Perancangan Sistem *Cluster Server* untuk Jaminan Ketersediaan Layanan Tinggi pada Lingkungan Virtual

Yudi Restu Adi[1], Oky Dwi Nurhayati[2], Eko Didik Widianto[3]

***Abstract*—Computer based services are now used in various fields, such as business, health, and education. The objective is to improve the performance of the company, institution, or organization. Services and data are stored in a server machine. Therefore, the server machine becomes an important thing that supports the availability of services, beside network infrastructure and electricity. The server machine may have hardware failure or software crash. The server machine can be down because of a power failure, human error, or disaster. The server machine sometimes must be turned off for upgrades or maintenance purposes. When a server is down, services running on the server will stop and important data can be lost. The objective of this research is to build a server cluster system that supports high availability services and data integrity over a virtual environment. The result of this research is a cluster server system that supports high availability services over the virtual environment and guarantee data integrity using server virtualization Proxmox VE 3.4 on two computer machines, FreeNAS x86 9.2.1.9 as a NAS server, two units of the Cisco Catalyst 2960 switch, and DRBD for data synchronization.**

***Intisari*— Layanan berbasis komputer kini digunakan pada berbagai bidang seperti bisnis, kesehatan, dan pendidikan. Tujuannya adalah untuk meningkatkan kinerja dari perusahaan, instansi, atau organisasi. Layanan dan data disimpan pada mesin *server*, sehingga mesin *server* menjadi faktor penting pendukung ketersediaan layanan selain dari infrastruktur jaringan dan kelistrikan. Perangkat keras mesin *server* dapat mengalami kerusakan dan perangkat lunak dapat mengalami *crash*. Mesin *server* dapat mati disebabkan kegagalan daya, kesalahan manusia, atau bencana alam. Mesin *server* juga terkadang harus dimatikan untuk keperluan *upgrade* atau pemeliharaan. Apapun sebabnya, saat *server* mengalami *down*, data-data penting bisa hilang dan layanan yang berjalan di dalam *server* akan berhenti. Tujuan penelitian dalam makalah ini adalah membangun sistem *cluster server* yang dapat memberikan jaminan ketersediaan layanan tinggi dan integritas data di atas lingkungan virtual. dari penelitian dalam makalah ini yaitu sistem *cluster server* yang mendukung ketersediaan layanan tinggi di atas lingkungan virtual dan jaminan integritas data menggunakan *server* virtualisasi Proxmox VE 3.4 pada dua unit komputer, FreeNAS 9.2.1.9 x86 sebagai *server* NAS, dua unit *switch* Cisco Catalyst 2960, dan DRBD untuk sinkronisasi data.**

***Kata Kunci*— *cluster*, *server*, virtualisasi, Proxmox VE, *High Availability*.**

[1]*Mahasiswa, Program Studi Sistem Komputer Fakultas Teknik Universitas Diponegoro, Jl. Prof. H. Soedarto, SH, Tembalang, Kota Semarang 50275 INDONESIA (e-mail: yudirestuadi@ce.undip.ac.id)*

[2, 3] *Dosen, Program Studi Sistem Komputer Fakultas Teknik Universitas Diponegoro, Jl. Prof. H. Soedarto, SH, Tembalang, Kota Semarang 50275 INDONESIA*

I. PENDAHULUAN

Perkembangan teknologi komputer dan jaringan diikuti juga oleh perkembangan layanan pada *server*. Layanan-layanan pada berbagai bidang seperti bisnis, kesehatan, dan pendidikan banyak yang sudah berbasis komputer dan jaringan, dengan tujuan untuk meningkatkan kinerja dari perusahaan, instansi, atau organisasi. Layanan dan data disimpan pada mesin *server*, sehingga mesin *server* menjadi faktor penting pendukung ketersediaan layanan selain infrastruktur jaringan. Semakin banyak layanan *server* dan data, semakin banyak pula diperlukan mesin *server*. Hal ini mengakibatkan meningkatnya biaya dan ruang yang dibutuhkan. Berdasarkan masalah yang telah dijelaskan, solusi yang digunakan untuk efisiensi biaya dan ruang adalah menggunakan teknologi virtualisasi.

Virtualisasi adalah suatu teknologi pada sebuah perangkat lunak yang memungkinkan satu perangkat keras untuk menjalankan beberapa sistem operasi dan servis pada saat yang sama [1]. Layanan-layanan *server* dijalankan pada mesin-mesin *server* virtual di dalam mesin *server* fisik. Jumlah layanan yang banyak, data-data penting, dan tingkat ketergantungan kinerja dari perusahaan, instansi, atau organisasi yang tinggi terhadap layanan *server* membuat *server* harus dapat melayani secara terus menerus.

Dalam makalah ini, penulis bermaksud membangun sebuah sistem *cluster server* untuk jaminan ketersediaan tinggi di atas lingkungan virtual yang dapat digunakan sebagai metode untuk *Disaster Recovery* (DR). DR adalah kemampuan untuk melanjutkan layanan saat terjadi gangguan yang besar, dan terkadang mengakibatkan kemampuan dan kinerja berkurang. DR menangani kasus ketika operasi tidak dapat dilanjutkan pada sistem atau tempat yang sama, pengganti atau cadangan sistem diaktifkan, dan operasi dilanjutkan di sistem tersebut [2].

Dines [3] menjelaskan dalam penelitiannya, penyebab gangguan utama yaitu kegagalan daya sebesar 44%, kemudian kegagalan perangkat keras sebesar 24%, kegagalan jaringan sebesar 15%, badai musim dingin sebesar 14%, kesalahan manusia sebesar 13%, banjir sebesar 13%, kegagalan perangkat lunak sebesar 11%, kebakaran sebesar 6%, lainnya sebesar 5%, angin topan sebesar 4%, tornado sebesar 2%, gempa bumi sebesar 1%, terorisme sebesar 1%, dan 36% responden menjawab belum memiliki gangguan.

Tujuan penelitian dalam makalah ini adalah menghasilkan sebuah sistem *cluster server* yang dapat memberikan jaminan ketersediaan layanan tinggi di atas lingkungan virtual dan menghasilkan sebuah sistem penyimpanan yang digunakan oleh sistem *cluster server* agar dapat memberikan jaminan integritas data saat terjadi kegagalan pada salah satu komponen sistem.

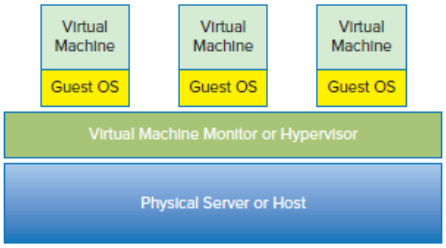
II. PERANGKAT KERAS DAN LUNAK PENYUSUN SISTEM

A. Komputer Cluster

Komputer *cluster* adalah sekumpulan komputer yang bekerja sama sehingga terlihat seolah-olah komputer tersebut adalah sebuah sistem tunggal. Komponen-komponen dalam *cluster* umumnya terhubung satu sama lain melalui jaringan berkecepatan tinggi seperti *Local Area Network* (LAN), dengan masing-masing *node* menjalankan sistem operasinya sendiri. *Cluster* umumnya digunakan untuk meningkatkan kinerja dan ketersediaan dibanding menggunakan komputer tunggal dan lebih hemat biaya daripada menggunakan komputer tunggal yang memiliki kecepatan atau ketersediaan sebanding. *Cluster* muncul sebagai konvergensi dari beberapa hal seperti ketersediaan mikroprosesor dengan harga rendah, jaringan berkecepatan tinggi, dan perangkat lunak untuk komputasi terdistribusi. Konsep *cluster* merupakan manajemen dari beberapa mesin fisik yang terhubung satu sama lain dengan menggabungkan sumber daya dari masing-masing mesin tersebut menjadi satu kesatuan [4].

