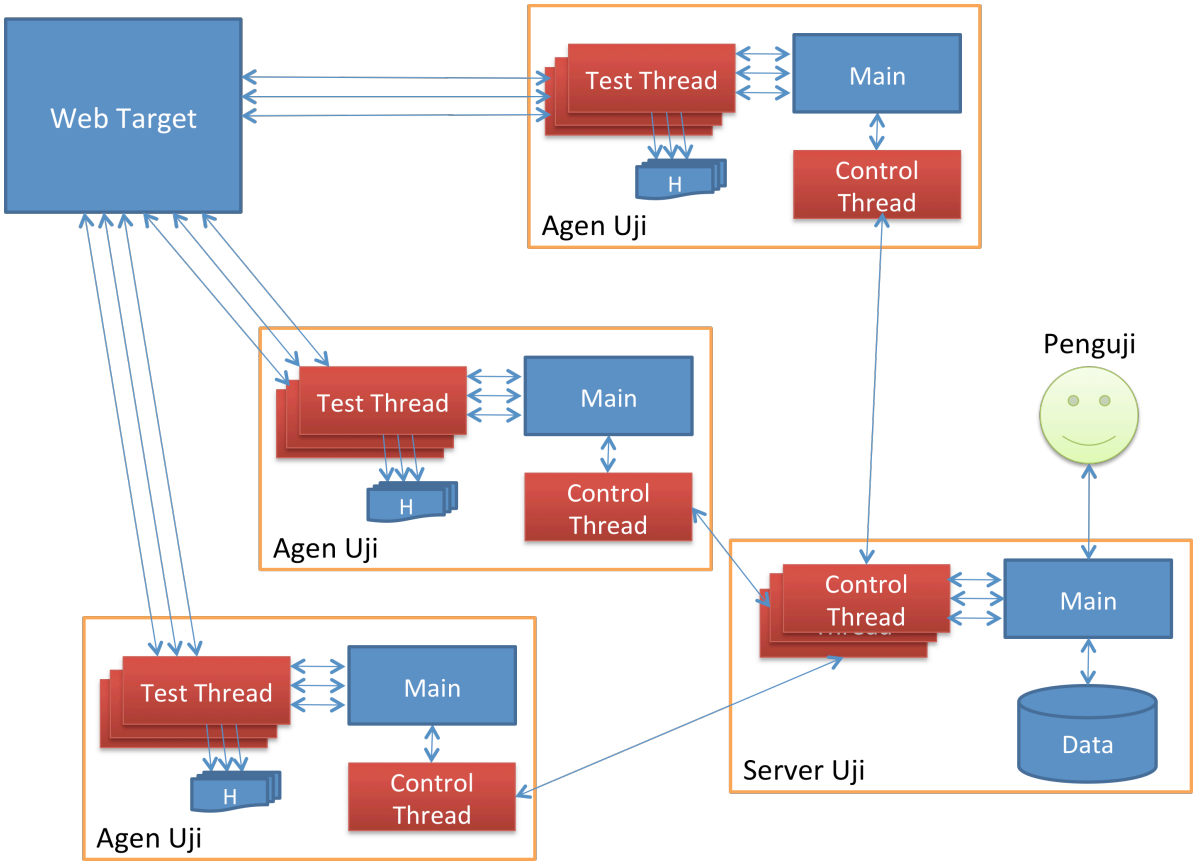
Indikator kapasitas, dapat dipantau melalui 2 sisi. Dari sisi AU dapat dilihat dari banyaknya AU yang mendapat response sukses pada suatu waktu (pada waktu yang sama). Sisi lain adalah dari WT sendiri. Pada WT dapat dipasang monitor terhadap koneksi konkuren yang sedang terjadi yang berasal dari lokasi AU. Pada sisi WT juga dimungkinkan mendapatkan informasi tersebut dari log web server (jika ada).

3.3. Disain Solusi

Gambara solusi pada bagian 3.2 diimplementasikan dengan menggunakan teknologi multi threading. Aplikasi yang dikembangkan ada 2, yaitu: SU dan AU. Disain lebih lanjut dapat dilihat pada Gambar 3.2.

Pada Gambar 2, penguji mengirimkan perintah kepada Server Uji (SU). Perintah ini disimpan pada basis data. SU adalah aplikasi server yang listen pada port tententu, misalnya port 5000. Agent Uji (AU) menghubungi SU menggunakan Thread Control (TC). Untuk setiap TC di AU ditangani oleh 1 TC di SU, dengan demikian koneksi dapat bersifat *dedicated*. AU secara periodik mengirmkan pesan permintaan perintah kepada SU. SU memberikan perintah melalui TC berupa URL yang diuji, banyaknya UA yang harus dibangkitkan dan frekuensi request. Setelah AU menerima perintah maka akan dieksekusi dengan membangkitkan Test Thread (TT) sebanyak UA yang diminta. Dengan demikian 1 UA diemulasi dengan 1 TT. Misalnya untuk menghasilkan trafik request yang mengemulasi 1000 UA dengan

menggunakan 10 AU selama 1 dengan frekuensi 10x, SU akan memberikan perintah URL + 100 UA + 10x.



