IMPLEMENTASI DAN ANALISA SISTEM FAILOVER VIRTUAL COMPUTER CLUSTER

SKRIPSI

TANIA RIZKY FEBRIANI

NPM. 0706276116

TEKNIK KOMPUTER

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

JUNI 2011

IMPLEMENTASI DAN ANALISA SISTEM FAILOVER VIRTUAL COMPUTER CLUSTER

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

TANIA RIZKY FEBRIANI

NPM. 0706276116

TEKNIK KOMPUTER

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

JUNI 2011

HALAMAN PERNYATAAN ORISINALITAS

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

Nama	**: Tania Rizky Febriani**
NPM	**: 0706276116**
Tanda Tangan	**:** 
Tanggal	**: 15 Juni 2011**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Tania Rizky Febriani
NPM	: 0706276116
Program Studi	: Teknik Komputer
Judul Skripsi	: Implementasi dan Analisa Sistem Failover Virtual Computer Cluster

Telah berhasil dipertahankan dihadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjanah Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

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

Penguji : Ir. Endang Sriningsih MT, Si ( )

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

Ditetapkan di : Depok

Tanggal : 23 Juni 2011

KATA PENGANTAR

Puji syukur saya panjatkan kepada Allah SWT., karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Program Studi Teknik Komputer Departemen Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

(1) Dr. Ir. Anak Agung Putri Ratna, M.Eng, selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(2) Orang tua dan keluarga saya yang telah memberikan bantuan dukungan material dan moral; dan
(3) Teman-teman Teknik Komputer 2007 dan Teknik Elektro 2007 yang telah banyak membantu saya dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap Allah SWT. berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 15 Juni 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Universitas Indonesia, saya bertanda tangan di bawah ini :

Nama : Tania Rizky Febriani
NPM : 0706276116
Program studi : Teknik Komputer
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Nonoksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul :

IMPLEMENTASI DAN ANALISA SISTEM FAILOVER VIRTUAL COMPUTER CLUSTER

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta sebagai pemegang Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 10 Juni 2010

Yang menyatakan

Tania Rizky Febriani

ABSTRAK

Nama	:	Tania Rizky Febriani
Program Studi	:	Teknik Komputer
Judul	:	Implementasi dan Analisa Sistem Failover Virtual Computer Cluster
Pembimbing	:	Dr. Ir. Anak Agung Putri Ratna, M.Eng

Saat ini sedang marak dibicarakan suatu teknologi yang menggabungkan beberapa sumber daya yang bekerja bersama-sama sehingga tampak seolah-olah merupakan satu *single system*, teknologi ini disebut *cluster*. Kebutuhan masyarakat akan informasi yang selalu *up to date* meningkatkan kebutuhan akan sistem yang memiliki *availability* tinggi. Hal ini semakin krusial karena masyarakat semakin haus akan informasi. Untuk itu dikembangkan suatu sistem yang disebut *failover cluster* yang memiliki tingkat *availability* tinggi. Skripsi ini membahas mengenai implementasi sistem *failover virtual computer cluster* dan analisa performanya dibandingkan dengan sistem yang tidak di-cluster. *Failover cluster* dirancang untuk sebuah sistem dengan *availability* tinggi. Sistem ini diimplementasikan menggunakan Windows Server 2008 R2 x64 sebagai OS-nya dan Hyper-V sebagai platform simulasinya. Sistem *failover virtual computer cluster* ini terdiri atas tiga buah *nodes* berupa *virtual machine* dengan sebuah *storage* yang terhubung dengan ketiga *nodes* menggunakan iSCSI. Analisa performa dilakukan terhadap lima parameter benchmark berbeda yaitu *processor arithmetic, .NET arithmetic, memory bandwidth, memory latency* dan *cache* dan memori. Hasil *benchmark* performa sistem *failover virtual computer cluster* untuk kelima parameter ini tidak lebih baik dibandingkan dengan sistem yang tidak di-*cluster*, dimana terjadi penurunan rata-rata keseluruhan sebesar 41% untuk skenario pertama dan 54 % untuk skenario kedua. Namun dari segi *availibility* sistem *failover virtual computer cluster* ini lebih unggul dibandingkan sistem yang tidak di-*cluster*.

Kata kunci*: Cluster, Virtualisasi,, Computer Cluster, Failover, komputer virtual*

ABSTRACT

Name	:	Tania Rizky Febriani
Study Program	:	Teknik Komputer
Title	:	Implementation and Analysis of Failover Virtual Computer Cluster System
Supervisor	:	Dr. Ir. Anak Agung Putri Ratna, M.Eng

Now a day there is a technology that much discuss for its ability to combine some resource that work together so that it looks like a single system, this technology is called cluster. The public need of information that up to date increase the public need of system that is high availability. This even more crucial because public is getting thirsty of information. This thesis discusses about the implementation of the failover virtual computer cluster system and analysis of its performance compared to system that is not part of a cluster. Failover cluster was designed for a high availability system. This failover virtual computer cluster system is implemented using Windows Server 2008 R2 x 64 as its operating system and Hyper-V as its virtualization platform. This failover virtual computer cluster system consists of three nodes that form by virtual machines with a storage that is connected to each nodes using iSCSI. Performance analysis carried out on five different parameters of benchmark they are, processor arithmetic, .NET arithmetic, memory bandwidth, memory latency, and cache and memory. Failover virtual computer cluster system performance benchmark results for the fifth parameter is not better than the system that is not part of a cluster. The results show that there are decreases of performance about 41% for the first scenario and 54% for the second scenario. But in terms of system availability failover virtual computer cluster system is better than the system that is not part of a cluster.

Key words: *Cluster, Virtualization, Failover, Computer Cluster, Virtual Computer*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Saat ini informasi merupakan kebutuhan yang sangat penting bagi masyarakat. Informasi dituntut untuk selalu *up to date* dan beberapa sektor kehidupan tidak mentolerir adanya kegagalan atau keterlambatan akibat adanya gangguan pada sistem. Contohnya saja sektor bisnis saham yang sedikit saja terjadi keterlambatan pada sistem komputerisasinya maka akan menanggung kerugian yang amat besar.

Suatu teknologi yang saat ini sedang dibicarakan adalah *cluster computer* yang merupakan suatu teknologi yang memanfaatkan beberapa sumber daya komputer tunggal yang kemudian bekerja bersama-sama sehingga tampak seperti satu sistem yang terintegrasi. Konsep cluster ini sedang banyak dikembangkan karena kelebihannya yaitu bisa menghasilkan suatu sistem dengan tingkat realibilitas tinggi dan sistem yang memiliki tingkat *availability* tinggi. Hal ini bergantung dari tujuan awal pembuatan sistem cluster computer tersebut. Dengan teknologi ini kebutuhan akan sistem yang memiliki tingkat *availability* tinggi dapat dicapai.

Apabila sedikit dikaitkan dengan isu lingkungan saat ini, suatu sistem cluster dapat mengurangi sampah elektronik yaitu komputer, karena sampah komputer yang awalnya tidak dapat digunakan karena kemampuannya sudah tertinggal dibandingkan komputer-komputer yang ada saat ini dapat digabungkan menjadi suatu cluster computer yang nantinya akan membentuk suatu super computer. Dengan ini salah satu konsep penghijauan lingkungan yaitu penggunaan kembali dapat diimplementasikan.

Teknologi *clustering* pada dasarnya adalah implementasi virtualisasi pada *hardware* untuk seolah-olah "menyatukan" dua atau lebih perangkat keras pada *server nodes* sehingga nantinya didapat satu buah sistem yang memiliki kemampuan setara dua atau lebih *nodes* yang tadi di-*cluster*-kan. Bentuk topologi seperti ini memang tidak mendukung sebuah sistem supercomputer, namun topologi sistem *cluster-enabled* ini bisa dimanfaatkan untuk melakukan komputasi-komputasi tingkat tinggi yang biasanya tidak sanggup dijalankan

sebuah sistem tunggal, dan tanpa harus mengeluarkan biaya untuk mengadakan sebuah supercomputer[1].

Konsep ini merupakan konsep yang baik untuk diterapkan, hanya saja masih belum banyak yang menggunakan konsep ini. Konsep virtualisasi yang dilakukan masih untuk skala kecil seperti di warung internet yaitu menggunakan konsep *thin client*. Apabila konsep virtualisasi berupa cluster ini diterapkan untuk skala yang lebih besar seperti di server maka diharapkan dapat mengurangi lebih banyak lagi energi yang digunakan dan juga sumber daya yang digunakan. Konsep virtualisasi dalam bentuk cluster ini sudah didukung dengan berbagai perangkat lunak seperti Hyper-V untuk Windows, VMware yang tersedia untuk Windows dan Linux dan Citrix Xen Server.

Pada skripsi ini penulis akan mengimplementasikan suatu sistem *failover virtual computer cluster* yang memanfaatkan teknologi virtualisasi yang sebelumnya telah dirancang pada tahap seminar. *S*istem ini kemudian akan dianalisis performansinya dengan membandingkan dengan sistem yang tidak di*cluster* menggunakan beberapa *software benchmark*.