B. Teknologi Virtualisasi

Pengertian virtualisasi dalam komputasi mengacu pada abstraksi dari komponen fisik menjadi objek logis. Dengan virtualisasi, dapat diperoleh utilitas yang lebih besar dari komponen fisik yang tersedia. Misalnya, Virtual LAN (VLAN) memberikan performa jaringan yang lebih baik dan mempermudah manajemen jaringan dengan membagi jaringan secara logis. Dalam virtualisasi komputer terdapat dua komponen yang berperan di dalamnya yaitu mesin virtual dan *hypervisor*. Gbr. 1 menunjukkan konsep virtualisasi komputer [5].



Gbr. 1 Konsep virtualisasi komputer.

Teknologi virtualisasi mengemulasi sumber daya komputasi fisik seperti komputer *desktop* dan *server*, prosesor dan memori, sistem penyimpanan, dan jaringan. *Server* virtualisasi menciptakan lingkungan virtual yang memungkinkan beberapa beban aplikasi atau *server* yang berjalan di satu komputer seolah-olah berjalan di komputer yang berbeda [6].

Hypervisor atau lebih dikenal dengan istilah *Virtual Machine Monitor* (VMM) adalah sebuah perangkat lunak virtualisasi yang dapat menjalankan beberapa sistem operasi pada sebuah mesin fisik secara bersamaan. Fungsi utama *hypervisor* adalah mengisolasi perangkat keras untuk masing-masing mesin virtual. *Hypervisor* juga berfungsi mengelola akses antara sistem operasi yang berjalan di atasnya dengan perangkat keras yang tersedia [5].

C. Proxmox Virtual Environment

Proxmox Virtual Environment atau Proxmox VE adalah *distro* Linux *server* dengan kode sumber terbuka untuk lingkungan virtualisasi berbasis distribusi Linux Debian. Proxmox VE digunakan untuk mengimplementasikan dan mengelola mesin virtual. Proxmox VE menggunakan *kernel* Red Hat Enterprise Linux (RHEL) yang dimodifikasi. Ada dua teknologi *hypervisor* yang didukung oleh Proxmox VE yaitu *container-based virtualization* dan *Kernel-based Virtual Machine* (KVM). Proxmox VE memiliki beberapa fitur utama sebagai berikut:

1) Open source: Proxmox VE sepenuhnya *open source* di bawah *General Public License*, *version* 3 (GNU AGPL, v3), yang artinya dapat dengan bebas melihat, mengubah, dan menghapus kode sumber.

2) Live migration: *Live migration* memungkinkan untuk memindahkan mesin virtual yang berjalan dari satu *server* fisik ke *server* fisik lain dengan *downtime* yang sangat kecil atau tanpa *downtime*.

3) High Availability: Dalam mode *cluster*, ketika terjadi kegagalan pada satu *node*, mesin virtual di dalamnya akan dipindahkan ke *node* lain untuk meminimalkan gangguan layanan.

4) Bridged networking: Proxmox VE memungkinkan pengguna untuk membuat jaringan privat antara mesin virtual. Selain itu, VLAN juga tersedia.

5) Flexible storage: Banyak pilihan penyimpanan yang tersedia termasuk penyimpanan lokal dan penyimpanan berbasis jaringan, seperti LVM, iSCSI, NFS, GFS, dan CEPH.

6) OS *template*: Proxmox VE memungkinkan pengguna untuk membangun *template* sistem operasi sendiri.

7) Scheduled backup: Tersedia antarmuka untuk pengguna yang digunakan untuk mengatur strategi *backup*. *File backup* dapat disimpan secara lokal atau penyimpanan lain yang sudah dikonfigurasi.

8) Command-line (CLI) *tool*: Proxmox VE menyediakan cara manajemen lain untuk pengguna mengatur sumber daya, mesin virtual, dan lainnya [7].

D. FreeNAS

FreeNAS adalah perangkat lunak gratis yang berfungsi untuk membuat *personal computer* (PC) menjadi *server Network Attached Storage* (NAS). FreeNAS mendukung koneksi dari Microsoft Windows, Apple OS X, Linux, dan FreeBSD. FreeNAS dikenal sebagai sistem operasi tertanam karena ringkas, efisien, dan spesifik hanya untuk satu tugas, dalam hal ini NAS. Setelah FreeNAS terpasang pada PC, maka PC menjadi *server* NAS, tidak dapat melakukan tugas-tugas umum lainnya pada waktu yang sama.

FreeNAS merupakan perangkat lunak dengan kode sumber terbuka, yang artinya pengguna bebas untuk memodifikasi dan mendistribusikannya. FreeNAS mendukung beberapa protokol diantaranya protokol *Common Internet File System* (CIFS) yang merupakan protokol milik Microsoft untuk mengakses

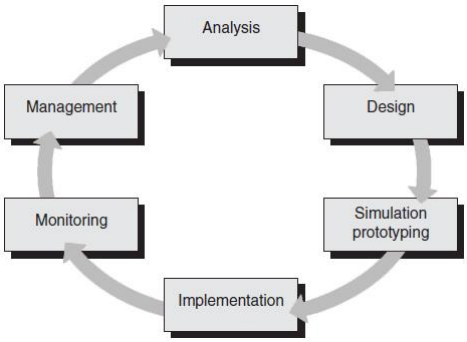
file melalui jaringan, protokol *Network File System* (NFS) yang banyak digunakan dalam sistem operasi Linux, protokol *File Transfer Protocol* (FTP) yang digunakan untuk mengirim *file* melalui jaringan, protokol *Remote Synchronization* (RSYNC) yang digunakan sebagai *backup server*, protokol *Secure Shell* (SSH) untuk enkripsi koneksi dan pertukaran data, dan protokol *Internet Small Computer System Interface* (iSCSI) yang menyimulasikan *hard disk* lokal melalui jaringan *Internet Protocol* (IP) [8].

E. DRBD

DRBD merupakan singkatan dari *Distributed Replicated Block Device*. Pada dasarnya DRBD merupakan *Redundant Array of Independent Disks* (RAID) *level* 1 yang berbasis jaringan. DRBD digunakan dalam lingkungan HA. DRBD memberikan ketersediaan tinggi dengan melakukan *mirroring* sistem ke mesin lain. Jika sistem pada *node* utama mengalami gangguan, dapat langsung dialihkan ke *node* cadangan untuk menghindari gangguan layanan. Dalam struktur DRBD, *primary node* dan *secondary node* terhubung melalui jaringan. Setiap perubahan pada *primary node* akan disinkronisasikan melalui *network interface card* (NIC) [7].