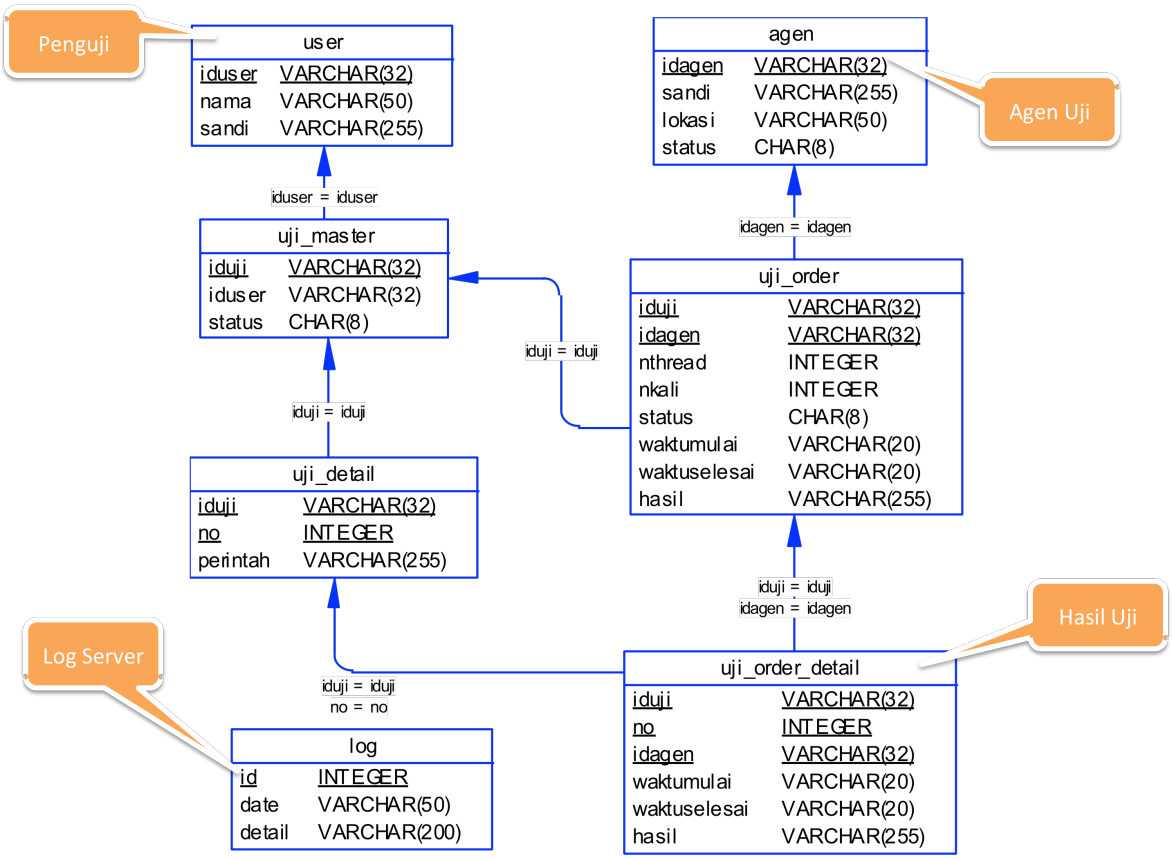
Gambar 3.2. Disain PL Uji

Banyaknya AU yang dapat dijalankan pada satu komputer sangat tergantung pada spesifikasi komputer yang digunakan. Demikian juga dengan banyaknya TT (emulasi UA) yang dapat dibangkitkan juga tergantung pada spesifikasi komputer, terutama terkait dengan *processor*, memori dan *network card* yang digunakan.

3.4. Disain Detail

3.4.1. Rancangan Basis Data di SU

Rancangan basis data pada SU dapat dilihat pada gambar



Gambar 3.3 Rancangan Basis Data Server Uji

Berikut adalah penjelasan dari rancangan basis data pada Gambar 3.3.

Nama: dbServerUji

Tabel:

1. user (iduser, nama, sandi)

   Contoh: (“gede”,”Gede Karya”,password(“gede!123”))

2. agen(idagen, sandi, lokasi, status)

   Contoh: (“fisip01”,password(“fisip!01”),”10.100.79.1”,”aktif”)

   Status = “aktif” | “nonaktif”

3. uji_master(iduji, iduser, status)

   contoh: (“ujibeban01”,”gede”,”aktif”)

status = “aktif” | “nonaktif”

4. uji_detail(iduji, no, perintah)

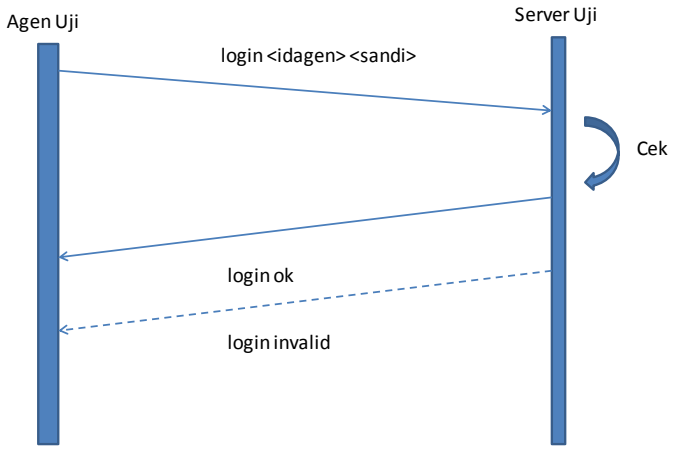
   contoh: (“ujibeban01”,1,”http://www.unpar.ac.id”)