1.2 Tujuan Penulisan

Tujuan dari skripsi ini adalah mengimplementasikan suatu sistem *failover virtual computer cluster* yang memanfaatkan teknologi virtualisasi yang sebelumnya telah dirancang pada tahap seminar untuk kemudian dijadikan bahan pembelajaran guna menguji performansi sistem *failover virtual computer cluster* berdasarkan parameter *processor arithmetic, .NET arithmetic, memory bandwidth, memory latency,* dan *cache* dan *memory* dibandingkan dengan sistem yang tidak di-*cluster*.

1.3 Pembatasan Masalah

Pada skripsi ini yang dibahas dibatasi pada implementasi rancangan sistem *cluster computer* dalam bentuk virtual menggunakan teknik virtualisasi yang disediakan oleh Windows Server 2008 R2 x64 melalui role Hyper-V. Masalah dibatasi hanya pada kegagalan pada server atau *virtual machine* tidak pada *storage*-nya.

1.4 Identifikasi Masalah

Dalam penulisan skripsi ini, penulis mengidentifikasikan masalah berdasarkan tema dan judul yang telah ditetapkan. Berikut ini adalah identifikasi masalah dalam skripsi ini :

1. Bagaimana merancang suatu jaringan tervirtualisasi dengan teknik *clustering*?
2. Bagaimana mengimplementasikan suatu failover cluster yang terdiri atas *virtual machine* sebagai *nodes*-nya?
3. Apakah clustering berhasil meningkatkan efektifitas dan performansi jaringan?
4. Apakah *cluster* memiliki performa yang lebih baik dibandingkan sistem yang tidak di-*cluster*?

1.5 Metodologi Penulisan

Metode penulisan pada skripsi ini adalah:

a. Studi literatur, yaitu mencari buku-buku, jurnal dan sumber-sumber lainnya untuk dijadikan bahan referensi.
b. Pendefinisian masalah, yaitu mendefinisikan masalah apa saja yang akan dibahas dalam skripsi ini.
c. Perancangan sistem, yaitu merancang desain sistem *failover virtual computer cluster* yang akan dibuat.
d. Implementasi dan Uji Coba, yaitu menerapkan rancangan yang telah dibuat dan kemudian diuji coba untuk diambil datanya sebagai bahan analisa.

1.6 Sistematika Penulisan

Skripsi ini terdiri dari empat bab, dimana masing-masing bab akan menjelaskan sebagai berikut:

a. Bab 1: Pendahuluan

   Pada bab ini, akan dijelaskan mengenai Latar Belakang, Tujuan, Pembatasan Masalah, Metodologi Penulisan, dan Sistematika penulisan.

b. Bab 2: Konsep Sistem *Failover Virtual Computer Cluster*

   Pada bab ini, akan dijelaskan tentang landasan atau dasar teori yang digunakan dalam penulisan skripsi ini yang berhubungan dengan masalah yang dibahas yaitu mengenai sistem *cluster computer,* virtualisasi dan hal-hal yang mendukungnya seperti *software* dan aplikasi pendukung.

c. Bab 3: Perancangan Sistem *Failover Virtual Computer Cluster*

Pada bab ini, akan dijelaskan mengenai perancangan sistem virtualisasi dengan teknik *clustering* yaitu rancangan topologi sistem jaringan virtualisasi dan algoritma pengerjaan.

d. Bab 4: Implementasi dan Analisa Sistem *Failover Virtual Computer Cluster*

Pada bab ini, akan dibahas mengenai proses implementasi dari rancangan sistem *failover virtual computer cluster* yang telah dibahas pada bab 3. Proses implementasi ini terdiri dari proses instalasi dan konfigurasi sistem serta analisa hasil *benchmark.*

e. Bab 5: Kesimpulan

Pada bab ini, akan dijelaskan mengenai kesimpulan yang dapat diambil dari pembahasan skripsi ini.

BAB 2
KONSEP SISTEM *FAILOVER VIRTUAL COMPUTER CLUSTER*

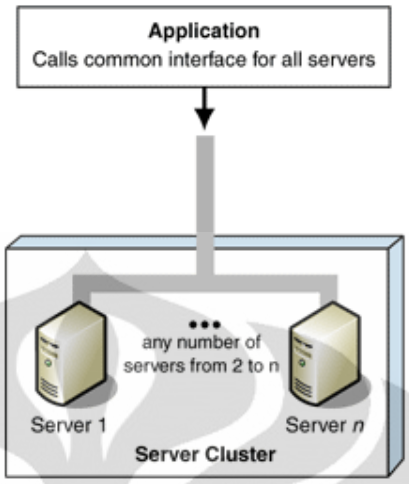
2.1 Konsep Virtualisasi

Virtualisasi merupakan suatu istilah umum yang mengacu kepada abstraksi dari sumber daya komputer. Definisi lainnya adalah "sebuah teknik untuk menyembunyikan karakteristik fisik dari sumber daya komputer dari bagaimana cara sistem lain, aplikasi atau pengguna berinteraksi dengan sumber daya tersebut. Hal ini termasuk membuat sebuah sumber daya tunggal (seperti server, sebuah sistem operasi, sebuah aplikasi, atau peralatan penyimpanan terlihat berfungsi sebagai beberapa sumber daya logikal; atau dapat juga termasuk definisi untuk membuat beberapa sumber daya fisik (seperti beberapa peralatan penyimpanan atau server) terlihat sebagai satu sumber daya logikal."[2]

Konsep virtualisasi yang digunakan dalam skripsi ini yaitu membuat beberapa mesin virtual dalam satu sumber daya fisik dalam hal ini komputer terlihat sebagai satu sumber daya logical melalui teknik *clustering* yang selanjutnya akan membentuk suatu *virtual computer cluster.* Teknik *clustering* ini akan dibahas lebih lanjut di sub bab berikutnya.

2.2 Pengenalan *Cluster Computer*

Cluster, dalam ilmu komputer dan jaringan komputer adalah sekumpulan komputer (umumnya server jaringan) independen yang beroperasi serta bekerja secara erat dan terlihat oleh klien jaringan seolah-olah komputer-komputer tersebut adalah satu buah unit komputer. Proses menghubungkan beberapa komputer agar dapat bekerja seperti itu dinamakan dengan *Clustering*. Komponen *cluster* biasanya saling terhubung dengan cepat melalui sebuah interkoneksi yang sangat cepat, atau bisa juga melalui jaringan lokal (LAN). Konsep dasar dari kerja *cluster computer* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Konsep Dasar *Cluster* [3].

Karena menggunakan lebih dari satu buah *server*, maka manajemen dan perawatan sebuah *cluster* jauh lebih rumit dibandingkan dengan manajemen *server mainframe* tunggal yang memiliki skalabilitas tinggi (semacam IBM AS/400), meski lebih murah.

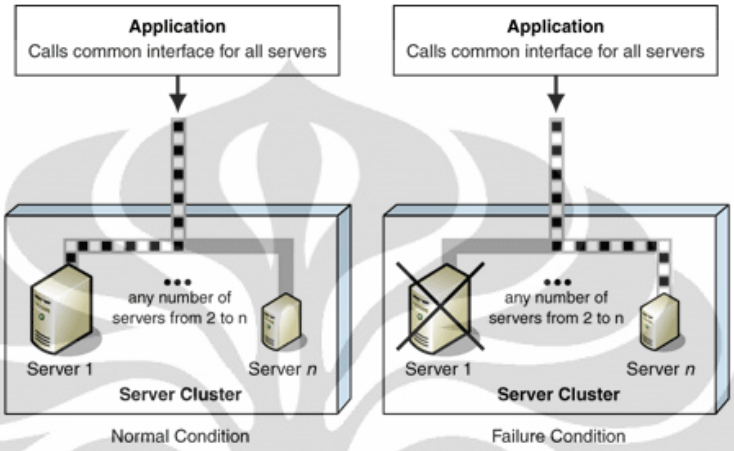
2.3 Prinsip Kerja *Cluster* Computer

Prinsip kerja *cluster* ada beberapa macam bergantung pada tujuan dari pembuatan *cluster* itu sendiri. Secara umum ada dua macam prinsip kerja *cluster* yang umum digunakan yaitu *cluster* dengan ketersedian tinggi dan *cluster* dengan reabilitas tinggi. Namun konsep dari *cluster* itu sendiri tetap sama yaitu sekumpulan komputer (umumnya server jaringan) independen yang beroperasi serta bekerja secara erat dan terlihat oleh klien jaringan seolah-olah komputer-komputer tersebut adalah satu buah unit komputer. Dimana yang membedakan hanyalah bagaimana prinsip kerja masing-masing *node*s dalam *cluster* tersebut.

2.3.1 Asymmetric *Cluster*s

Pada *asymmetric clusters*, sebuah *standby server* yang ada hanya untuk mengambil alih *server* lain apabila terjadi *failure*. *Cluster* tipe ini biasanya digunakan untuk mendukung *availability* dan skalabilitas tinggi untuk *read/write stores* seperti layanan *databases, messaging systems, file* dan *print*. Jika salah satu dari *nodes* di *cluster* tidak tersedia, dikarenakan *downtime* yang direncanakan

untuk pemeliharaan atau *downtime* yang tidak direncanakan yaitu kegagalan sistem, maka *node* lain akan mengambil alih fungsi dari *node* yang *down*. Salah satu contoh *asymmetric cluster* adalah *failover cluster* [3].