III. METODOLOGI

Metode pengembangan yang digunakan mengacu pada kerangka kerja metode *Network Development Life Cycle* (NDLC). NDLC terdiri atas enam tahap yaitu analisis, desain, simulasi, implementasi, *monitoring*, dan manajemen, seperti ditunjukkan pada Gbr. 2.



Gbr. 2 Metode NDLC.

Setiap tahapan merupakan sebuah proses yang menghasilkan keluaran, di mana keluaran tersebut menjadi dasar untuk tahapan selanjutnya. Tahap analisis mencakup analisis kebutuhan fungsional, kebutuhan perangkat keras, dan kebutuhan perangkat lunak. Tahap desain mencakup desain jaringan dan desain sistem berdasarkan spesifikasi kebutuhan yang dibuat. Tahap simulasi dan atau *prototyping* mencakup pembuatan sistem dalam lingkungan percobaan atau perangkat lunak simulasi. Tahap *monitoring* dan manajemen merupakan proses untuk memastikan kelancaran sistem setelah implementasi [9].

A. Analisis Kebutuhan Perangkat Keras

Perangkat keras yang digunakan untuk membangun sistem *cluster server* pada makalah ini adalah sebagai berikut.

- Tiga unit komputer dengan pembagian fungsi dua unit sebagai *server* virtualisasi dan satu unit sebagai *server* NAS. Spesifikasi masing-masing komputer dijelaskan pada Tabel I.
- Dua unit *switch* Cisco Catalyst 2960 sebagai perangkat *fencing* dan interkoneksi antar *node*.
- Satu unit laptop difungsikan untuk mengonfigurasi sistem *cluster server* dan sebagai klien.
- Kabel *console* Cisco + USB *to serial converter* sebagai media untuk akses ke *console switch* Cisco Catalyst 2960.
- Kabel UTP *category* 5e sebagai media komunikasi antar *node*.

TABEL I
SPESIFIKASI KOMPUTER

Komponen	**Server**		
	Virtualisasi 1	**Virtualisasi 2**	**NAS**
Prosesor	Intel Pentium CPU G620 2.60GHz	Intel Pentium CPU G620 2.60GHz	Intel Pentium CPU E2200 2.20Ghz
Motherboard	ECS H61H2-M12	ECS H61H2-M12	HP FT917AA-AR6
RAM	4 GB	4 GB	1 GB
HDD	HDD1 320 GB HDD2 500 GB	HDD1 320 GB HDD2 500 GB	HDD1 160 GB HDD2 160 GB
NIC	2 *Fast Ethernet*	2 *Fast Ethernet*	2 *Fast Ethernet*

B. Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan untuk membangun sistem *cluster server* yang mendukung ketersediaan tinggi di atas lingkungan virtual dan perangkat lunak pendukung lainnya untuk penelitian ini adalah sebagai berikut.

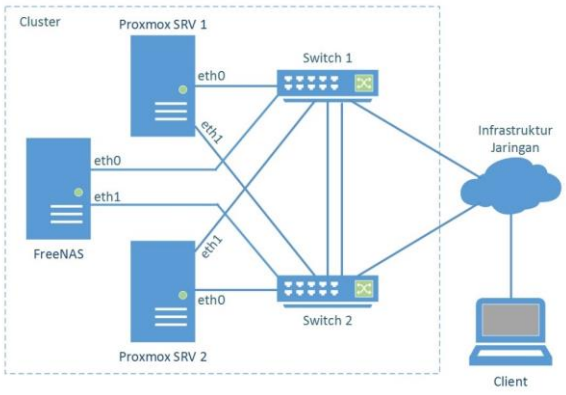
- Proxmox VE 3.4, digunakan untuk mengimplementasikan dan mengelola mesin virtual.
- FreeNAS 9.2.1.9 x86, digunakan untuk membuat *personal computer* (PC) menjadi *server* NAS.
- Ubuntu Server 14.04.3 LTS x64, digunakan sebagai sistem operasi pada mesin virtual. Ubuntu Server 14.04.3 LTS x64 akan didukung hingga tahun 2019. Pembaruan akan meliputi fitur baru perangkat keras komputer, pembaruan keamanan, dan pembaruan Ubuntu Stack (perangkat lunak *cloud computing*).
- Windows Server 2012 R2 Standard x64, digunakan sebagai sistem operasi pada mesin virtual. Windows Server 2012 R2 Standard x64 merupakan salah satu dari produk Microsoft Windows Server. Windows Server 2012 R2 Standard x64 adalah sebuah sistem operasi yang digunakan khusus untuk *server*.
- Putty, digunakan untuk melakukan *remote access* ke *server* virtualisasi dan *switch*. Putty adalah perangkat lunak *remote console*/*terminal* yang digunakan untuk

melakukan *remote access* ke komputer dan perangkat jaringan dengan menggunakan SSH, Telnet, atau *serial*.

- Fping, digunakan untuk mengukur *downtime* pada *server*. Fping merupakan program *ping* sederhana yang memiliki kemampuan melakukan *ping* dengan interval lebih cepat dibanding *ping* standar.
- VSFTPD, digunakan sebagai FTP *server* yang dipasang pada sistem operasi Ubuntu Server 14.04 x64 untuk pengujian integritas data.
- WinMD5, digunakan untuk menghitung nilai MD5 *checksum* dari suatu *file*. Dalam hal ini, WinMD5 digunakan untuk menguji integritas *file* hasil sinkronisasi DRBD.
- FileZilla Client, digunakan sebagai FTP *client* untuk mengirimkan *file dummy* ke FTP *server* dalam pengujian integritas data.
- Google Chrome sebagai *web browser* untuk mengakses *web management* Proxmox VE.
- Windows 10 Education x64, yaitu sistem operasi yang terpasang pada laptop untuk menjalankan Putty dan *web browser* serta sebagai sistem operasi klien.

C. Desain Topologi Fisik

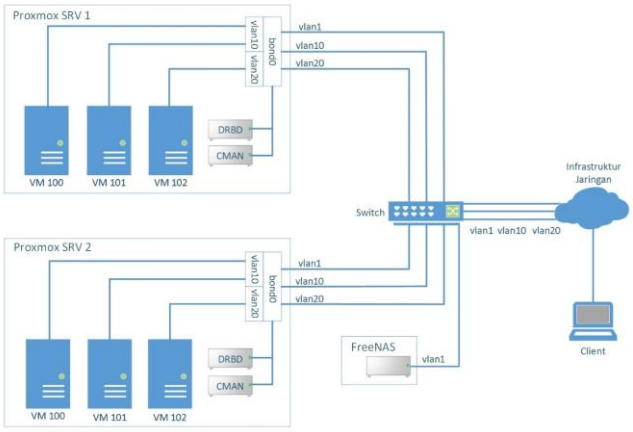
Topologi jaringan secara fisik menggambarkan bagaimana perangkat jaringan terhubung satu sama lain. Desain topologi fisik sistem ini membutuhkan tiga unit *server* di mana dua unit digunakan sebagai *server* virtualisasi dan satu unit digunakan sebagai *server* NAS. Sebagai perangkat *fencing*, sistem ini menggunakan *switch* Cisco Catalyst 2960 karena mendukung protokol SNMP. *Switch* yang digunakan berjumlah dua unit, bertujuan untuk menjaga konektivitas jaringan jika salah satu *switch* mengalami kerusakan. Di luar sistem *cluster server*, agar *server* dapat terhubung ke jaringan lain, dibutuhkan sebuah *router*. *Router* menjadi *gateway* agar terhubung ke jaringan luar. Media komunikasi yang digunakan untuk menghubungkan antar perangkat jaringan adalah kabel UTP. Gbr. 3 menunjukkan desain topologi fisik sistem *cluster server*.