5. uji_order(iduji, idagen, nthread, nkali, status, waktumulai, waktuselesai, hasil)
   Contoh:
   (“ujibeban01”,”fisip01”,1,1,”order”,”2011-03-08 00:00:00”,null,null)
   (“ujibeban01”,”fisip01”,1,1,”eksekusi”,”2011-03-08 00:00:00”,null,null)
   (“ujibeban01”,”fisip01”,1,1,”selesai”,”2011-03-08 00:00:00”,”2011-03-08 10:00:01”,”25000”)
   Status = “order” | “eksekusi” | “selesai”
   Nthread = banyaknya thread uji
   Nkali= banyaknya transaksi simultan (sequensial), perintah akan dijalankan sebanyak berapa kali.
6. Uji_order_detail (iduji, no,idagen, waktumulai, waktuselesai, hasil)
   Contoh:
   (“ujibeban01”,1,”fisip01”,”2011-03-08 00:00:00”,null, null)
   (“ujibeban01”,1,”fisip01”,”2011-03-08 00:00:00”,”2011-03-08 01:01:01” ,”2500”)

3.4.2. Rancangan Protokol Komunikasi SU – AU.

Rancangan protokol dapat dilihat pada gambar 3.4.a sampai 3.4.c.

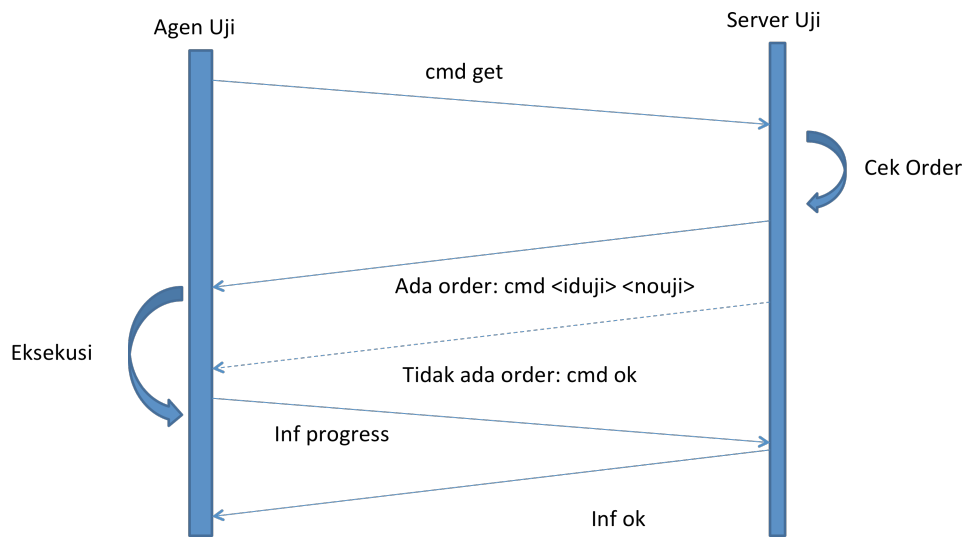
Otentifikasi



Gambar 3.4.a. Rancangan protokol pada fase Otentifikasi

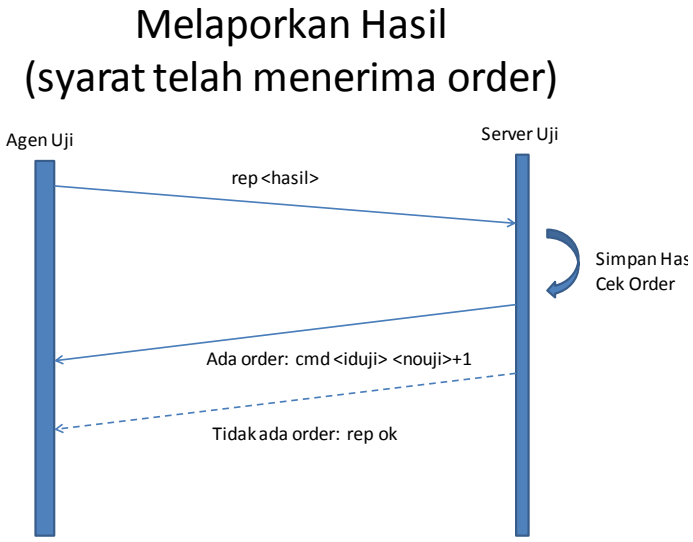
Ada 3 status aplikasi, yaitu: otentifikasi, meminta perintah dan melaporkan hasil. Berikut adalah penjelasan proses/ aksi masing-masing status:

1. Otentifikasi (Gambar 3.4.a)
   a. Semua data di atas diinisiasi melalui aplikasi berbasis web “Controler” ke basis data “dbServerUji”, dengan otorisasi menggunakan login/password pada tabel “user”.
   b. Aplikasi “Server Uji” membaca basis data “dbServerUji”.
   c. Aplikasi “Agen Uji” menghubungi “Server Uji” melalui jaringan TCP/IP sesuai dengan parameter eksekusi. Setelah terhubung, wajib melakukan otentifikasi dengan mengirimkan pasangan “login idagen sandi”. Setelah itu “Server Uji” memvalidasi berdasarkan data di tabel “agen”. Jika valid, maka “Server Uji” akan mengisi field “lokasi” dengan alamat IP dari agen dan menset “status” = “aktif”, kemudian memberikan pesan “login ok”.
   d. “Agen uji” akan memasuki siklus “Meminta perintah” dengan mengirimkan pesan “cmd get”.



Gambar 3.4.b Fase Meminta Perintah

2. Meminta perintah (Gambar 3.4.b)
   a. Setelah “Server Uji” mendapat pesan “cmd get”, akan mengecek ke tabel “uji_order”, apakah ada order yang waktu >= waktu sekarang dan status=”order”.
   b. Jika ya, maka akan dibalas dengan pesan “cmd 1 <perintah ke 1>”
   c. Jika tidak ada perintah yang berstatus “order” maka akan menjawabnya dengan pesan “cmd ok”.



