BAB IV
PENGUJIAN DAN ANALISA

Pada bab ini akan dilakukan pengujian dan analisa terhadap kernel linux pada sistem LTSP yang dibangun, hasil pengujian akan dibandingkan dengan kernel linux *default* dengan kernel linux 2.6.32-22 dan 2.6.33.

4.1 Metode Pengujian dan Pengambilan Data

Pengujian dari sistem yang dirancang yaitu untuk mendapatkan tingkat kestabilan kernel linux dengan sistem LTSP di sisi *client.* Dalam hal ini untuk mengetahui apakah proses booting dan proses menjalankan program aplikasi dapat dengan cepat dan stabil.

Komputer yang digunakan dalam pengujian adalah 4 unit, 1 komputer digunakan untuk *server* dan 3 komputer lainya digunakan untuk *client* dengan spesifikasi komputer yang sama. Pada saat proses *benchmarking* akan diambil *sample* data, data yang diambil adalah 10 kali proses booting dan proses menjalankan program aplikasi secara bersamaan pada *client*, karena tidak ada *utilitas benckmarking* yang memungkinkan untuk menggambil data secara bersamaan maka proses *benckmarking* dilakukan dengan cara manual dengan menggunakan *stopwatch* sebagai alat bantu pengambilan data.

4.2 Skenario Pengujian

Pengujian yang akan dilakukan adalah dengan melihat perbandingan proses booting dan proses menjalankan program aplikasi pada kernel linux *default* dan pada kernel linux versi 2.6.32-22, 2.6.33 yang telah di optimalisasi pada semua *client* secara bersamaan, program aplikasi yang akan digunakan dalam pengambilan data adalah :

1. Open Office 3.1 dan
2. Web Browser Firefox 3.5.3

Batasan ini dilakukan karena aplikasi-aplikasi tersebut adalah aplikasi yang dirancang untuk digunakan oleh *client* dan selain itu juga menuntut penggunaan sumber daya yang cukup besar, sehingga layak untuk dijadikan aplikasi untuk pengujian.

4.3 Persiapan Pengujian

Sebelum melakukan pengujian terhadap *client* ada beberapa tahapan yang harus diperiksa dalam tugas akhir ini yaitu :

1. Memeriksa kernel linux.
2. Mengaktifkan LTSP.
3. Memeriksa *service* LTSP.
4. Memeriksa *service-service* DHCP *server*.

4.3.1 Memeriksa Kernel Linux

Sebelum melakukan pengujian lakukan pengecekan apakah kernel linux yang akan di uji coba sudah terinstal pada *server*. Untuk memeriksa kernel linux pada *server* lakukan perintah sebagi berikut :

```
~# dpkg --get-selections | grep ‘2.6’
```

Opsi tersebut merupakan untuk pengecekan kernel linux versi 2.6.xx. Jika keluaran nya seperti dibawah ini maka sudah terinstal :

```
libedata-cal1.2-6                                    install
libpython2.6                                         install
linux-header-2.6.31-14                               install
linux-header-2.6.31-14-generic                       install
linux-header-2.6.32-22                               install
linux-header-2.6.32-22-generic                       install
linux-header-2.6.33-rendra-custom-2                  install
linux-image-2.6.31-14-generic                        install
```

```
linux-image-2.6.32-22-generic                    install
linux-image-2.6.33-rendra-custom-2               install
python2.6                                        install
python2.6-minimal                                install
```

4.3.2 Mengaktifkan LTSP

Untuk mengaktifkan LTSP di sisi *server* dengan cara menyalakan komputer *server*, karena perancangan pada sistem LTSP menggunakan pengalamatan IP *address* DHCP (*Dynamic Host Configuration Protocol*) maka ketika komputer *client* dinyalakan akan mendapatkan IP *address* secara otomatis.

4.3.3 Memeriksa service LTSP

Dalam memeriksa *service-service* LTSP yang akan dijalankan dalam sistem tersebut dapat berjalan dengan baik. *Service* yang paling utama adalah *service* DHCP *server.*

4.3.4 Memeriksa service DHCP Server

Sebelum pengujian dilakukan periksa terlebih dahulu *service* DHCP *server* yang digunakan, untuk mengetahui *service* tersebut sudah berfungsi dapat dilihat dengan menggunakan perintah :

```
~# tail –f /var/log/messages
```

Jika sudah, dengan demikian semua *client* yang terhubung dengan *server* akan mendapatkan IP *address* secara dinamis.

4.4 Pengujian dan Pengambilan Data

Setelah semua sistem dipersiapkan dan berfungsi secara baik, selajutnya adalah proses pengujian dan pengambilan data dengan metode dan skenario yang telah ditetapkan, berdasarkan kernel linux yang digunakan.