Gambar 2.2: Asymmetric *cluster* [3].

Gambar 2.2 mengilustrasikan bagaimana *asymmetric cluster* menampilkan sumber daya virtual ke aplikasi. Pada kondisi normal, server utama akan melayani semua permintaan. Pada saat kondisi *down, standby server* akan mengambil alih kerja server utama dan melayani semua permintaan.

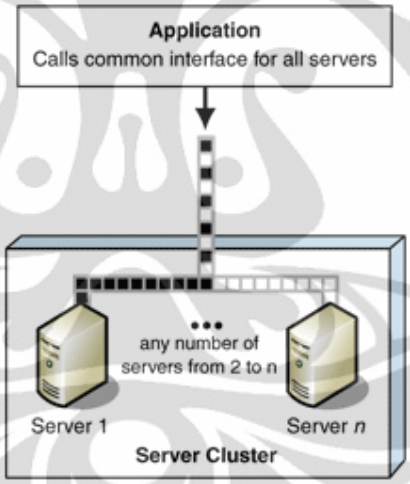
Konsep *cluster* ini adalah sebuah kumpulan komputer yang independen yang bekerja bersama-sama sebagai sebuah *single system* untuk menjamin suatu aplikasi dan sumber daya yang penting dan sifatnya *critical* sebisa mungkin berada pada keadaan *high availability.* Kumpulan komputer ini dikelola sebagai sebuah *single system*, ia akan saling berbagi *namespace* yang sama, dan kumpulan komputer ini secara khusus dirancang untuk mengurangi *downtime* yang disebabkan kegagalan *software* dan *hardware*.

Pada *cluster* jenis ini seperti yang dapat dilihat pada Gambar 2.2 ada beberapa *node*s berupa komputer yang merupakan anggota dari *cluster* yang bekerja baik aktif maupun pasif. *Node*s yang aktif akan menjalankan sebuah servis atau aplikasi dimana pasif *node*s berada dalam kondisi siaga (standby) namun tetap saling berkomunikasi dengan *node*s yang aktif sehingga mereka dapat

mengidentifikasi apabila terjadi kegagalan (*failure)*. Pada saat terjadi kegagalan, *node* pasif kemudian menjadi aktif dan mulai menjalankan servis dan aplikasi.

2.3.2 Symmetric *Cluster*s

Pada *symmetric clusters*, setiap server dalam *cluster* menjalankan kerja yang sama berguna. Umumnya, setiap server merupakan server utama untuk satu set aplikasi tertentu. Jika satu server gagal, server sisanya akan melanjutkan proses yang ditugaskan kepadanya begitu pula dengan aplikasi yang ada di server yang gagal. *Symmetric clusters* lebih hemat biaya karena mereka lebih sering memanfaatkan lebih banyak sumber daya yang ada di *cluster*. Namun, pada kondisi *failure,* beban tambahan yang ada di *server* yang tersisa dapat menyebabkan kegagalan sistem juga [3].



Gambar 2.3. Symmetric *cluster* [3]

Gambar 2.3 mengilustrasikan bagaimana *symmetric cluster* menampilkan sumber daya virtual ke aplikasi. Permintaan dibagikan kepada server dengan kondisi baik untuk mendistribusikan bebab dan skalabilitas.

Salah satu contoh *symmetric cluster* adalah *load-balanced cluster. Load-balanced clusters* meningkatkan performa, *availability* dan skalabilitas dari layanan seperti *Web servers, media servers, VPN servers,* dan *read-only stores* oleh distribusi permintaan diantara seluruh sever yang kondisinya baik di dalam area *cluster*.

2.4 Macam-Macam *Cluster*

Cluster komputer terbagi ke dalam beberapa kategori, sebagai berikut:

a. *High-availability clusters*

High-availability cluster, yang juga sering disebut sebagai *Failover Cluster* pada umumnya diimplementasikan untuk tujuan meningkatkan *availability* layanan yang disediakan oleh *cluster* tersebut. Elemen *cluster* akan bekerja dengan memiliki *node-node* redundan, yang kemudian digunakan untuk menyediakan layanan saat salah satu elemen *cluster* mengalami kegagalan. Ukuran yang paling umum dari kategori ini adalah dua *node*, yang merupakan syarat minimum untuk melakukan redundansi. Implementasi *cluster* jenis ini akan mencoba untuk menggunakan redundansi komponen *cluster* untuk menghilangkan kegagalan di satu titik (*Single Point of Failure*).

b. *Load-balancing clusters*

Cluster kategori ini beroperasi dengan mendistribusikan beban pekerjaan secara merata melalui beberapa *node* yang bekerja di belakang *(back-end node)*. Umumnya *cluster* ini akan dikonfigurasikan sedemikian rupa dengan beberapa *front-end load-balancing* redundan. Karena setiap elemen dalam sebuah *cluster load-balancing* menawarkan layanan penuh, maka dapat dikatakan bahwa komponen *cluster* tersebut merupakan sebuah *cluster* aktif/*cluster* HA aktif, yang bisa menerima semua permintaan yang diajukan oleh klien.

c. *Compute clusters*

Umumnya, penggunaan utama *cluster* komputer adalah untuk tujuan komputasi, dibandingkan penanganan operasi yang berorientasi I/O seperti layanan Web atau basis data. Perbedaan utama untuk kategori ini dengan kategori lainnya adalah seberapa eratkah penggabungan antar *node*-nya. Sebagai contoh, sebuah tugas komputasi mungkin membutuhkan komunikasi yang sering antar *node* ini berarti bahwa *cluster* tersebut menggunakan sebuah jaringan terdedikasi yang sama, yang terletak di lokasi yang sangat berdekatan, dan mungkin juga merupakan *node-node* yang bersifat homogen. Desain *cluster* seperti ini, umumnya disebut juga

sebagai *Beowulf Cluster*. Ada juga desain yang lain, yakni saat sebuah tugas komputasi hanya menggunakan satu atau beberapa *node* saja, dan membutuhkan komunikasi antar-*node* yang sangat sedikit atau tidak ada sama sekali. Desain *cluster* ini, sering disebut sebagai "Grid". Beberapa *compute cluster* yang dihubungkan secara erat yang didesain sedemikian rupa, umumnya disebut dengan "Supercomputing". Beberapa perangkat lunak *Middleware* seperti MPI atau *Parallel Virtual Machine* (PVM) mengizinkan program *compute clustering* agar dapat dijalankan di dalam *cluster-cluster* tersebut.

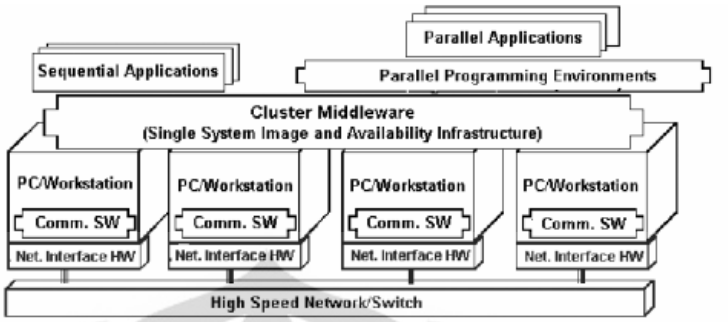
d. *Grid computing*

Grid pada umumnya adalah *compute cluster*, tapi difokuskan pada *throughput* seperti utilitas perhitungan ketimbang menjalankan pekerjaan-pekerjaan yang sangat erat yang biasanya dilakukan oleh *Supercomputer*. Seringnya, *grid* memasukkan sekumpulan komputer, yang bisa saja didistribusikan secara geografis, dan kadang diurus oleh organisasi yang tidak saling berkaitan.

Grid computing dioptimalkan untuk beban pekerjaan yang mencakup banyak pekerjaan independen atau paket-paket pekerjaan, yang tidak harus berbagi data yang sama antar pekerjaan selama proses komputasi dilakukan. *Grid* bertindak untuk mengatur alokasi pekerjaan kepada komputer-komputer yang akan melakukan tugas tersebut secara independen. Sumber daya, seperti halnya media penyimpanan, mungkin bisa saja digunakan bersama-sama dengan komputer lainnya, tapi hasil sementara dari sebuah tugas tertentu tidak akan mempengaruhi pekerjaan lainnya yang sedang berlangsung dalam komputer lainnya.

2.5 *Cluster Computer Architecture*

Sebuah *cluster computer* adalah salah satu jenis sistem pem-prosesan paralel (*parallel processing system*) atau sistem pem-prosesan terdistribusi (*distributed processing system*), yang terdiri dari kumpulan komputer yang berdiri sendiri yang saling terhubung untuk melakukan kerja sama sebagai sebuah sumber daya yang terintegrasi. Gambar 2.4 menggambarkan bentuk arsitektur dari *cluster computer* [4].