Gambar 3.4.c. Fase Melaporkan Hasil

3. Melaporkan hasil (Gambar 3.4.c)
    a. Setelah “Agen Uji” menyelesaikan suatu perintah, maka akan melaporkannya dengan mengirim pesan “rep <no uji> <waktu dalam mili second>”.
    b. “Server Uji” akan menyimpan hasilnya di tabel “uji_order_detail”, kemudian akan mengecek ke tabel “uji_detail”, apakah ada perintah dengan no = <no uji> + 1.
    c. Jika ya, maka akan dikirim pesan “cmd <no> <perintah ke no>”.
    d. Jika tidak, maka akan mengirip pesan “rep ok”, kemudian “Server Uji” memasukkanya ke tabel “uji_order” pada field “waktuselesai” dan “hasil”. Dimana hasil diakumulasi dari isi tabel “uji_order_detail” dengan iduji dan idagen yang sama.
    e. Setelah menerima “rep ok”, maka “Agen Uji” akan masuk ke siklus “meminta perintah” dengan mengirimkan pesan “cmd get” lagi.

Demikianlah hasil analisis dan rancangan dari perangkat lunak uji kapasitas dan performansi.

BAB 4 Hasil Implementasi dan Pengujian

Berikut adalah hasil implementasi basis data, program dan pengujian yang telah dilakukan.

4.1. Implementasi Basis Data

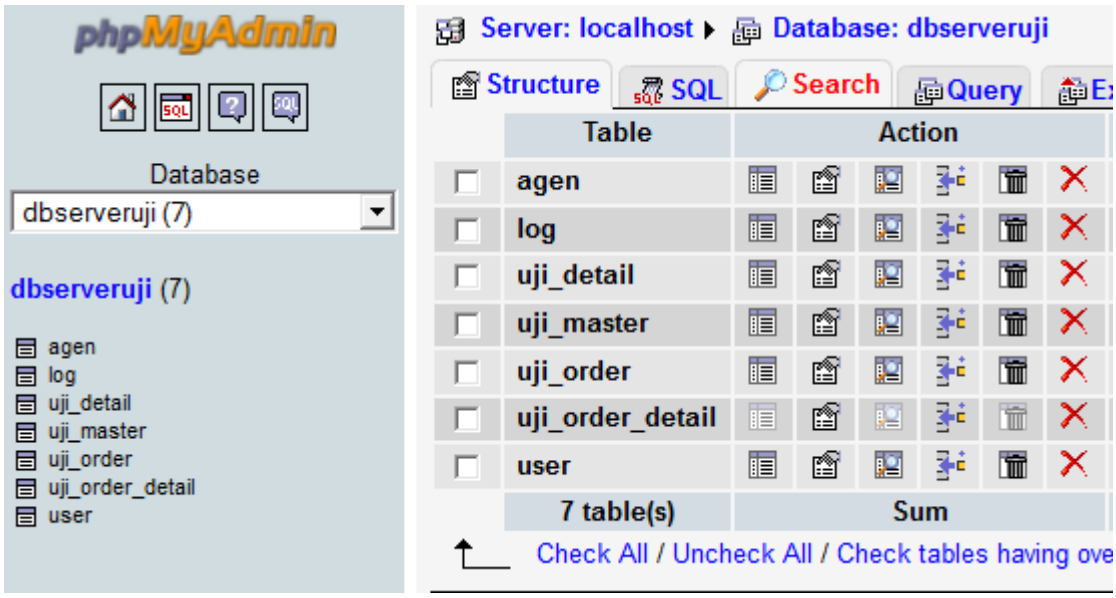
Dari rangangan pada bab 3, diimplementasikan pada lingkungan MySQL versi 5.1. Berikut adalah *data definition language* (DDL) dari rancangan basis data pada Gambar 3.3:

```
Create table user (iduser char(32) not null, nama varchar(50), sandi varchar(255),
        primary key(iduser));
Create table agen (idagen char(32) not null, sandi varchar(255), lokasi varchar(50), status
varchar(10),
        Primary key (idagen));
Create table uji_master (iduji char(32) not null, iduser char(32) not null, status varchar(10),
        Primary key(iduji), foreign key(iduser) references user(iduser));
Create table uji_detail(iduji  char(32) not null, no int not null, perintah varchar(255),
        Primary key (iduji, no));
Create table uji_order(iduji char(32) not null, idagen char(32) not null, nthread int, nkali int,
        status varchar(10),  waktumulai datetime, waktuselesai datetime, hasil varchar(255),
        Primary key (iduji, idagen)); // foreign key tolong ditambahkan sesuai nama
Create table uji_order_detail(iduji char(32) not null, no int not null, idagen char(32) not null,
        nthread int, nkali int,  status varchar(10),  waktumulai datetime,
        waktuselesai datetime, hasil varchar(255),
```

Primary key (iduji, no, idagen));

7. Uji_order_detail (iduji, no,idagen, waktumulai, waktuselesai, hasil)

Pada gambar 4.1 ditampilkan visualisasi implementasi basis data dengan menggunakan PHPMyAdmin.