4.4.1 Pengujian Proses Booting

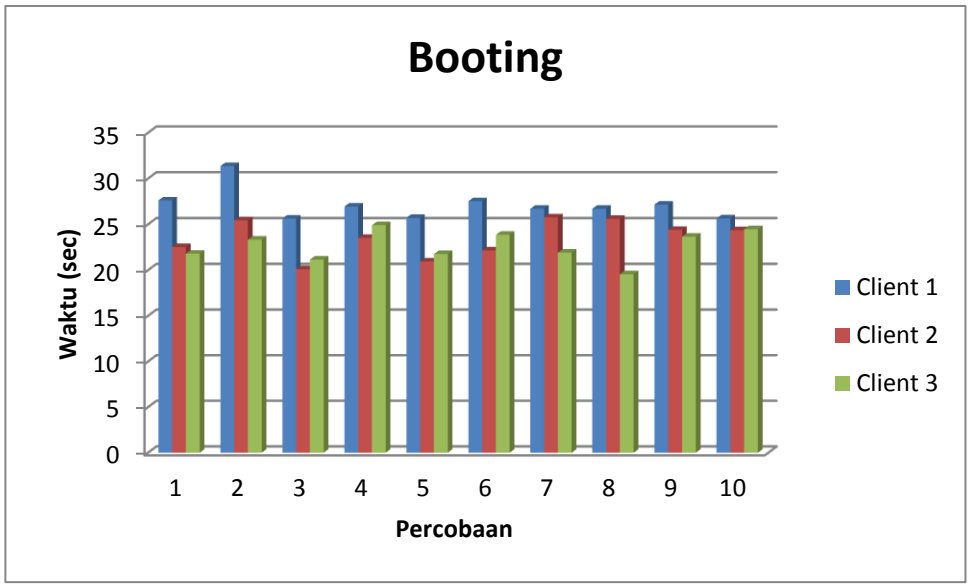
Distribusi Ubuntu Desktop 9.10 yang menggunakan kernel linux *default* merupakan kernel yang tidak mengalami modifikasi, tabel dibawah menunjukkan proses booting pada keadaan tiga *client* secara bersamaan, dalam satuan detik.

Tabel 4.1 Tabel *Benchmark* Booting dengan Kernel *Default*

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	27,60	22,52	21,79
ke-2	31,35	25,42	23,32
ke-3	25,62	20,10	21,15
ke-4	26,93	23,52	24,90
ke-5	25,70	20,93	21,75
ke-6	27,52	22,15	23,85
ke-7	26,69	25,75	21,90
ke-8	26,70	25,60	19,55
ke-9	27,15	24,38	23,65
ke-10	25,64	24,36	24,45

Dari tabel diatas memperlihatkan *benchmark* booting dengan kernel linux *default* (satuan dalam detik), proses pengambilan data dihitung pada saat pemberian IP *address* oleh *server*. Seperti yang terlihat pada tabel proses booting menunjukkan kurang stabil nya proses booting pada *client*. Perbedaan pada setiap *client* adalah proses tidak dilayani sekaligus walaupun waktu dinyalakan bersamaan, dari proses booting pertama pada *client* 3 lebih cepat lalu *client* 2 dan *client* 1 dikarenakan waktu melayani dilakukan satu per satu, pada proses booting ke dua sama seperti proses waktu booting pertama, proses booting ke tiga *client* 2 yang pertama dilayani sehingga lebih cepat proses booting, dan dikarenakan pada proses pengambilan data dilakukan me-*restart* setiap melakukan pengambilan data sehingga modul-modul yang di *load* itu tidak tersimpan di dalam memory maka terjadi perubahan-perubahan tersebut dapat dilihat dari percobaan satu di

Tabel 4.1 pada *client 3* selalu lebih cepat dengan waktu 21,79" setelah itu *client 2* dan *client 1* tetapi tidak hanya pada *client 3* saja proses booting lebih cepat pada percobaan lima *client 2* dengan waktu 20,93" dikarenakan pada proses tersebut terjadi proses inisialisasi hardware dilakukan saat booting. Selatjutnya data yang didapat diubah dalam bentuk grafik dapat dilihat pada gambar 4.1.



Gambar 4.1 Grafik *Benchmark* Booting dengan Kernel *Default*

Pada grafik perbedaan menunjukkan berbeda-beda dikarenakan kernel linux yang digunakan adalah kernel *default* tampa optimalisasi, jadi tidak stabil untuk lingkungan LTSP.

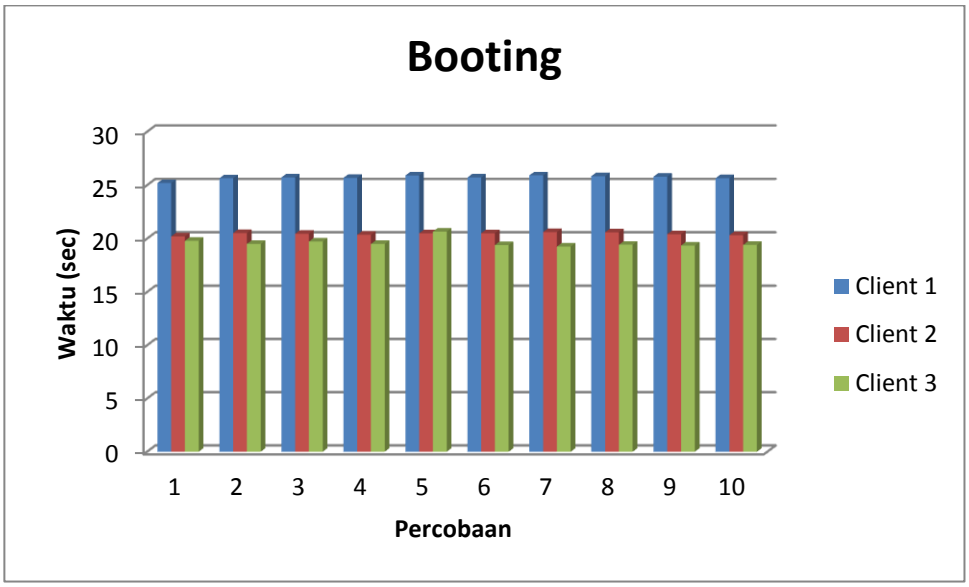
Selanjutnya dilakukan pada kernel linux versi 2.6.32-22 dengan skenario yang sama pada sebelumnya. Data dapat dilihat pada tabel berikut.