Gbr. 3 Desain topologi fisik.

D. Desain Topologi Logis

Topologi jaringan secara logis menggambarkan bagaimana perangkat berkomunikasi, atau dengan kata lain bentuk komunikasi. Berbeda dengan desain topologi jaringan secara fisik yang dilihat bentuknya secara fisik seperti kabel atau perangkat jaringan fisik lainnya, desain topologi jaringan secara logis lebih terkait dengan protokol jaringan. Topologi jaringan logis bisa sama, lebih sederhana, ataupun lebih rumit dari topologi jaringan fisiknya. Dalam infrastruktur terdapat perangkat *router*. *Router* dalam desain topologi logis ini berfungsi untuk menghubungkan VLAN di dalam *cluster server* dan menghubungkan *cluster server* dengan jaringan luar. Gbr. 4 menunjukkan desain topologi logis dari sistem *cluster server* untuk jaminan ketersediaan layanan tinggi di atas lingkungan virtual.



Gbr. 4 Desain topologi logis.

Mesin virtual yang digunakan dalam penelitian ini ada tiga, yaitu VM 100, VM 101, dan VM 102. Mesin virtual tersebut tersimpan pada kedua *server* virtualisasi hasil dari sinkronisasi DRBD. Spesifikasi detail dari mesin virtual yang dibuat ditunjukkan pada Tabel II.

TABEL II
SPESIFIKASI DETAIL MESIN VIRTUAL

Komponen	Mesin Virtual		
	VM 100	**VM 101**	**VM 102**
Nama	Ubuntu-Server-14.04-OpenVZ	Ubuntu-Server-14.04-KVM	Windows-Server-2012-R2-KVM
Hypervisor	OpenVZ Container	KVM	KVM
Sistem operasi	Ubuntu Server 14.04 x64	Ubuntu Server 14.04 x64	Windows Server 2012 R2 Standard x64
HDD *size*	32 GB	32 GB	32 GB
HDD *bus*	-	IDE	IDE
HDD *format*	-	*Raw*	*Raw*
Storage	drbd-ext3	drbd-vg	drbd-vg
CPU	2 *Core*	2 *Core*	2 *Core*
Memory	1 GB	1 GB	1 GB
NIC	-	*Intel* E1000	*Intel* E1000

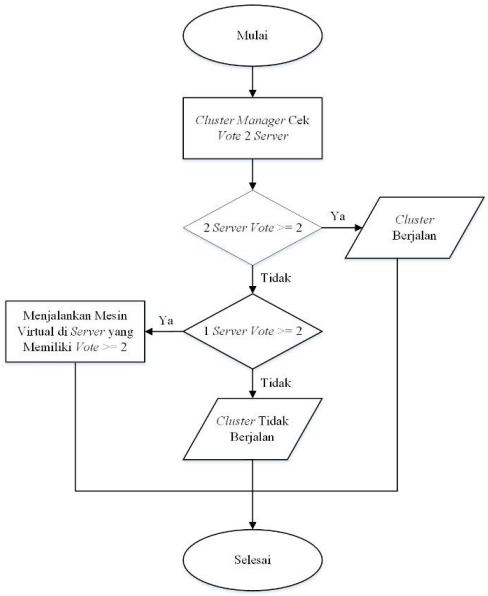
Desain topologi logis sistem *cluster server* ini menggunakan alamat IP privat yang berbeda untuk masing-masing VLAN. Tabel III menunjukkan alamat IP yang digunakan dalam desain topologi logis sistem *cluster server*.

TABEL III
ALAMAT IP SISTEM *CLUSTER SERVER*

Node	***Interface***	**Alamat IP**	***Gateway***
Proxmox SRV 1	bond0	172.16.1.10/24	172.16.1.1
	vlan10	192.168.10.253/24	-
	vlan20	192.168.20.253/24	-
Proxmox SRV 2	bond0	172.16.1.20/24	172.16.1.1
	vlan10	192.168.10.254/24	-
	vlan20	192.168.20.254/24	-
Switch 1	vlan1	172.16.1.11/24	172.16.1.1
Switch 2	vlan1	172.16.1.21/24	172.16.1.1
FreeNAS	lagg0	172.16.1.30/24	172.16.1.1
VM 100	eth0	192.168.10.2/24	192.168.10.1
VM 101	eth0	192.168.10.3/24	192.168.10.1
VM 102	eth0	192.168.20.2/24	192.168.20.1
Klien	eth0	10.10.10.2/24	10.10.10.1

E. Cara Kerja Sistem

Masing-masing *node* dalam *cluster* memiliki satu *vote*. Syarat sebuah *cluster* dapat berjalan adalah terdapat dua *vote* dalam sistem *cluster* dengan tiga *node*. Kondisi tersebut dinamakan kuorum karena jumlah *vote* sudah melebihi setengah dari total *vote*. Berdasarkan Gbr. 5, saat sistem dimulai, *cluster manager* pada masing-masing *server* virtualisasi akan mengecek jumlah *vote* pada *server* virtualisasi. Jika *vote* pada kedua *server* virtualisasi berjumlah tiga, menunjukkan bahwa *cluster* dalam kondisi baik karena semua *node* terhubung dalam *cluster*, maka hasilnya *cluster* akan berjalan dan mesin virtual berjalan normal.



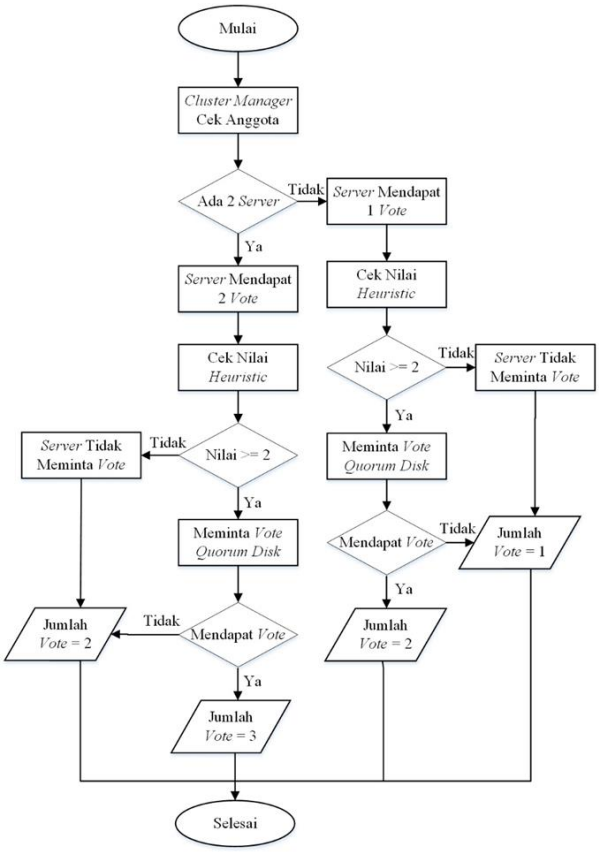
Gbr. 5 Diagram alir cara kerja sistem *cluster server*.

