



UNIVERSITAS INDONESIA

***Analisa Kualitas Live Migration Virtual Machine Pada
Peer-to-Peer Network Menggunakan Xen***

SKRIPSI

**SYAMSUDIN DANIL SURYADI
0806339364**

**FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN TEKNIK ELEKTRO
DEPOK
JUNI 2012**



UNIVERSITAS INDONESIA

***Analisa Kualitas Live Migration Virtual Machine Pada
Peer-to-Peer Network Menggunakan Xen***

SKRIPSI

Diajukan sebagai salah satu syarat memperoleh gelar sarjana

**SYAMSUDIN DANIL SURYADI
0806339364**

**FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN TEKNIK ELEKTRO
DEPOK
JUNI 2012**

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar**

Nama : Syamsudin Danil Suryadi

NPM : 0806339364

Tanda Tangan : 


Tanggal : 13 Juni 2012

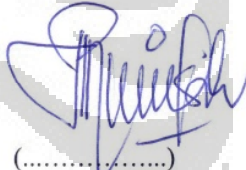
HALAMAN PENGESAHAN

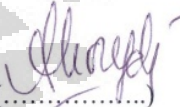
Skripsi ini diajukan oleh :
Nama : Syamsudin Danil Suryadi
NPM : 0806339364
Program Studi : Teknik Komputer
Judul Skripsi : Analisa Kualitas *Live Migration Virtual Machine*
pada *Peer-to-Peer Network* Menggunakan Xen

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang dilakukan untuk memperoleh gelar Sarjana Teknik pada program studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Dr. Ir. Anak Agung Putri Ratna M.Eng.  (.....)

Penguji : Ir. A. Endang Sriningsih MT., Si.  (.....)

Penguji : Prima Dewi Purnamasari ST., MT., MSc.  (.....)

Ditetapkan di : Depok
Tanggal : 29 Juni 2012

KATA PENGANTAR

Puji syukur saya panjatkan kehadirat Allah SWT, karena atas segala rahmat dan hidayah-Nya saya dapat menyelesaikan skripsi ini. Saya menyadari bahwa skripsi ini tidak akan terselesaikan tanpa bantuan dari berbagai pihak. Mulai dari proses pembelajaran, analisa yang telah dijalani dan proses penyusunan dari buku skripsi ini, saya ingin mengucapkan terima kasih kepada:

1. Dr. Ir. Anak Agung Putri Ratna M.Eng., selaku pembimbing telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini.
2. Mama yang tersayang atas segala doa dan pengorbanannya, maka aku bisa mencapai segala apa yang ada pada diriku saat ini.
3. Terima kasih kepada Yulius Dimas, Bagus Wira, Rian, Wega, James, Diamond, dan Rhaka, rekan-rekan seperjuangan dalam pengerjaan skripsi di Digital hingga pulang larut malam. Terima kasih kepada Rani Kumalasari atas segala bantuannya selama pengerjaan skripsi ini.
4. Terima kasih kepada Alifandi, Shaugi, Dyani, Henry, Noni, dan Asep. Teman-teman satu bimbingan saya, tim skripsi yang luar biasa. Terima kasih kepada Imam Bahari atas segala saran dan bantuannya.
5. Terima kasih kepada Michelle, atas kepergiannya disaat aku membutuhkannya sehingga aku harus bertahan keras menyelesaikan skripsi ini.
6. Terima kasih kepada Imam Bahari, Tio, Dodi, dan Cobra atas bantuannya berbagi keceriaan dan menemani di saat-saat sulit pengerjaan skripsi ini
7. Dan juga terima kasih untuk teman-teman saya dari Teknik Komputer angkatan 2008 yang tiada hentinya mendukung saya baik secara langsung maupun tidak langsung.

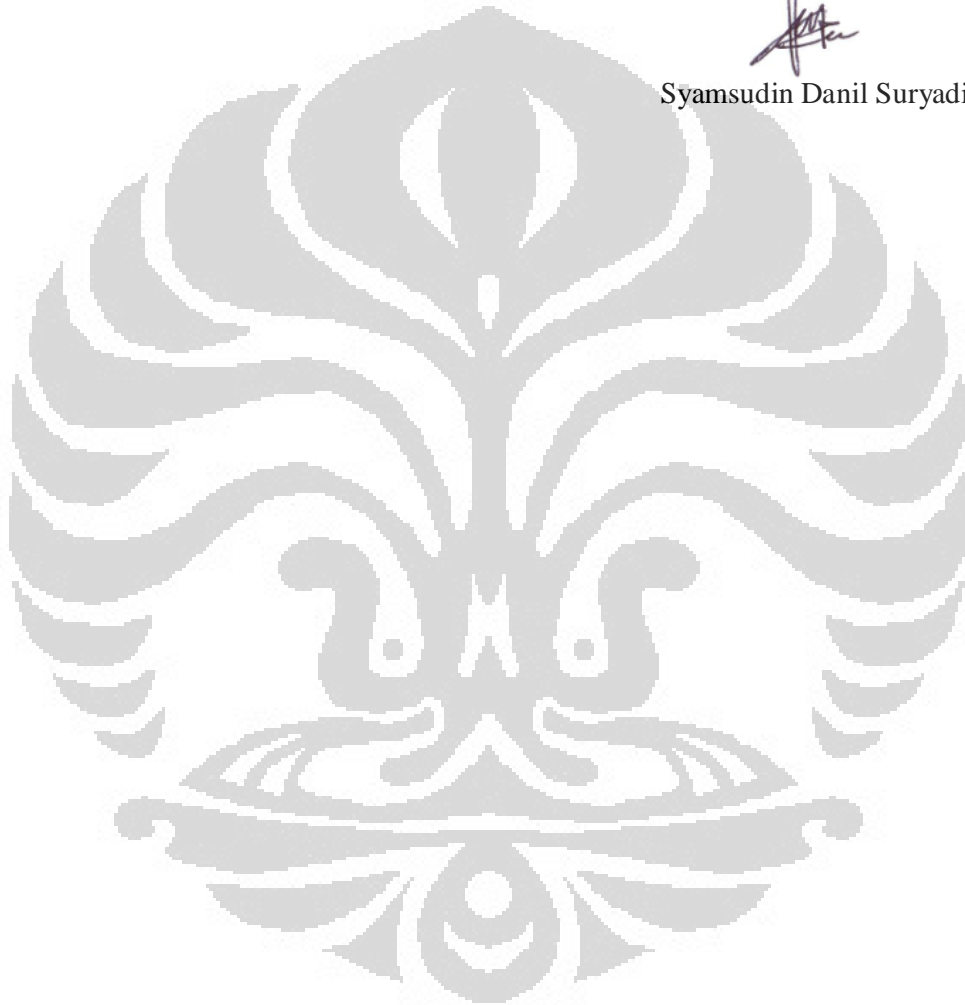
Saya memohon maaf apabila terdapat kesalahan dan berbagai kekurangan lain dalam penulisan Skripsi ini. Kritik dan saran yang membangun sangat saya harapkan sehingga dapat membantu saya dalam melakukan penyusunan makalah

yang lebih baik lagi di kemudian hari. Akhir kata, semoga Allah SWT berkenan membalas kebaikan semua pihak yang telah membantu. Semoga skripsi ini bermanfaat bagi perkembangan ilmu pengetahuan.

Depok, 12 Juni 2012



Syamsudin Danil Suryadi



**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai civitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Syamsudin Danil Suryadi
NPM : 0806339364
Program Studi : Teknik Komputer
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, meyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Non Eksklusif (Non-exclusive Royalty Free Right)** Atas karya ilmiah saya yang berjudul:

“Analisa Kualitas Live Migration Virtual Machine pada Peer-to-Peer Network Menggunakan Xen”

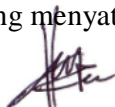
Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia Berhak menyimpan, mengalihmediakan/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 12 Juni 2012

Yang menyatakan


(Syamsudin Danil Suryadi)

ABSTRAK

Nama : Syamsudin Danil Suryadi
Program Studi : Teknik Komputer
Judul : Analisa Kualitas *Live Migration Virtual Machine* Pada *Peer-to-Peer Network* Menggunakan XEN

Skripsi ini berisi tentang perancangan, implementasi serta analisa performa dari *Live Migration* dari *Virtual Machine* terhadap perubahan datanya menggunakan kernel Xen. *Live migration* yang merupakan salah satu fitur virtualisasi menjadi teknologi penting dalam dunia IT (*Information Technology*).

Pengukuran dilakukan dengan beberapa parameter, yaitu *throughput*, *delay*, paket yang hilang, dan beban CPU yang dihasilkan selama migrasi. Hasilnya selama tidak ada perubahan data di *virtual machine* nilai *throughput* stabil di nilai sekitar 7 Mbit/sec. Nilai *delay* bila tidak ada perubahan data di *virtual machine* didapatkan stabil dinilai 0%. Jumlah paket yang hilang selama migrasi bila tidak terjadi perubahan data di *virtual machine* nilai stabil pada rata-rata 5 paket. Nilai beban CPU selama migrasi tidak terjadi perubahan yang berarti masih stabil di bawah 2.00. Hal ini mengindikasikan perubahan data ternyata berefek buruk terhadap kualitas *live migration* karena begitu terjadi perubahan data di *virtual machine* maka nilai dari parameter-parameter pengujian menjadi lebih buruk, yaitu menjadi lebih rendah pada nilai *throughput*, lebih tinggi pada nilai *delay*, dan lebih besar pada jumlah paket yang hilang.

Kata kunci: *Live Migration*, Perubahan Data, *Virtual Machine*

ABSTRACT

Nama : Syamsudin Danil
Program study : Computer Engineering
Title : Analysis Quality Of Virtual Machine Live Migration in
Peer to Peer Network Using Xen

This thesis contains the design, implementation and performance analysis of the Live Migration of Virtual Machine to change its data using the Xen kernel. Live migration is one of the features of virtualization become an important technology in the world of IT (Information Technology).

Measurements were performed with several parameters, ie throughput, delay, packet loss, and the CPU load generated during migration. The result as long as no changes to the data in a virtual machine throughput value is stable at a value of about 7 Mbit / sec. Delay values when no changes to the data obtained in the virtual machine is stable rated 0%. Number of packets lost during the migration does not occur when data changes in the virtual machine so stable in 5 value pack, CPU load values for the migration is not significant change was stable below 2.00. This indicates a change in the data turned out to affect adversely the quality of live migration because when it changes the data in the virtual machine, then the value of the test parameters become worse that is become lower in the value throughput, higher on the value of delay, and greater on number of packets lost.

Keyword: Live Migration, Changes Data, Virtual Machine

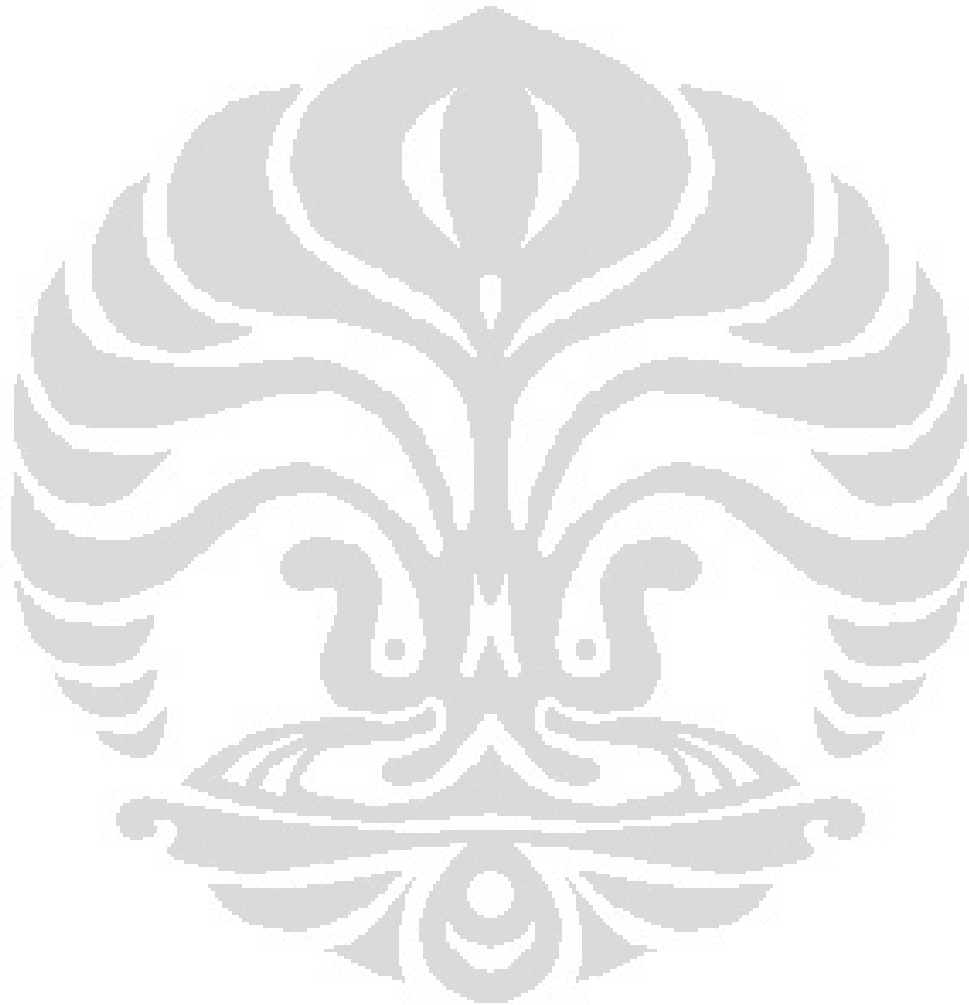
DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PERNYATAAN ORISINALITAS	ii
HALAMAN PENGESAHAN	iii
KATA PENGANTAR	iv
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI	vi
ABSTRAK	vii
ABSTRACT	viii
DAFTAR GAMBAR	xii
DAFTAR TABEL	xiii
DAFTAR ISTILAH	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Tujuan	2
1.3 Pembatasan Masalah	2
1.4 Metodologi Penelitian	3
1.5 Sistematika Penulisan	3
BAB II VIRTUALISASI: <i>LIVE MIGRATION OF VIRTUAL MACHINE</i>	4
2.1 Pengertian Virtualisasi	4
2.2 Konsep Virtualisasi	4
2.3 Manfaat Virtualisasi	5
2.4 Xen	6
2.5 <i>Virtual Machine</i>	6
2.6 <i>Live Migration Virtual Machine</i>	7
2.6.1 Faktor Live Migration	10
2.6.2 Virtualisasi I/O	11
2.6.3 Manfaat <i>Live Migration</i>	12
2.7 Penggunaan DRBD Sebagai Sistem Sinkronisasi Antar <i>Host</i>	12
2.7.1 Pengertian DRBD	12
2.7.2 Kerja DRBD	13
2.7.2.1 <i>Fully Synchronous</i>	13
2.7.2.2 <i>Asynchronous</i>	14
2.7.3 Aksesibilitas data hanya di aktif node	14
2.7.4 Fitur DRBD	14
2.7.4.1 <i>Single – Primary Mode</i>	15
2.7.4.2 <i>Dual-Primary Mode</i>	15
2.7.4.3 <i>Efficient Synchronization</i>	15
2.7.5 Konfigurasi DRBD	16
2.7.5.1 Konfigurasi tempat penyimpanan sumber daya DRBD	16
2.7.5.2 Konfigurasi Jaringan	16
2.7.5.3 Konfigurasi Sumber Daya	17
2.8 <i>Logical Volume Manager (LVM) dan Redundant Array of Independent Disks (RAID)</i>	17
2.8.1 <i>Logical Volume Manager (LVM)</i>	18
2.8.1.1 Konfigurasi LVM	19

2.8.2	Redundant Array of Independent Disk (RAID)	19
-------	--	----

BAB III PERANCANGAN <i>LIVE MIGRATION</i> PADA XEN SERVER		
DENGAN <i>PEER-TO-PEER NETWORK</i> 20		
3.1	Topologi Jaringan	20
3.2	Algoritma Rancangan	21
3.2.1	Metode <i>Pre-Copy</i>	21
3.3	Rancangan Perangkat Keras dan Perangkat Lunak	23
3.3.1	Rancangan Perangkat Keras	23
3.3.2	Rancangan Perangkat Lunak	24
3.4	Konfigurasi Jaringan	25
3.4.1	Konfigurasi Jaringan Xen	25
3.4.2	Konfigurasi IP Statik	25
3.5	Aplikasi Uji	26
3.4	Skenario Uji Coba	27
3.4.1	Perubahan Data <i>Virtual Machine</i>	27
3.4.2	Skenario Satu	27
3.4.3	Skenario Dua	28
BAB IV UJICоба DAN ANALISA DATA KUALITAS PERPINDAHAN		
VIRTUAL MACHINE PADA JARINGAN PEER TO PEER		
MENGGUNAKAN XEN 30		
4.1	Implementasi dan Konfigurasi Sistem	30
4.1.1	Instalasi Ubuntu 11.10	30
4.1.1.1	Konfigurasi Ubuntu	30
4.1.1.2	Konfigurasi Jaringan Ubuntu	30
4.1.1.3	Konfigurasi Nama <i>Host</i>	31
4.1.2	Instalasi XEN 4.1	32
4.1.3	Instalasi GParted	32
4.1.4	Instalasi LVM	33
4.1.4.1	Konfigurasi LVM	33
4.1.5	Konfigurasi <i>Xen-tools</i>	33
4.1.6	Instalasi <i>Virtual Machine</i>	34
4.1.7	Instalasi DRBD	35
4.1.7.1	Konfigurasi DRBD	35
4.1.8	Konfigurasi VM Untuk Menggunakan DRBD	36
4.1.9	Salin Konfigurasi <i>Virtual Machine</i> Pada <i>Host</i> Dua	37
4.1.10	Konfigurasi <i>Live Migration</i> pada XEN	37
4.1.11	<i>Live Migration</i>	38
4.2	Analisa Kualitas Perpindahan Virtual Machine	38
4.3	Analisa Dan Ujicoba Pada Skenario Satu	38
4.3.1	Analisa <i>Throughput</i>	39
4.3.2	Analisa <i>Delay</i>	42
4.4	Analisa Pada Skenario Dua	44
4.4.1	Analisa <i>Ping Test</i>	45
4.4.2	Analisa Beban CPU	48
4.5	Analisa Lama Waktu Migrasi Dan <i>Downtime</i>	50
BAB V KESIMPULAN 53		