Gambar 4.1. Visualisasi Hasil Implementasi Basis data

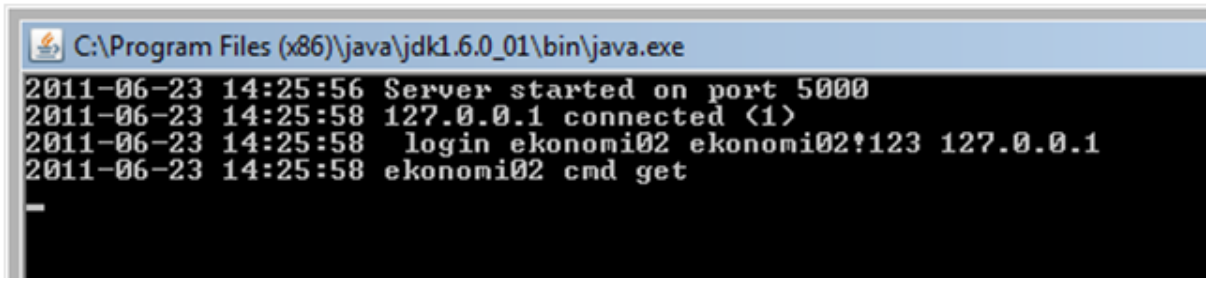
4.3. Implementasi Program Aplikasi

Implementasi rancangan pada bab 3 telah dilakukan. Hasilnya berupa 2 program yaitu:

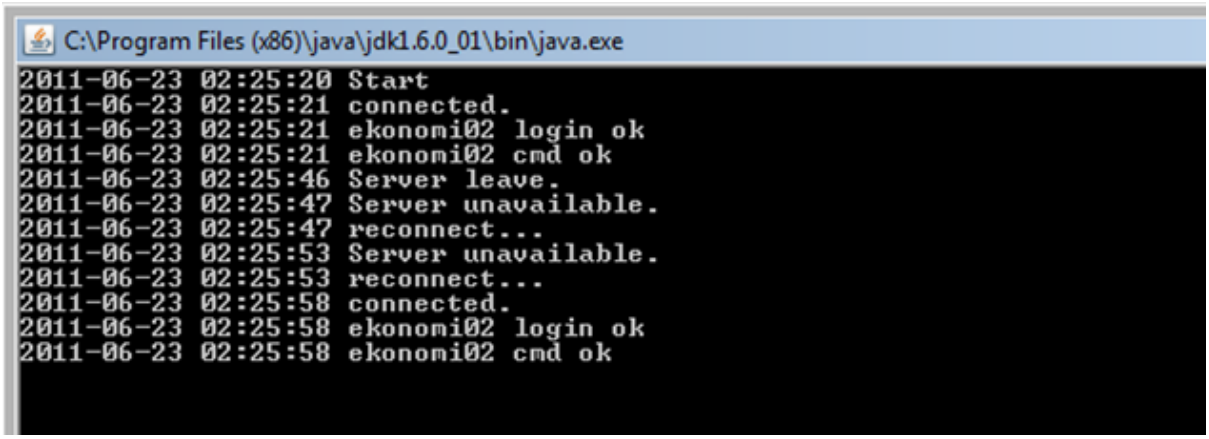
1. WebTest sebagai hasil implementasi dari Agent Uji (AU).

2. WebTestServer sebagai hasil implementasi dari Server Uji (SU).

Baik AU dan SU diimplementasikan dengan bahasa pemrogramn Java. Pada gambar 4.2 dapat dilihat contoh program server (WebTestServer) dan 4.3 dapat dilihat contoh program agen uji (WebTest) .



Gambar 4.2. Hasil Running dari WebTestServer sebagai Implementasi SU



Gambar 4.3. Hasil Running dari WebTest sebagai implementasi AU

4.3. Pengujian

Pengujian telah dilakukan dengan berbagai konfigurasi, yaitu:

1. Menggunakan 1 komputer yang terinstall dengan 1 – 10 agen uji.
2. Menggunakan 18 komputer, dengan masing-masing ada 1-5 agen uji.
3. Menggunakan sistem operasi Windows dan Linux dengan menggunakan virtualisasi VirtualBox.

Kasus yang diuji juga ada beberapa yaitu:

1. Kasus 1 perintah sederhana berupa request URL saja.
2. Kasus 3 perintah berurutan.

3. Kasus terjadwal, dimana perintah dieksekusi pada jam tertentu.

Pada gambar 4.4 dapat dilihat contoh perintah yang digunakan dalam pengujian yang diperlihatkan dari perintah yang tersimpan di basis data MySQL.

	iduji	no	perintah
	ujibeban01_local	1	http://localhost/xampp/
	ujibeban02_unpar	1	http://www.unpar.ac.id/main.php?sub=02705&sub_cont...
	ujibeban02_unpar	2	http://www.unpar.ac.id/main.php?sub=02705&sub_cont...
	ujibeban02_unpar	3	http://www.unpar.ac.id/main.php?sub=0270506&sub_co...
	ujibeban03_detik	1	http://www.detik.com/
	ujibeban03_detik	2	http://www.detiknews.com/
	ujibeban03_detik	3	http://www.detiknews.com/read/2011/06/24/084627/16...
	ujibeban03_detik	4	http://www.detiknews.com/read/2011/06/24/084120/16...