Tabel 4.2 Tabel *Benchmark* Booting dengan Kernel versi 2.6.32-22

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	25,79	20,60	19,79
ke-2	25,64	20,52	19,50

ke-3	25,72	20,45	19,72
ke-4	25,68	20,35	19,49
ke-5	25,90	20,49	20,65
ke-6	25,71	20,50	19,39
ke-7	25,91	20,61	19,25
ke-8	25,84	20,59	19,42
ke-9	25,79	20,41	19,35
ke-10	25,65	20,32	19,40

Dari tabel diatas menunjukkan ada perbedaan, yaitu proses booting lebih cepat dan konstan, menunjukan bahwa proses booting dengan kernel linux versi 2.6.32-22 cukup stabil. Perbedaan pada setiap *client* menunjukan yang tidak berubah disetiap melayani semua *client*, pada percobaan satu pada *client* 3 lebih cepat dengan waktu 19,79 detik lalu melayani *client* 2 dan *client* 1 selalu sama pada setiap booting sampai booting ke-10, pada percobaan lima pada *client* 3 dan *client* 2 yang dilayani terlebih dahulu adalah *client* 2 dengan waktu 20,49 detik walaupun waktu booting sama dikarenakan pada proses tersebut ada pengecekan *local drive*, seperti hal nya pada kernel *default* perubahan-perubahan tersebut dan begitu responsip dengan kecepatan booting dikarenakan inisialisasi perangkat atau modul-modul yang di *load* hanya modul-modul yang telah di *costumisasi* sehingga proses booting lebih cepat dan stabil, data dapat dilihat dalam bentuk grafik pada gambar 4.2.



Gambar 4.2 Grafik *Benchmark* Booting dengan Kernel versi 2.6.32-22

Pada grafik menujukkan bahwa kernel linux versi 2.6.32-32 cukup stabil dengan perbedaan dan perubahan di setiap *client* secara konstan dikarenakan sudah dilakukan *patch* sebelum di optimalisasi.

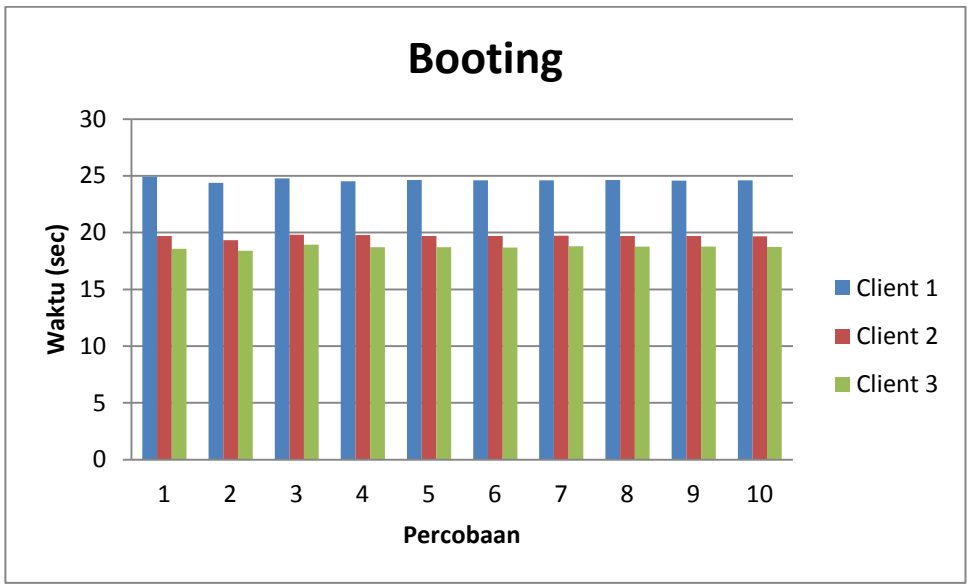
Selanjutnya yang terakhir dengan menggunakan kernel linux versi 2.6.33 pada *benchmarking* booting.

Tabel 4.3 Tabel *Benchmark* Booting dengan Kernel versi 2.6.33

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	24,91	19,71	18,58
ke-2	24,38	19,34	18,40
ke-3	24,78	19,80	18,93
ke-4	24,52	19,77	18,70
ke-5	24,64	19,69	18,72
ke-6	24,60	19,70	18,69
ke-7	24,59	19,72	18,79
ke-8	24,62	19,71	18,76
ke-9	24,58	19,69	18,78

ke-10	24,60	19,68	18,75

Pada tabel diatas perubahan juga terjadi dan bahkan lebih cepat dibandingan dengan kernel linux versi 2.6.32-22, menunjukkan bahwa kernel linux versi 2.6.33 lebih stabil dan cepat karena *patch* dilakukan semua secara manual bersamaan dengan optimalisasi pada saat membangun paket binary kernel. Pada proses booting tetap *client* 3 dilayani pertama kali dengan 18,58 detik, pada *client 2* waktu proses booting 19,17 detik sedangkan pada *client 1* waktu booting 24,91 detik kabel juga akan mempengaruhi di setiap proses booting, perbedaan di setiap *client 3* dan *client* 2 tidak berbeda jauh dibanding pada *client* 1 tetapi dari proses booting ke-1 hingga proses booting ke-10 dari semua *client* menujukan secara konstan stabil dengan waktu yang sama di setiap *client*, *patch* yang dilakukan akan memperkaya *driver* yang sudah tersedia sama seperti pada kernel versi 2.6.32-22 proses inisialisasi dan modul-modul tertentu saja yang hanya diaktifkan ketika proses tersebut, perangkat keras langsung dikenali sehingga proses booting lebih cepat berikut grafik yang tunjukan pada gambar 4.3.