Gambar 2.4 *Cluster* Computer Architecture [4]

Berikut ini bagian-bagian dalam arsitektur *cluster computer*:

- *Multiple high performance computers (PCs, Workstations, SMP servers).*
- *Operating systems (Layered or Micro-Kernel based).*
- *High Performance Networks (Switched network fabrics, Gigabit Ethernet).*
- *Cluster Middleware (Single System Image, and System Availability Infrastructure).*
- *Parallel Programming Environments.*
- Aplikasi.

2.6 Komponen dalam *Cluster Computer*

Cluster Computer terdiri atas empat komponen utama. Setiap komponen memiliki fungsi khusus yang dibutuhkan agar *hardware* yang ada dapat bekerja. Berikut ini keempat komponen utama dari *cluster computer*[1].

1. *Network*:

- Mendukung komunikasi antara *nodes, server,* dan *storage*
- Terdiri atas Fast Ethernet, switch, cables, dan beberapa *hardware* jaringan.

2. *Nodes*:

- Bekerja sebagai *processor* dalam *cluster*.
- Setiap *node* dapat dipertukarkan, karena tidak ada perbedaan fungsi diantara *nodes* tersebut.
- Terdiri dari semua komputer dalam *cluster* selain *gateway* dan *server*

3. *Server*

- Mendukung layanan jaringan untuk *cluster*.

- DHCP
- NFS (*Node image* dan *shared file system*)
- Menjalankan program secara parallel dan menjalankan proses di *nodes*
- Harus memiliki persyaratan minimum.

4. *Storage*:

- Bekerja sebagai tempat penyimpanan aplikasi dan service yang akan di-share kepada *node-node*.
- Aplikasi dan service yang akan dijalankan oleh *node*s diambil dari storage.

2.7 *Software*

2.7.1 Windows Server 2008 Hyper-V

Windows Server 2008 R2 x64 merupakan *operating system* yang digunakan dalam implementasi prototype sistem *cluster computer* ini. Windows Server 2008 merupakan sistem operasi untuk server yang dikembangkan oleh Microsoft dan merupakan hasil pengembangan lanjutan dari Windows Server 2003. Sama halnya dengan Windows Vista dan Windows 7, Windows Server 2008 berbasis pada Windows NT 6.x. Windows Server 2008 merupakan nama baru dari Windows Server "Longhorn".

Pemilihan Windows Server 2008 sebagai OS dari sistem ini dikarenakan sistem operasi ini mendukung teknologi virtualiasi dengan fitur-fiturnya. Beberapa fitur yang diperlukan dalam implementasi sistem ini antara lain Failover *Clustering*, Hyper-V dan iSCSI *Initiator*, dimana ketiga fitur ini sangat penting dalam pembuatan *cluster*.

Untuk membangun suatu sistem *cluster computer* dibutuhkan suatu *software* pendukung. Ada beberapa *software* yang dapat digunakan untuk membangun suatu sistem *cluster computer* seperti VM-Ware, Xen Server, dan lain sebagainya. Namun dalam sistem yang akan dibuat, yang akan digunakan adalah Hyper-V. Seperti yang telah disebutkan sebelumnya Hyper-V merupakan fitur dari Windows Server 2008 R2 x64 yang bisa menggantikan *software* yang telah disebut di atas, sehingga memudahkan proses implementasinya.

Hyper-V merupakan suatu teknologi virtualisasi *hypervisor-based* untuk x64 versi Windows Server 2008. *Hypervisor* merupakan platform virtualisasi berbasis-

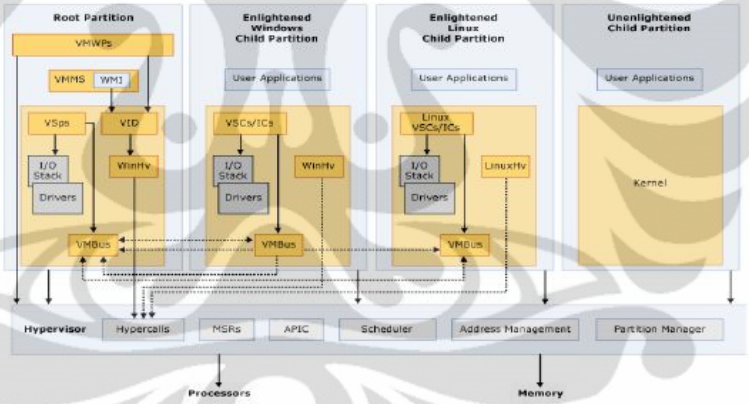
prosesor yang memungkinkan beberapa sistem operasi terisolasi untuk berbagi platform perangkat keras tunggal (*single hardware*). Hyper-V merupakan fitur utama dalam pembuatan virtualisasi dalam *cluster*.

Hyper-V mendukung isolasi dalam hal partisi. Partisi adalah unit logic dari isolasi, yang didukung oleh *hypervisor*, di mana sistem operasi mengeksekusi. Microsoft *Hypervisor* harus memiliki minimal satu partisi parent, yang menjalankan Windows Server 2008 Edition 64-bit. Stack virtualisasi berkerja dipartisi parent dan memiliki akses langsung ke perangkat keras. Partisi parent ini kemudian akan membuat partisi child yang akan menjadi *host* dari GuestOS. Sebuah partisi parent membuat partisi child menggunakan interface pemrograman aplikasi hypercall (API).

Partisi tidak memiliki akses ke prosesor fisik, juga tidak menangani interupsi prosesor. Sebaliknya, mereka memiliki *virtual view* atas prosesor dan dijalankan di alamat memori virtual yang bersifat *private* untuk setiap partisi tamu. *Hypervisor*-lah yang menangani intrupsi terhadap prosesor, dan mengarahkan mereka ke partisi masing-masing. Hyper-V juga bisa mempercepat terjemahan alamat antara ruang tamu berbagai alamat virtual secara hardware dengan menggunakan *Input Output Memory Management Unit* (IOMMU) yang beroperasi independen dari *Memory Management Hardware* yang digunakan oleh CPU. Sebuah IOMMU digunakan untuk memetakan alamat memori fisik ke alamat yang digunakan oleh partisi *child*.

Partisi *child* juga tidak memiliki akses langsung ke sumber daya perangkat keras lainnya namun disajikan virtual view atas resource, sebagai *Virtual Device* (VDevs). Permintaan pada VD akan diarahkan melalui VM Bus atau *hypervisor* untuk perangkat di partisi *parent*, yang akan menangani permintaan tersebut. VM Bus merupakan saluran komunikasi logikal antar-partisi. Partisi *parent* menyediakan *Virtualization Service Provider* (VSPs) yang berkomunikasi melalui VM Bus untuk menangani permintaan akses perangkat dari partisi *child*. Partisi anak menyediakan *Virtualization Service Consumers* (VSCs) yang mengarahkan permintaan perangkat ke VSPs di partisi parent melalui VM Bus. Seluruh proses ini bersifat transparan untuk GuestOS.

VD juga dapat mengambil keuntungan dari fitur virtualisasi Windows Server, bernama *Enlightened* I/O, untuk *storage*, *network*, *graphics*, dan *input subsystem*. *Enlightened* I/O adalah implementasi khusus virtualisasi pada protocol komunikasi tingkat tinggi (seperti SCSI), yang langsung memanfaatkan VM Bus, dan melewati setiap lapisan emulasi pada perangkat. Hal ini membuat komunikasi lebih efisien, tetapi membutuhkan *Enlightened Guest* yang kompatibel terhadap *hypervisor* dan VM Bus. Hyper-V *Enlightened* I / O dan kernel yang kompatibel dengan *hypervisor* disediakan melalui layanan instalasi Hyper-V. Integrasi komponen, yang mencakup *virtual client server* (VSC) driver, juga tersedia untuk sistem operasi klien yang lain. Hyper-V membutuhkan prosesor yang sudah mendukung virtualisasi *Hardware*, seperti Intel VT atau AMD Virtualization (AMDV). Gambar 2.5 memberikan gambaran tingkat tinggi dari arsitektur lingkungan Hyper-V yang berjalan pada Windows Server 2008 [6].



Gambar 2.5 Arsitektur Hyper-V [6].

Singkatan dan istilah yang digunakan dalam diagram di atas dijelaskan sebagai berikut:

- APIC - *Advanced Programmable Interrupt Controller* - Sebuah perangkat yang memungkinkan tingkat prioritas diatur sesuai *Interrupt*.
- *Child Partition* - Partisi yang menghandle GuestOS.
- *Hypercall - Interface* untuk komunikasi dengan *hypervisor*.
- *Hypervisor* - Suatu lapisan perangkat lunak yang berada *hardware* dan sistem operasi satu atau lebih. Pekerjaan utamanya adalah untuk menyediakan lingkungan eksekusi yang terisolasi yang disebut partisi.