Gambar 4.4 Contoh Perintah yang digunakan dalam Pengujian

Pada gambar 4.5 dapat dilihat agen-agen uji yang digunakan dalam pengujian pada 1 komputer.

	idagen	sandi	lokasi	status
	agenuji01	agenuji01!123	127.0.0.1	nonaktif
	agenuji02	agenuji02!123	127.0.0.1	nonaktif
	agenuji03	agenuji03!123	127.0.0.1	nonaktif
	agenuji04	agenuji04!123	127.0.0.1	nonaktif
	agenuji05	agenuji05!123	*NULL*	nonaktif
	agenuji06	agenuji06!123	*NULL*	nonaktif
	agenuji07	agenuji07!123	*NULL*	nonaktif
	agenuji08	agenuji08!123	*NULL*	nonaktif
	agenuji09	agenuji09!123	*NULL*	nonaktif
	agenuji10	agenuji10!123	*NULL*	nonaktif

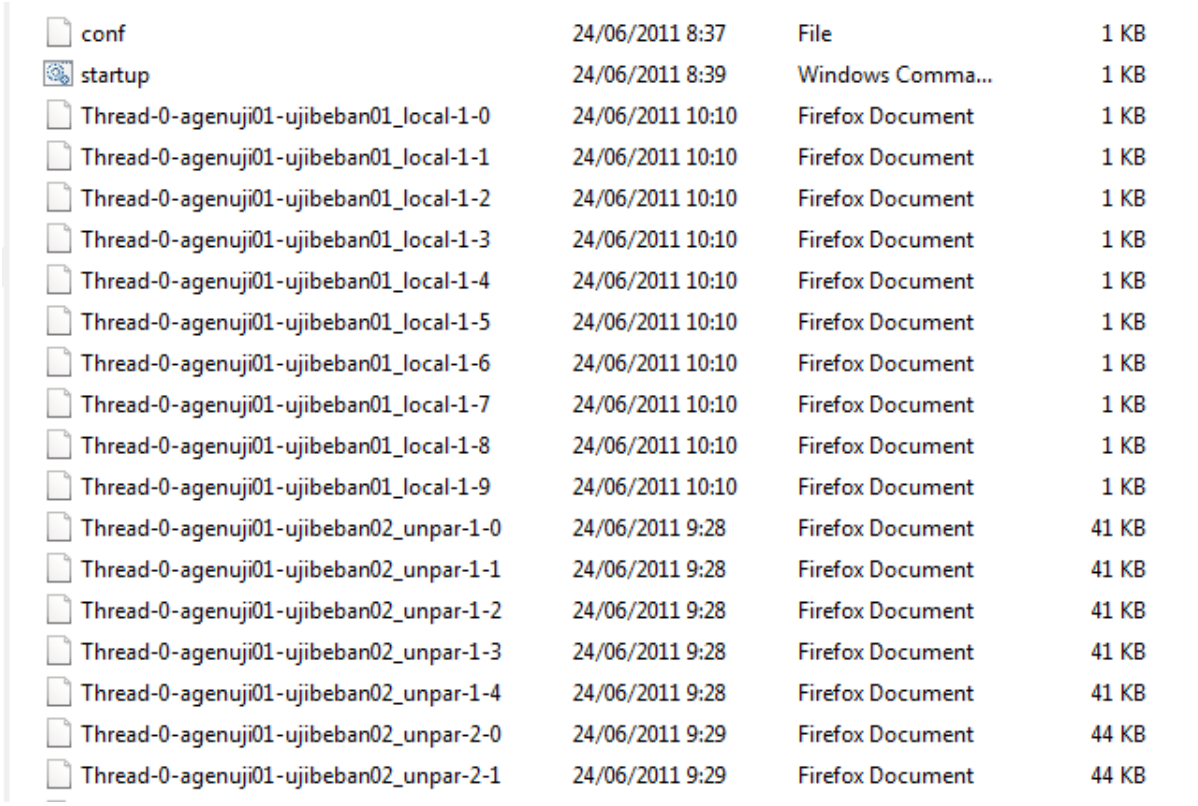
Gambar 4.5. Contoh Agen Uji yang digunakan dalam Pengujian

Sedangkan pada gambar 4.6 dapat dilihat hasil pengujian.

			iduji	idagen	nthread	nkali	status	waktumulai	waktuselesai	hasil
☐	✎	✕	ujibeban01_local	agenuji01	10	10	selesai	2011-06-24 10:10:04	2011-06-24 10:10:14	34
☐	✎	✕	ujibeban02_unpar	agenuji01	10	5	selesai	2011-06-24 09:29:38	2011-06-24 09:30:18	17776
☐	✎	✕	ujibeban03_detik	agenuji01	50	2	selesai	2011-06-24 09:42:53	2011-06-24 09:45:18	92810
☐	✎	✕	ujibeban03_detik	agenuji02	50	2	selesai	2011-06-24 09:40:46	2011-06-24 09:44:31	80204
☐	✎	✕	ujibeban03_detik	agenuji03	5	10	eksekusi	2011-06-23	*NULL*	*NULL*
☐	✎	✕	ujibeban03_detik	agenuji04	10	10	selesai	2011-06-24 09:39:21	2011-06-24 09:42:46	78692

Gambar 4.6 Hasil pengujian untuk beberapa Kasus Uji

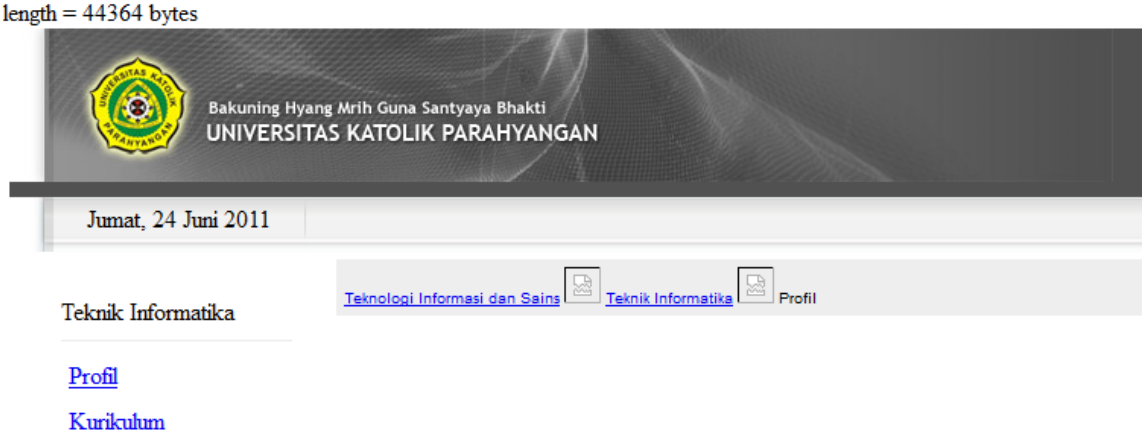
Pada gambar 4.7 ditampilkan isi folder dari salah satu Agen Uji, setelah melakukan proses pengujian.