DAFTAR ACUAN	54
DAFTAR PUSTAKA	56

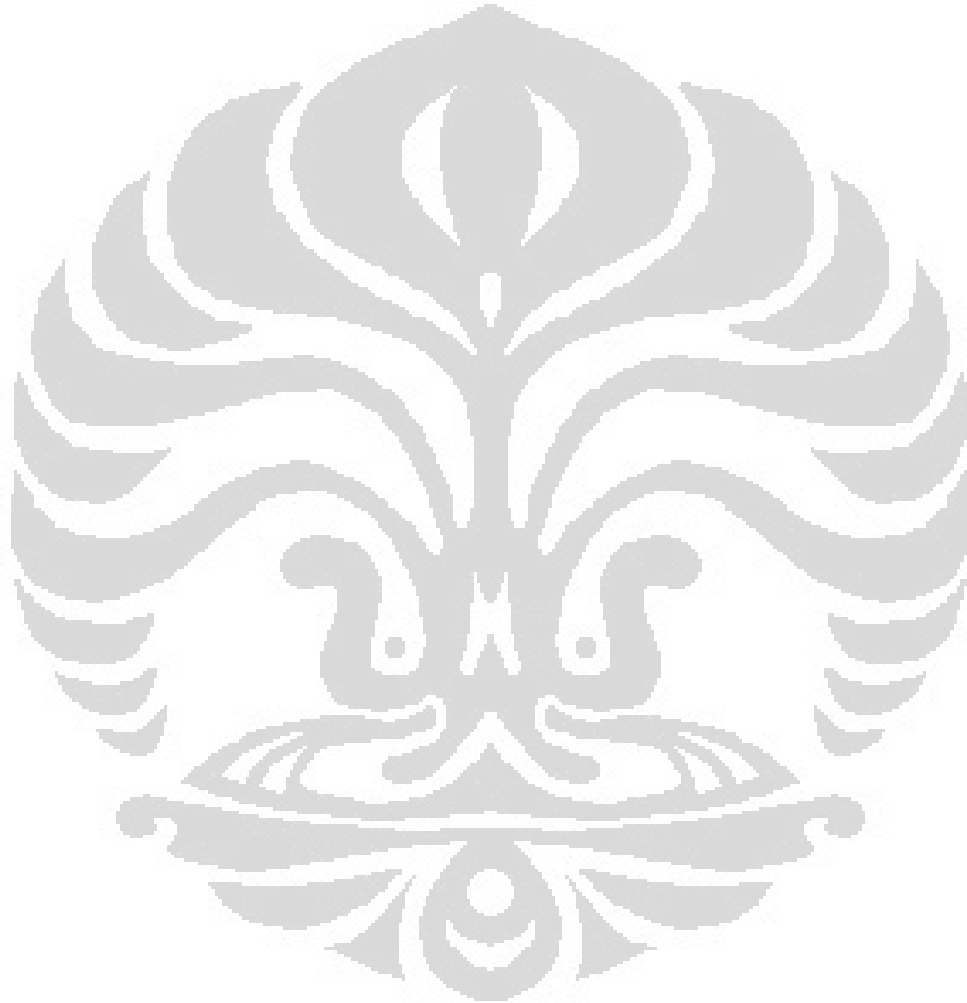


DAFTAR GAMBAR

Gambar 2.1. Konsep Dasar Virtualisasi	4
Gambar 2.2 Virtualisasi I/O Dalam <i>Virtual Machine</i>	11
Gambar 2.3. Sistem DRBD	13
Gambar 2.4 Contoh Konfigurasi DRBD	17
Gambar 2.5 <i>Logical Volume Group</i>	18
Gambar 3.1 Topologi Jaringan <i>Live Migration</i>	20
Gambar 3.2 Algoritma <i>Pre-Copy</i>	21
Gambar 3.3 Konfigurasi Jaringan Xen	25
Gambar 3.6. Skenario 1	27
Gambar 3.7 Skenario 2	28
Gambar 4.1 Konfigurasi IP Pada Host	31
Gambar 4.2 Konfigurasi Host	32
Gambar 4.3 Konfigurasi <i>Xen-tools</i>	34
Gambar 4.4 Konfigurasi DRBD	35
Gambar 4.5 Konfigurasi VM Untuk DRBD	36
Gambar 4.6 Konfigurasi <i>Live Migration</i> Pada Xen	37
Gambar 4.7 <i>Capturing TCP Port 8002</i>	39
Gambar 4.8 <i>Throughput</i> pada Summary Wireshark	40
Gambar 4.9 <i>Throughput VS Jumlah Migrasi</i>	41
Gambar 4.10 Delay Pada Summary Wireshark	42
Gambar 4.11 <i>Delay vs Jumlah Migrasi</i>	44
Gambar 4.12 <i>Capturing Ping Di Terminal Ubuntu</i>	45
Gambar 4.13 Hasil <i>Ping Test</i> Pada Percobaan Ketiga	46
Gambar 4.14 Paket Hilang VS Jumlah Migrasi	48
Gambar 4.15 Analogi CPU Load Linux [11]	49
Gambar 4.16 Beban CPU vs Waktu Migrasi	49
Gambar 4.17 Grafik IO Graph TCP Port 8002	51
Gambar 4.18 Downtime <i>Virtual Machine</i>	51

DAFTAR TABEL

Tabel 3.1 Rancangan Perangkat Keras Untuk <i>Live Migration</i>	23
Tabel 3.2 Rancangan Perangkat Lunak Untuk <i>Live Migration</i>	24
Tabel 3.3 Konfigurasi IP Statik	26
Tabel 4.1 Data Nilai <i>Throughput</i>	40
Tabel 4.2 Data Perhitungan <i>Delay</i>	43
Tabel 4.4 Data Paket Yang Hilang	47



DAFTAR ISTILAH

1. DRBD (*Distributed Replication Block Device*) adalah sistem distribusi penyimpanan untuk Linux.
2. *Dirty Pages* adalah *page* yang belum tertulis di dalam *disk*.
3. *Guest* adalah nama lain *virtual machine*.
4. *Host* adalah sistem fisik *server*.
5. *Live Migration* adalah sebuah teknologi dimana seluruh VM yang sedang berjalan dipindahkan dari mesin fisik yang satu ke mesin fisik yang lain.
6. LVM (*Logical Volume Manager*) adalah sebuah perangkat lunak bantu untuk manajemen *logical volume* (tipe penyimpanan *volume* yang bukan sebenarnya seperti partisi namun secara logikal) termasuk melakukan alokasi *disk*, *mirroring*, dan perubahan ukuran *logical volumes*.
7. Peer-to-Peer adalah jaringan komputer yang hanya menghubungkan dua komputer dimana kedua komputer bisa menjadi *server* maupun *client*, jadi tidak ada perbedaan antara *client* dan *server*.
8. RAID (*Redundant Array of Independent Disks*) adalah kombinasi *disk drive*.
9. VM (*Virtual Machine*) adalah implementasi perangkat lunak dari lingkungan komputasi dimana sistem operasi atau program dapat diinstal dan berjalan.
10. VMM (*Virtual Machine Monitor*) adalah sebuah program *host* yang memungkinkan satu fisik *host* untuk mendukung adanya lebih dari satu operating sistem berjalan di atasnya.
11. Xen adalah *open source Virtual Machine Monitor* (VMM) ,dikembangkan di Universitas Cambridge, yang mampu untuk menjalankan sampai dengan seratus sistem operasi.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dewasa ini bisnis dunia IT (*Information Technology*) dan bisnis *professional* telah berkembang untuk menghadapi tantangan dari lebih banyak devais serta perubahan layanan dan permintaan dari pelanggan. Tantangannya yang lebih sulit adalah untuk mengatur kekompleksitasannya namun menghasilkan kelenturan sehingga tetap dapat menjaga kestabilan biaya yang harus dikeluarkan. Satu kata yang mendeskripsikan lingkungan yang cocok untuk tipe bisnis seperti ini adalah kata dinamik. Teknologi sistem yang dinamik menolong dunia IT dan bisnis *professional* menghadapi tantangan ini. Demi usaha menciptakan sistem yang dinamik, perusahaan-perusahaan membutuhkan strategi virtualisasi untuk memobilisasi sumberdaya dari infrastruktur untuk bertemu dengan permintaan bisnis yang dinamik. Kemampuan bisnis dan perusahaan tidak hanya untuk bertahan dari banyaknya pesaing dan ekonomi, namun kesiapan menghadapi permintaan yang dinamik ini adalah fungsi kapabilitas dan kebugaran infrastruktur sistem bisnis yang mendukung kerja mereka. Hasilnya bagi dunia IT dan bisnis *professional* adalah peningkatan tingkatan layanan, membebaskan sumber daya yang kritis bagi sistem untuk mengambil tantangan bisnis yang lebih besar, dan penghematan biaya.

Virtualisasi telah menjadi fasilitas penting di dalam modernisasi instalasi komputasi dan pusat data. Virtualisasi menyediakan kesempatan untuk peningkatan efisiensi dengan cara meningkatkan utilitas *hardware* dan isolasi aplikasi sebaik meringkas alokasi sumber daya dan manajemen. Salah satu fitur yang membuat virtualisasi menarik adalah *live migration*. *Platform live migration* (seperti XenMotion atau VMotion) memungkinkan administrator untuk memindahkan *virtual machine* yang sedang berjalan ke fisik *host* yang baru. Ini merupakan keuntungan untuk para *service provider* untuk menyediakan *high*

availability pada aplikasi-aplikasinya. Ada tingkatan layanan dimana sebuah service provider berkomitmen kepada pengguna dalam hal penyewaan dan menjalankan aplikasi yang dideskripsikan dalam SLA (*Service Level Agreement*) berhubungan langsung *high availability* dari aplikasi. Berkaitan dengan bisnis, aktifitas seperti *restarting* sebuah mesin untuk perawatan hardware sehingga menghentikan layanan aplikasi adalah suatu hal yang sangat dilarang di zaman sekarang ini. *Live migration* meringankan masalah itu dengan mengizinkan administrator untuk memindahkan *virtual machine* dengan sedikit interupsi. Hal ini membuat dapat terjadinya perawatan fisik *hardware* secara regular, mendukung dinamik rekonfigurasi, dan mendukung pemindahan beban komputasi serta mendinginkan pusat data.

Bagaimanapun, sedikit interupsi pada layanan aplikasi masih tidak dapat diijinkan selama migrasi untuk mendukung *high availability*. Oleh karena itu, skripsi ini merancang *live migration* dengan topologi sederhana dan jaringan *peer-to-peer* sekaligus memberikan analisa pengaruh data dinamik pada *virtual machine* terhadap berbagai parameter pendukung perpindahan.

1.2 Tujuan

Tujuan dari penelitian ini adalah untuk menganalisa kualitas dari perpindahan *virtual machine* menggunakan kernel Xen dengan adanya perubahan data pada *virtual machine* sehingga dapat diambil kesimpulan apakah perubahan data pada *virtual machine* mempengaruhi kestabilan dari Xen *Live Migration*.

1.3 Pembatasan Masalah

Penelitian ini membahas mengenai konsep virtualisasi, *virtual machine*, dan gambaran perpindahan *virtual machine* secara langsung (*live migration*) menggunakan kernel Xen dan *Distributed Replication Block Device* (DRBD) serta implementasi *live migration* pada topologi sederhana untuk mendapatkan analisa kualitas *throughput*, *delay*, dan jumlah paket yang hilang terhadap dilakukannya

perubahan data pada *virtual machine* serta beban CPU, lama waktu migrasi, dan waktu *downtime*.

1.4 Metodologi Penelitian

Untuk membantu dalam melengkapi penulisan skripsi ini digunakan metode studi literatur, yaitu dengan mencari buku-buku jurnal-jurnal ilmiah artikel, *website* di internet yang digunakan untuk referensi, merancang skenario pengujian, kemudian melakukan implementasi, pengamatan, analisa, dan penarikan kesimpulan.

1.5 Sistematika Penulisan

Seminar ini akan dibagi menjadi 5 bab, yaitu :

- a. Bab 1 : Pendahuluan
Bab ini akan dijelaskan Latar Belakang, Tujuan, Pembatasan Masalah, Metodologi Penulisan dan Sistematika Penulisan.
- b. Bab 2 : Virtualisasi: *Live Migration Of Virtual Machine*
Bab ini akan dijelaskan mengenai Xen *Live Migration* serta sistem-sistem pendukungnya.
- c. Bab 3 : Perancangan *Live Migration* pada Peer-to-Peer Network.
Bab ini akan dijelaskan bagaimana perancangan *live migration* pada *peer-to-peer network*, topologi, algoritma serta perangkat lunak pendukungnya.
- d. Bab 4 : Implementasi, Ujicoba dan Analisa Kualitas Xen *Live Migration* dengan *Peer-to-Peer Network*
Bab ini akan dijelaskan tentang implementasi, ujicoba, dan analisa kualitas Xen *Live Migration* dengan *peer-to-peer network*.
- e. Bab 5 : Kesimpulan
Bab ini berisi mengenai kesimpulan dari uraian bab-bab sebelumnya.

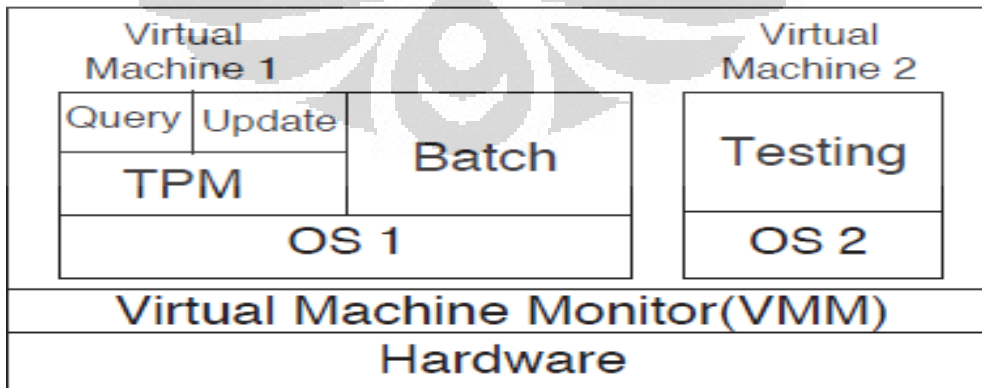
BAB II
VIRTUALISASI: LIVE MIGRATION OF VIRTUAL MACHINE

2.1 Pengertian Virtualisasi [2]

Virtualisasi ditemukan sekitar 30 tahun yang lalu untuk mengizinkan *main-frame* yang sangat mahal dahulu agar dapat saling berbagi melewati berbagai lingkungan aplikasi yang berbeda. Bila dapat disimpulkan virtualisasi adalah kemampuan untuk menjalankan lebih dari satu sistem operasi dalam sebuah fisik sistem dan saling berbagi sumber daya perangkat kerasnya.

2.2 Konsep Virtualisasi

Virtualisasi seperti yang telah dijelaskan pada pengertiannya dapat mengizinkan berjalannya dua lingkungan sistem operasi (sebagai *guest*) yang berbeda dalam mesin yang sama seperti pada Gambar 2.1. Kedua sistem operasi tersebut berjalan diatas *Virtual Machine Monitor* (VMM). VMM akan melakukan virtualisasi pada semua sumber daya seperti CPU, memori, penyimpanan, jaringan dan mengalokasikannya ke berbagai *virtual machines* (sebuah implementasi perangkat lunak dari lingkungan komputasi dimana sebuah sistem operasi dapat diinstall dan berjalan) yang berjalan di atas VMM. Maka kedua sistem operasi tersebut akan saling terisolasi satu dengan yang lain.



Gambar 2.1. Konsep Dasar Virtualisasi [2]

2.3 Manfaat Virtualisasi [3]

Ada beberapa manfaat dari virtualisasi yang salah satunya menjadi dasar dari percobaan skripsi ini

- Kehandalan dan Ketersediaan: Karena terisolasinya kedua aplikasi meski berada pada sebuah fisik yang sama maka kegagalan sebuah aplikasi dalam *virtual machines* tidak akan berpengaruh pada *virtual machines* lainnya.
- Biaya: Ada banyak penghematan yang dapat terjadi antara lainnya dengan penggabungan *server-server* kecil menjadi *powerful server*, pengurangan luas ruang *server*, dan lisensi perangkat lunak itu sendiri.
- Keamanan: Dengan kemampuan berjalannya lebih dari satu *guest* sistem operasi serangan keamanan sistem dapat diatasi karena adanya isolasi antar lingkungan operasi. Bila salah satu *guest* sistem operasi diserang maka *guest* yang lainnya tidak akan terkena efeknya atau berpengaruh.
- Kemampuan adaptasi ke variasi beban kerja: Perubahan dalam level intensitas beban kerja dapat dengan mudah direndahkan dengan cara *shifting resources* dan alokasi prioritas.
- Penyeimbangan beban: Status perangkat lunak yang secara keseluruhan terenkapsulasi oleh *Virtual Machines Monitor* (VMM). Selanjutnya, mudah untuk memindahkan atau migrasi mesin *virtual* ke *platforms* lain dalam tujuan untuk meningkatkan daya guna melalui penyeimbangan beban yang lebih baik atau kebutuhan perawatan pada mesin fisik.
- Peninggalan aplikasi: Meski jika sebuah organisasi memutuskan untuk memindahkan ke operating sistem yang berbeda, aplikasi masih dapat

berjalan pada operating sistem (OS) yang lama sebagai *guest OS* dalam VM. Hal ini dapat mengurangi biaya migrasi.

2.4 Xen

Ketika berbicara virtualisasi, VMWare dan Xen adalah dua nama yang paling dikenal. Xen adalah *open source Virtual Machine Monitor (VMM)* ,dikembangkan di Universitas Cambridge, yang mampu untuk menjalankan sampai dengan seratus sistem operasi. VMWare lebih tua dari Xen dan sebagai konsekuensinya sudah sangat terkenal. VMWare juga telah mengembangkan basis pengguna yang luas bersama dengan sistem pendukung yang berdedikasi. Namun ketika berbicara tentang VMWare tampak seperti pilihan yang lebih mahal karena VMWare merupakan aplikasi berbayar.

Xen kaya akan fitur, *open source*, dan *hypervisor* berbasis virtualisasi yang telah meskipun baru kemunculannya tetapi telah diterima dunia dan memiliki reputasi yang tinggi di dunia IT (*Information Technology*). Xen beroperasi pada *paravirtualization* mana ia memodifikasi sistem operasi yang sedang berjalan pada sehingga instruksi akan langsung dikirim ke perangkat keras. Xen juga mengharuskan *hardware* yang digunakan dengan perangkat lunak mereka adalah baik Intel-VT atau AMD-V. Ini berarti bahwa pengguna dengan *hardware* yang tidak kompatibel perlu meng-*upgrade* terlepas dari berapa kuat *hardware* yang dimiliki.

2.5 Virtual Machine [4]

Virtual machine (VM) adalah implementasi perangkat lunak dari lingkungan komputasi dimana sistem operasi atau program dapat diinstal dan berjalan. Secara tipikal *virtual machine* mengemulasikan lingkungan komputasi tetapi meminta sumber daya CPU, memori, *hard disk*, dan jaringan yang akan diatur oleh lapisan virtualisasi yang akan menerjemahkan permintaan ini ke fisik

hardware. *Virtual machine* diciptakan dengan lapisan virtualisasi seperti *hypervisor* atau *platform* virtualisasi yang berjalan diatas *server* sistem operasi. Sistem operasi ini dikenal sebagai *Host*. Lapisan virtualisasi dapat digunakan untuk menciptakan banyak lingkungan *virtual machine* yang terisolasi. Secara tipikal, *guest operating system*, *virtual machine*, dan programnya sadar bahwa mereka berjalan pada platform *virtual*, sepanjang *platform virtual* mendukung, perangkat lunak ini dapat diinstal dengan cara yang sama seperti menempatkan ke fisik server. Contohnya, *guest OS* mungkin mempunyai fisik *hard disk* tetapi sebenarnya permintaan IO (*Input Output*) diterjemahkan oleh lapisan virtualisasi sehingga dapat diakses oleh *host*.