Jika *vote* di kedua *server* virtualisasi berjumlah dua, hal tersebut menandakan bahwa *quorum disk* terputus. Kondisi tersebut tidak mengganggu jalannya cluster karena kedua *server* masih dalam kondisi kuorum. Sistem *cluster* tetap berjalan dan mesin virtual tetap berjalan normal. Jika hanya terdapat satu *server* virtualisasi yang memiliki *vote* lebih dari atau sama dengan dua, hal tersebut menandakan bahwa ada satu *server* virtualisasi yang terputus dari *cluster*.

Jika ada mesin virtual yang berjalan pada *server* virtualisasi yang terputus, maka akan dijalankan ulang pada *server* virtualisasi yang memiliki *vote* lebih dari atau sama dengan dua untuk menjaga mesin virtual tetap dapat diakses oleh klien. Jika tidak ada *server* virtualisasi yang memiliki *vote* lebih dari atau sama dengan dua, maka terjadi kegagalan *cluster* karena tidak ada *server* virtualisasi di dalam *cluster* yang berada pada kondisi baik. Mesin virtual tidak dapat diakses dan layanan terhenti.

Server virtualisasi memiliki proses untuk menentukan jumlah *vote* yang didapat. Dalam proses tersebut terdapat tahapan yang disebut *heuristic*. *Heuristic* dalam sistem *cluster* adalah sebuah algoritme yang menentukan suatu *server* virtualisasi berada dalam kondisi baik atau tidak. Jumlah *heuristic* yang dapat dikonfigurasi maksimal adalah sepuluh. Masing-masing *heuristic* dapat diberi nilai yang berbeda. Sebuah *server* virtualisasi dianggap dalam kondisi baik jika nilai *heuristic* lebih dari setengah total nilai.

Sistem *cluster server* dalam makalah ini menggunakan tiga *heuristic* dengan nilai *heuristic* masing-masing adalah 1. Server virtualisasi dianggap dalam kondisi baik jika mendapatkan nilai *heuristic* 2. Jika *server* virtualisasi memiliki kondisi baik dari hasil *heuristic*, maka *server* tersebut berhak untuk meminta *vote* kepada *quorum disk*. Jika *server* virtualisasi memiliki kondisi tidak baik dari hasil *heuristic*, *server* tersebut tidak dapat meminta *vote* kepada *quorum disk*. Gbr. 6 menjelaskan langkah-langkah dari proses untuk menentukan *vote*.



Gbr. 6 Diagram alir proses menentukan *vote*.

Proses dimulai dari *cluster manager* pada *server* virtualisasi mengecek anggota *cluster*, jika ada dua *server* virtualisasi maka *server* tersebut akan mendapat dua *vote*. Satu *vote* dari *server* virtualisasi itu sendiri dan satu *vote* lainnya dari *server* virtualisasi lain. Selanjutnya, *server* virtualisasi tersebut mengecek nilai *heuristic*, jika nilai *heuristic* lebih dari atau sama dengan 2, *server* virtualisasi akan meminta *vote* tambahan kepada *quorum disk*. Jika *quorum disk* memberikan *vote*, maka total *vote* menjadi tiga, tetapi jika *quorum disk* tidak memberikan *vote* dikarenakan jaringan terputus atau terjadi kegagalan perangkat keras total *vote* tetap dua. Jika nilai *heuristic* kurang dari 2, *server* virtualisasi tidak meminta *vote* tambahan kepada *quorum disk* dan total *vote* tetap dua.

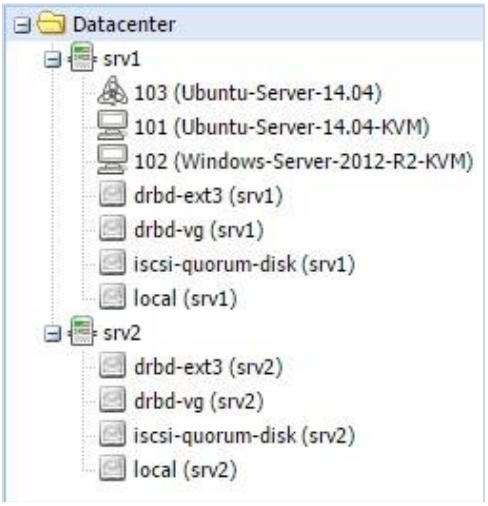
Saat hasil pengecekan anggota *cluster* oleh *cluster manager* pada *server* virtualisasi hanya terdapat satu *server* virtualisasi yaitu *server* virtualisasi yang melakukan pengecekan itu sendiri, maka *server* virtualisasi tersebut mendapatkan satu *vote*. Selanjutnya *server* virtualisasi tersebut mengecek nilai *heuristic*. Jika nilai *heuristic* lebih dari atau sama dengan 2, maka akan meminta *vote* tambahan dari *quorum disk*. jika mendapatkan *vote*, total *vote server* virtualisasi tersebut menjadi dua, tetapi jika tidak mendapatkan *vote* dari *quorum disk*, total *vote* tetap satu. Jika nilai *heuristic* kurang dari 2, *server* virtualisasi tidak meminta *vote* tambahan kepada *quorum disk* dan total *vote* tetap satu.

IV. HASIL PENGUJIAN DAN ANALISIS

A. *Pengujian Kegagalan Komponen Sistem Cluster Server*

Pengujian dilakukan dengan kondisi awal seluruh komponen sistem *cluster server* berjalan normal, kemudian salah satu komponen di dalamnya dimatikan. Kondisi sistem *cluster server* saat terjadi kegagalan di salah satu komponen diamati. Dalam kondisi normal seluruh mesin virtual berjalan pada *server* Proxmox SRV 1 seperti ditunjukkan Gbr. 7.

Perubahan kondisi yang diamati antara lain status *cluster*, jumlah *vote*, status *network bonding*, dan status mesin virtual. Hasil pengujian kegagalan salah satu komponen sistem *cluster server* disajikan pada Tabel IV.



Gbr. 7 Kondisi normal mesin virtual yang berjalan.

B. *Pengukuran Downtime Saat Jalur Aktif Terputus*

Pengukuran dilakukan menggunakan aplikasi Fping dan *stopwatch*. Aplikasi Fping diatur agar melakukan *ping* ke *server* dengan interval 0,1 sekon. Ukuran paket yang dikirimkan menggunakan ukuran terkecil yaitu 1 *byte* untuk menghindari kegagalan pengiriman paket karena disebabkan *server* tidak dapat menjawab permintaan yang besar. *Downtime* dapat dilihat dari jumlah paket yang gagal kemudian dikali 0,1 sekon.