- IC – *Integrated Component* - Komponen yang memungkinkan partisi child untuk berkomunikasi dengan partisi lain dan *hypervisor* nya.
- I/O *stack - Input / Output stack.*
- MSR – *Memory Service Routine* - Layanan Memori Rutin.
- Partisi *Parent* - Mengatur *machine-level function*, seperti *device driver*, manajemen daya, dan penambahan perangkat panas / penghapusan. Partisi *parent* (atau *root*) adalah satu-satunya partisi yang memiliki akses langsung ke memori fisik dan perangkat.
- VID - *Virtualization Infrastructure Driver* - Menyediakan jasa manajemen alam di antara partisi, jasa prosesor virtual manajemen, dan jasa manajemen memori untuk partisi.
- VM Bus – Mekanisme komunikasi berbasis kanal yang digunakan untuk komunikasi antar-partisi dan penghitungan perangkat pada sistem dengan beberapa partisi virtualisasi aktif. VMBus diinstal melalui *installer* Hyper-V yang terintegrasi.
- VMMS - *Virtual Machine Management Service* - Bertanggung jawab untuk mengelola keadaan dari semua mesin virtual pada partisi *child*.
- VMWP - *Virtual Machine Worker Procces* – komponen *user mode* virtualisasi stack. Proses pekerja ini menyediakan jasa manajemen mesin virtual dari Windows Server 2008 instance di partisi parent untuk GuestOS di partisi *child*. *Virtual Machine Management Service* menumbuhkan proses pekerja terpisah untuk setiap mesin virtual yang berjalan.
- VSC – *Virtualization Service Client* - Sebuah perangkat sintetis yang berada di partisi child. VSCs memanfaatkan sumber daya perangkat keras yang disediakan oleh *Virtualization Service Provider* (VSPs) di partisi parent. Mereka berkomunikasi dengan VSPs pada partisi *parent* melalui VMBus untuk memenuhi permintaan I/O di partisi *child*.
- VSP - *Virtualisasi Service Provider* - Berada di partisi parent dan memberikan dukungan untuk partisi *child* melalui *Virtual Machine Bus* (VMBus).
- WinHv - *Windows Hypervisor Interface Library* - WinHv pada dasarnya adalah sebuah jembatan antara driver sistem operasi dan *driver hypervisor*

yang memungkinkan untuk memanggil *hypervisor* menggunakan *Windows Calling Convention*.

- WMI - VMMS memperlihatkan satu set *Windows Management Instrumentation* (WMI) berbasis API untuk mengelola dan mengendalikan VM.

Fitur kedua adalah Failover *Cluster*. Fitur ini dibutuhkan untuk membangun *cluster* itu sendiri. Penggabungan tiap *node*s anggota *cluster*, validasi *cluster*, testing pada *cluster* dan manajemen *cluster* dilakukan oleh fitur ini untuk itu fitur ini sangat dibutuhkan untuk implementasi skripsi ini.

Fitur ketiga adalah iSCSI *initiator*. Fitur ini penting karena seperti yang telah disebutkan sebelumnya bahwa salah satu komponen dalam *cluster* adalah storage. Untuk menghubungkan storage dengan *node*s *cluster* digunakan iSCSI. Sedangkan iSCSI itu ada yang merupakan *target* dan ada yang merupakan inisiator. iSCSI *target* adalah storage itu sendiri dan iSCSI *initiator* adalah para *node*s dalam *cluster*. Jadi iSCSI *initiator* ini akan menginisiasikan tiap *node*s sebagai *initiator*. Sedangkan untuk membuat suatu iSCSI *target* dibutuhkan *software* tambahan lain yaitu *Starwind* yang akan dibahas lebih lanjut di sub bab selanjutnya.

2.7.2 *Starwind Software*

Dalam membuat suatu *cluster* dibutuhkan sebuah storage area network. Storage area network (SAN) merupakan suatu jaringan yang *dedicated* yang menyediakan akses gabungan pada level *storage*. Pada dasarnya SAN digunakan untuk membuat perangkat storage dapat diakses oleh server sehingga perangkat ini tampak seperti tersedia secara lokal pada *operating system*. Sebuah SAN umumnya memiliki network khusus yang secara umum tidak dapat diakses melalui regular network dan regular device. *Starwind* merupakan suatu *software* untuk membuat storage area network dan iSCSI *target*.

Seperti yang telah disebutkan sebelumnya dalam menghubungkan storage dan *node*s dibutuhkan suatu iSCSI *target* dan iSCSI *initiator*. SAN akan terbentuk dengan meng-*install Starwind* iSCSI SAN *software* dan *host* tempat di-*install*-nya *software* disanalah iSCSI *target* berada. Jadi pada implementasinya nanti *software*

ini akan di-*install* pada VM yang akan berperan sebagai storage. Sehingga VM itu dapat diidentifikasi oleh *node*s melalui fitur iSCSI *initiator* sebagai iSCSI *target*. Dan keduanya akan terhubung melalui network khusus untuk koneksi iSCSI.

2.7.3 *SiSoftware Sandra*

Si*Software Sandra* (***S**ystem **AN**alyser, **D**iagnostic and **R**eporting **A**ssistant*) merupakan *software* utilitas yang garis besar modul-modul yang dimilikinya meliputi diagnosis, pelaporan informasi, dan *benchmarking*. Kapabilitas ini sesuai dengan nama Sandra sendiri yang merupakan akronim dari *System Analyser, Diagnostic and Reporting Assistant.*

Sandra sendiri seperti disebutkan di atas bukan menunjukkan nama seseorang tapi merupakan suatu akronim. Namun Sandra sendiri merupakan nama dari bahasa Yunani yang apabila diartikan memiliki makna "*pelindung*" atau "*penyelamat umat manusia*" yang dirasa sesuai dengan perannya.

SiSoft Sandra ini digunakan untuk kebutuhan *benchmark*ing pada skripsi ini. Sisoft Sandra ini nantinya akan digunakan untuk menghitung beberapa parameter yaitu *processor arithmetic, .NET arithmetic, memory bandwidth, memory latency,* dan *cache and memory*. Detail mengenai parameter ini akan dijelaskan lebih lanjut pada subbab selanjutnya.

Salah satu metode *benchmark* yang digunakan oleh *Sisoft Sandra* adalah metode *benchmarking* sintetis yaitu Whetstone dan Dhrystone. Whetstone merupakan jenis *benchmark* yang dimaksudkan untuk mengukur kinerja komputer dalam mengolah bilangan *floating point* dan digunakan untuk membandingkan arsitektur maupun kompilator teroptimisasi yang dijalankannya. Sedangkan Dhrystone merupakan *benchmark* sintetik difokuskan untuk mengukur kinerja komputer atas bilangan integer dan string.

2.8 Kelebihan dan Kekurangan *Cluster Computer*

Setiap sistem pasti memiliki kekurangan dan kelebihan masing-masing. Sama halnya dengan *cluster computer, cluster computer* memiliki beberapa kelebihan yang memotivasi banyak pihak untuk membangun sistem seperti ini. Kelebihan tersebut antara lain:

- Meningkatkan skalabilitas.

Suatu sistem *cluster computer* memungkinkan aplikasi untuk mengatasi lebih banyak beban, dengan begitu skalabilitas dari sistem akan meningkat.

- Ketersedian yang lebih tinggi (*Higher Availability*).

  Cluster computer membantu aplikasi menghindari interupsi yang terjadi pada layanan, sehingga tingkat *availability*-nya menjadi lebih tinggi.

- Fleksibilitas yang lebih besar (*Greater flexibility*).

  Kemampuan *clustering* untuk menyajikan suatu kesatuan sumber daya komputer virtual yang mendukung personnel IT yang memiliki lebih banyak pilihan untuk mengkonfigurasi infrastruktur untuk mendukung performa, *availability*, dan skalabilitas (*scalability*) aplikasi sesusai dengan kebutuhan.

Selain kelebihan di atas ada juga beberapa kekurangan dari *computer cluster* yang perlu diketahui dan dipertimbangkan dalam membangun sistem ini. Kekurangan dari *cluster computer* antara lain:

- Meningkatkan kompleksitas dari infrastruktur.

  Beberapa desain *clustering* secara signifikan meningkatkan kompleksitas dari infrastruktur yang dibangun, dimana hal ini dapat mempengaruhi kebutuhan operasional dan *support*. Contohnya, *clustering* dapat meningkatkan jumlah *server* untuk mengatur, *storage devices* untuk memelihara, dan *network connections* untuk konfigurasi dan monitoring.

- Dibutuhkan desain dan kode tambahan.

  Suatu aplikasi kadang membutuhkan desain khusus dan perubahan *coding* agar berhungsi dengan baik saat digunakan pada infrastruktur yang menggunakan *clustering*. Contohnya, kebutuhan untuk mengelola dapat menjadi lebih sulit untuk beberapa server dan dapat membutuhkan perubahan *coding* untuk mengakomodasi pengelolaan sehingga informasi tidak hilang apabila sebuah *server* gagal (*fail*).

- *Incompatibility*.

  Aplikasi yang sudah ada atau komponen aplikasi kemungkinan tidak mendukung teknologi *clustering* Contohnya, sebuah keterbatasan dalam teknologi yang digunakan untuk mengembangkan aplikasi atau komponen tidak dapat mendukung *clustering* walaupun sudah dilakukan perubahan *code*.