Gambar 4.3 Grafik *Benchmark* Booting dengan Kernel versi 2.6.33

4.4.2 Proses Pengujian Program Aplikasi

Berikutnya akan dilakukan *benchmark* proses eksekusi program aplikasi Open Office 3.1 dan Web Browser Firfox 3.5.3, pengujian dilakukan dengan skenario pengujian sama seperti proses *benchmark* booting dari ketiga kernel linux yang digunakan.

4.4.2.1 Pengujian Eksekusi Program Aplikasi Open Office 3.1

Tabel dan grafik dibawah ini eksekusi program aplikasi secara bersamaan pada ketiga *client* aktif, dengan menggunakan kernel *default*.

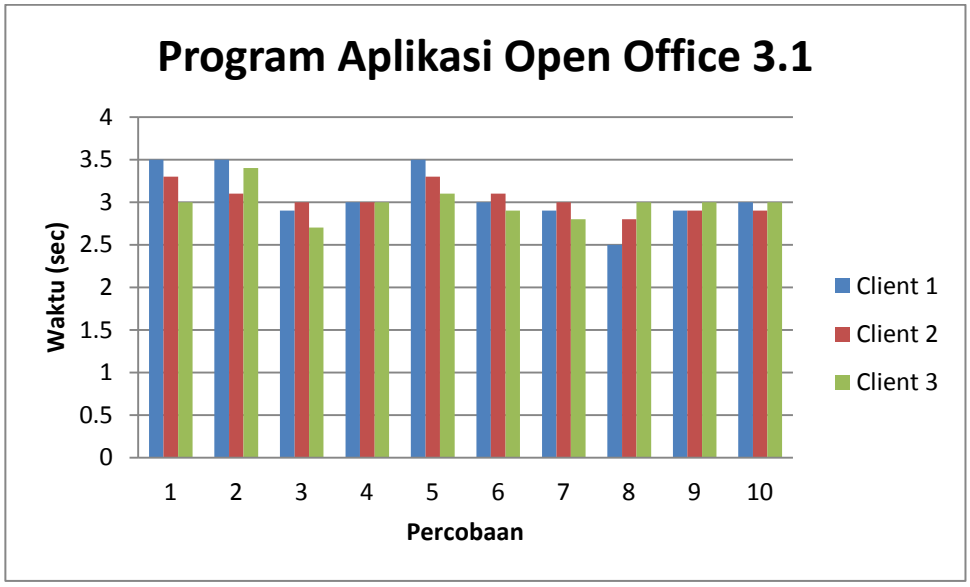
Tabel 4.4 Tabel *Benchmark* Eksekusi Program Aplikasi Open Office 3.1 dengan Kernel Linux *Default*

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	3,5	3,3	3,4
ke-2	3,5	3,1	3,3
ke-3	2,9	3	2,7
ke-4	3,1	3,3	3
ke-5	3,5	3,3	3,1
ke-6	3	3,1	2,9
ke-7	2,9	3	2,8
ke-8	2,5	2,8	3
ke-9	2,9	2,9	3
ke-10	3	2,9	3

Pada tabel diatas perubahan yang berbeda-beda di setiap *client* sama pada proses booting dengan kernel linux *default* pada *client* 3 diesekusi lebih cepat dengan waktu 3,4 detik karena dilayani pertama kali pada proses eksekusi ke-1, sedangkan pada proses eksekusi ke-2 yang pertama dilayani adalah *client* 2 dengan waktu 3,1 detik, dikarenakan pada saat proses eksekusi tidak hanya mengeksekusi program aplikasi open office saja *library-library* yang dibutuhkan

untuk mengeksekusi program aplikasi open office sehingga pada proses yang ke-1 lebih lambat dibanding proses-proses selanjutnya, karena proses yang ke-2 *library-library* tersebut masih terimpan di dalam memory.

Data proses esekusi program aplikasi yang didapatkan dari hasil *benchmark* dapat dilihat dalam bentuk grafik pada gambar 4.4.



Gambar 4.4 Grafi *Benchmark* Eksekusi Program Aplikasi Open Office 3.1 dengan Kernel Linux *Default*

Selanjutnya pengujian menggunakan kernel linux versi 2.6.32-22 dengan skenario yang sama pada sebelumnya, berikut tabel dan grafik.