TABEL IV
PENGUJIAN KEGAGALAN SALAH SATU KOMPONEN *CLUSTER*

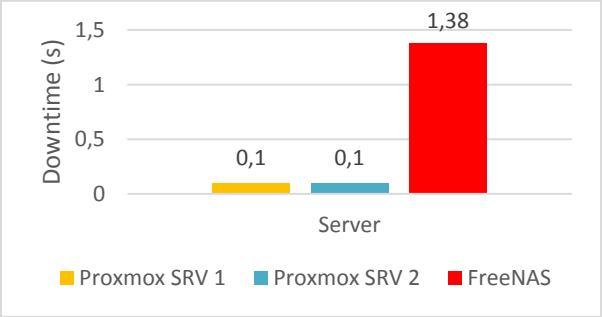
Komponen Gagal	Hasil
Server Proxmox SRV 1	Sistem *cluster* tetap berjalan dengan jumlah *vote* pada *server* Proxmox SRV 2 adalah 2. Mesin virtual berjenis KVM dijalankan ulang secara otomatis pada *server* Proxmox SRV 2 dan mesin virtual berjenis OpenVZ Container gagal dijalankan ulang pada *server* Proxmox SRV 2.
Server Proxmox SRV 2	Sistem *cluster* tetap berjalan dengan jumlah *vote* pada *server* Proxmox SRV 1 adalah 2. Seluruh mesin virtual tetap berjalan pada *server* Proxmox SRV 1.
Server FreeNAS	Sistem *cluster* tetap berjalan dengan jumlah *vote* pada kedua *server* virtualisasi adalah 2. Seluruh mesin virtual tetap berjalan pada *server* Proxmox SRV 1.
Switch 1	Sistem *cluster* tetap berjalan dengan jumlah *vote* pada kedua *server* virtualisasi adalah 3. Seluruh mesin virtual tetap berjalan pada *server* Proxmox SRV 1. Status *active port* pada *server* Proxmox SRV 1 yang sebelumnya adalah eth0 berubah menjadi eth1, artinya trafik *server* Proxmox SRV 1 dilewatkan melalui *Switch* 2.
Switch 2	Sistem *cluster* tetap berjalan dengan jumlah *vote* pada kedua *server* virtualisasi adalah 3. Seluruh mesin virtual tetap berjalan pada *server* Proxmox SRV 1. Status *active port* pada *server* Proxmox SRV 2 yang sebelumnya adalah eth0 berubah menjadi eth1 artinya trafik *server* Proxmox SRV 2 dilewatkan melalui *Switch* 1.

Pengukuran ini dilakukan pada *server* Proxmox SRV 1, *server* Proxmox SRV 2,dan *server* FreeNAS dengan memutus jalur aktif secara bergantian. Pengukuran dilakukan untuk menguji *network bonding* yang sudah dikonfigurasi. Hasil pengukuran *downtime* saat jalur aktif terputus disajikan pada Tabel V.

TABEL V
HASIL PENGUKURAN *DOWNTIME* SAAT JALUR AKTIF TERPUTUS

Pengukuran ke-	*Downtime Server* (s)		
	Proxmox SRV 1	Proxmox SRV 2	FreeNAS
1	0,1	0,1	0,1
2	0,1	0,1	2,3
3	0,1	0,1	0,1
4	0,1	0,1	3,0
5	0,1	0,1	0,1
6	0,1	0,1	2,7
Rerata	0,1	0,1	1,38

Berdasarkan nilai rata-rata pengukuran *downtime* saat jalur aktif terputus pada Tabel V, dapat dibuat grafik agar terlihat lebih jelas hasil pengukuran *downtime* dari ketiga *server*. Grafik rata-rata *downtime* saat jalur aktif terputus ditunjukkan pada Gbr. 8.



Gbr. 8 Grafik *downtime* saat jalur aktif terputus.

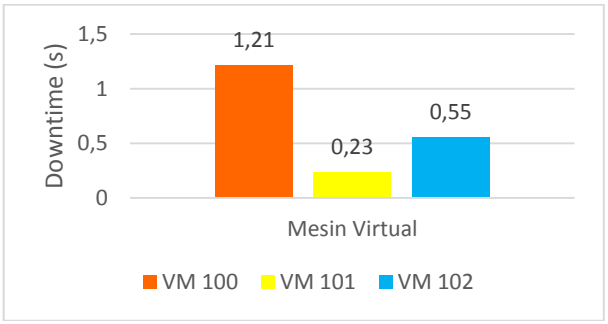
C. *Pengujian Downtime Terjadwal Live Migration*

Pengukuran kedua yaitu pengukuran *downtime* terjadwal dari mesin virtual saat proses *live migration* dan waktu yang dibutuhkan untuk migrasi mesin virtual dari satu *server* virtualisasi ke *server* virtualisasi lainnya. Pengukuran dilakukan pada ketiga mesin virtual yaitu mesin virtual berjenis OpenVZ Container dengan sistem operasi Ubuntu Server 14.04 x64, mesin virtual berjenis KVM dengan sistem operasi Ubuntu Server 14.04 x64, dan mesin virtual berjenis KVM dengan sistem operasi Windows Server 2012 R2 Standard x64. Mesin virtual dipindahkan secara bergantian dari *server* Proxmox SRV 1 menuju *server* Proxmox SRV 2, kemudian kembali ke *server* Proxmox SRV 1. Hasil pengukuran *downtime* saat proses *live migration* dan waktu yang dibutuhkan untuk migrasi mesin virtual disajikan pada Tabel VI.

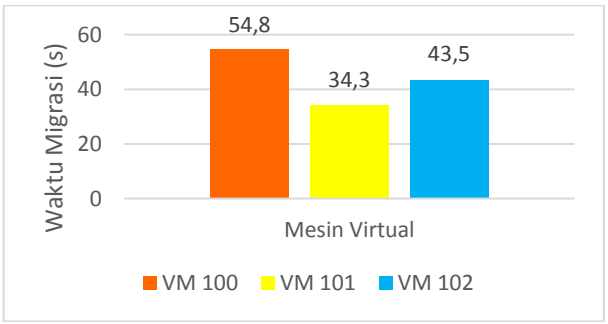
TABEL VI
HASIL PENGUKURAN *DOWNTIME* DAN WAKTU MIGRASI PROSES *LIVE MIGRATION*

Pengukuran ke-	VM 100		VM 101		VM 102	
	***Down time* (s)**	**Waktu Migrasi (s)**	***Down time* (s)**	**Waktu Migrasi (s)**	***Down time* (s)**	**Waktu Migrasi (s)**
1	1,2	51	0,3	23	0,6	42
2	1,2	56	0,2	24	0,5	44
3	1,3	58	0,2	23	0,4	40
4	1,2	56	0,2	26	0,6	42
5	1,2	50	0,3	26	0,7	46
6	1,2	58	0,2	24	0,5	47
Rerata	1,21	54,8	0,23	34,3	0,55	43,5

Berdasarkan nilai rata-rata pengukuran *downtime* saat proses *live migration* dan waktu migrasi mesin virtual pada Tabel VI, dapat dibuat grafik agar terlihat lebih jelas hasil pengukuran dari ketiga mesin virtual. Grafik rata-rata *downtime* saat proses *live migration* ditunjukkan pada Gbr. 9. Grafik rata-rata waktu migrasi ditunjukkan pada Gbr. 10.



Gbr. 9 Grafik *downtime* saat proses live migration.



Gbr. 10 Grafik waktu migrasi mesin virtual.