Virtual machine dapat menyediakan berbagai manfaat dari instalasi OS dan program yang langsung ke fisik *hardware*. Isolasi menjamin aplikasi dan layanan yang berjalan dengan *virtual machine* tidak akan berinterferensi dengan *host OS* atau *virtual machine* yang lain. *Virtual machine* juga dapat dipindahkan, disalin, dan ditetapkan kembali antara *server host* untuk optimasi sumber daya *hardware*. Administrator dapat juga mengambil keuntungan untuk menyederhanakan *backup* dan *disaster recovery*.

2.6 *Live Migration Virtual Machine* [5]

Live migration adalah sebuah teknologi dimana seluruh VM yang sedang berjalan dipindahkan dari mesin fisik yang satu ke mesin fisik yang lain. Secara lebih jelas, hal tersebut berarti seluruh VM termasuk memori yang sedang aktif dan status terakhir dari lingkungan tersebut dipindahkan dari mesin fisik asal ke mesin fisik tujuan (antar *hosts*). Hal seperti ini dimaksudkan agar pada saat terjadi perpindahan VM yang berisi servis *online* pengguna tidak harus terputus dalam melakukan kegiatan *online*-nya. Pada tahap akhir perpindahannya *virtual I/O devices* akan diputus dari sumber dan akan dihubungkan kembali ke mesin fisik tujuan.

Ada dua pertimbangan parameter penting dalam melakukan *live migration*, yaitu total waktu migrasi (*total migration time*) dan waktu henti (*downtime*). Total waktu migrasi adalah jumlah waktu total yang dibutuhkan untuk perpindahan VM antar fisik mesin dan di dalamnya termasuk waktu henti. Waktu henti adalah waktu saat VM berhenti berjalan untuk sesaat, tergantung sistem yang diimplementasi, akibat perpindahan tersebut.

Perpindahan VM antar fisik mesin dapat dilakukan dengan beberapa teknik antara lain adalah

- *Stop-and-copy*, yaitu menghentikan layanan VM dan menyalin seluruh memori ke fisik tujuan. Teknik ini meminimalkan total waktu migrasi namun menyebabkan banyaknya waktu henti karena VM ditunda selama proses transfer tersebut.
- *On-demand*, yaitu perpindahan dengan menghentikan VM dan hanya menyalin data kernel penting ke fisik tujuan. Sisa dari pengalaman VM ditransfer ketika dapat diakses di fisik tujuan. Teknik ini memiliki waktu henti yang sangat singkat namun menyebabkan total waktu migrasi yang besar.

Bila di amati dari kedua teknik tersebut keduanya memiliki kualitas yang tidak baik. *Stop-and-copy* menyebabkan banyak waktu henti dan pada *on-demand* menyebabkan banyak waktu total migrasi. Kedua hal tersebut tidak dapat ditolerir mengingat jika VM sedang menjalankan aplikasi yang sedang banyak digunakan pengguna. Maka muncullah teknik *pre-copy* untuk mengakomodir kelemahan dua teknik sebelumnya.

- *Pre-copy*

Perpindahan secara *pre-copy* berupaya mengatasi masalah yang berhubungan dengan dua teknik awal tersebut dengan menggabungkan

iterasi yang dibatasi hingga tahap akhir perpindahan dan secara tipikal memiliki fase *stop-and-copy* yang singkat.

Inti dari adanya ide ini adalah konvergensi. Ini melibatkan iterasi penyalinan yang terus menerus dimana halaman memori (*memory pages*) VM yang telah dimodifikasi selama proses penyalinan sebelumnya dikirimkan kembali ke fisikal tujuan dengan asumsi bahwa sejumlah poin dari halaman yang termodifikasi akan cukup kecil untuk menghentikan VM sementara menyalin ,sejumlah kecil, sisa *pages* dan melakukan *restart* di fisikal tujuan. Desain seperti ini yang meminimalkan total waktu migrasi dan waktu henti.

Migrasi *pre-copy* melibatkan 6 tingkat , yaitu

- 1) *Initialisation*: pemilihan fisikal mesin tujuan.
- 2) *Reservation*: sumber daya di fisikal mesin asal dicadangkan
- 3) *Iterative pre-copy*: halaman yang dimodifikasikan selama iterasi sebelumnya ditransfer ke tujuan. Seluruh RAM telah dikirim dalam iterasi yang pertama.
- 4) *Stop-and-copy*: VM dihentikan untuk melakukan *transfer* akhir.
- 5) *Commitment*: *Host* tujuan mengindikasikan bahwa telah menerima dengan lengkap salinan dari VM.
- 6) *Activation*: Sumber daya pada fisikal yang baru ditempelkan kembali ke VM yang telah pindah pada fisikal tujuan tersebut.

Definisi dari kondisi terhentinya VM merupakan hal yang kritikal mengingat pentingnya VM untuk terus berjalan atau menghasilkan waktu henti yang sesingkat-singkatnya. Kondisi tersebut sangat tergantung dari desain *hypervisor* dan sistem dari *live migration* itu sendiri namun secara keseluruhan untuk mengurangi jumlah data yang disalin antar fisikal sambil meminimalkan waktu henti VM. Bagaimanapun juga kondisi terhentinya VM ini memiliki efek yang signifikan dalam kualitas proses migrasi.

2.6.1 Faktor *Live Migration*

Untuk memindahkan *virtual machine* dari *host* asal ke *host* tujuan ada beberapa hal yang harus diperhatikan agar perpindahan berjalan sesuai yang diinginkan

- Migrasi status CPU
- Migrasi isi memori
- Migrasi isi penyimpanan
- Migrasi koneksi jaringan

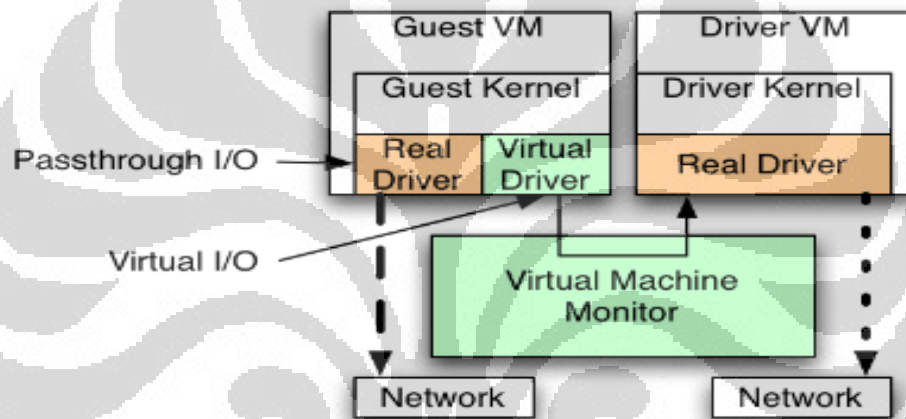
Migrasi isi memori adalah hal yang sedikit rumit mengingat *virtual machine* pada *host* asal masih berjalan dan membuat modifikasi pada status memori. Idanya adalah untuk melakukan penyalinan isi memori berulang kali, dan mengirim hanya "*delta*" perubahan ke *host* tujuan. Ada saat, ketika hanya sedikit "*delta*" memori yang perlu disalin. Pada tahap ini *virtual machine* pada *host* sumber akan dijeda, memori *delta* tersebut disalin, dan *virtual machine* dilanjutkan pada *host* tujuan. Jeda singkat inilah yang menyebabkan *downtime*.

Migrasi isi penyimpanan mirip dengan memori, tetapi akan memerlukan lebih banyak waktu dan migrasi dapat berlangsung pada urutan menit. Mungkin tidak mudah untuk menjamin *downtime* yang kecil kedua dengan migrasi penyimpanan. Semua produk komersial saat ini mengesampingkan masalah ini dengan menggunakan penyimpanan terpusat (misalnya NFS, iSCSI, *Fibre Channel* berdasarkan SAN) yang menempatkan *image virtual machine*. Penyimpanan konten tidak harus bermigrasi jika kedua sumber dan target *node* yang terhubung ke penyimpanan terpusat.

Migrasi koneksi jaringan merupakan hal yang cukup sederhana, jika berasumsi bahwa semua *host* berada dalam *subnet* IP yang sama. Ketika VM yang bermigrasi ke *node* target, VM hanya harus mengirim *broadcast Address Resolution Protocol* (ARP) mengatakan bahwa alamat IP telah dipindahkan ke

lokasi fisik baru (alamat MAC). Karena ini terjadi pada hubungan antara Layer 2 dan Layer 3 dari *stack* jaringan dan koneksi TCP mempertahankan migrasi. Akibatnya, aplikasi melihat tidak ada gangguan pada koneksi jaringan. Namun, pendekatan ini tidak bekerja jika VM harus menyeberang *subnet*. Keberhasilan dan popularitas *live migration* terletak pada kenyataan bahwa memiliki *downtime* sangat kecil.

2.6.2 Virtualisasi I/O



Gambar 2.2 Virtualisasi I/O Dalam *Virtual Machine* [6]

Monitor *virtual machine* dan *hypervisor* harus menyediakan akses *guest virtual machine* untuk mengakses *hardware*. Dengan *virtual I/O*, operasi di *virtual machine* dihentikan oleh VMM dan diselenggarakan oleh *hypervisor*, sistem operasi *host*, atau eksekusi *driver* dalam *privileged virtual machine*. Kinerja *virtual I/O* lebih rendah dari *I/O* pada perangkat keras fisik karena *hypervisor* harus memperantarai lebih dahulu semua permintaan *I/O*. Gambar 2.2 menunjukkan contoh sistem virtualisasi dengan dua kartu jaringan meninggalkan akses melalui *I/O* dan dapat diakses melalui *virtual I/O* menggunakan *driver* dalam *driver virtual machine*.

2.6.3 Manfaat *Live Migration*

Salah satu kasus penggunaan utama untuk *live migration* adalah untuk pengelolaan sumber daya dalam komputasi awan. Sebagai contoh, penyedia komputasi awan seperti Amazon EC2 memiliki ribuan *virtual machine* berjalan di pusat datanya. Untuk menghemat energi, biaya, dan untuk *load balancing* mereka dapat menggerakkan *virtual machine* menggunakan *live migration*, tanpa mengganggu aplikasi pelanggan mereka yang berjalan di *virtual machine*.

2.7 Penggunaan DRBD Sebagai Sistem Sinkronisasi Antar Host. [7]

Seperti yang telah dijelaskan sebelumnya *live migration* adalah perpindahan mesin *virtual* antar dua *node* yang berbeda fisik. Untuk mendukung hal tersebut dibutuhkan sinkronisasi data antar kedua *node* atau terdapat *sharing storage* yang dapat digunakan oleh kedua *node* agar *live migration* dapat berjalan dengan baik. Ada berbagai jenis teknik sinkronisasi antar *node* dan *sharing storage* yang dapat digunakan antara lain *Distributed Replicated Block Device* (DRBD) pada sinkronisasi antar *node* atau *ISCSI* dan *NFS* pada *sharing storage*.

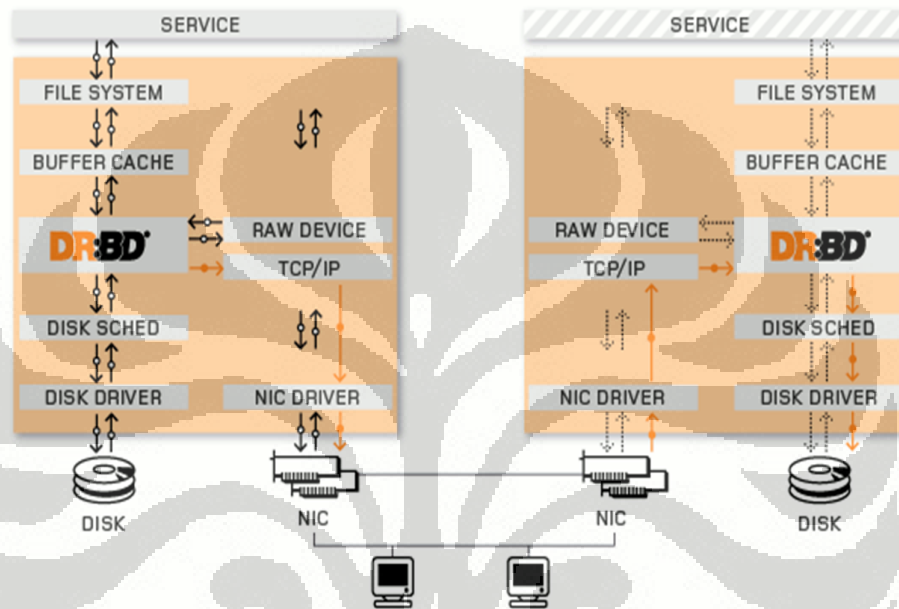
2.7.1 Pengertian DRBD

Distributed Replicated Block Device (DRBD) adalah sistem distribusi penyimpanan untuk Linux.

DRBD *mirroring* data

- *Real Time*: Replikasi data terjadi terus menerus antar *node*
- *Transparan*: Aplikasi – aplikasi yang menyimpan data di *mirroring* devais akan dapat diakses.
- *Synchronously* atau *Asynchronously*: Ada dua tipe pilihan *mirroring* data secara bersamaan antar *node* atau berkelanjutan.

DRBD mengacu kepada blok devais yang didesain sebagai bangunan blok untuk *high availability* kluster dengan cara melakukan *mirroring* (sinkronisasi antar node) seluruh blok devais melalui jaringan seperti yang ditunjukkan pada Gambar 2.3.



Gambar 2.3. Sistem DRBD [7]

2.7.2 Kerja DRBD

DRBD bekerja pada blok devais seperti partisi *hard disk* atau LVM (*logical volume*) dengan melakukan *mirroring* setiap blok data yang dituliskan ke disk *node* lain.

Ada dua tipe mirroring pada DRBD, yaitu *fully synchronous* dan *asynchronous*.

2.7.2.1 Fully Synchronous

Mirroring dapat dilakukan dengan sinkronisasi data pada kedua *node*. Hal tersebut berarti sistem *file* berada pada aktif *node* yang akan menulis setiap

transaksi pada blok yang akan selesai bila keduanya telah sama dan akan kembali aktif menuliskannya bila terjadi transaksi kembali. Protokol C digunakan pada tipe *mirroring* ini.

2.7.2.2 *Asynchronous*

Tipe ini biasanya dilakukan bila *node* berada pada jarak yang jauh dibanding *node* pasangannya. Hal ini berarti setiap *entity* pada blok devais menginformasikan permintaan untuk melakukan penulisan secepatnya pada *node* berikutnya setelah data telah dituliskan pada *node* lokal. Secara singkat dapat dikatakan metode ini melakukan penulisan data pada *node* pasangannya setelah *node* lokal telah dituliskan (tidak bersamaan) sedangkan pada *synchronous* kedua *node* langsung terjadi penulisan pada blok devais secara bersamaan.

2.7.3 Aksesibilitas data hanya di aktif node

Konsekuensi dari *mirroring* data pada blok devais adalah data dapat diakses hanya pada *node* yang aktif (akses melalui sistem *file*) hanya pada *node* yang aktif.

Ada beberapa cara untuk mengakses data pada *node* kedua (pasangannya):

- Menggunakan DRBD pada *logical volume* dan menggunakan kemampuan LVM
- Menggunakan mode *primary* dengan sistem *file* yang terdistribusi (GFS, OCFS2)

2.7.4 Fitur DRBD

Ada beberapa fitur penting DRBD yang dapat digunakan oleh pengguna

- *Single – Primary Mode*
- *Dual – Primary Mode*
- *Replication Mode*
- *Efficient Synchronous*

2.7.4.1 Single – Primary Mode

Pada fitur ini setiap sumber daya ,setiap saat, hanya bisa dimanipulasi pada satu kluster yang menjadi *primary*. Mode ini menggunakan sistem *file* konvensional seperti ext3 dan ext4.

2.7.4.2 Dual-Primary Mode

Pada fitur ini kedua node menjadi *primary* sehingga terjadinya sinkronisasi data bersamaan dapat terjadi. Mode ini memerlukan kluster sistem *file* yang akan memakai *lock manager* seperti GFS dan OCFS2. *Dual-Primary mode* diperuntukkan agar terjadi penyeimbangan beban pada kluster yang membutuhkan konkurensi akses data pada kedua *node*.

2.7.4.3 Efficient Synchronization

Sinkronisasi dibutuhkan jika koneksi replikasi node terputus karena berbagai kendala. Efisiensi sinkronisasi adalah DRBD tidak akan melakukan sinkronisasi blok yang terjadi perubahan data baru apabila tertulis secara asli. Sebuah *node* dengan data yang tidak konsisten secara keseluruhan tidak dapat terjadi operasi dengan begitu untuk menjaga periode waktu sinkronisasi selama *node* tersebut masih tidak konsisten sesingkat-singkat mungkin.

Perkiraan waktu sinkronisasi dapat menggunakan rumus sebagai berikut

$$t_{sync} = \frac{D}{R}$$

t_{sync} adalah waktu perkiraan sinkronisasi

D adalah jumlah data yang akan di sinkronisasi

R adalah kecepatan sinkronisasi data

Efisiensi sinkronisasi DRBD dapat ditingkatkan menggunakan *data digest* yang disebut juga *checksums*. Ketika menggunakan *checksums*, DRBD akan membaca blok devais terlebih dahulu sebelum melakukan sinkronisasi pada kedua *node* kemudian menghitung *hash* yang terdapat pada *disk*. Selanjutnya akan terjadi komparasi antar *hash* kedua *node* dan akan menulis data kembali apabila *hash* tersebut tidak cocok. Hal ini dapat mempersingkat waktu sinkronisasi khususnya bila koneksi pasangan *node* sering terputus.

2.7.5 Konfigurasi DRBD

Ada beberapa persiapan penting agar DRBD dapat berjalan setelah proses instalasi

- Konfigurasi tempat penyimpanan untuk sumber daya DRBD
- Konfigurasi jaringan
- Konfigurasi sumber daya

2.7.5.1 Konfigurasi tempat penyimpanan sumber daya DRBD

Sebagai tempat penyimpanan sumber daya DRBD dapat dipilih pada beberapa tipe blok devais yang biasa terdapat di sistem seperti partisi *hard drive* biasa, perangkat lunak devais RAID, dan LVM

2.7.5.2 Konfigurasi Jaringan

Secara konvensional DRBD menggunakan port TCP dari 7788 hingga 7790. DRBD menggunakan dua koneksi TCP untuk setiap sumber daya yang dikonfigurasi. Selain memastikan *firewall* untuk mengizinkan koneksi ini, TCP yang akan digunakan harus dipastikan bahwa tidak digunakan oleh aplikasi lainnya.