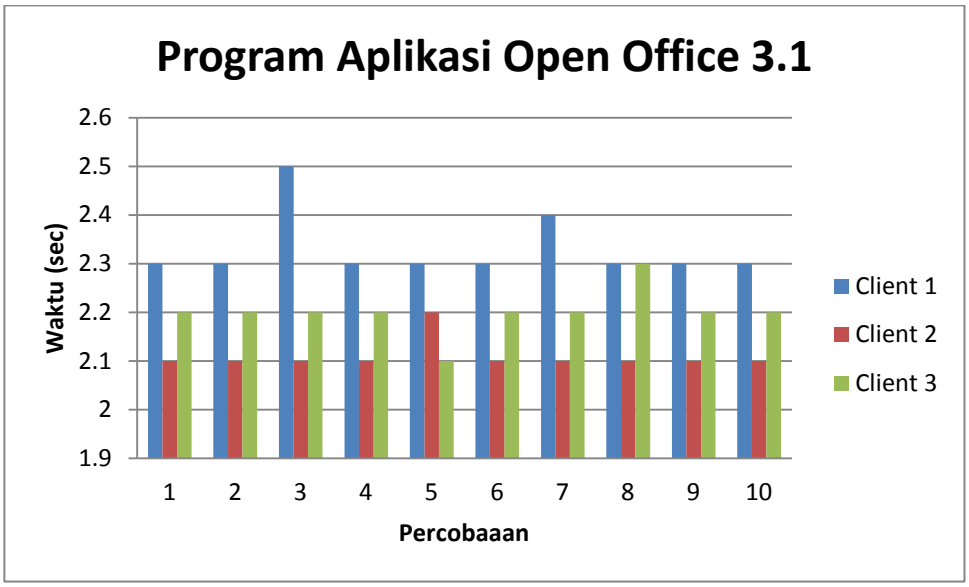
Tabel 4.5 Tabel *Benchmark* Eksekusi Program Aplikasi Open Office 3.1 dengan Kernel Linux versi 2.6.32-22

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	2,5	2,3	2,4
ke-2	2,3	2,1	2,2
ke-3	2,5	2,1	2,2
ke-4	2,3	2,1	2,2
ke-5	2,3	2,2	2,1

ke-6	2,3	2,1	2,2
ke-7	2,4	2,1	2,2
ke-8	2,3	2,1	2,2
ke-9	2,3	2,1	2,2
ke-10	2,3	2,1	2,2

Pada tabel diatas perubahan di setiap *client* tidak seperti pada kernel linux *default*, pada *client* 2 yang lebih cepat dan pertama dengan waktu 2,3 detik selanjutnya *client 3* dengan waktu eksekusi 2,4 detik dan pada *client 1* waktu eksekusi 2,5 detik, pada saat eksekusi ke-5 *client* 3 pertama diesekusi dengan waktu eksekusi 2,1 detik lalu *client 2* dengan waktu eksekusi 2,2 detik selanjutnya pada *client 1* dengan waktu eksekusi 2,3 detik, karena proses eksekusi tidak akan dilakukan secara bersamaan walaupun waktu eksekusi dilakukan secara bersamaan, pada proses eksekusi sama hal nya pada kernel *default* akan mengeksekusi banyak *library* yang dibuntuhkan sehingga pada proses selanjutnya akan lebih responsif dan proses eksekusi akan lebih cepat dan stabil.

Data dalam bentuk grafik pada proses *benchmark* eksekusi program aplikasi Open Office 3.1 dengan kernel linux versi 2.6.32-22, ditunjukan pada gambar 4.5.



Gambar 4.5 Grafik *Benchmark* Eksekusi Program Aplikasi Open Office 3.1 dengan Kernel Linux versi 2.6.32-22

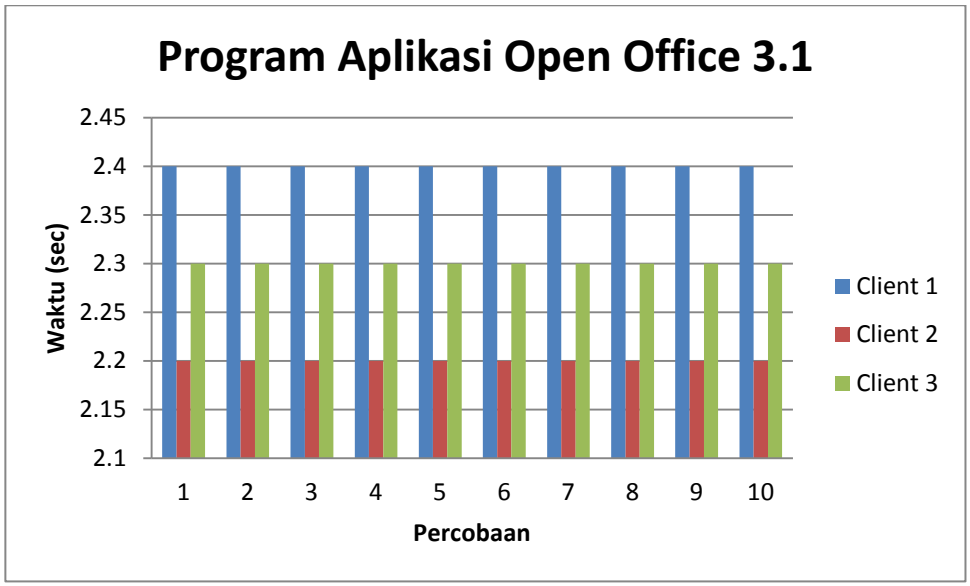
Selanjutnya yang terakhir pengujian pada kernel linux versi 2.6.33, tabel dan grafik sebagai berikut.