Gambar 4.7. Contoh Folder hasil Uji

Sedangkan pada gambar 4.8 dan 4.9 dapat dilihat contoh dari halaman web hasil pengujian. Pada bagian atas terdapat ringkasan hasil berupa waktu (performansi) dan ukuran file hasil.

Thread-0 3 Https.get(http://www.unpar.ac.id/main.php?sub=02705&sub_content=0270505)(2845 ms)
Content-type = text/html
Content-length = -1
Actual length = 44364 bytes



Gambar 4.8. Contoh Halaman Hasil Uji dari suatu Agen Uji

Thread-0 9 Https.get(http://www.detiknews.com/read/2011/06/24/084120/1667550/10/peraturan-pemerintah-soal-erp-diteken-sby?n991102605)(7821 ms)
Content-type = text/html
Content-length = -1
Actual length = 10792 bytes

Gambar 4.9. Contoh Halaman Hasil Uji oleh Agen Uji

Pada gambar 4.10 ditampilkan log dari SU yang dapat menggambarkan proses uji dilakukan oleh masing-masing AU.

			id	date	detail
☐	✎	✕	1	2011-06-24 09:28:03	Server started on port 5000
☐	✎	✕	2	2011-06-24 09:28:05	127.0.0.1 connected (1)
☐	✎	✕	3	2011-06-24 09:28:05	login agenuji04 agenuji04!123 127.0.0.1
☐	✎	✕	4	2011-06-24 09:28:05	agenuji04 cmd get
☐	✎	✕	5	2011-06-24 09:28:05	127.0.0.1 connected (2)
☐	✎	✕	6	2011-06-24 09:28:05	login agenuji02 agenuji02!123 127.0.0.1
☐	✎	✕	7	2011-06-24 09:28:05	agenuji02 cmd get
☐	✎	✕	8	2011-06-24 09:28:08	127.0.0.1 connected (3)
☐	✎	✕	9	2011-06-24 09:28:08	login agenuji01 agenuji01!123 127.0.0.1
☐	✎	✕	10	2011-06-24 09:28:08	agenuji01 cmd get
☐	✎	✕	11	2011-06-24 09:28:08	127.0.0.1 connected (4)
☐	✎	✕	12	2011-06-24 09:28:08	login agenuji03 agenuji03!123 127.0.0.1
☐	✎	✕	13	2011-06-24 09:28:08	agenuji03 cmd get
☐	✎	✕	14	2011-06-24 09:28:10	agenuji04 inf progress
☐	✎	✕	15	2011-06-24 09:28:10	agenuji02 inf progress
☐	✎	✕	16	2011-06-24 09:28:13	agenuji01 inf progress
☐	✎	✕	17	2011-06-24 09:28:13	agenuji01 rep 48 ms for http://localhost/xampp/(10...
☐	✎	✕	18	2011-06-24 09:28:14	agenuji03 inf progress

Gambar 4.10 Contoh Log Hasil Uji

Dengan demikian semua implementasi sudah dilakukan dan berhasil dengan baik. Perangkat lunak ini juga digunakan untuk melakukan uji beban pada sistem aplikasi StudentPortal pada SI Akademik Unpar dalam rangka mempersiapkan pendaftaran kuliah mahasiswa (FRS).

BAB 5 KESIMPULAN DAN POTENSI PENGEMBANGAN

Pada bagian ini dijelaskan tentang kesimpulan yang diambil berdasarkan metodologi, kajian pustaka, dan hasil-hasil penelitian. Bagian ini diakhiri dengan potensi pengembangan yang dapat dilakukan untuk penelitian selanjutnya.