2.7.5.3 Konfigurasi Sumber Daya

Semua aspek DRBD dikontrol dalam konfigurasi file, */etc/drbd.conf*. Secara awal konfigurasi *file* ini berupa

```
include "/etc/drbd.d/global_common.conf";
include "/etc/drbd.d/*.res";
```

Namun, dua buah baris ini tidak cukup untuk melakukan konfigurasi DRBD. Oleh karena itu, konfigurasi *file* pada */etc/drbr.conf* harus diubah dan disesuaikan sesuai kebutuhan yang diinginkan seperti yang ditunjukkan pada Gambar 2.4.

```
global {
  usage-count yes;
}
common {
  protocol C;
}
resource r0 {
  on alice {
    device /dev/drbd1;
    disk /dev/sda7;
    address 10.1.1.31:7789;
    meta-disk internal;
  }
  on bob {
    device /dev/drbd1;
    disk /dev/sda7;
    address 10.1.1.32:7789;
    meta-disk internal;
  }
}
```

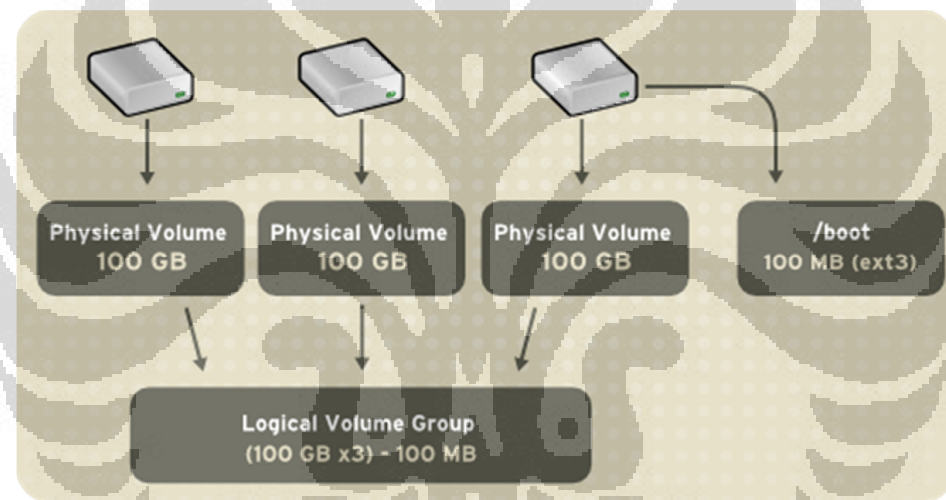
Gambar 2.4 Contoh Konfigurasi DRBD

2.8 Logical Volume Manager (LVM) dan Redundant Array of Independent Disks (RAID)

Ada dua tipe pendukung penyimpanan pada Linux yaitu RAID dan LVM yang menawarkan ketangguhan dan fleksibilitas.

2.8.1 Logical Volume Manager (LVM) [8]

LVM adalah sebuah perangkat lunak bantu untuk manajemen *logical volume* (tipe penyimpanan *volume* yang bukan sebenarnya seperti partisi namun secara logikal) termasuk melakukan alokasi *disk*, *mirroring*, dan pengubahan ukuran *logical volumes*. Sebuah *hard drive* atau sejumlah *hard drive* dapat di alokasikan untuk satu atau lebih fisik *volume*. Fisikal *volume* LVM dapat dikombinasikan dalam *logical volume* dengan pengecualian partisi */boot*. Partisi */boot* tidak dapat menjadi sebuah *logical volume* karena *boot loader* tidak akan dapat membacanya.



Gambar 2.5 Logical Volume Group

Kelebihan dari LVM ini sendiri karena merupakan *logical volume* maka dapat melakukan penciptaan ukuran *hard disk* yang secara fisik melebihi ukuran aslinya seperti pada Gambar 2.5.

2.8.1.1 Konfigurasi LVM

LVM dapat dikonfigurasi melalui terminal linux atau secara grafikal. Ada beberapa langkah yang dibutuhkan dalam melakukan konfigurasi dan dapat menggunakan perintah *system-config-lvm* untuk melakukan konfigurasi lebih lanjut

- Buat *physical volume* dari *hard disk*
- Buat *volume group* dari *physical volume*
- Ubah tipe *volume* menjadi *lvm2*
- Buat *logical volume* dari *volume group* tersebut.

2.8.2 Redundant Array of Independent Disk (RAID) [9]

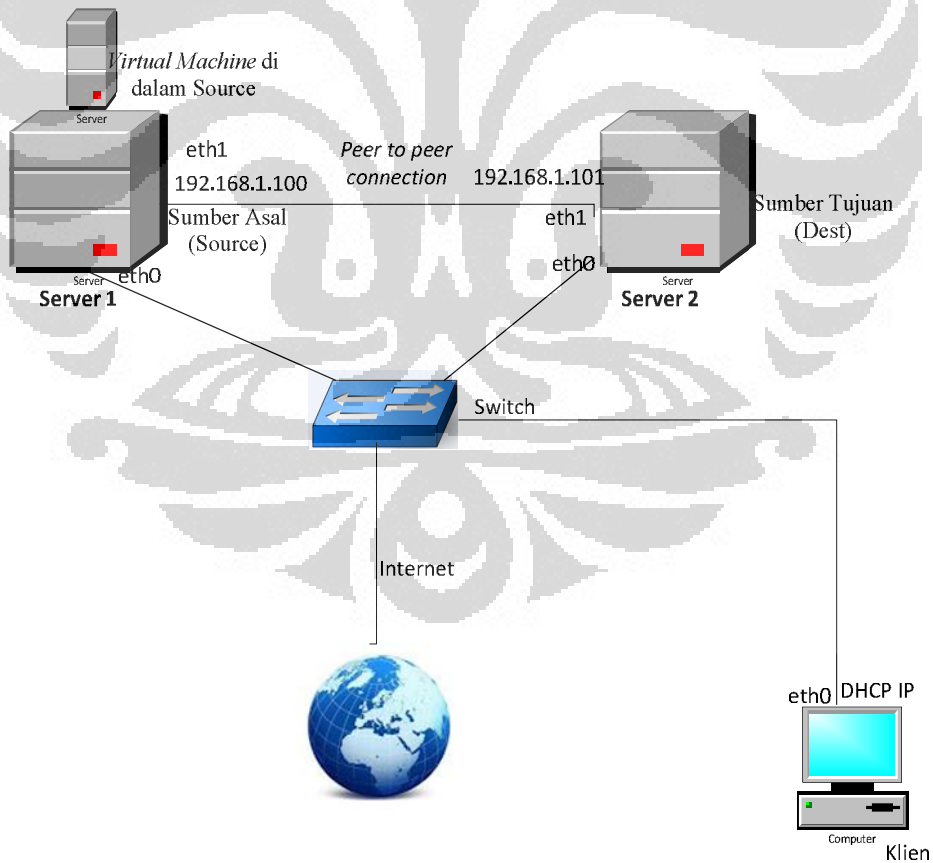
Konsep dari RAID adalah mengkombinasikan *disk drives* menjadi deretan (*array*) *disk* untuk menciptakan kualitas yang lebih baik dibandingkan sebuah *disk drives* yang besar. Deretan *drives* akan terlihat sebagai sebuah *drive*. Salah satu kelebihan dari deretan *disk* ini adalah dapat dibuatnya menjadi *fault-tolerant* (secara singkat dapat diartikan sebagai pencegah kesalahan dalam hal ini kehilangan data dalam *disk drive*). Jika salah satu *hard drive* rusak data masih dapat disimpan pada *drive* yang masih sehat menggunakan *re-created* lokasi data (salah satu fitur RAID) pada *drive*.

Bagaimanapun porsi data yang terdapat pada *drive* yang rusak tersebut harus segera dilakukan *re-created* dari *drive* yang masih sehat. Oleh karena itu, jika terjadi *drive* rusak maka harus segera diganti segera. Namun saat *drive* gagal untuk kedua kalinya ,jika tidak terdapat *drive* lainnya, maka kehilangan data tidak dapat terhindarkan. Ada mode RAID yang dapat mengatasi hal ini yaitu “*Hot – Spare*” membuat layer perlindungan tambahan.

BAB III
PERANCANGAN *LIVE MIGRATION* PADA XEN SERVER DENGAN
PEER-TO-PEER NETWORK

3.1 Topologi Jaringan

Skripsi ini menggunakan jaringan berskala kecil yang digunakan sebagai uji coba. Topologi *live migration* ini terdiri dari dua PC yang masing-masing memiliki konfigurasi yang sama dan satu laptop sebagai klien. Dua PC dan satu laptop ini akan terhubung dengan *switch* yang akan menghubungkannya ke internet agar kedua *server* dan *virtual machine* mendapatkan akses internet. Secara lebih lengkap rancangan topologi yang akan dibuat sebagai skripsi ini seperti pada Gambar 3.1.

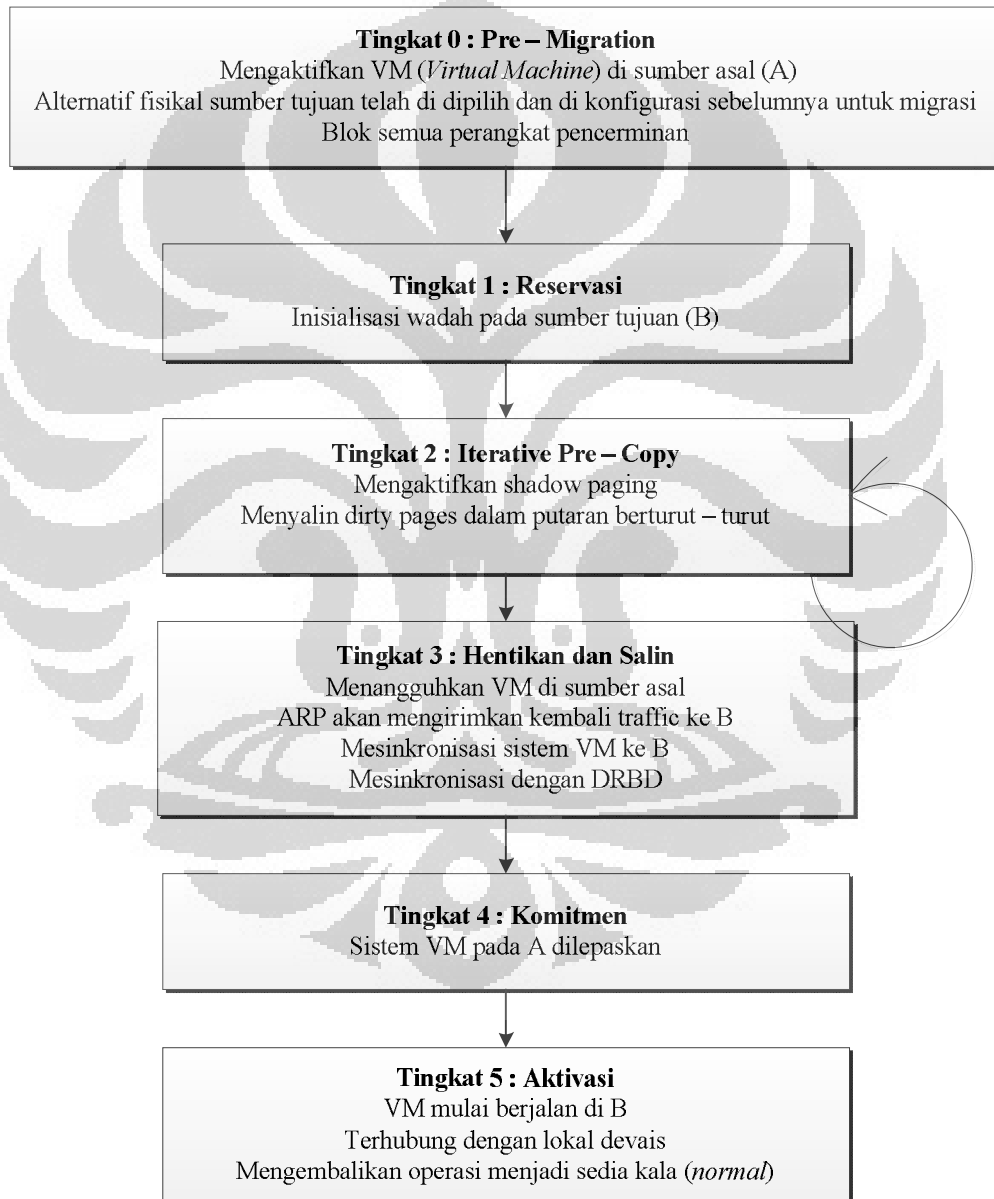


Gambar 3.1 Topologi Jaringan *Live Migration*

3.2 Algoritma Rancangan

Pada perancangan ini menggunakan metode *Pre-Copy* yang dapat menghasilkan waktu *downtime* yang kecil.

3.2.1 Metode *Pre-Copy*



Gambar 3.2 Algoritma *Pre-Copy*

Untuk mengetahui proses transaksi perpindahan antara A dan B dengan metode *Pre-Copy* ini dapat dilihat pada Gambar 3.2 diatas yang terbagi atas beberapa tingkatan hingga akhirnya terjadi migrasi VM dari A ke B.

Tingkat 0 : Merupakan tingkatan paling awal atau inisialisasi dimana pada tingkat ini VM telah diaktifkan pada fisik *host* A dan telah ditentukan *host* target B sebagai tempat migrasi dari A.

Tingkat 1 : Merupakan proses inisialisasi pada *host* B sebagai wadah berikutnya. *Host* B akan dilakukan penyiapan semua sumber daya yang sesuai untuk sebagai wadah VM dari A agar tidak terjadi kegagalan saat proses migrasi ,misalnya, hanya karena kekurangan jumlah memori yang seharusnya disediakan untuk menampung VM dari A.

Tingkat 2 : Merupakan proses *iteration* yang berarti pengulangan proses untuk mencapai tujuan tertentu. Pada *iteration* pertama semua halaman A di salin ke B dan pada proses *iteration* selanjutnya menyalin *dirty pages* terus menerus sesuai *update* dari A.

Tingkat 3 : Merupakan proses *suspend* penghentian VM di A serta pengalihan *traffic* ke B. Status CPU , *memory pages* yang tersisa disalinkan, dan sinkronisasi dengan DRBD. Saat proses inilah terjadinya penghentian sementara *downtime* pada aplikasi di A. Namun, masih di tingkatan ini *host* A masih menjadi *host* utama untuk VM meskipun telah menyalin sumber daya dan statusnya ke B.

Tingkat 4: Merupakan proses komitmen perpindahan. *Host* B mengirimkan pemberitahuan kepada A bahwa telah menerima dengan baik *OS image* secara sempurna.

Tingkat 5 : Merupakan proses aktivasi *host* B sebagai fisik baru bagi VM. *Host* B telah diaktifkan dan menjadi *host* utama VM yang telah sukses berpindah dari A

Dengan pengertian tingkat – tingkat tersebut dapat pula di ketahui empat *timeline* dimana proses tingkatan berlangsung pad metode ini

1. VM masih berjalan normal di A : Tingkat 0 dan Tingkat 1

2. Penyalain *OS Image* dalam *pages* yang berlangsung berkali – kali :
Tingkat 2
3. VM terhenti terjadinya *downtime* : Tingkat 3 dan Tingkat 4
4. VM telah berpindah dan berjalan normal di B : Tingkat 5

3.3 Rancangan Perangkat Keras dan Perangkat Lunak

3.3.1 Rancangan Perangkat Keras

Pada perancangan ini digunakan dua PC sebagai *server* seperti pada Gambar 3.1, satu laptop sebagai klien, satu switch *Ethernet* 100 Mbps, dua USB LAN sebagai penghubung ke *intefaces Ethernet Card* 1 (*eth1*), satu kabel *cross* dan empat kabel UTP.

Tabel 3.1 Rancangan Perangkat Keras Untuk *Live Migration*

	CPU	Memory (MB)	HDD (GB)
Source	Intel Dual Core	2048	160
Dest	Intel Dual Core	2048	160
<i>Virtual Machine</i>	Intel Dual Core	1024	8
<i>Klien</i>	Intel Core i3	2048	320

3.3.2 Rancangan Perangkat Lunak

Perangkat lunak yang digunakan pada perancangan ini adalah

Tabel 3.2 Rancangan Perangkat Lunak Untuk *Live Migration*

	OS	Perangkat Lunak Tambahan
Source	Ubuntu 11.10	XEN 4.1, lvm2, DRBD
Dest	Ubuntu 11.10	XEN 4.1, lvm2, DRBD
<i>Virtual Machine</i>	Ubuntu 10.10	VSFTPD, FTP
Klien	Ubuntu 11.10	

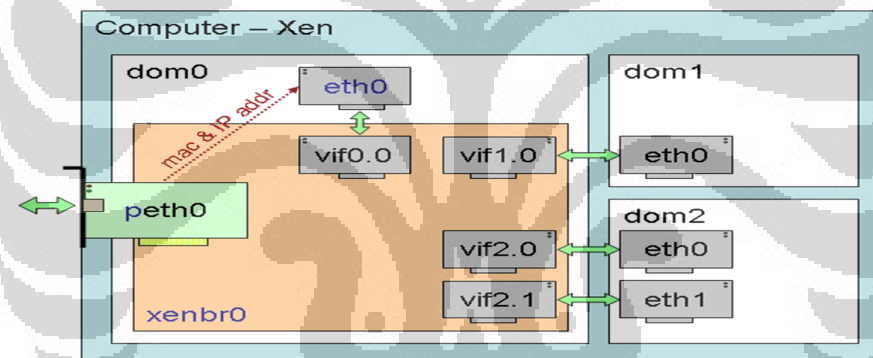
1. Sistem operasi yang digunakan pada kedua *server* dan klien adalah Ubuntu 11.10 dengan kernel 3.0.1 generic.
2. Xen adalah *open source virtual machine monitor*, dikembangkan di University of Cambridge. Dibuat dengan tujuan untuk menjalankan sampai dengan seratus sistem operasi ber-fitur penuh (*full featured OS*) di hanya satu komputer. Virtualisasi Xen menggunakan teknologi paravirtualisasi menyediakan isolasi yang aman, pengatur sumber daya, garansi untuk *quality-of-services* dan *live migration* untuk sebuah mesin virtual. Xen *Hypervisor* yang digunakan pada kedua host adalah Xen 4.1.0
3. LVM2 adalah *tool* yang akan menyediakan fasilitas manajemen *logical volume* penyimpanan di linux.
4. DRBD (*Distributed Replicated Block Device*) adalah aplikasi yang dapat digunakan sebagai solusi *cluster mirroring* dari sebuah blok devais seperti *hard disk*, partisi, dan *logical volume* antar server. DRBD yang digunakan pada kedua *host* adalah drbd8.

3.4 Konfigurasi Jaringan

Agar DomU sebagai *virtual machine* dapat diakses selama perpindahan maka diperlukan konfigurasi jaringan Xen untuk mendukung relokasi IP dan konfigurasi IP statik.

3.4.1 Konfigurasi Jaringan Xen

Pada konfigurasi ini diperlukan pengaktifan Xen bridge yang akan menjembatani *traffic* dari DomU (*Guest*) ke mesin fisik (server) dan sebaliknya. Caranya adalah mengaktifkan fungsi (*network script network bridge*) dan (*vif-script vif bridge*). Sebagai ilustrasi terdapat pada Gambar 3.3 ini



Gambar 3.3 Konfigurasi Jaringan Xen [10]

Sebagai keterangan vif1.0 adalah *virtual interfaces* yang menghubungkan eth0 dari *virtual machine* (dom1) tersebut sementara xenbr0 yang akan saling menghubungkan *virtual interfaces* dan eth0 pada host dengan peth0 yang akan menghubungkan ke internet.