Tabel 4.6 Tabel *Benchmark* Eksekusi Program Aplikasi Open Office 3.1 dengan Kernel Linux versi 2.6.33

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	2,6	2,4	2,5
ke-2	2,4	2,2	2,3
ke-3	2,4	2,2	2,3
ke-4	2,4	2,2	2,3
ke-5	2,4	2,2	2,3
ke-6	2,4	2,2	2,3
ke-7	2,4	2,2	2,3
ke-8	2,4	2,2	2,3
ke-9	2,4	2,2	2,3
ke-10	2,4	2,2	2,3

Pada kernel linux versi 2.6.33 lebih lambat dibandingkan dengan versi 2.6.32-22 di setiap *client* karena pada kernel linux versi 2.6.33 *patch* dilakukan secara manual, tetapi tidak ada perubahan di setiap *client* pada saat eksekusi ke-1 sama hal nya pada kernel versi sebelumnya proses eksekusi tidak hanya program aplikasi open office saja tetapi *library* program tersebut akan dieksekusi sehingga pada awal penggambilan data tidak begitu responsif, sedangkan proses eksekusi ke-2 sampai ke-10 proses eksekusi lebih responsif dikarenakan *library-library* masih tersimpan di dalam memory, pada proses eksekusi percobaan satu *client 2* lebih cepat dengan waktu 2,4 detik lalu *client 3* dilayani dengan waktu eksekusi 2,5 detik. Pada tetap percobaan dua hingga percobaan sepuluh *client 2* lebih responsif dengan waktu eksekusi lebih cepat yaitu 2,2 detik lalu pada *client 3* dengan waktu eksekusi 2,3 detik dan yang terakhir pada *client 1* dengan waktu eksekusi 2,4 detik waktu eksekusi stabil dan konstan pada setiap percobaan setelah percobaan satu dilakukan.

Data dalam bentuk grafik pada proses *benchmark* eksekusi program aplikasi Open Office 3.1 dengan kernel linux versi 2.6.33, ditunjukan pada gambar 4.6.



Gambar 4.6 Grafik *Benchmark* Eksekusi Program Aplikasi Open Office 3.1 dengan Kernel Linux versi 2.6.33

4.4.2.2 Pengujian Eksekusi Program Aplikasi Web Browser Firefox 3.5.3

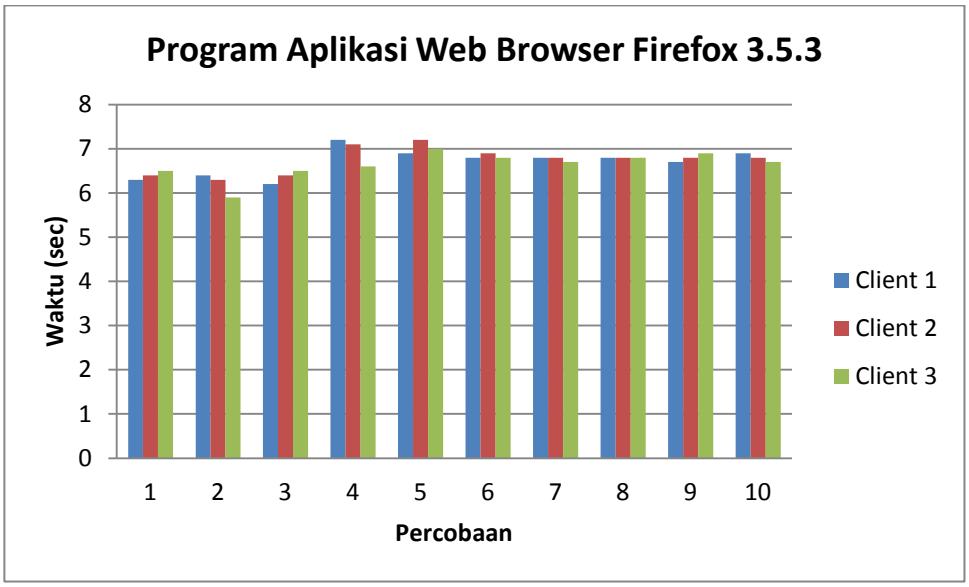
Tabel dan grafik dibawah ini eksekusi program aplikasi secara bersamaan pada ketiga *client* aktif, dengan menggunakan kernel *default*.

Tabel 4.7 Tabel *Benchmark* Eksekusi Program Aplikasi Web Browser Firefox 3.5.3 dengan Kernel *Default*

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	6,3	6,4	6,5
ke-2	6,2	6,3	5,9
ke-3	6,2	6,4	6,5
ke-4	7,2	7,1	6,6
ke-5	6,9	7,2	7
ke-6	6,8	6,9	6,8
ke-7	6,8	6,8	6,7
ke-8	6,9	6,8	6,8
ke-9	6,7	6,8	6,9
ke-10	6,9	6,8	6,7

Pada tabel diatas perubahan yang berbeda-beda di setiap *client* sama halnya pada proses booting dan eksekusi program aplikasi open office dengan kernel linux *default* pada *client* 2 dieksekusi lebih cepat karena dilayani pertama kali pada proses eksekusi ke-1, sedangkan pada proses eksekusi ke-2 yang pertama dilayani adalah *client* 3. Dengan perubahan berbeda-beda disetiap *client* tidak konstan pada karena pada kernel linux *default* tidak dilakukan perubahaan dan optimalisasi.