2.9 Aspek yang akan diukur

Performansi dari suatu sistem komputer dapat diukur berdasarkan beberapa indikator performa. Terkait dengan keterlibatan tingkat kompleksitas, tidak ada satu pun pengukuran yang dapat memberikan ukuran yang benar-benar akurat dari suatu performa sistem komputer. Indikator yang berbeda dibutuhkan untuk mengukur aspek yang berbeda. Berikut ini beberapa indicator untuk pengukuran performa yang akan diukur:

- *Processor Arithmetic*

Benchmarks pada bagian ini dilakukan terhadap unit ALU dan FPU dari *processor*. Hasilnya akan menunjukan bagaimana *processor* memproses instruksi *arithmetic* dan *floating point*.

- .NET *Arithmetic*

Pada bagian ini yang dilakukan adalah menguji performa aritmatika (integer dan *floating point*) dari .NET CLR (run-time engine). Hasilnya akan menunjukan bagaimana *processor* memproses operasi .NET.

- *Memory Bandwidth*

*Benchmark*s pada bagian ini dilakukan terhadap *memory bandwidth*. Hasilnya akan menunjukan bagaimana kinerja dari *memory sub-systems*

- *Memory Latency*

Pada bagian ini yang dilakukan adalah menguji *latency (response time)* dari *cache*s dan *memory processor*. Hasilnya akan menunjukan bagaimana kinerja *cache*s and *memory sub-systems* dari *processor*.

Latensi dari *cache* diukur berdasarkan *clock processor* yaitu berapa lama *clock* yang dibutuhkan suatu data sampai siap digunakan dan hal ini bergantung pada kecepatan *clock processor*. Sedangkan latensi dari memori diukur dalam nanoseconds dan tidak bergantung pada kecepatan *clock processor*. Pengukuran latensi ini dilakukan dalam dua cara yaitu:

- ***Random Memory Access***: proses pengaksesan memori dilakukan secara random.
- ***Linear Memory Access***: proses pengaksesan memori dilakukan secara sekuensial.

- *Cache* dan *Memory*

Benchmark dilakukan terhadap *caches* dan *memory access* dari *processor* (*transfer speed*). Hasilnya akan menunjukan bagaimana kinerja *caches* and *memory sub-systems* dari *processor* dalam hal kecepatan transfer.

Tabel 2.1 Parameter yang akan diukur beserta satuannya [7].

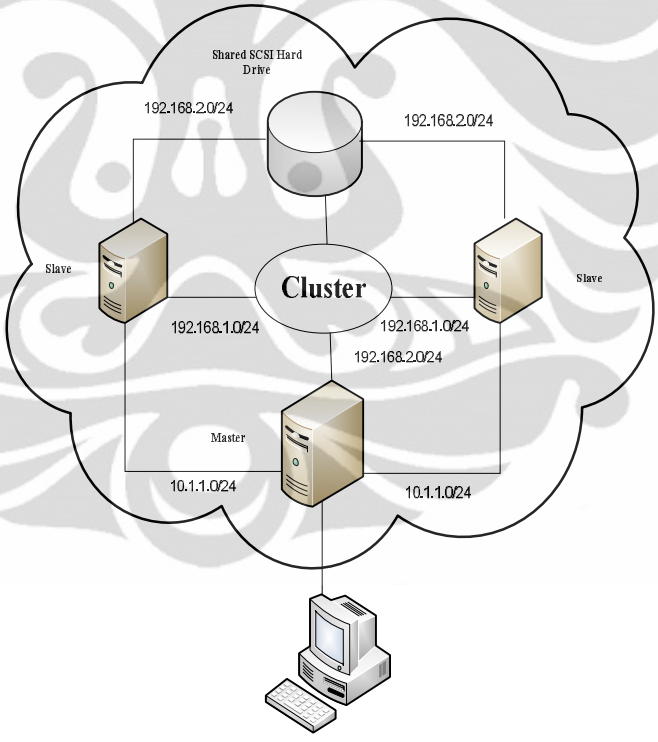
Category of Measure	Unit of Measure	Unit Description	Description of Test
		Processor Arithmetic	
Aggregate *Arithmetic* Performance	GOPS	Giga Operations Per Second	Aggregate *Arithmetic* Performance (GOPS)
Dhrystone ALU	GIPS	Giga Instructions Per Second	Dhrystone ALU (GIPS)
Whetstone iSSE3	GFLOPS	Giga *Floating Point* Operations Per Second	Whetstone iSSE3 (GFLOPS)
		.NET *Arithmetic*	
Aggregate .NET Performance	GOPS	Giga Operations Per Second	Aggregate .NET Performance (GOPS)
Dhrystone .NET	GIPS	Giga Instructions Per Second	Dhrystone .NET(GIPS)
Whetstone .NET	MFLOPS/ GFLOPS	Mega *Floating Point* Operations Per Second / Giga *Floating Point* Operations Per Second	Whetstone .NET (GFLOPS)
		***Memory* Bandwidth**	
Aggregate *Memory* Performance	GB/s	Gigabytes Per Second	Aggregate *Memory* Performance (GB/s)
Int Buff'd iSSE2 *Memory* Bandwidth	GB/s	Gigabytes Per Second	Int Buff'd iSSE2 *Memory* Bandwidth (GB/s)
Float Buff'd iSSE2 *Memory* Bandwidth	GB/s	Gigabytes Per Second	Float Buff'd iSSE2 *Memory* Bandwidth (GB/s)
		Memory Latency	
Random	ns	Nanosecond	Random (ns)
Linear	ns	Nanosecond	Linear (ns)
		Cache* and *Memory	
***Cache*/*Memory* Bandwidth**	GB/s	Gigabytes Per Second	*Cache*/*Memory* Bandwidth (GB/s)
Integrated Data *Cache*	GB/s	Gigabytes Per Second	Integrated Data *Cache* (GB/s)
L2 On-board *Cache*	GB/s	Gigabytes Per Second	L2 On-board *Cache* (GB/s)

Tabel 2.1 merupakan tabel yang menunjukkan detail parameter-parameter yang akan diukur pada tahap implementasi. Seperti yang dapat dilihat pada tabel di atas tiap parameter memiliki unit pengukuran yang bermacam-macam disesuaikan dengan pengukuran yang dilakukan.

BAB 3
PERANCANGAN SISTEM *FAILOVER VIRTUAL COMPUTER CLUSTER*

3.1 Topologi Sistem *Cluster Computer*

Perancangan Sistem *Cluster Virtual Computer* yang akan dibuat dimulai dari rancangan topologi dari sistem *cluster virtual computer* tersebut. Gambar 3.1 menunjukkan rancangan topologi sistem *cluster virtual computer*, dimana sudah diketahui bahwa konsep *cluster* adalah sekumpulan komputer yang terkoneksi seluruhnya bekerja bersama sebagai satu kesatuan sumber daya atau sistem. Hanya saja pada rancangan ini dan implementasinya yang akan di-*cluster* adalah bentuk *virtual* dari komputer yang sebenarnya yaitu *virtual machine*. *Cluster* yang akan dirancang ini secara *default* akan terbentuk sebagai *failover cluster*.



Gambar 3.1 Rancangan Topologi Sistem *Cluster Virtual Computer*

Dari gambar 3.1 dapat dilihat bahwa sistem *cluster* tersebut dibuat dalam sebuah *personal computer* (PC) yang didalamnya dibuat tiga buah *virtual machine* dimana ketiganya akan menjadi *nodes* dari *cluster* yang akan dibuat. Dalam pembuatan *cluster* tiap *node* memiliki perannya masing-masing yaitu satu sebagai

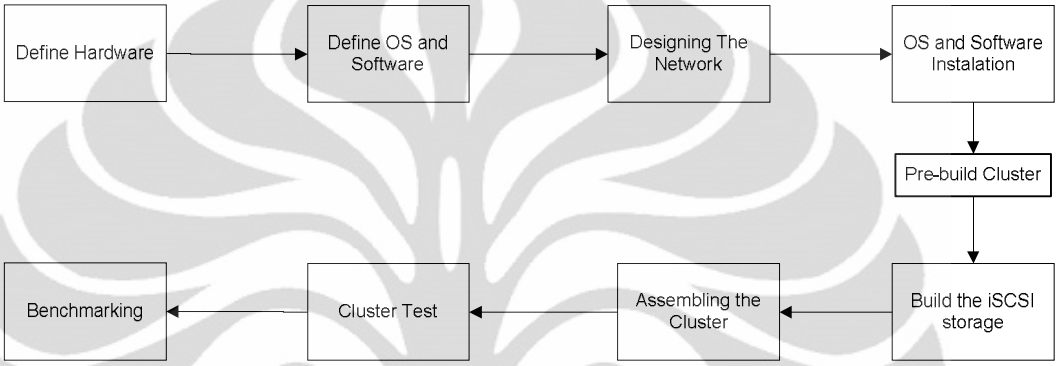
domain controller atau dapat juga disebut sebagai *master node* dan dua sebagai *member node* atau dapat juga disebut *slave.* Prinsip kerja yang akan dijalankan oleh ketiga *nodes* ini sama dengan yang dijelaskan pada bagian prinsip kerja *cluster* di bab sebelumnya. Ketiga *nodes* akan terhubung dengan *storage* melalui *simulated iSCSI* [4][8].