3.4.2 Konfigurasi IP Statik

Pada konfigurasi ini akan diberikan IP statik dalam satu subnet pada setiap mesin fisik dan interfaces. Sementara virtual machine akan mendapatkan IP

192.168.102.50 yang tetap berada pada satu *subnet* agar dapat tetap saling terhubung. IP privat lainnya digunakan pada jaringan peer-to-peer di *Ethernet Card 1* (eth1) yang berada diluar subnet pada jaringan *Ethernet Card 0* (eth0) agar tidak terjadi kebingungan di jaringan. Sementara klien tidak menggunakan IP statik melainkan DHCP (*Domain Host Configuration Protocol*) karena mesin klien akan mendapatkan otomatis IP dari *server* yang secara otomatis akan berada pada satu *subnet*. Konfigurasinya akan terlihat seperti Tabel 3.3

Tabel 3.3 Konfigurasi IP Statik

	Ethernet Card 0	Ethernet Card 1
Server 1	192.168.102.61	192.168.1.100
Server 2	192.168.102.79	192.168.1.101

3.5 Aplikasi Uji

Aplikasi uji ini merupakan aplikasi yang diimplementasikan pada setiap host target perpindahan *virtual machine* baik dari *node* satu ke *node* dua atau sebaliknya

1. Wireshark adalah satu dari sekian banyak *tool Network Analyzer* yang dipakai untuk menganalisa paket jaringan, pengembangan protokol jaringan serta edukasi bagi yang ingin memperdalam ilmunya dalam jaringan komputer. Kelebihan bagi wireshark adalah lisensi nya yang *free* alias *open source*. Selain itu Wireshark juga dibuat dengan berbasiskan GUI yang cukup baik dan bagus.
2. VisualVM adalah visual tool yang mengintegrasikan beberapa *command line JDK tools* dan *light weight profiling capabilities*. Didesain untuk digunakan pada saat *production* dan *development*, serta meningkatkan lebih lanjut kemampuan dalam monitoring dan analisa untuk *platform Java SE*. VisualVM adalah alat yang memberikan informasi detail tentang aplikasi Java saat (aplikasi tersebut) sedang dijalankan. Dengan GUI (*Graphical User Interface*) yang intuitif memungkinkan kita untuk dengan

mudah melihat informasi mengenai beberapa aplikasi Java yang sedang dijalankan.

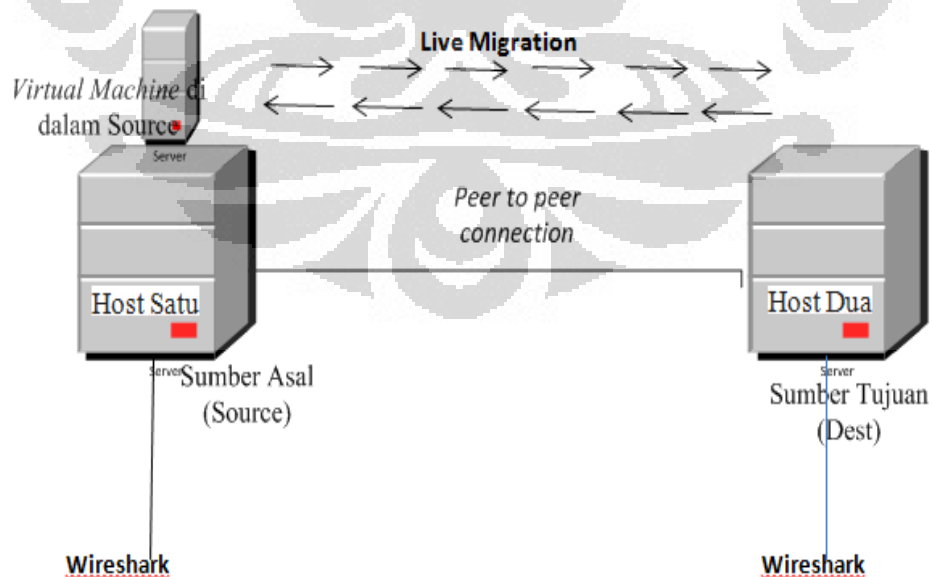
3.4 Skenario Uji Coba

Pada perancangan ini akan dianalisa dan pengukuran bagaimana performa Xen *live migration* pada *peer-to-peer network* dengan dua skenario ataupun penggabungan dari dua skenario tersebut. Analisa akan dilakukan dengan hasil dari pengukuran *throughput*, *delay*, dan paket yang hilang saat *ping test* terhadap dilakukannya perubahan data serta analisa beban CPU selama migrasi.

3.4.1 Perubahan Data *Virtual Machine*

Perubahan data virtual machine dilakukan dengan cara melakukan *update* pada Ubuntu 10.10 yang merupakan sistem operasi dari *virtual machine* tersebut. Hal itu dilakukan dengan mengetikkan *apt-get update* di terminal *virtual machine*.

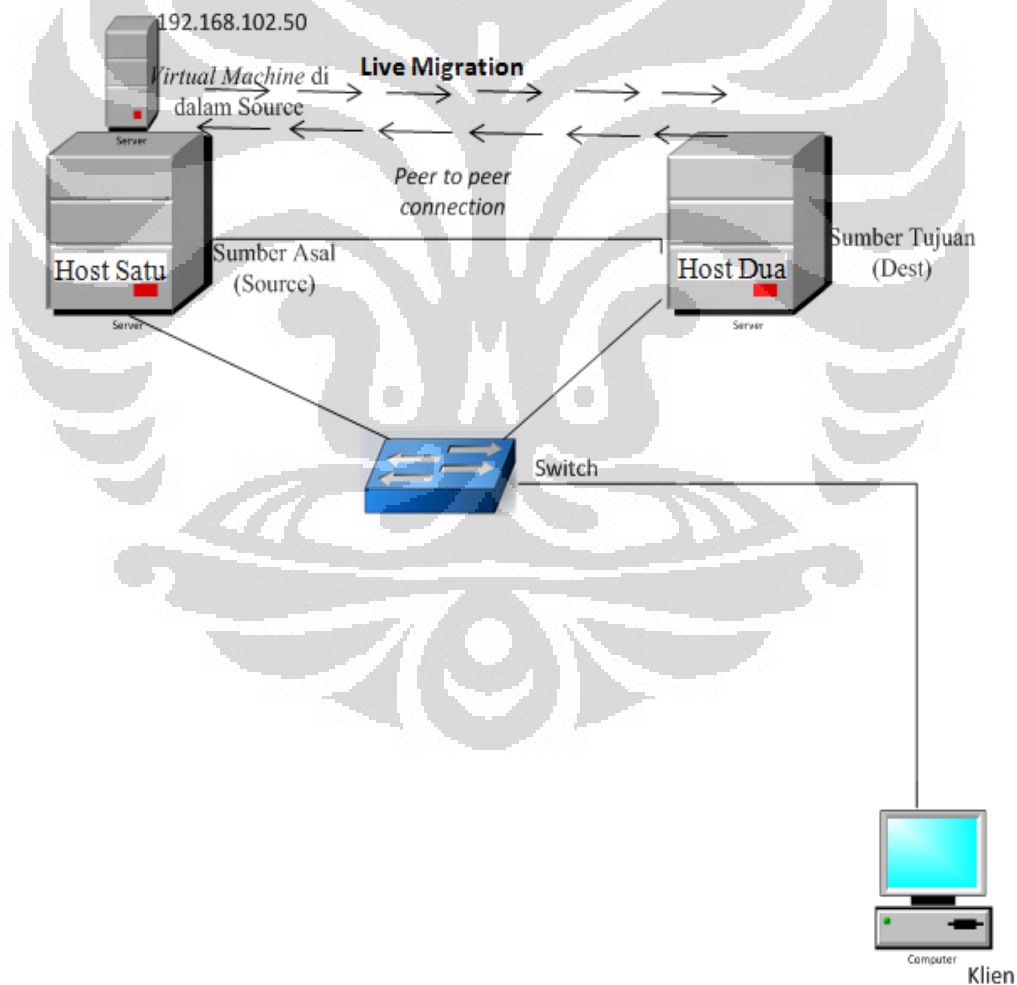
3.4.2 Skenario Satu



Gambar 3.6. Skenario 1

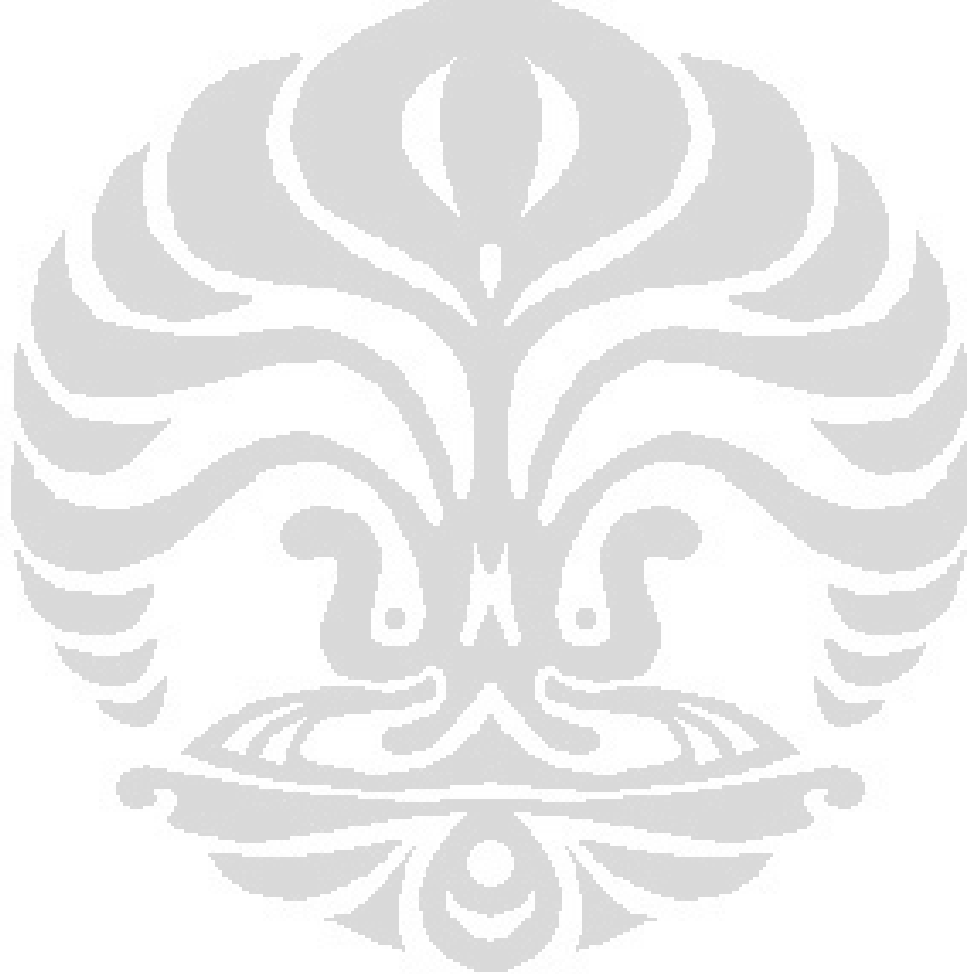
Pada skenario satu ini *virtual machine* akan berpindah dari *host* satu ke *host* dua dan begitupun sebaliknya. Setiap perpindahan dari *host* satu ke *host* dua atau *host* dua ke *host* satu maka aplikasi wireshark akan meng-*capture throughput* dan *delay* di mesin fisikal tujuan. Seperti yang ditunjukkan pada Gambar 3.6 bila *virtual machine* akan berpindah dari *host* satu ke *host* dua maka aplikasi wireshark akan melakukan *capturing* di sisi dest sebagai tujuan dari perpindahan.

3.4.3 Skenario Dua



Gambar 3.7 Skenario 2

Pada skenario satu ini seperti Gambar 3.7 *virtual machine* akan berpindah dari *host* satu ke *host* dua dan begitupun *host* dua ke *host* satu. Saat perpindahan *virtual machine*, komputer klien akan melakukan ping IP 192.168.102.50 terus-menerus hingga *virtual machine* dapat berjalan normal di *host* tujuannya. Nilai paket yang hilang akan di-*capture* oleh komputer klien. Pada skenario ini juga akan dihitung beban CPU yang digunakan saat perpindahan *virtual machine*.



BAB 4

UJICOBAN DAN ANALISA DATA KUALITAS PERPINDAHAN *VIRTUAL MACHINE* PADA JARINGAN PEER TO PEER MENGGUNAKAN XEN

4.1 IMPLEMENTASI DAN KONFIGURASI SISTEM

Tahap–tahap pembuatan sistem untuk mendukung adanya *live migration virtual machine* dilakukan pada kedua *host* dengan beberapa proses instalasi dan implementasi.

4.1.1 Instalasi Ubuntu 11.10

Instalasi Ubuntu *desktop* pada kedua *host* menggunakan *live CD* Ubuntu 11.10 dengan memori swap 4 Gb dan daya tampung Ubuntu sebesar 100 Gb.

4.1.1.1 Konfigurasi Ubuntu

Konfigurasi *Repository* sebagai mirroring lokal di Universitas Indonesia dengan perintah `gedit /etc/apt/source.list` pada terminal agar prosesnya proses update dan instalasi dapat berjalan lebih cepat. *Repository* yang digunakan secara lokal melalui `kambing.ui.ac.id`.

4.1.1.2 Konfigurasi Jaringan Ubuntu

Konfigurasi jaringan pada kedua *host* perlu dilakukan agar kedua *host* dapat saling terhubung dan dapat menjalin komunikasi antar keduanya dengan perintah `gedit /etc/network/interfaces` di terminal. Kedua *host* menggunakan interfaces Ethernet card 0 (`eth0`) sebagai interfaces yang akan menghubungkan *host* ke internet melalui switch dan Ethernet card 1 (`eth1`) sebagai interfaces yang akan menghubungkan *host* satu ke *host* dan sebaliknya melalui USB LAN. Konfigurasi IP (*Internet Protocol*) yang digunakan kedua interfaces berupa IP statik agar tidak terjadi perubahan meski dilakukan *restart* pada sistem. Pemberian IP pada `eth0` akan berada pada satu subnet sedangkan `eth1` harus berada diluar subnet `eth0` agar tidak terjadi intereferensi antara `eth0` dan `eth1`. Hasilnya konfigurasi pada kedua *host* dapat dilihat pada Gambar 4.1.

<p><i>Host 1:</i></p> <pre># Interfaces <u>ethernet</u> card 0 auto <u>eth0</u> #<u>Otomatis</u> <u>mengaktifkan</u> <u>eth0</u> <u>iface</u> <u>eth0</u> <u>inet</u> <u>static</u> #<u>Konfigurasi</u> <u>statik</u> <u>IP</u> <u>address</u> <u>192.168.102.61</u> <u>netmask</u> <u>255.255.255.0</u> <u>network</u> <u>192.168.102.0</u> <u>broadcast</u> <u>192.168.102.255</u> <u>gateway</u> <u>192.168.102.1</u></pre> <p>Berada pada subnet yang sama</p>	<p><i>Host 2:</i></p> <pre># Interfaces <u>Ethernet</u> card 0 auto <u>eth0</u> #<u>Otomatis</u> <u>mengaktifkan</u> <u>eth0</u> <u>iface</u> <u>eth0</u> <u>inet</u> <u>static</u> #<u>Konfigurasi</u> <u>statik</u> <u>IP</u> <u>address</u> <u>192.168.102.79</u> <u>netmask</u> <u>255.255.255.0</u> <u>network</u> <u>192.168.102.0</u> <u>broadcast</u> <u>192.168.102.255</u> <u>gateway</u> <u>192.168.102.1</u></pre>
<pre># Interfaces <u>ethernet</u> card 1 auto <u>eth1</u> #<u>Otomatis</u> <u>mengaktifkan</u> <u>eth1</u> <u>iface</u> <u>eth1</u> <u>inet</u> <u>static</u> #<u>Konfigurasi</u> <u>statik</u> <u>IP</u> <u>address</u> <u>192.168.1.100</u> <u>netmask</u> <u>255.255.255.0</u> <u>network</u> <u>192.168.1.0</u> <u>broadcast</u> <u>192.168.1.255</u></pre> <p>Berada pada subnet yang sama</p>	<pre># Interfaces <u>Ethernet</u> card 1 auto <u>eth1</u> #<u>Otomatis</u> <u>mengaktifkan</u> <u>eth1</u> <u>iface</u> <u>eth1</u> <u>inet</u> <u>static</u> #<u>Konfigurasi</u> <u>statik</u> <u>IP</u> <u>address</u> <u>192.168.1.101</u> <u>netmask</u> <u>255.255.255.0</u> <u>network</u> <u>192.168.1.0</u> <u>broadcast</u> <u>192.168.1.255</u></pre>

Gambar 4.1 Konfigurasi IP Pada Host

4.1.1.3 Konfigurasi Nama *Host*

Setelah IP telah diberikan pada kedua host akan dilakukan proses inialisasi nama atas IP – IP yang telah diberikan pada setiap interfaces yang ada pada kedua host untuk memudahkan proses live migration sehingga tidak diperlukan menuliskan IP hanya perlu menuliskan nama inialisasi tersebut saja. Hal ini dapat dilakukan dengan perintah gedit /etc/hosts di terminal. Hasilnya dapat dilihat pada Gambar 4.2.

```

127.0.0.1      localhost
192.168.102.61 source.domain.local source
#Interfaces eth0 pada host 1 dengan nama source
192.168.102.79 dest.domain.local dest
#Interfaces eth0 pada host 2 dengan nama dest
192.168.1.100 sourceX
#Intefaces eth1 pada host 1 dengan nama sourceX untuk membedakan dengan eth0
192.168.1.101 destX
#Interfaces eth1 pada host 2 dengan nama destX untuk membedakan dengan eth0

```

Gambar 4.2 Konfigurasi Host

4.1.2 Instalasi XEN 4.1

Instalasi *hypervisor* Xen 4.1 dengan arsitektur inter processor menggunakan terminal dengan perintah `sudo apt-get install xen-hypervisor-4.1-i386`. Kemudian dilanjutkan dengan instalasi *image* linux untuk meng-update gnome pada linux sehingga akan terdapat Ubuntu dengan kernel Xen. Instalasi dapat dilakukan dengan perintah `apt-get install linux-image-server` di terminal. Setelah proses instalasi selesai lakukan *reboot* untuk masuk ke Ubuntu dengan kernel Xen yang ada di dalamnya agar dapat menggunakan *Xen Live Migration*.

4.1.3 Instalasi GParted

GParted adalah program aplikasi untuk melakukan partisi pada *hard disk*. Instalasi dilakukan dengan perintah `sudo apt-get install GParted` di terminal. Instalasi Gparted diperlukan untuk mendapatkan sebuah partisi *hard disk* baru yang akan digunakan sebagai penyimpanan *disk* dan *swap virtual machine* dalam format LVM. Partisi dilakukan pada kedua host sehingga didapatkan sebuah partisi baru `/dev/sda7` pada *host* satu sebesar 25 Gb dan partisi `/dev/sda9` pada *host* dua sebesar 24 Gb.