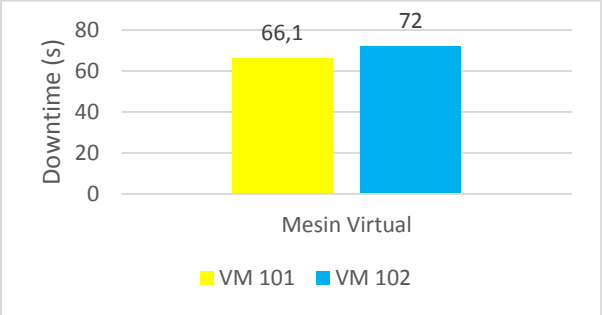
D. *Pengujian Downtime Tidak Terjadwal*

Pengukuran ketiga yaitu pengukuran *downtime* tidak terjadwal dari mesin virtual saat terjadi kegagalan pada salah satu *server* virtualisasi yang menjalankan mesin virtual. Pengukuran dilakukan pada dua mesin virtual yaitu mesin virtual berjenis KVM dengan sistem operasi Ubuntu Server 14.04 x64 dan mesin virtual berjenis KVM dengan sistem operasi Windows Server 2012 R2 Standard x64. Pengukuran tidak dilakukan pada mesin virtual berjenis OpenVZ Container karena tidak dapat dilakukan proses *failover* saat terjadi kegagalan pada *server* virtualisasi. Hasil pengukuran *downtime* tidak terjadwal saat terjadi kegagalan pada *server* virtualisasi disajikan pada Tabel VII.

TABEL VII
HASIL PENGUKURAN *DOWNTIME* TIDAK TERJADWAL

Pengukuran ke-	***Downtime* Mesin Virtual (s)**	
	VM 101	**VM 102**
1	64	76
2	68	68
3	64	70
4	60	75
5	76	70
6	65	73
Rerata	66,1	72

Berdasarkan nilai rata-rata pengukuran *downtime* tidak terjadwal pada Tabel VII, dapat dibuat grafik agar terlihat lebih jelas hasil pengukuran dari kedua mesin virtual. Grafik rata-rata *downtime* saat terjadi *downtime* tidak terjadwal ditunjukkan pada Gbr. 11.



Gbr. 11 Grafik *downtime* tidak terjadwal.

E. Pengujian Integritas Data

Pengujian terhadap integritas data sistem *cluster server* pada makalah ini berkaitan dengan proses sinkronisasi data pada DRBD. Integritas data disebut juga keakuratan data, keutuhan data, atau kebenaran data. Keutuhan data pada *disk* DRBD dapat rusak disebabkan *server* virtualisasi *down* saat data tersebut belum sepenuhnya tersinkronisasi. Tujuan pengujian integritas data ini untuk mengetahui sejauh mana sistem *cluster server* dapat menjamin keutuhan dan keselamatan data pada mesin virtual.

Perangkat lunak yang digunakan yaitu VSFTPD sebagai aplikasi FTP *server* yang dipasang dalam mesin virtual dengan sistem operasi Ubuntu Server 14.04 x64 berjenis KVM, WinMD5 sebagai aplikasi untuk mengecek integritas data menggunakan fungsi MD5 *checksum*, dan FileZilla Client sebagai aplikasi FTP *client* untuk mengirim *file* ke FTP *server*.

Skenario pengujian integritas data adalah dengan mengirim sejumlah *file dummy* dengan ukuran tertentu ke FTP *server* menggunakan FTP *client*. Di tengah proses pengiriman, *server* virtualisasi tempat mesin virtual berjalan dikondisikan *down* dan proses pengiriman terhenti. Setelah proses *failover* dan mesin virtual kembali berjalan, pengiriman dilanjutkan sampai seluruh *file* terkirim. Selanjutnya adalah mengecek jumlah *file* yang ada pada *server* dan mengecek integritas dari masing-masing *file* menggunakan aplikasi WinMD5.

TABEL VIII
HASIL PENGUJIAN INTEGRITAS DATA

Pengujian ke-	Ukuran *File* (KB)	Interval Integritas Data Sebelum *Server Down* (s)
1	1	1
2	1	1
3	2	3
4	2	3
5	1024	4
6	1024	4
7	2048	5
8	2048	5
9	10240	5
10	10240	5
11	20480	5
12	20480	5

Dari waktu *file* diterima sebelum terjadi *down* pada *server* dapat diukur interval waktu di mana data akan tetap utuh saat terjadi gangguan pada *server* virtualisasi yang sedang menjalankan mesin virtual. Untuk *file* berukuran 2 KB yang tetap terjaga integritasnya adalah *file* yang terkirim 3 sekon sebelum *server* virtualisasi mengalami *down*. Pengujian dilakukan untuk *file* dengan ukuran lain. Seluruh hasil pengujian disajikan pada Tabel VIII.

F. Analisis dan Pembahasan

Hasil pengujian kegagalan pada salah satu komponen sistem *cluster server* menunjukkan sistem *cluster server* tetap berjalan meskipun terjadi kegagalan pada salah satu komponennya. Penggunaan *network bonding* pada *server* Proxmox SRV 1, *server* Proxmox SRV 2, dan *server* FreeNAS yang terhubung ke *Switch* 1 dan *Switch* 2 mendukung fungsi *failover* perangkat *switch*. Saat salah satu perangkat *switch* mengalami kegagalan, masing-masing *server* masih memiliki jalur yang terhubung ke *switch* aktif.

Cluster berjalan menggunakan menggunakan sistem *vote*. *Cluster* dapat berjalan jika *vote* yang didapat lebih dari setengah jumlah anggota *cluster*. Agar *cluster* mendukung toleransi kegagalan pada salah satu anggotanya, maka ditambahkan *quorum disk* sebagai *vote* tambahan. Total *vote* dalam *cluster* adalah tiga. Jadi saat salah satu anggota *cluster* mengalami kegagalan, baik itu *server* virtualisasi ataupun *server* NAS, total *vote* dalam *cluster* masih berjumlah dua dan *cluster* tetap berjalan.

Hasil pengukuran *downtime* saat jalur aktif terputus yang dilakukan pada ketiga *server* menunjukkan bahwa untuk *server* Proxmox SRV 1 dan *server* Proxmox SRV 2 tidak ada perbedaan. *Downtime* yang terjadi dari seluruh hasil pengukuran adalah 0,1 sekon. Cara kerja *network bonding* pada Proxmox yaitu saat jalur utama yang aktif (eth0) terputus maka jalur cadangan (eth1) akan menjadi aktif maksimal setelah 100 ms, sesuai dengan interval pemantauan yang dilakukan oleh MII. Saat jalur utama kembali terhubung, jalur aktif tetap pada jalur cadangan, dan jika jalur cadangan terputus maka jalur aktif kembali ke jalur utama dengan *downtime* yang sama. Untuk *server* FreeNAS, terdapat perbedaan *downtime* setiap kali jalur utama kembali terhubung. Hal tersebut dikarenakan cara kerja *network bonding* FreeNAS yang berbeda dengan Proxmox. Saat jalur utama terputus maka jalur aktif berpindah ke jalur cadangan dengan *downtime* 0,1 sekon, tetapi saat jalur utama kembali terhubung, jalur aktif langsung berpindah kembali ke jalur utama. Saat proses tersebut terjadi *downtime* lebih dari 2 sekon dikarenakan *interface* pada *switch* masih dalam status *blocking* sebelum proses STP pada *switch* selesai dilakukan.