Antar *nodes* pada *cluster* akan terhubung satu sama lain melalui suatu *network*, dikarenakan sistem ini bentuknya *virtual* maka *network* yang dibentuk pun bentuknya *virtual*. Pada implementasinya nanti akan ada tiga buah *network* yang digunakan yaitu public *network* yang diwakilkan oleh VLAN 2, heartbeat *network* yang diwakilkan oleh VLAN 3 dan iSCSI *network* yang diwakilkan oleh VLAN 4. *Domain controller node* hanya terhubung dengan VLAN 2 dan VLAN 4 karena tugasnya sebagai *master, domain controller* merupakan *node* yang terhubung langsung dengan dunia luar sehingga dibutuhkan public *network* dalam melakukan komunikasinya serta sebagai bagian dari *cluster* sudah tentu dibutuhkan koneksi ke *storage*. Untuk member *nodes*, keduanya terhubung dengan ketiga *network* yaitu VLAN 2, VLAN 3 dan VLAN 4. VLAN 3 berfungsi untuk komunikasi antara kedua member *nodes*, sedangkan VLAN 4 seperti disebutkan sebelumnya nama *network*-nya adalah iSCSI sehingga sudah jelas fungsinya adalah sebagai *network* khusus yang menghubungkan seluruh *cluster nodes* dengan *storage*. Setiap *node* dihubungkan pada storage, namun storage tidak dibagi. Tetapi dapat diakses oleh setiap *node*. Hanya aktif *node* yang mengakses storage dan pasif *node*s dikunci oleh layanan *cluster* untuk mengakses storage. Detail mengenai IP dan konfigurasi dari tiap-tiap *network* akan dijelaskan pada sub bab selanjutnya[9].

Private network digunakan oleh layanan *cluster* sehingga *node*s dapat berkomunikasi satu sama lain dan memverifikasi bahwa satu sama lain berada dalam kondisi *up* dan sedang berjalan. Proses verifikasi ini menggunakan dua proses yang disebut *looksalive resource check* dan *isalive resource check.* Jika sumber daya di dalam *cluster* gagal merespon pada saat *resource check* dalam jangka waktu tertentu, pasif *node* akan berasumsi bahwa aktif *node* mengalami kegagalan dan akan mulai menjalankan servis dan aplikasi yang berjalan pada *cluster*[10].

iSCSI adalah singkatan dari *Internet Small Computer System Interface* yang merupakan sebuah protokol penyimpanan jaringan pada jaringan TCP/IP. Konsep dasar dari iSCSI adalah menggunakan perintah perintah SCSI dan membungkusnya kedalam paket TCP/IP untuk mentransmisikan data dari media penyimpanan ke komputer dalam jaringan.

3.2 Algoritma Perancangan Sistem *Cluster Computer*



Gambar 3.2 Blok Diagram Perancangan Sistem *Cluster Computer*

Gambar 3.2 menunjukan tahapan dalam mendesain suatu sistem *cluster computer*. Dimana dapat dilihat pembuatan sistem *cluster computer* dimulai dengan menentukan spesifikasi komputer yang akan digunakan dimulai dari memori, *processing power* sampai dengan *disk space* [11]. Berikut ini *hardware requirement* dari sistem yang akan diimplementasikan [12]:

- **Processor:** x64 *compatible processor* dengan Intel VT atau AMD-V *technology enabled. Hardware Data Execution Prevention* (DEP), khususnya Intel XD bit (execute disable bit) atau AMD NX bit (*no execute bit*), harus tersedia dan di-*enabled*.
- **Kecepatan CPU minimal:** 1.4 GHz, disarankan 2 GHz atau yang lebih cepat.
- **RAM:** Minimum: 1 GB RAM, disarankan 2 GB RAM atau lebih besar. Tambahan RAM dibutuhkan untuk setiap *guest* OS
- **Available disk space:** Minimum: 8 GB, disarankan 20 GB atau lebih besar . Tambahan *disk space* dibutuhkan untuk setiap *guest* OS.
- **DVD ROM drive**
- **Display:** Super VGA (800 × 600) atau monitor dengan resolusi lebih tinggi.
- **Other:** Keyboard dan Microsoft Mouse atau *compatible pointing device*.

Tahap kedua adalah menentukan OS dan *software* yang akan digunakan dalam rancangan ini. Sistem yang akan dibuat direncanakan menggunakan OS Windows Server 2008 R2 x64 bit. OS ini dipilih dikarenakan memiliki fitur-fitur yang dibutuhkan untuk *virtual*isasi dan *clustering,* yaitu Hyper-V dan *Failover Cluster*. Sedangkan *software* yang digunakan hanya tiga yaitu *Starwind* untuk keperluan instalasi *storage, Folder2iso* yaitu suatu *software converter* untuk mengubah suatu folder dalam bentuk *.iso* untuk keperluan pertukaran data ke dalam *virtual machine* dan *Sisoftware SANDRA* untuk keperluan *benchmarking*.

Tahap ketiga adalah mendesain *network* yang akan digunakan. Rancangan *cluster* yang dibuat adalah suatu *failover cluster* dimana suatu *failover cluster* itu membutuhkan tingkat ketersediaan yang tinggi, sehingga dibutuhkan lebih dari satu buah *network*. Jadi, jika satu *network* mengalami kondisi *down* maka masih ada *network* lain yang menghubungkan *cluster* tersebut. Seperti telah dijelaskan di subbab sebelumnya ada tiga buah *network* yang akan dibuat, detail ketiga *network* tersebut sebagai berikut[9]:

- VLAN 2: Public 10.1.1.x/24
  VLAN 2 merupakan *network* yang menghubungkan ketiga *cluster nodes*.
- VLAN 3: Heartbeat 192.168.1.x/24
  VLAN 3 merupakan *network* yang menghubungkan dua *nodes* yang menjadi member *nodes*.
- VLAN 4: iSCSI 192.168.2.x/24
  VLAN 4 merupakan *network* yang menghubungkan seluruh *cluster nodes* dengan iSCSI *storage*.

Tahap keempat adalah instalasi OS pada *host computer*. Untuk *software* yang akan di-install di *host* hanya *Folder2iso*, karena *software* ini dibutuhkan untuk pertukaran data antara *host* dan *virtual machine* yang akan dibuat. Pada tahap ini OS di-install sekaligus dengan fitur-fitur tambahan yang dibutuhkan.

Tahap kelima adalah tahap *pre-build cluster* yaitu tahap dibuatnya *nodes-nodes cluster*. Pada tahap ini dilakukan pembuatan *virtual machine* serta konfigurasi dari masing-masing VM untuk dijadikan *nodes*. Konfigurasi yang dilakukan antara lain menghubungkan VM dengan *virtual network* yang telah dibentuk sebelumnya, melakukan instalasi OS pada VM, konfigurasi *network*

masing-masing *node* serta instalasi *software* benchmark. Pada tahap ini juga dilakukan konfigurasi untuk menentukan *domain* user untuk tiap *nodes* yang berpengaruh nantinya terhadap perannya di *cluster*.

Tahap keenam adalah membangun *storage*, karena sistem yang dibangun adalah suatu sistem *virtual* maka *storage*-nya pun bentuknya adalah *virtual machine* yang dihubungkan dengan *virtual network* yang telah dibentuk sebelumnya sehingga dapat terhubung dengan *cluster nodes*. Sama halnya seperti *nodes cluster* yang lain pada *storage* juga di install OS yang sama yaitu Windows Server 2008 R2 x64 bit. Hanya saja pada *storage* tidak perlu dilakukan konfigurasi *domain* dan instalasi fitur OS. Tambahan dibagian *storage* hanyalah *Starwind software* untuk menghubungkan iSCSI target *storage* (*storage* itu sendiri) dengan iSCSI initiator (*cluster nodes*). Dan lagi dikarenakan perannya sebagai *storage* maka VM ini memerlukan *virtual disk* tambahan yaitu VHD IDE dan VHD SCSI. Kedua *disk* tambahan ini yang nantinya akan menjadi share *storage* untuk ketiga *nodes*.

Tahap ketujuh dari perancangan ini adalah membangun *cluster* itu sendiri. Tahap pembangunan *cluster* ini terdiri dari tahap validasi *cluster* yaitu tahap untuk mengetahui apakah seluruh kebutuhan *cluster* sudah terpenuhi atau belum seperti *nodes, network,* dan *storage*. Apabila telah dilakukan validasi maka *cluster* dapat dibuat. Pada pembuatannya yang dilakukan antara lain menentukan nama *cluster*, menentukan *network* dan ip *cluster*, menambahkan *nodes* ke dalam *cluster*. Setelah berhasil dibuat, tahap selanjutnya adalah instalasi *service* dan aplikasi dari *cluster* tersebut. *Service* dan aplikasi yang akan diinstall adalah *File Server.*

Selanjutnya masuk kedua tahap terakhir dari rancangan pembuatan sistem ini yaitu *cluster test* dan *benchmarking*. *Cluster test* merupakan tahap dimana dilakukan *test* pada *cluster* untuk mengetahui apakah fungi *failover cluster*nya sudah bekerja baik atau belum. Sedangkan tahap terakhir yaitu *benchmarking* merupakan tahap untuk melakukan *test* terhadap performa *cluster* dilihat dari beberapa parameter untuk selanjutnya dibandingkan dengan performa sistem yang tidak di *cluster* sebagai keperluan analisa.