4.1.4 Instalasi LVM

LVM adalah sebuah program bantu untuk manajemen *logical volume* termasuk melakukan alokasi disk yang akan digunakan untuk alokasi *disk* dan *swap virtual machine* dan *mirroring* yang akan digunakan untuk sinkronisasi antar *host*. Instalasi LVM dilakukan perintah `apt-get install lvm2` di terminal.

4.1.4.1 Konfigurasi LVM

Setelah proses instalasi LVM selesai, diperlukan pembuatan volume group sebagai wadah penyimpanan *virtual machine*. Pembuatan *volume group* dengan nama “vg” pada `/dev/sda7` di *host* satu dan `/dev/sda9` di *host* dua dilakukan dengan dua perintah dibawah ini

- Pembuatan volume group dengan nama “vg” di *host* satu
`pvcreate /dev/sda7`
`vgcreate vg /dev/sda7`
- Pembuatan volume group dengan nama “vg” di *host* dua
`pvcreate /dev/sda9`
`vgcreate vg /dev/sda9`

4.1.5 Konfigurasi Xen-tools

Konfigurasi pada Xen-tools diperlukan sebagai konfigurasi awal pembentuk *virtual machine* karena di Xen-tools akan dideskripsikan bagaimana *virtual machine* dibentuk, memori apa yang digunakan, ukuran memorinya, tipe sistem file, distro yang digunakan, konfigurasi jaringan yang digunakan, password, dan *mirroring* yang digunakan. Konfigurasi ini akan dilakukan pada kedua *host* dengan perintah `gedit /etc/xen-tools/xen-tools.conf` di terminal. Hasilnya tampak pada Gambar 4.3


```

lvm = vg #nama LVM yang digunakan "vg" seperti yang dijelaskan pada sub bab 4.1.4.1
size = 8Gb #Besar penyimpanan virtual machine dalam bentuk image
memory = 1024Mb #Ukuran memori virtual machine
swap = 384Mb #Ukuran swap memori virtual machine
fs = ext3 #Tipe sistem file pada image
dist = lucid #Distro linux yang digunakan, dalam hal ini digunakan lucid, yaitu Ubuntu 10.10

#Konfigurasi gateway jaringan virtual machine menggunakan gateway host agar dapat
berkomunikasi dengan dunia luar (internet)
gateway = 192.168.102.1
netmask = 255.255.255.0
broadcast = 192.168.1.255

passwd = 1 #Password untuk masuk ke root virtual machine

#Mirroring yang digunakan untuk mengunduh distro lucid langsung ditujukan pada repository
lokal di Universitas Indonesia yaitu kambing.ui.ac.id
mirror_lucid=http://kambing.ui.ac.id/ubuntu

```

Gambar 4.3 Konfigurasi Xen-tools

4.1.6 Instalasi Virtual Machine

Setelah melakukan konfigurasi di Xen-tools, instalasi virtual machine berdasarkan konfigurasi tersebut dapat dilakukan. Instalasi hanya dilakukan pada host satu agar tidak terjadi duplikasi pada kedua host dengan perintah `xen-create-image --hostname=(nama virtual machine yang akan dibuat) --ip=(IP virtual machine yang akan dibuat)` di terminal. Pada hostname diberikan nama VM1 dan pada IP diberikan 192.168.102.50 agar masih berada pada satu *subnet* dengan eth0 kedua *host* sehingga dapat saling terhubung. Hasilnya perintah instalasi virtual machine di terminal menjadi `xen-create-image --hostname=VM1 --ip=192.168.102.50`. Proses instalasi ini berjalan cukup lama tergantung kecepatan jaringan internet pada *host*. Setelah instalasi ini selesai, hasilnya akan tercipta file

VM1.cfg di folder Xen. Hal ini mengindikasikan bahwa instalasi telah berhasil dan untuk membuktikannya *virtual machine* dapat dijalankan pada host satu dengan perintah `xm create VM1.cfg`

4.1.7 Instalasi DRBD

DRBD adalah sistem distribusi penyimpanan untuk Linux. Instalasi ini berfungsi sebagai sistem *mirroring* yang akan digunakan untuk sinkronisasi *disk* dan *swap virtual machine* antar kedua *host* di eth1. Perintah yang digunakan adalah `sudo apt-get install drbd8-utils` di terminal.

4.1.7.1 Konfigurasi DRBD

Instalasi DRBD tidak cukup untuk membuat sinkronisasi dapat berjalan maka diperlukan konfigurasi pada DRBD. Konfigurasi dapat dilakukan pada file `drbd.conf` dengan cara mendeskripsikan *port-port* yang digunakan untuk *disk* dan *swap virtual machine*, meta data, dan protokol yang digunakan. Hasilnya seperti Gambar 4.4.

<pre>resource VM1-disk { <u>protocol C</u>; syncer { } }</pre>	<p>Konfigurasi pada sumber daya disk VM1 dengan protocol C yang merupakan protokol untuk sinkronisasi pada DRBD</p>	<pre>resource VM1-swap { <u>protocol C</u>; syncer { } }</pre>	<p>Konfigurasi pada sumber daya swap VM1 dengan protocol C yang merupakan protokol untuk sinkronisasi pada DRBD</p>
<pre>on source.domain.local { address 192.168.1.100 <u>7789</u>; device /dev/drbd1; disk /dev/vg/VM1-disk; meta-disk /dev/vg/meta[0]; }</pre>	<pre>on source.domain.local { address 192.168.1.100 <u>7790</u>; device /dev/drbd2; disk /dev/vg/VM1-swap; meta-disk /dev/vg/meta[1]; }</pre>	<p>Inisialisasi sinkronisasi disk dan swap VM1 (virtual machine) di port 7789 dan 7790 pada host 1</p>	
<pre>on dest.domain.local { address 192.168.1.101 <u>7789</u>; device /dev/drbd1; disk /dev/vg/VM1-disk; meta-disk /dev/vg/meta[0]; };</pre>	<pre>on dest.domain.local { address 192.168.1.101 <u>7790</u>; device /dev/drbd2; disk /dev/vg/VM1-swap; meta-disk /dev/vg/meta[1] }</pre>	<p>Inisialisasi sinkronisasi disk dan swap VM1 (virtual machine) di port 7789 dan 7790 pada host 2</p>	

Gambar 4.4 Konfigurasi DRBD

Bentuk meta data sebesar 1 Gb untuk *disk* dan *swap* di volume group “vg” dengan perintah `lvcreate -L 1G -n meta vg` dan jalankan dengan perintah `drbdadm create -md VM1-disk` dan `drbdadm create-md VM1-swap`. Pembentukan meta data untuk disk dan swap dimaksudkan agar dapat menyimpan setiap informasi tentang devais. Setelah semua langkah inisialisasi dan konfigurasi pada DRBD selesai, DRBD dapat dijalankan agar kedua host tersinkronisasi dengan perintah `/etc/init.d/drbd start`. Prosesnya akan berjalan hingga pengecekan kedua status drbd kedua host dengan perintah `/etc/drbd status` menjadi *up to date* seperti berikut :

```
1:VM1-disk Connected Primary/Secondary UpToDate/UpToDate C
2:VM1-swap Connected Primary/Secondary UpToDate/UpToDate C
```

4.1.8 Konfigurasi VM Untuk Menggunakan DRBD

Ubah direktori disk pada virtual machine di VM1.cfg agar sumber dayanya langsung mengacu ke sumber daya DRBD agar sinkronisasi berlangsung terus menerus. Pada awalnya sumber daya disk *virtual machine* akan mengacu pada *volume group* yang dibuat seperti Gambar 4.5.

```
disk = [
  'phy:/dev/vg/test-swap,xvda1,w',
  'phy:/dev/vg/test-disk,xvda2,w',
]
```

Di ubah menjadi

```
disk = [
  'drbd:test-swap,xvda1,w',
  'drbd:test-disk,xvda2,w',
]
```

Gambar 4.5 Konfigurasi VM Untuk DRBD

4.1.9 Salin Konfigurasi *Virtual Machine* Pada *Host* Dua

Konfigurasi VM1.cfg yang hanya terdapat di *host* satu perlu disalinkan pada *host* dua agar *host* dua memiliki file konfigurasi yang sama. Salinan file tersebut pun harus ditempatkan pada folder */etc/xen* sesuai asal file tersebut di *host* satu.

4.1.10 Konfigurasi *Live Migration* pada XEN

Secara *default* Xen tidak mengizinkan *live migration* oleh karena itu, konfigurasi Xen di kedua *host* harus diubah agar *live migration* dapat berjalan dengan perintah *gedit /etc/xen/xend-config* di terminal. Ada beberapa konfigurasi di *xend-config* yang ,secara default tidak diaktifkan, harus diaktifkan seperti konfigurasi pada Gambar 4.6.

```
#Aktivasi jaringan bridge Xen yang akan mendukung bypass jaringan virtual
machine dan host
(vif-script vif-bridge)
(network-script network-bridge)

#Aktivasi alokasi host dengan TCP port 8002
(xend-relocation-hosts-allow ")
(xend-relocation-address ")
(xend-relocation-port 8002)

#Aktivasi alokasi relokasi server
(xend-relocation-server yes)
(xend-unix-server yes)
```

Gambar 4.6 Konfigurasi *Live Migration* Pada Xen

4.1.11 *Live Migration*

Setelah semua implementasi dan konfigurasi yang diperlukan dilakukan proses *live migration* dapat dilakukan dengan perintah `xm migrate` (nama virtual machine) (tujuan migrasi) `-live` (untuk *live migration*)

4.2 **Analisa Kualitas Perpindahan *Virtual Machine***

Kualitas uji yang akan digunakan dalam menganalisa perpindahan *virtual machine* dengan Xen ini berdasarkan berbagai parameter uji. Parameter ubah yang digunakan adalah *throughput*, *delay*, dan jumlah paket yang hilang terhadap dilakukannya perubahan data pada *virtual machine* serta pengaruh *live migration* terhadap beban CPU pada *host*. Sementara parameter tetapnya adalah berupa banyaknya migrasi, yaitu sebanyak sepuluh kali setiap pengujian. Pengujian akan dibagi menjadi dua skenario pengukuran. Pada pengujian dengan skenario satu untuk mengukur *throughput* dan *delay* yang akan diukur selama sepuluh kali migrasi dengan dilakukannya perubahan *virtual machine*. Pada pengujian dengan skenario dua untuk mengukur paket yang hilang akibat *ping test* selama sepuluh kali migrasi serta beban CPU yang dihasilkan selama proses *live migration*. Aplikasi yang digunakan pada skenario satu adalah Wireshark dan pada skenario dua adalah terminal Ubuntu dan VisualVM. Pada akhirnya akan menganalisa lama waktu migrasi *virtual machine* ini dan lama waktu *downtime* yang terjadi dengan memanfaatkan hasil-hasil pengujian dari skenario satu dan skenario dua.

4.3 **Analisa dan Ujicoba pada Skenario Satu**

Analisa pada skenario satu ini dilakukan saat terjadi perpindahan virtual machine sebanyak sepuluh kali dari *host* satu ke *host* dua dan sebaliknya. Proses *capturing* dilakukan dengan menangkap paket – paket TCP Port 8002 karena di port tersebut terjadinya relokasi *host* oleh Xen. Proses *capturing* oleh wireshark seperti yang terlihat pada Gambar 4.7

No.	Time	Source	Destination	Protocol	Length	Info
437361	645.134891	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421293201 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780
437582	645.154166	192.168.1.101	192.168.1.100	TCP	66	teradataoridms > 42205 [ACK] Seq=17 Ack=421293829 Win=551328 Len=0 TSval=786887 TSecr=77356
437583	645.156001	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421293829 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780
437584	645.158016	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421295277 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780
437585	645.158046	192.168.1.101	192.168.1.100	TCP	66	teradataoridms > 42205 [ACK] Seq=17 Ack=421296725 Win=551328 Len=0 TSval=786888 TSecr=77356
437586	645.160034	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421296725 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780
437587	645.161013	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421298173 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780
437588	645.161044	192.168.1.101	192.168.1.100	TCP	66	teradataoridms > 42205 [ACK] Seq=17 Ack=421299621 Win=551328 Len=0 TSval=786889 TSecr=77356
437589	645.163008	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421299621 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780
437590	645.165000	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421301059 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780
437591	645.165037	192.168.1.101	192.168.1.100	TCP	66	teradataoridms > 42205 [ACK] Seq=17 Ack=421302517 Win=551328 Len=0 TSval=786890 TSecr=77356
437592	645.167008	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421302517 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780
437593	645.169009	192.168.1.100	192.168.1.101	TCP	1514	42205 > teradataoridms [ACK] Seq=421303965 Ack=17 Win=14608 Len=1448 TSval=773568 TSecr=780

▶ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
 ▶ Ethernet II, Src: Realtek5_53:44:58 (00:e0:4c:53:44:58), Dst: Realtek5_53:44:58 (00:e0:4c:53:44:58)
 ▶ Internet Protocol Version 4, Src: 192.168.1.101 (192.168.1.101), Dst: 192.168.1.100 (192.168.1.100)
 ▶ Transmission Control Protocol, Src Port: 34091 (34091), Dst Port: 7790 (7790), Seq: 1, Ack: 1, Len: 8
 ▶ Data (8 bytes)

Gambar 4.7 Capturing TCP Port 8002

Dengan melakukan analisa ini, hasilnya diharapkan dapat mengetahui kualitas *throughput* dan *delay* saat proses perpindahan *virtual machine* pada jaringan peer-to-peer dengan dilakukannya perubahan data pada *virtual machine*. Perubahan data yang dimaksud pada skenario ini adalah melakukan *update* pada sistem operasi Ubuntu di *virtual machine* dengan perintah *apt-get update* di terminal.

4.3.1 Analisa Throughput

Throughput merupakan kecepatan rata-rata transfer data yang artinya jumlah data yang dapat dibawa dari sebuah titik ke titik lain dalam jangka waktu tertentu (pada umumnya dalam detik) yang dalam jaringan komputer sering disamakan sebagai *bandwidth*. *Bandwidth* adalah jumlah bit yang dapat dikirimkan dalam satu detik yang dapat ditunjukkan dengan rumus sebagai berikut:

$$\text{Bandwidth} = \text{jumlah bit} / \text{waktu}$$

Sedangkan *throughput* walaupun memiliki satuan dan rumus yang sama dengan *bandwidth*, tetapi *throughput* lebih menggambarkan *bandwidth* yang sebenarnya (aktual) pada suatu waktu tertentu dan pada kondisi jaringan tertentu [11]. Satuan yang dinyatakan sebagai sebuah *throughput* dapat berupa *bytes per second*, *packet per second* atau *bits per second*. Pengambilan data untuk analisa ini dilakukan saat *live migration* terjadi melalui *interface eth1* yang merupakan *interfaces* koneksi *peer to peer* antar kedua *host*. Penangkapan paket-paket dilakukan dengan aplikasi

wireshark di *host* tujuan migrasi. Pada proses *capture* paket-paket di *interfaces* terjadi penyaringan untuk meng-*capture* hanya paket-paket pada TCP port 8002 sehingga rata-rata *throughput* setiap kali migrasi dapat diketahui. *Throughput* dapat diketahui dengan melihat *summary* pada menu statistics di Wireshark. Hasil *throughput* dapat dilihat pada Gambar 4.8

Traffic	Captured	Displayed	Marked
Packets	1136717	1112106	0
Between first and last packet	14054.146 sec	1310.771 sec	
Avg. packets/sec	80.881	848.436	
Avg. packet size	1005.141 bytes	1022.677 bytes	
Bytes	1142561278	1137324707	
Avg. bytes/sec	81297.097	867675.956	
Avg. MBit/sec	0.650	6.941	

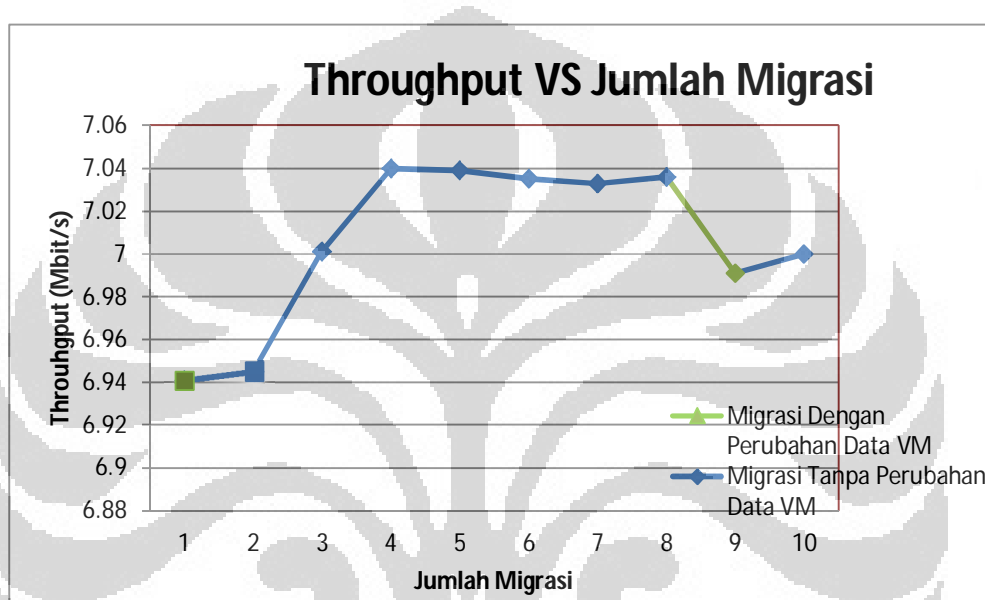
Gambar 4.8 *Throughput* pada Summary Wireshark

Data yang ditangkap pada Gambar 4.8 merupakan data pertama *live migration* setelah percobaan keberhasilan *live migration* tanpa pengambilan data. Hasil data yang didapat selama sepuluh kali migrasi untuk perhitungan nilai rata – rata *throughput* dapat ditampilkan pada Tabel 4.1.

Tabel 4.1 Data Nilai *Throughput*

Percobaan <i>Live Migration</i>	<i>Throughput</i> (Mbit/sec)	<i>Data Virtual Machine</i>
1	6.941	Melakukan Perubahan Data
2	6.945	Tidak Melakukan Perubahan Data
3	7.001	Tidak Melakukan Perubahan Data
4	7.04	Tidak Melakukan Perubahan Data
5	7.039	Tidak Melakukan Perubahan Data
6	7.035	Tidak Melakukan Perubahan Data
7	7.033	Tidak Melakukan Perubahan Data
8	7.036	Tidak Melakukan Perubahan Data
9	6.991	Melakukan Perubahan Data
10	7	Tidak Melakukan Perubahan Data

Dari Tabel 4.1 terlihat bahwa nilai *throughput* dari setiap percobaan *live migration* tanpa dilakukannya perubahan data pada *virtual machine* terlihat cukup stabil sedangkan dilakukannya perubahan data pada *virtual machine* nilai *throughput* migrasi didapatkan mengecil dibanding nilai sebelumnya. Secara lebih jelas hasil analisis *throughput* ini dapat dilihat pada Gambar 4.9.