Data dalam bentuk grafik pada *benchmark* eksekusi program aplikasi web browser firefox 3.5.3 dengan kernel *default.*



Gambar 4.7 Grafik *Benchmark* Eksekusi Program Aplikasi Web Browser Firefox 3.5.3 dengan Kernel *Default*

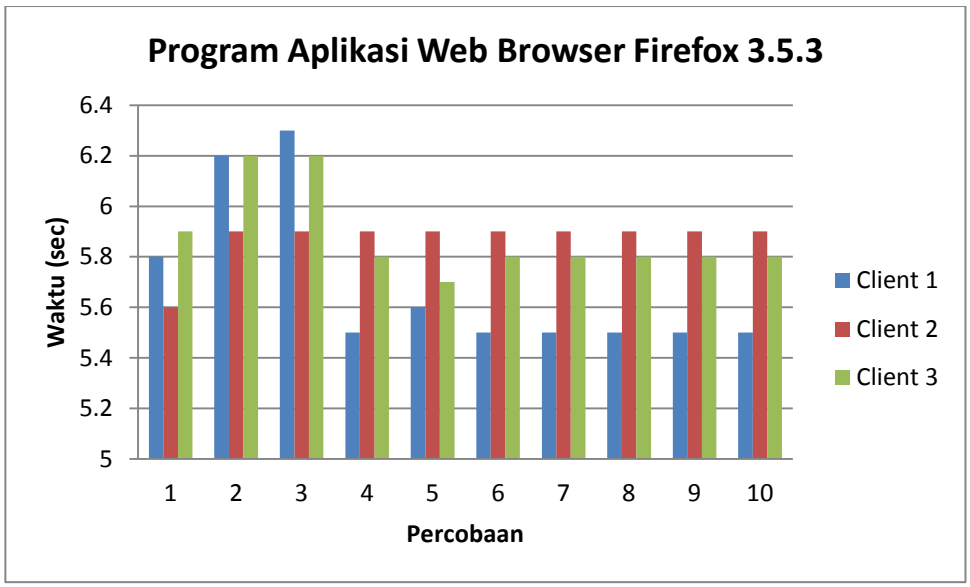
Selanjutnya pengujian menggunakan kernel linux versi 2.6.32-22 dengan skenario yang sama pada sebelumnya, berikut tabel dan grafik.

Tabel 4.8 Tabel *Benchmark* Eksekusi Program Aplikasi Web Browser Firefox 3.5.3 dengan Kernel Linux versi 2.6.32-22

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	6,5	6.2	6,4
ke-2	6,3	5,9	6,2
ke-3	6,3	5,9	6,2
ke-4	5,5	5,9	5,8
ke-5	5,6	5,9	5,7
ke-6	5,5	5,9	5,8
ke-7	5,5	5,9	5,8
ke-8	5,5	5,9	5,8
ke-9	5,5	5,9	5,8
ke-10	5,5	5,9	5,8

Pada tabel diatas perubahan di setiap *client* tidak seperti pada kernel linux *default*, pada *client* 2 yang lebih cepat dan pertama di layani, pada saat eksekusi ke-1 sampai proses eksekusi ke-5 secara bergantian antara *client* 1 dan *client* 3 walaupun waktu eksekusi dilakukan secara bersamaan karena proses tersebut dilakukan diawal eksekusi, dan *library* juga ikut dieksekusi setelah eksekusi ke-6 pada setiap *client* tidak mengalami perubahan dengan menggunakan kernel linux versi 2.6.32-22 eksekusi cukup konstan dan stabil dikarenakan pada proses eksekusi selanjutnya *library-library* masih tersimpat di dalam memory sehingga dengan proses eksekusi menjadi lebih cepat dan stabil.

Data dalam bentuk grafik pada *benchmark* eksekusi program aplikasi web browser firefox 3.5.3 dengan kernel linux versi 2.6.32-22.



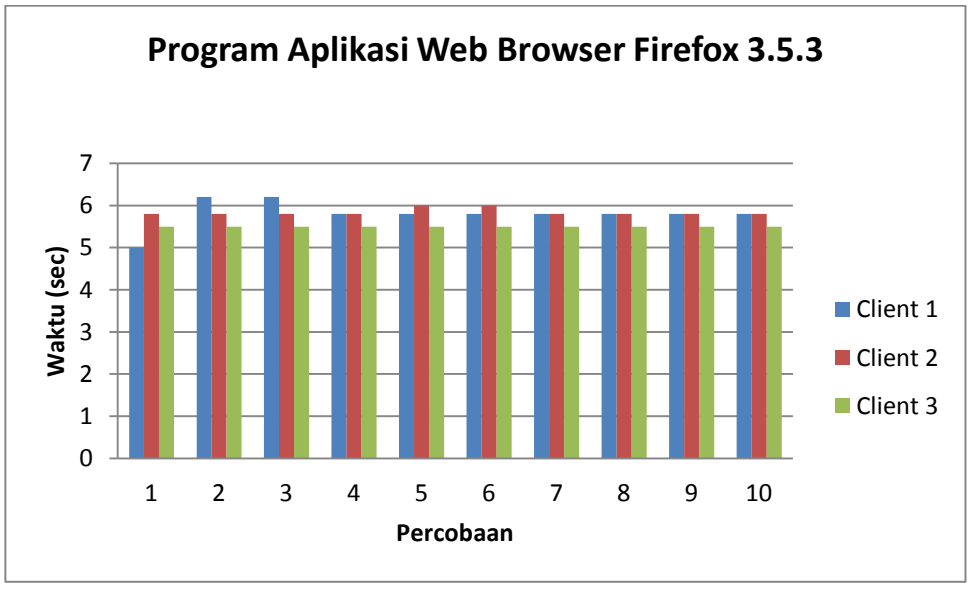
Gambar 4.8 Grafik *Benchmark* Eksekusi Program Aplikasi Web Browser Firefox 3.5.3 dengan Kernel Linux versi 2.6.32-22

Selanjutnya pengujian yang terakhir pada kernel linux versi 2.6.33, berikut tabel dan grafik.