Hasil pengukuran *downtime* terjadwal dan waktu migrasi saat proses *live migration* pada tiga mesin virtual, yaitu VM 100 dengan sistem operasi Ubuntu Server 14.04 x64 dan jenis mesin virtual OpenVZ Container, VM 101 dengan sistem operasi Ubuntu Server 14.04 x64 dan jenis mesin virtual KVM, serta VM 102 dengan sistem operasi Windows Server 2012 R2 Standard x64 dan jenis mesin virtual KVM menunjukkan beberapa perbedaan.

Downtime yang terjadi dan waktu migrasi dari mesin virtual berjenis OpenVZ Container lebih lama dibandingkan dengan mesin virtual berjenis KVM. Hal tersebut disebabkan oleh perbedaan proses pemindahan data antara dua jenis mesin virtual tersebut. Proses *live migration* pada OpenVZ Container

perlu memindahkan seluruh isi *container* dari satu *server* virtualisasi ke *server* virtualisasi lainnya. Setelah itu mesin virtual pada *server* virtualisasi sumber dihentikan dan *file* konfigurasi OpenVZ Container dipindahkan ke *server* virtualisasi tujuan. Mesin virtual dilanjutkan kembali dan *memory page* dari *server* virtualisasi sumber dikirim ke *server* virtualiasi tujuan. Waktu proses pemindahan file konfigurasi merupakan *downtime* yang terjadi.

Proses *live migration* pada KVM adalah dengan menyalin seluruh *memory page* mesin virtual dari *server* virtualiasi sumber ke *server* virtualisasi tujuan saat mesin virtual berjalan. Jika saat proses menyalin terdapat perubahan pada *memory page* sumber, maka *memory page* pada *server* virtualisasi tujuan akan ditandai. Mesin virtual pada *server* virtualisasi sumber dihentikan, dan *memory page* yang ditandai akan disalin ulang. Waktu proses menyalin ulang sampai mesin virtual dilanjutkan merupakan *downtime* yang terjadi. *Downtime* dan waktu migrasi mesin virtual dengan sistem operasi Ubuntu Server 14.04 x64 lebih singkat dibandingkan mesin virtual dengan sistem operasi Windows Server 2012 R2 Standard x64, karena dalam kondisi konfigurasi *default, memory page* sistem operasi Ubuntu Server 14.04 x64 lebih kecil dari sistem operasi Windows Server 2012 R2 Standard x64.

Hasil pengukuran *downtime* tidak terjadwal, yaitu saat *server* virtualisasi mengalami kegagalan menunjukkan hasil bahwa mesin virtual dengan sistem operasi Ubuntu Server 14.04 x64 memiliki rata-rata *downtime* lebih singkat dibandingkan mesin virtual dengan sistem operasi Windows Server 2012 R2 x64. Perbedaan terjadi disebabkan waktu *booting* masing-masing sistem operasi sampai *services networking* dijalankan berbeda.

Hasil pengujian integritas data pada sinkronisasi DRBD menunjukkan hasil bahwa secara keseluruhan sistem *cluster server* dalam penelitian dalam makalah ini dapat menjamin integritas suatu data pada interval 5 sekon sebelum *server* virtualisasi *down* dan proses *failover* mesin virtual dilakukan. Pengecualian terjadi untuk *file* dengan ukuran di bawah 1 KB yang tetap utuh pada interval 1 sekon sebelum *server* virtualisasi *down*, dikarenakan proses sinkronisasi yang cepat. Penggunaan protokol C pada konfigurasi DRBD masih memberi *delay* maksimal 5 sekon pada *file* dengan ukuran di atas 1 KB untuk selesai tersinkronisasi.

V. KESIMPULAN

Berdasarkan hasil pengujian dan analisis, dapat diambil beberapa kesimpulan. Rancang bangun sistem *cluster server* yang dapat memberikan jaminan ketersediaan layanan tinggi di atas lingkungan virtual berhasil dibangun dengan menggunakan Proxmox VE 3.4 sebagai *server* virtualisasi pada dua unit komputer, FreeNAS 9.2.1.9 x86 sebagai *server* NAS, dan dua unit *switch* Cisco Catalyst 2960. Integritas dan keamanan data saat terjadi kegagalan pada salah satu *server* virtualisasi dapat dijaga dengan menggunakan DRBD untuk mereplikasi *disk* lokal *server* virtualisasi. *Hypervisor* atau jenis mesin virtual yang didukung oleh Proxmox VE adalah OpenVZ Container dan KVM. *Hypervisor* yang mendukung HA saat terjadi kegagalan *server* virtualisasi adalah KVM.

Penggunaan *network bonding*/*link aggregation* memberikan HA pada server saat jalur utama terputus dengan *downtime* 0,1 sekon. Proses *live migration* menggunakan sistem operasi yang sama, yaitu Ubuntu Server 14.04 x64 untuk mesin virtual jenis KVM memiliki rata-rata *downtime* 0,23 sekon, dan waktu migrasi 34,3 sekon lebih singkat dibanding mesin virtual jenis OpenVZ Container yang memiliki rata-rata *downtime* 1,21 sekon dan waktu migrasi 54,8 sekon. Waktu yang dibutuhkan untuk melakukan *live migration* dan lama *downtime* yang terjadi dipengaruhi oleh cara migrasi dari masing-masing *hypervisor* dan *memory page* sistem operasi yang berjalan. *Downtime* yang terjadi saat proses *failover* mesin virtual dipengaruhi waktu yang butuhkan sistem operasi untuk *booting*. Sistem *cluster server* dalam makalah ini dapat menjamin integritas data pada interval 5 sekon sebelum terjadi kegagalan pada *server* virtualisasi.

REFERENSI

[1] A. Arfriandi, "Perancangan, Implementasi, dan Analisis Kinerja Virtualisasi Server Menggunakan Proxmox, VMware ESX, dan Openstack," *Jurnal Teknologi*, vol. 5, no. 2, pp. 182–191, 2012.

[2] K. Schmidt, *High Availability and Disaster Recovery: Concepts, Design, Implementation*. Heidelberg: Springer, 2006.

[3] R. Dines, "The State Of Disaster Recovery Preparedness," *Disaster Recovery Journal*, vol. 24, no. 1, pp. 1–3, 2011.

[4] A. B. M. Moniruzzaman, M. Waliullah, and M. Rahman, "A High Availability Clusters Model Combined with Load Balancing and Shared Storage Technologies for Web Servers," *International Journal of Grid Distribution Computing*, vol. 8, no. 1, pp. 109–120, 2015.

[5] M. Portnoy, *Virtualization Essentials*. Indianapolis: John Wiley & Sons, Inc., 2012.

[6] Oracle and L. C. Miller, *Server Virtualization For Dummies®, Oracle Special Edition*. Hoboken: John Wiley & Sons, Inc., 2012.

[7] S. M. C. Cheng, *Proxmox High Availability*. Birmingham: Packt Publishing Ltd., 2014.

[8] G. Sims, *Learning FreeNAS*. Birmingham: Packt Publishing Ltd., 2008.

[9] J. E. Goldman and P. T. Rawles, *Applied Data Communications, A Bussiness-Oriented Approach*. John Wiley & Sons, Inc., 2001.