Gambar 4.9 *Throughput* VS Jumlah Migrasi

Lebih jelas terlihat pada Gambar 4.9 dengan garis dan poin berwarna hijau bahwa setiap dilakukannya perubahan data pada *virtual machine* nilai *throughput* akan turun dan kemudian akan kembali naik begitu tidak dilakukannya perubahan data yang ditandakan dengan poin dan garis berwarna biru. Semakin lama *virtual machine* tidak terjadi perubahan data maka nilai *throughput* akan semakin stabil seperti yang diperlihatkan pada migrasi empat hingga delapan. Hal ini telah sesuai dan dapat dianalisis dengan tidak dilakukannya perubahan data pada *virtual machine* maka *dirty pages* pada *live migration* pun akan tetap sehingga bisa didapatkan *throughput* yang stabil. Dirty Pages adalah page yang belum tertulis di dalam *disk*. Akumulasi *page* dapat terjadi karena adanya perubahan data pada *virtual machine*. Dirty pages terjadi karena aplikasi mengotorinya dengan data baru. Ketika data tersebut telah dituliskan kedua *disk* dan memori maka tidak ada

lagi *dirty page*. Hal ini mengindikasikan bahwa kapasitas *link* menjadi hal yang harus diperhatikan juga. Bila kapasitas link lebih besar maka adanya *dirty pages* bisa lebih teratasi karena dapat langsung dikirimkan ke host tujuan dalam beberapa iterasi atau bahkan dalam satu kali iterasi meski data *virtual machine* terus dinamik untuk melayani kebutuhan user. Oleh karena itu, pada dewasa ini sudah banyak *vendor* dan *provider* menggunakan *Gigabyte Ethernet* dengan *fiber optic*.

4.3.2 Analisa Delay

Delay merupakan waktu tunda paket yang diakibatkan oleh proses transmisi dari satu titik ke titik lain yang menjadi tujuannya [8]. *Delay* diperoleh dari selisih waktu kirim antara satu paket TCP dengan paket yang lainnya. Perhitungan rata-rata delay di Wireshark dilakukan dengan cara membagi lamanya paket pertama dan paket terakhir yang di-*captured* dengan jumlah paket. Namun, keduanya menggunakan data pada kolom *displayed* di *summary* Wireshark seperti yang tampak pada Gambar 4.10.

Traffic	Captured	Displayed	Marked
Packets	1136717	1112106	0
Between first and last packet	14054.146 sec	1310.771 sec	
Avg. packets/sec	80.881	848.436	
Avg. packet size	1005.141 bytes	1022.677 bytes	
Bytes	1142561278	1137324707	
Avg. bytes/sec	81297.097	867675.956	
Avg. MBit/sec	0.650	6.941	

Gambar 4.10 Delay Pada *Summary* Wireshark

Pada Gambar 4.10 menampilkan hasil *summary* pada percobaan *live migration* pertama setelah percobaan keberhasilan *live migration* tanpa pengambilan data menunjukkan lamanya waktu antara paket pertama dan paket terakhir, *between first and last packet*, sebesar 1310.771 sec dan jumlah paket, *packets*, sebanyak 1112106. Hasilnya didapatkan nilai rata-rata delay sebagai berikut :

$$\begin{aligned} \text{Rata - rata delay} &= \textit{between first and last packet} / \textit{Packets} \\ &= 1310.771 / 1112106 \end{aligned}$$

= 0.0011736 sekon

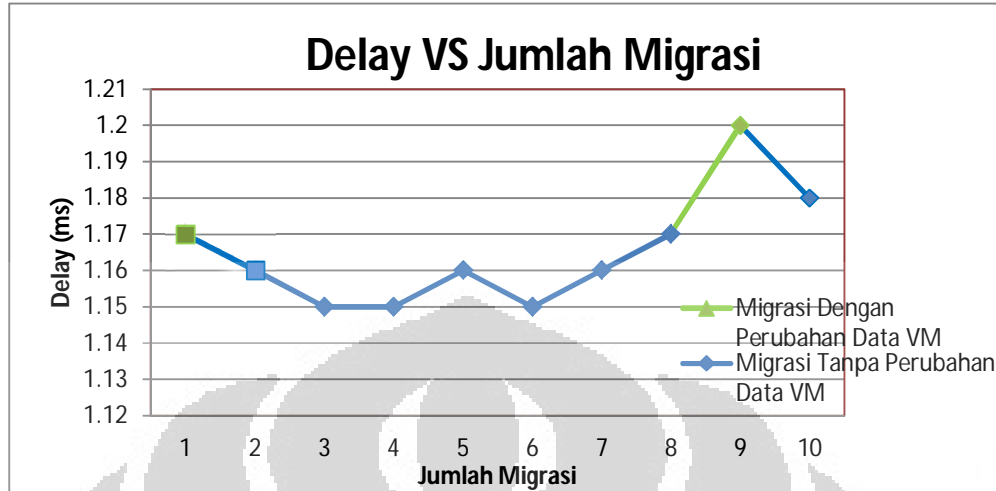
= 1.17 ms

Pengukuran *delay* tetap dilakukan pada *interfaces eth1* jaringan *peer-to-peer* sebanyak sepuluh kali migrasi. *Filtering* dengan paket-paket *delay* pada TCP *port* 8002. Perhitungan nilai rata-rata *delay* untuk percobaan *live migration* seterusnya hingga sepuluh kali dilakukan dengan cara yang sama seperti perhitungan pada percobaan *live migration* pertama. Hasil pengambilan data untuk parameter *delay* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Data Perhitungan *Delay*

Percobaan	<i>Delay(ms)</i>	<i>Data Virtual Machine</i>
1	1.17	Melakukan Perubahan Data
2	1.16	Tidak Melakukan Perubahan Data
3	1.15	Tidak Melakukan Perubahan Data
4	1.15	Tidak Melakukan Perubahan Data
5	1.16	Tidak Melakukan Perubahan Data
6	1.15	Tidak Melakukan Perubahan Data
7	1.18	Tidak Melakukan Perubahan Data
8	1.17	Tidak Melakukan Perubahan Data
9	1.2	Melakukan Perubahan Data
10	1.18	Tidak Melakukan Perubahan Data

Dari Tabel 4.2 terlihat bahwa nilai *delay* dari setiap percobaan *live migration* tanpa adanya perubahan data pada *virtual machine* terlihat cukup stabil sedangkan bila terjadi perubahan data pada *virtual machine* nilai *delay* migrasi didapatkan membesar dari nilai sebelumnya. Secara lebih jelas hasil analisis *delay* ini dapat dilihat pada Gambar 4.11.



Gambar 4.11 Delay vs Jumlah Migrasi

Terlihat pada Gambar 4.11 terlihat nilai *delay* menjadi stabil bila pada *virtual machine* tidak terjadi perubahan data atau tidak terjadi penambahan data baru. Nilai *delay* menjadi besar bila terjadi perubahan data pada *virtual machine* sebelum terjadinya migrasi kembali. Hal ini terjadi akibat adanya *dirty pages* yang bertambah pada *link eth1* sementara kapasitas *link eth1* tidak berubah dalam mengirimkan memory *virtual machine* dalam beberapa iterasi atau pun iterasi terakhir sehingga menyebabkan *delay* meningkat pada saat dilakukannya perubahan data di *virtual machine* sebelum terjadinya *live migration*.

4.4 Analisa Pada Skenario Dua

Analisa pada skenario ini dilakukan saat terjadi perpindahan *virtual machine* sebanyak sepuluh kali dari *host* satu ke *host* dua atau sebaliknya namun dengan penambahan yang sedikit berbeda dari skenario satu. Pada skenario ini selain melakukan *capturing* beban CPU dengan Visual VM ditambahkan juga komputer klien yang melakukan ping ke IP *virtual machine* terus-menerus hingga proses *live migration* selesai melalui terminal Ubuntu. Proses *capturing* di terminal Ubuntu seperti yang terlihat pada Gambar 4.12.

```

64 bytes from 192.168.102.50: icmp_req=1465 ttl=64 time=0.295 ms
64 bytes from 192.168.102.50: icmp_req=1466 ttl=64 time=0.299 ms
64 bytes from 192.168.102.50: icmp_req=1467 ttl=64 time=0.271 ms
64 bytes from 192.168.102.50: icmp_req=1468 ttl=64 time=0.316 ms
64 bytes from 192.168.102.50: icmp_req=1469 ttl=64 time=0.280 ms
64 bytes from 192.168.102.50: icmp_req=1470 ttl=64 time=0.274 ms
64 bytes from 192.168.102.50: icmp_req=1471 ttl=64 time=0.524 ms
64 bytes from 192.168.102.50: icmp_req=1472 ttl=64 time=0.276 ms
64 bytes from 192.168.102.50: icmp_req=1473 ttl=64 time=0.214 ms
64 bytes from 192.168.102.50: icmp_req=1474 ttl=64 time=0.296 ms
64 bytes from 192.168.102.50: icmp_req=1475 ttl=64 time=0.322 ms
64 bytes from 192.168.102.50: icmp_req=1476 ttl=64 time=0.262 ms
64 bytes from 192.168.102.50: icmp_req=1477 ttl=64 time=0.277 ms

```

Gambar 4.12 *Capturing Ping Di Terminal Ubuntu*

Dengan melakukan analisa ini, hasilnya diharapkan dapat mengetahui *availability virtual machine* dengan *Xen live migration* saat proses perpindahan *virtual machine* pada jaringan *peer-to-peer* dengan dilakukannya perubahan data pada *virtual machine* dan pengaruh *live migration* terhadap beban CPU

4.4.1 Analisa Ping Test

Ping adalah standar utilitas administrasi jaringan digunakan untuk melakukan test ketercapaian host dalam sebuah *Internet Protocol (IP)*. Ping beroperasi dengan mengirimkan *Internet Control Message Protocol (ICMP)* kepada host dan menunggu respon balik oleh ICMP. Hasilnya dapat diketahui berapa kali permintaan dikirimkan, berapa besar permintaan dikirim, berapa lama untuk menunggu setiap balasan, dan berapa paket yang hilang selama proses. Pengambilan data untuk analisa ini dilakukan selama proses *live migration* berlangsung hingga *virtual machine* telah berjalan di *host* tujuan. Pengukuran dilakukan selama sepuluh kali perpindahan oleh komputer klien untuk mengetahui berapa banyak paket yang hilang selama perpindahan. Paket yang hilang dapat diketahui setelah menghentikan proses ping dengan mengurangi jumlah paket yang dikirim dengan paket yang diterima. Hasilnya dapat dilihat seperti Gambar 4.13.

```

64 bytes from 192.168.102.50: icmp_req=1465 ttl=64 time=0.295 ms
64 bytes from 192.168.102.50: icmp_req=1466 ttl=64 time=0.299 ms
64 bytes from 192.168.102.50: icmp_req=1467 ttl=64 time=0.271 ms
64 bytes from 192.168.102.50: icmp_req=1468 ttl=64 time=0.316 ms
64 bytes from 192.168.102.50: icmp_req=1469 ttl=64 time=0.280 ms
64 bytes from 192.168.102.50: icmp_req=1470 ttl=64 time=0.274 ms
64 bytes from 192.168.102.50: icmp_req=1471 ttl=64 time=0.524 ms
64 bytes from 192.168.102.50: icmp_req=1472 ttl=64 time=0.276 ms
64 bytes from 192.168.102.50: icmp_req=1473 ttl=64 time=0.214 ms
64 bytes from 192.168.102.50: icmp_req=1474 ttl=64 time=0.296 ms
64 bytes from 192.168.102.50: icmp_req=1475 ttl=64 time=0.322 ms
64 bytes from 192.168.102.50: icmp_req=1476 ttl=64 time=0.262 ms
64 bytes from 192.168.102.50: icmp_req=1477 ttl=64 time=0.277 ms
64 bytes from 192.168.102.50: icmp_req=1478 ttl=64 time=0.292 ms
64 bytes from 192.168.102.50: icmp_req=1479 ttl=64 time=0.202 ms
64 bytes from 192.168.102.50: icmp_req=1480 ttl=64 time=0.287 ms
64 bytes from 192.168.102.50: icmp_req=1481 ttl=64 time=0.265 ms
64 bytes from 192.168.102.50: icmp_req=1482 ttl=64 time=0.284 ms
64 bytes from 192.168.102.50: icmp_req=1483 ttl=64 time=0.267 ms
^C
192.168.102.50 ping statistics:
1483 packets transmitted, 1478 received, 0% packet loss, time 1482005ms
rtt min/avg/max/mdev = 0.175/0.299/15.119/0.458 ms
root@daniel-Presario-C042-Notebook-PC: /home/daniel#

```

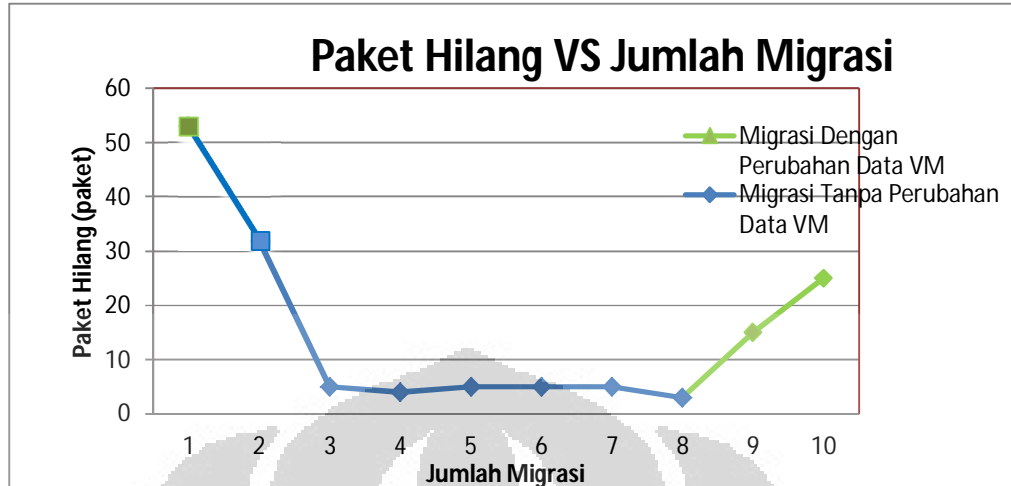
Gambar 4.13 Hasil *Ping Test* Pada Percobaan Ketiga

Pada Gambar 4.13 yang merupakan hasil ping test pada percobaan migrasi ketiga dapat diketahui jumlah paket yang hilang dengan mengurangi jumlah paket yang ditransmisikan (*packets transmitted*) dengan paket yang diterima (*received*), yaitu sebesar 5 paket. Dengan melakukan analisis ini, hasilnya diharapkan dapat mengetahui bagaimana *availability virtual machine* selama migrasi. Perhitungan jumlah paket yang hilang selama sepuluh migrasi dilakukan dengan cara yang sama seperti pada perhitungan jumlah paket yang hilang di Gambar 4.13. Hasil data yang didapat selama sepuluh kali migrasi untuk perhitungan paket yang hilang setiap kali migrasi dapat ditampilkan pada Tabel 4.4

Tabel 4.4 Data Paket Yang Hilang

Percobaan <i>Live Migration</i>	Paket Hilang	Data <i>Virtual Machine</i>
1	53	Melakukan Perubahan Data
2	32	Tidak Melakukan Perubahan Data
3	5	Tidak Melakukan Perubahan Data
4	4	Tidak Melakukan Perubahan Data
5	5	Tidak Melakukan Perubahan Data
6	5	Tidak Melakukan Perubahan Data
7	5	Tidak Melakukan Perubahan Data
8	3	Tidak Melakukan Perubahan Data
9	15	Melakukan Perubahan Data
10	25	Melakukan Perubahan Data

Dari Tabel 4.3 terlihat bahwa banyaknya paket hilang berpengaruh terhadap dilakukannya perubahan data pada *virtual machine*. Setiap kali dilakukannya perubahan data maka pada akhir proses ping akan bertambah banyak paket yang hilang. Namun, sebaliknya selama tidak dilakukannya perubahan data maka paket yang hilang akan stabil dengan nilai yang kecil sekitar 5 paket. Secara lebih jelas hasil analisis ping ini dapat dilihat pada Gambar 4.14.

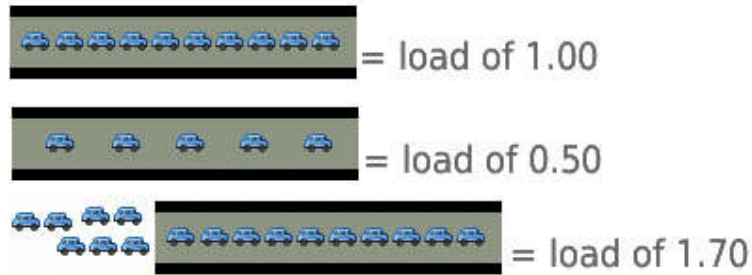


Gambar 4.14 Paket Hilang VS Jumlah Migrasi

Pada Gambar 4.14 terlihat jelas bahwa garis berwarna hijau, yaitu adanya perubahan data pada *virtual machine* mengakibatkan besar paket yang hilang meningkat dan menyebabkan ketidakstabilan *live migration*. Banyaknya paket yang hilang mengindikasikan turut membesarnya waktu *downtime*. Waktu *downtime* adalah proses dihentikannya *virtual machine* untuk sinkronisasi akhir *traffic*, *disk* dan *swap*, dan sistem ke *host* tujuan. Hal ini disebabkan karena adanya *dirty pages* yang bertambah seiring dilakukannya perubahan data dan penambahan data sinkronisasi ke *host* tujuan sedangkan kapasitas *link eth1* antar *host* tetap.

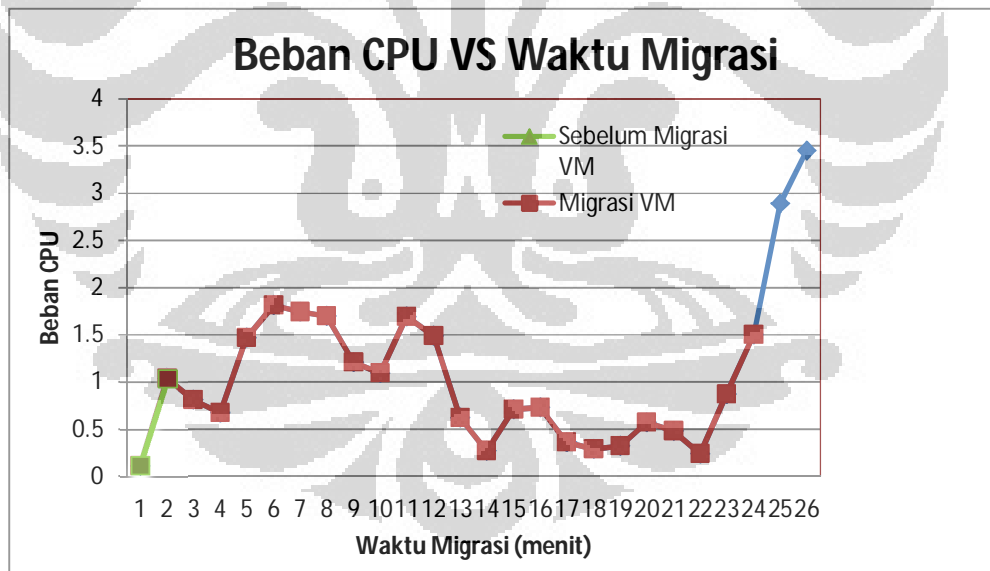
4.4.2 Analisa Beban CPU

Beban CPU (*CPU Load*) pada Linux adalah antrian *traffic* proses CPU. Lebih jelasnya dapat dianalogikan sebagai antrian mobil untuk masuk ke dalam gerbang dimana untuk *single processor* nilai *load* efektif adalah 1.00 yang dianalogikan hanya dapat menampung 10 mobil. Bila nilai *load* lebih dari itu dikatakan ada antrian mobil lainnya diluar yang tidak dapat tertampung karena gerbang sudah terisi sepuluh mobil. Analogi ini digambarkan pada Gambar 4.15.