Tabel 4.9 Tabel *Benchmark* Eksekusi Program Aplikasi Web Browser Firefox 3.5.3 dengan Kernel Linux versi 2.6.33

Percobaan	*Client* 1 (detik)	*Client* 2 (detik)	*Client* 3 (detik)
ke-1	6,4	6	5,9
ke-2	6,2	5,8	5,5
ke-3	6,2	5,8	5,5
ke-4	5,8	5,8	5,5
ke-5	5,8	6	5,5
ke-6	5,8	6	5,5
ke-7	5,8	5,8	5,5
ke-8	5,8	5,8	5,5
ke-9	5,8	5,8	5,5
ke-10	5,8	5,8	5,5

Data dalam bentuk grafik pada *benchmark* eksekusi program aplikasi web browser firefox 3.5.3 dengan kernel linux versi 2.6.33.



Gambar 4.9 Grafik *Benchmark* Eksekusi Program Aplikasi Web Browser Firefox 3.5.3 dengan Kernel Linux versi 2.6.33

Dari hasil pengujian dan perbandingan grafik yang didapatkan, sama hal nya seperti kernel versi *default* dan versi 2.6.32-22 dapat dilihat bahwa kernel

linux versi 2.6.33 dengan menggunakan Ubuntu Desktop 9.10 memiliki efisiensi yang lebih baik dibandingkan menggunkan kernel linux versi 2.6.32-22 dan kernel *default*, hal ini disebabkan kernel linux versi 2.6.33 yang telah di optimalisasi sendiri dan *pacth* dilakukan secara manual tidak seperti pada kernel linux versi 2.6.32-22 yang sudah di *patch* sebelum di kompilasi walaupun telah di optimalisasi yang dikhususkan untuk sistem LTSP.

Kinerja yang lebih baik pada kernel linux versi 2.6.33 juga disebabkan ada perbaikan *bug* pada kernel linux tersebut, dengan modul-modul yang lebih baru. Ada beberapa faktor-faktor penyebab terjadi perubahan-perubahan pada proses booting dan eksekusi program aplikasi.

1. Hardware, cukup mempengaruhi karena pada saat proses booting ada inisialisasi untuk mengenali modul-modul apa saja yang akan diaktifkan.
2. Kabel, pada saat *client* me-*request* untuk mendapatkan IP *Address* DHCP agar *client* memuat kernel dan melakukan inisialisasi hardware sering tidak terpenuhi.

4.5 Pengujian Live CD/DVD

Setelah melakukan pengujian dan analisa, selanjutnya untuk menguji *Live* CD/DVD yang telah dibangun dengan mem-*bundel* paket LTSP dan kernel linux versi 2.6.33 apakah *Live* CD/DVD berjalan dengan baik dan melakukan pengecekan apakah kernel linux dan paket LTSP sudah ada di dalam *Live* CD/DVD yang telah dibangun.

1. Komputer dihidupkan kemudian BIOS mendeteksi semua hardware yang terpasang kemudian mencari urutan *device* yang bisa diboot, dengan terlebih dahulu *setting* boot *priority* CD/DVD pada komputer atau pada Virtual Box .
2. Selanjutnya untuk sistem yang dijalankan lewat *Live* CD/DVD maka BIOS akan mencari boot *sector* pada CD-ROM dan meloadnya yang ada

didalam *Live* CD/DVD. Umumnya, yang akan diload adalah yang ada di boot *sector* adalah boot loader. Boot loader yang nantinya dipakai dalam sistem ini adalah GRUB, lalu memilih opsi ***Try and Run Ubuntu Desktop Without Change Disk*** setelah memilih opsi tersebut selesai dan masuk kedalam *desktop.*

3. Buka program *Terminal* dan lakukan pengecekan kernel linux apakah sudah masuk atau tidak dengan mengetikan perintah `uname –a` maka keluaran menunjukan seperti ini `Linux linux 2.6.31-14-generic #48-Ubuntu SMP Fri Oct 16 14:04:26 UTC 2009 GNU/Linux` maka lakukan pengecekan dengan melakukan perintah `sudo dpkg --get-selections | grep '2.6'` maka akan menampilakan seperti dibawah ini:

```
libedata-cal1.2-6                              install
libpython2.6                                   install
linux-header-2.6.33-rendra-custom-2            install
linux-image-2.6.33-rendra-custom-2             install
python2.6                                      install
python2.6-minimal                              install
```

Maka kernel sudah masuk ke dalam *Live* CD/DVD secara baik, selanjutnya pengecekan paket LTSP apakah sudah berhasil masuk atau belum lakukan pengecekan dengan perintah `sudo aptitude search ltsp` maka keluaran akan seperti di bawah :

```
i   ltsp-server             -Basic LTSP server environm
i   ltsp-server-standalone  -Complete LTSP server envir
i A ltspfs                  -Fuse based remote filesyst
```

4. Lakukan juga dengan pengujian dengan menginstal secara permanen pada *Harddisk.*

5. Setelah proses instalasi pada *Harddisk* lakukan pengecekan dengan cara seperti pada pengujian pada saat *running* secara *Live* CD/DVD.