5.1. Kesimpulan

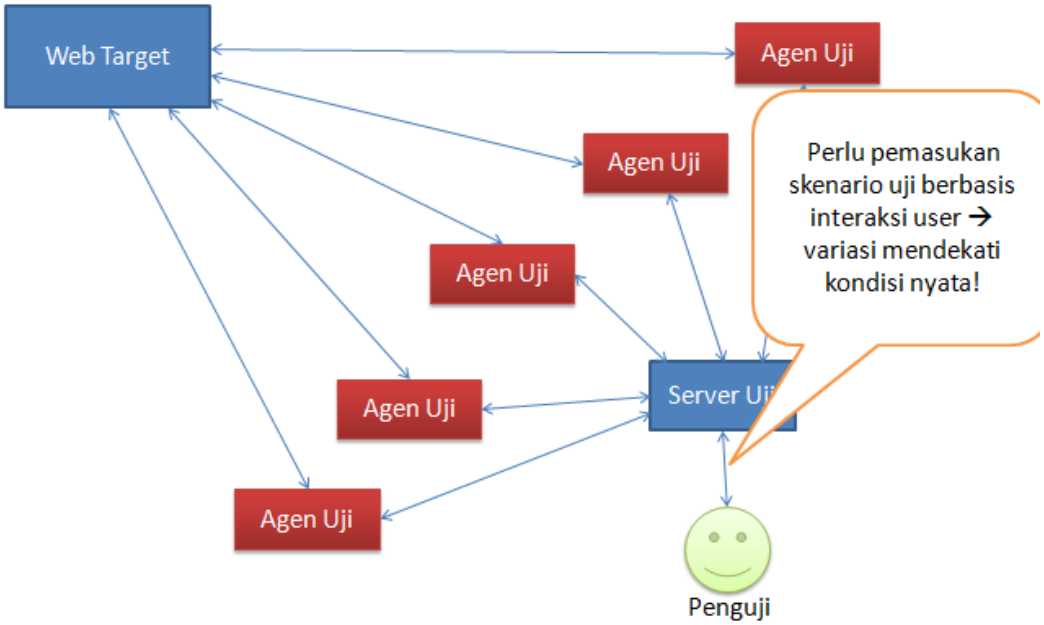
Berdasarkan hasil penelitian di atas, dapat disimpulkan:

1. Perangkat lunak uji yang dikembangkan sudah dapat berfungsi dengan baik, mencakup fitur: pendaftaran perintah uji terjadwal oleh penguji, eksekusi multi agent dan Pelaporan kinerja. Di dalamnya sudah termasuk otomasi dan multi agen.
2. Berdasarkan hasil pengujian, kapasitas *User Agent* yang bisa diemulasi tergantung pada spesifikasi komputer (prosessor, memori, *network card*).

5.2. Potensi Pengembangan

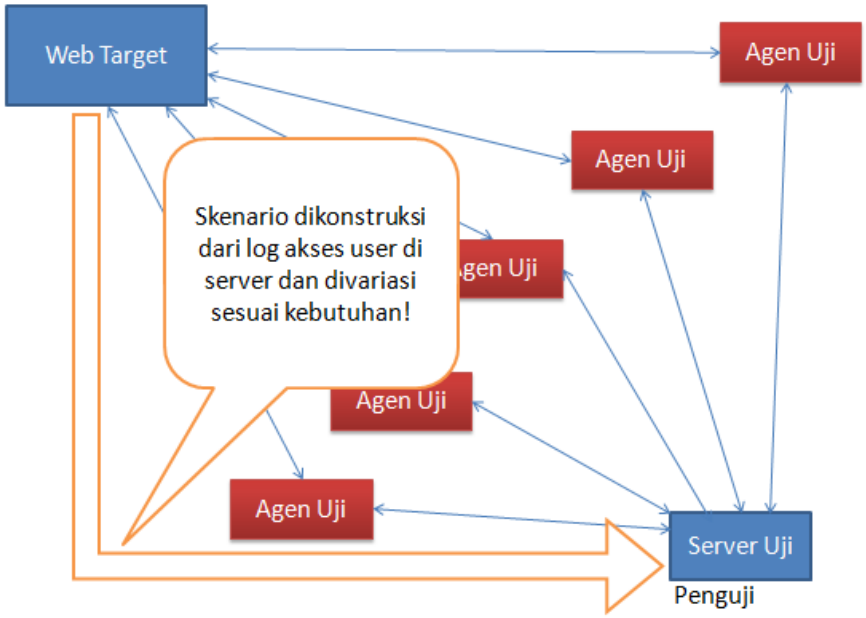
Dari hasil penelitian yang telah dilakukan, ada beberapa hal yang perlu dikembangkan yaitu:

1. Mendukung operasi POST. Hal ini penting, mengingat pada realitanya operasi yang digunakan oleh user mencakup GET dan POST. Hanya saja, operasi ini kemungkinan lebih sulit untuk diimplementasikan, karena tidak hanya berupa URL, akan tetapi juga perlu menggali dan mengemulasi jalur data yang digunakan.
2. Fungsi konfigurasi dan pelaporan yang *user friendly*. Saat ini semua konfigurasi dan pelaporan dapat dilihat di basis data menggunakan tools PHPMyAdmin. Hal ini cukup menyulitkan dalam operasional, oleh karena itu kedepan disarankan untuk dikembangkan sehingga lebih *user friendly.*
3. Pemasukan skenario mendekatan interaksi user sebenarnya. Hal ini kemungkinan dapat dilakukan dengan 2 pendekatan seperti pada Gambar 5.1 dan 5.2.



Gambar 5.1. Perekaman Data Uji melalui Pemasukan Data

Pada gambar 5.1 data uji direkam dari penguji, kemudian diberi variasi dan diduplikasi sesuai dengan kebutuhan.



Gambar 5.2. Data Uji dibangkitkan berdasarkan log dari Server yang dijadikan target

Dengan model pada Gambar 5.2 dimungkinkan untuk membangkitkan data uji berdasarkan log akses user yang ada di Server Target. Tentunya untuk kasus operasi GET dapat diperolah dari URL nya sedangkan untuk POST perlu dilihat lebih lanjut.

DAFTAR PUSTAKA

1. David G. Messerschmitt. *Understanding Networked Applications: A First Course.* Morgan Kaufmann. 2000
2. George Coulouris, Jean Dollimore, Tim Kindberg. *Distributed Systems Concepts and Design.* Addison Wesley. 2001.
3. Bruce Eckel, *Thinking in Java*, Second Edtion, Printice-Hall, 2000.
4. Chaffee, Alexander Day, *"Building a Robust Multithreaded Server in Java"*, Jguru Java Training, 1998