Gambar 4.15 Analogi CPU Load Linux [11]

Namun, pada *live migration* ini digunakan *dual-core processor* sehingga *load* yang efektif adalah 2.00. Pengambilan data untuk analisa ini dilakukan selama proses *live migration* berlangsung hingga *virtual machine* telah berjalan di *host* tujuan. Nilai beban CPU dapat diketahui dengan bantuan VisualVM yang akan memperlihatkan nilai *load* untuk tiap menitnya selama proses *live migration* tanpa perubahan data. Hasilnya dapat dilihat pada Gambar 4.16.

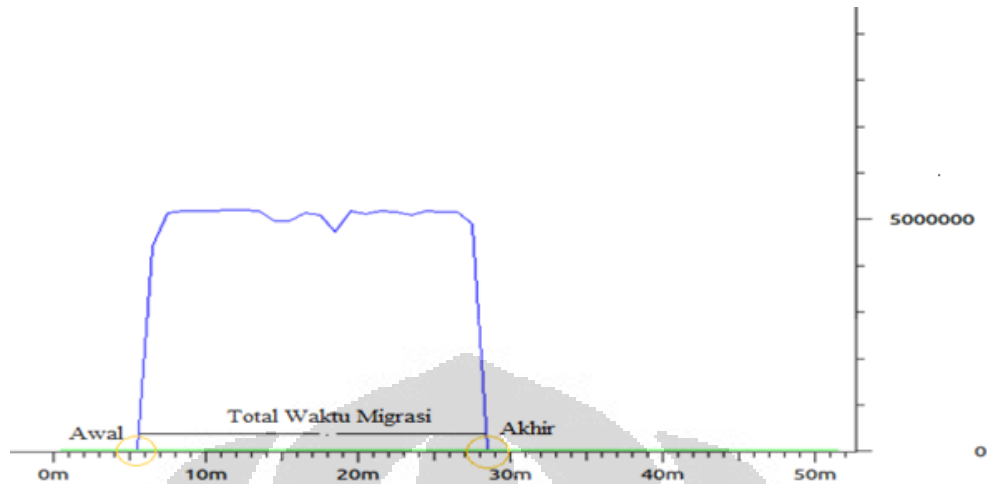


Gambar 4.16 Beban CPU vs Waktu Migrasi

Pada Gambar 4.16 perubahan beban CPU yang cukup fluktuatif selama *live migration*. Selama masa *idle* sebelum terjadinya proses migrasi beban CPU masih sangat rendah sekitar 0.5. Saat proses berlangsung ada peningkatan beban CPU namun hingga proses berlangsung berakhir nilai beban CPU masih berada di bawah nilai 2.00 dan setelah migrasi berakhir yang diakhiri dengan aktifnya *virtual machine* terlihat adanya peningkatan beban CPU hingga 3.5 yang kemudian akan kembali normal kembali nilai 0.5. Dari penjelasan diatas dapat dianalisa bahwa proses migrasi tidak menggunakan banyak konsumsi beban CPU yang berarti meski menunjukkan peningkatan namun masih dalam tahap wajar untuk beban CPU *dual-core* karena tidak ada interferensi antar *host* dan *virtual machine* dalam *host* karena diatur oleh lapisan virtual atau masih melimpahnya sumber daya CPU di *host* yang menggunakan *dual-core*. Hal ini didukung pula dengan tidak adanya proses berarti saat proses migrasi pada CPU. Adanya peningkatan beban saat *virtual machine* aktif berjalan yang kemudian akan turun kembali adalah kewajaran adanya pengaktifan kernel, sistem root, memori, dan lonjakan CPU dari *virtual machine* itu sendiri.

4.5 Analisa Lama Waktu Migrasi Dan Downtime

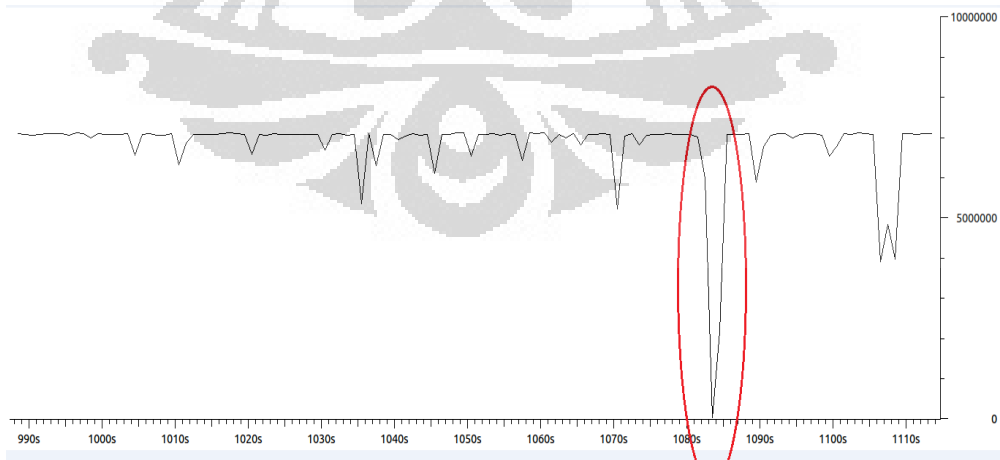
Lama waktu migrasi adalah lamanya waktu *virtual machine* melakukan perpindahan dari satu *host* ke *host* lainnya yang berbeda secara fisik. Sementara *downtime* adalah waktu berhentinya *virtual machine* untuk melakukan sinkronisasi akhir dengan *host* tujuan. Lama waktu migrasi dapat dengan mudah ditentukan dengan melihat grafik hasil IO *graph* yang dihasilkan oleh wireshark. IO *graph* akan menampilkan grafik TCP port 8002 yang merupakan port yang digunakan dalam proses relokasi host di Xen. Setelah capturing pada TCP port 8002 sudah berhenti berjalan, mengindikasikan migrasi telah selesai, grafiknya dapat dilihat dengan melihat IO *graph* pada *statistic*. Hasilnya dapat dilihat pada Gambar 4.17 dengan sumbu x dalam satuan menit dan sumbu y dalam satuan bit/s.



Gambar 4.17 Grafik IO Graph TCP Port 8002

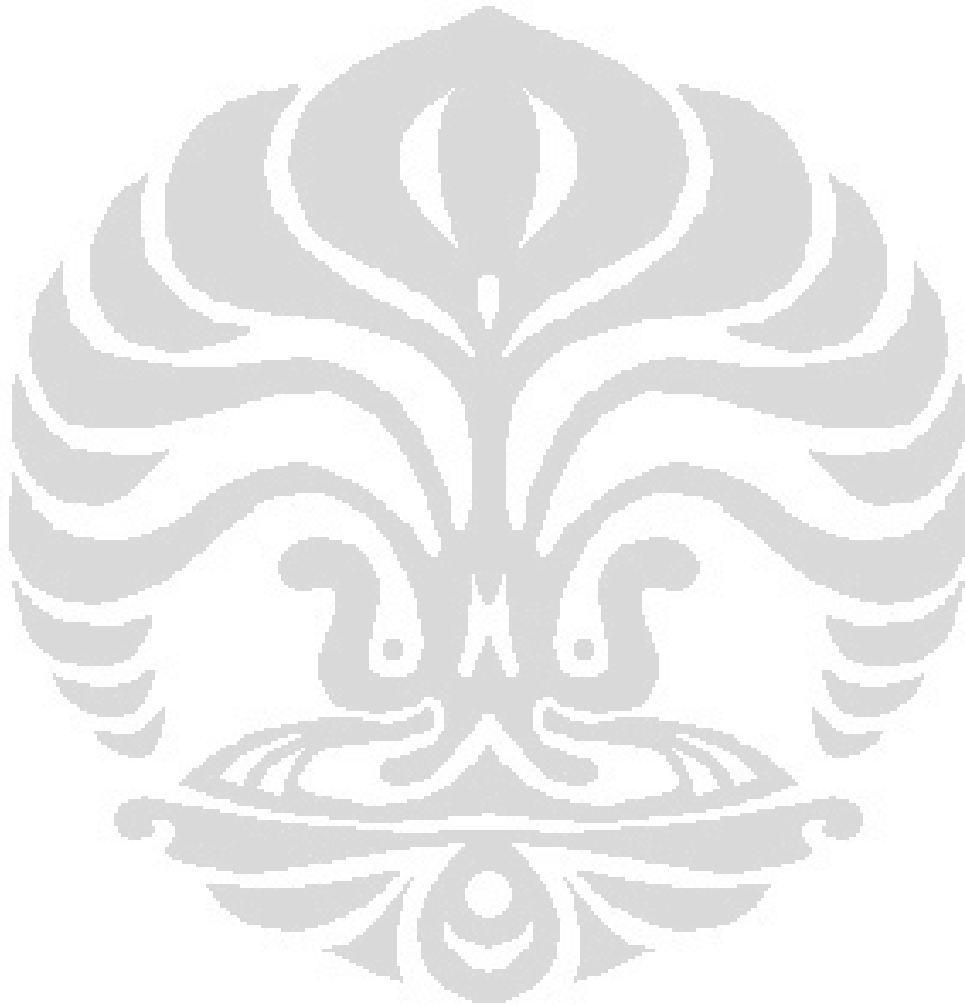
Gambar 4.17 menggunakan percobaan migrasi pertama karena merupakan migrasi pertama yang diuji. Gambar 4.17 memperlihatkan batas dari munculnya *traffic* TCP port 8002 dan hilangnya *traffic* tersebut yang dapat dianalisa merupakan lamanya waktu migrasi. Hasilnya lama waktu *live migration virtual machine* ini adalah 22 menit.

Lama waktu *downtime* dapat ditentukan dengan melihat *throughput* terendah pada IO *graph* Gambar 4.18. Namun, kali ini grafik tersebut harus diperbesar untuk melihat lebih jelas *traffic throughput* terendah yang mengindikasikan saat-saat *downtime* dari *virtual machine*. Hasilnya dapat dilihat pada Gambar 4.18



Gambar 4.18 Downtime Virtual Machine

Pada Grafik 4.18 terlihat adanya *throughput* yang terhenti beberapa saat yang ditunjukkan dengan grafik yang turun hingga menyentuh titik 0 selama 5 detik. Hal ini menggambarkan adanya proses ‘*stop*’ pada *live migration virtual machine* untuk menyelesaikan proses iterasi akhir pada *host* tujuan.



BAB V

KESIMPULAN

Berdasarkan hasil pengukuran dan analisa data pada skenario–skenario yang ada untuk penelitian ini didapatkan kesimpulan sebagai berikut :

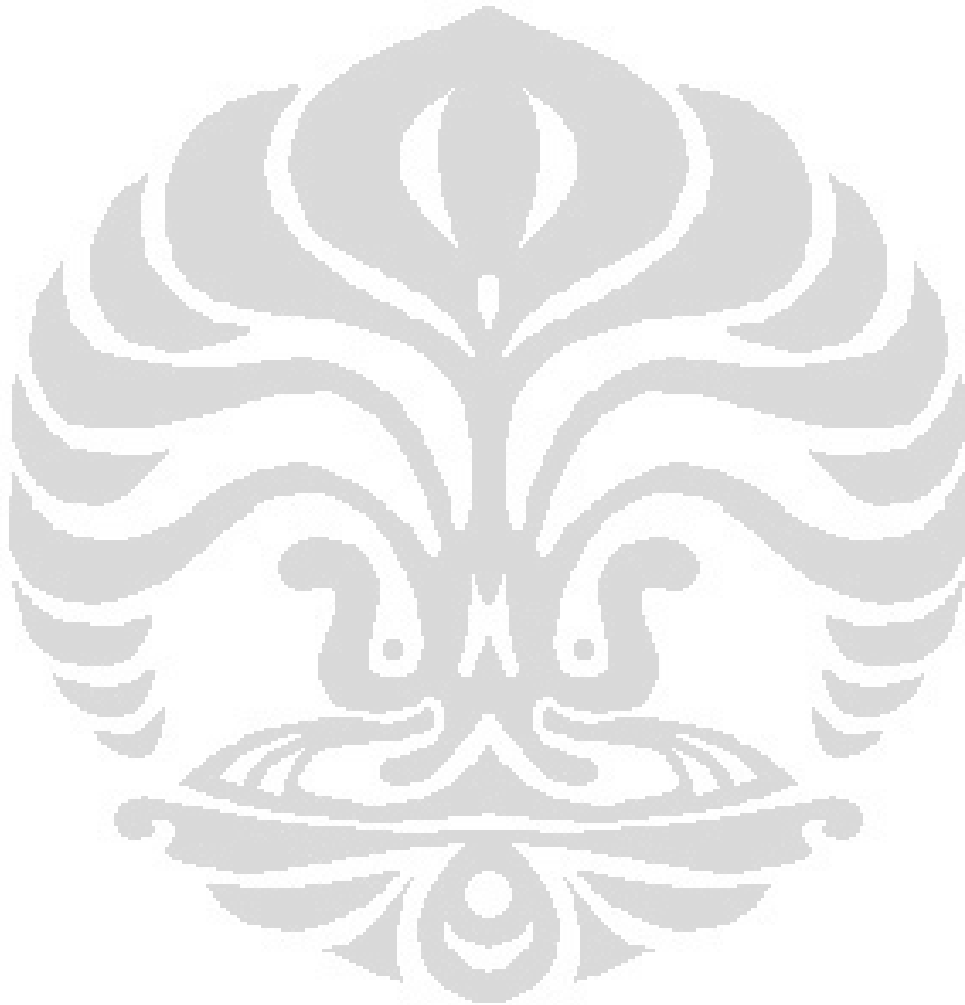
1. Berdasarkan parameter *throughput* dan dilakukannya perubahan data, perpindahan *virtual machine* dengan adanya perubahan data menyebabkan *throughput* mengecil dari perpindahan sebelumnya dan stabil di sekitar 7 Mbit/sec begitu beberapa kali perpindahan tidak dilakukan perubahan data. Nilai rata – rata *throughput* selama sepuluh kali migrasi sebesar 7.002 Mbit/sec.
2. Berdasarkan parameter delay, adanya perubahan data pada virtual machine menyebabkan naiknya nilai delay namun begitu tidak adanya perubahan data pada *virtual machine* selama beberapa kali perpindahan nilai *delay* akan stabil di sekitar 1.17 ms. Nilai rata–rata *delay* selama sepuluh kali migrasi sebesar 1.167 ms.
3. Berdasarkan parameter paket yang hilang, adanya perubahan data pada *virtual machine* menyebabkan terjadinya peningkatan jumlah paket yang hilang namun begitu tidak adanya perubahan data pada *virtual machine* selama beberapa kali perpindahan jumlah paket yang hilang akan stabil di sekitar 5 paket. Nilai rata–rata paket yang hilang selama sepuluh kali migrasi sebesar 15.2 paket.
4. Berdasarkan parameter beban CPU, perpindahan *virtual machine* tidak menguras sumber daya CPU yang berarti. Nilai bebannya masih berada di bawah 2.00 selama migrasi berlangsung yang berarti masih normal untuk *processor dual-core*.
5. Berdasarkan analisa grafik pada TCP port 8002 dapat diketahui lama waktu migrasi dan total *downtime*.
6. Berdasarkan analisa yang ada ternyata dilakukannya perubahan data pada *virtual machine* mempengaruhi *throughput*, *delay*, dan jumlah paket yang hilang selama perpindahan ini.

DAFTAR ACUAN

- [1] Furht, B., & Escalante, A. (2010). *Handbook of Cloud Computing*. New York: Springer.
- [2] Concept Virtualization Server. Diakses tanggal: Desember, 21 2011. <http://UWF.com/concept-of-virtualization>.
- [3] Menasce, D. A. (2005). *Department Computer Science: George Mason University*. Diakses Januari 27, 2012, dari Department Computer Science: www.cs.gmu.edu/faculty/menasce.html
- [4] Techtarget Corporation. (2010, Juni 2). *SearchServerVirtualization*. Diakses januari 20, 2012, dari virtual-machine: <http://searchservervirtualization.techtarget.com/virtual-machine>
- [5] Akoush, S., Sohan, R., Rice, A., Moore, A. W., & Hopper, A. (2010). Predicting Performance Of Virtual Machine. *18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication System*, 1 -10.
- [6] Paul Willmann, Scott Rixner, and Alan L. Cox. Protection Strategies for Direct Access to Virtualized I/O Devices. *Proceedings of the annual conference on USENIX. Annual Technical Conference*, 2008
- [7] Linbit. (2011, Februari 21). What is DRBD. Di akses januari 5, 2012, from DRBD.org: <http://drbd.org>
- [8] *Logical Volume Manager: redhat*. (2007 - 2012). Diakses Januari 28, 2012, dari redhat Inc: http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/ch-lvm.htm
- [9] Techtarget Corporation. (2010, Juni 2). *RAID (redundant array of independent disk)*. Diakses januari 28, 2012, dari storage resources: <http://searchstorage.techtarget.com/definition/RAID>
- [10] Spark Support Infotech Pvt Ltd. (2010, Desember 19). *Xen Bridge Networking : sparksupport*. Diakses Februari 15, 2012, dari sparksupport: <http://www.sparksupport.com/blog/xen-bridging>

[11] Andre. (2011, Juli 31). *Understanding Linux CPU Load - when should you be worried?* Diakses Maret 12, 2012, dari scoutapp web site:

<http://blog.scoutapp.com/articles/2009/07/31/understanding-load-average>



DAFTAR PUSTAKA

- Akoush, S., Sohan, R., Rice, A., Moore, A. W., & Hopper, A. (2010). Predicting Performance Of Virtual Machine. *18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication System*, 1 -10.
- Asplund. (2011, Februari 27). *Xen CLuster*. Diakses januari 5, 2012, dari ASPLUN.NU: <http://asplund.nu/xencluster/xen-cluster-howto.html>
- Clark, C., Fraser, K., Hand, S., Hanshen, J. G., Pratt, I., & Warfield, A. (2007). Live Migration Of Virtual Machines. *IEEE Conference*, 1-14.
- Polze , A., Troger, P., & Salfner, F. (2011). Timely Virtual Machine Migration for Pro-Active Fault Tolerance. *14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops* (pp. 1-10). Berlin: IEEE.
- Wu, Y., & Zhao, M. (2011). Performance Modelling Of Virtual Machine Live Migration. *IEEE 4th International Conference on Cloud Computing* (pp. 1-8). Miami: IEEE.