3.3 Tahap-Tahap Pengujian

Tugas utama dari pengujian performa atau *testing* adalah untuk mengumpukan informasi dalam rangka untuk membantu mengetahui kualitas secara keseluruhan dari sistem yang akan diuji. Sebagai tambahan, hasil dari uji performa dan analisis dapat membantu dalam mengestimasi konfigurasi *hardware* yang dibutuhkan untuk mendukung aplikasi yang akan dijalankan.

Pada sistem ini ada dua pengujian yang dilakukan yaitu pengujian *cluster* dan pengujian performa. Untuk pengujian *cluster* ada dua scenario yang diakan dilakukan yaitu *unplanned failover test* dan *planned failover test*. *Unplanned failover test* yaitu *testing* terhadap *cluster* dengan membuat salah satu *node cluster* down kemudian dilihat apakah *nodes* yang lain dapat mengambil alih kerja *nodes* tersebut secara otomatis. Sedangkan *planned failover test* yaitu scenario *testing* dimana perancang yang mengarahkan *nodes* mana yang akan mengambil alih saat terjadi *failure* pada *nodes* yang sedang menjalankan aplikasi dan dilihat apakah *nodes* tersebut bisa mengambil alih kerja *nodes* yang mengalami *failure* itu. Jika kedua *test* ini berhasil maka *cluster* dapat dianggap sudah bekerja baik.

Pada pengujian performa, *testing* dilakukan menggunakan *Sisoftware SANDRA*, parameter yang diukur yaitu *processor arithmetic*, *.NET arithmetic*, *memory bandwidth, cache and memory, memory latency*, dan *file system*. Skenarionya *testing* ini dilakukan sebanyak sepuluh kali untuk masing-masing parameter. *Testing* dilakukan pada saat semua *nodes* dalam kondisi *up* dan *testing* dilakukan disetiap *nodes*, saat salah satu *nodes* dalam kondisi *failure* dan *testing* dilakukan hanya pada *nodes* yang tidak mengalami *failure*, dan saat kondisinya hanya tinggal satu *nodes* yang bekerja. Hasil *testing* performa *cluster* ini nantinya dibandingkan dengan hasil *testing* performa dari sebuah mesin VM dengan spesifikasinya sama dengan satu *nodes*. Untuk selanjutnya dianalisa performa *cluster* tersebut apakah lebih baik atau lebih buruk.

BAB 4
IMPLEMENTASI DAN ANALISA SISTEM FAILOVER COMPUTER CLUSTER

Pada bab sebelumnya telah dilakukan perancangan dari sistem yang akan dibuat yaitu sistem *virtual computer cluster*. Setelah tahap perancangan di masuk ke tahap implementasi. Skenario yang akan dilakukan adalah membuat kondisi *virtual computer cluster* sebagai prototype dari sistem *cluster* computer yang sebenarnya.

Seperti yang telah disebutkan pada tahap perancangan sistem ini akan dibangun pada sebuah PC yang didalamnya akan di-*install* sistem operasi Windows Server 2008 R2 x64 bit sebagai sistem operasi *host* dari sistem *virtual computer cluster* ini. Lalu dengan fitur yang ada pada Windows Server 2008 R2 x64 bit ini (Hyper-V) akan dibentuk tiga buah *virtual machine*, dimana satu *virtual machine* berperan sebagai *domain controller* dan dua *virtual machine* lainnya berperan sebagai *nodes*. Kemudian akan dibuat *storage area network* dengan simulasi iSCSI menggunakan software Starwind, untuk menghubungkannya dengan tiap *nodes* pada *cluster*. Setelah semua kebutuhan *cluster* telah dibangun selanjutnya adalah membangun *cluster* dari tiga buah *virtual machine* itu sendiri menggunakan fitur failover *cluster* dari Windows Server 2008. Terakhir dilakukan *benchmark* pada sistem virtual *cluster* ini untuk dianalisa performa-nya dan dibandingkan dengan performa dari satu buah *virtual machine*.

4.1 Spesifikasi *Hardware*

Dalam membangun sistem *virtual computer cluster* ini beberapa komponennya membutuhkan spesifikasi khusus agar dapat berjalan. Berikut ini adalah spesifikasi *hardware* yang digunakan untuk mengimplementasikan sistem *virtual computer cluster* disesuaikan dengan kebutuhan sistem yaitu:

Tabel 4.1 Spesifikasi *Hardware* yang digunakan.

Component	Description
Model	• Compaq Presario CQ4168L
Processor	• Intel® Core™2 Duo *Processor* E7500 • Speed 2.93GHz • Cores per *Processor* 2 Unit(s) • Threads per Core 2 Unit(s) • Type Core 2 Duo • L2 On-board *Cache* 3MB • FSB speed 1066Mhz • Instruction Set 64 bit
Memory	• 4 GB DDR3 • Speed: PC3-10600 MB/sec (message as PC3-8500)

4.2 Tahap Implementasi

Pada bagian ini akan dijelaskan tahapan implementasi berupa instalasi dan konfigurasi dari tiap komponen sistem dari tahap instalasi sistem operasi sampai dengan pengambilan data. Sebelum dilakukan instalasi, perlu dipastikan terlebih dahulu apakah semua persyaratan yang dibutuhkan telah terpenuhi, salah satunya pastikan fitur virtualisasi pada BIOS sudah dalam kondisi *enable*. Selanjutnya setelah telah dipastikan semuanya telah sesuai maka lakukan tahapan dari implementasi sistem *virtual computer cluster* berikut ini:

Step 1: Instalasi Sistem Operasi Windows Server 2008 R2 x64

Instalasi OS Windows Server 2008 R2 x64 ini tidak berbeda dengan proses instalasi OS lain. Tahap instalasi cukup mengikuti petunjuk yang muncul di layar. Hanya saja yang perlu diperhatikan adalah saat muncul window untuk memilih operating system apa yang akan di-*install* maka pilih *Windows Server 2008 Data Center (Full Instalation)* x64. Perlu dipastikan *installer* yang

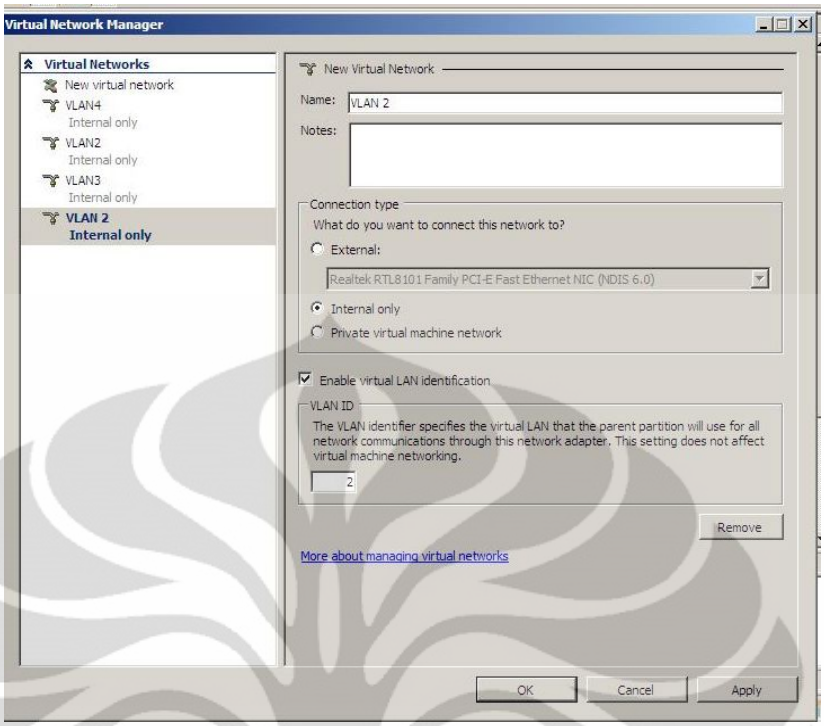
digunakan merupakan x64 bit bukan x86 bit agar semua fitur yang dibutuhkan untuk pembentukan *cluster* dapat digunakan.

Step 2: Instalasi Role Hyper-V

Proses instalasi fitur dan role pada Windows Server 2008 dilakukan pada menu server *manager*. Untuk instalasi Hyper-V dapat dilakukan pada bagian *role summary* pada server *manager*. Role Hyper-V ini tidak akan bekerja apabila fitur virtualisasi pada BIOS dalam kondisi *disable* dan tidak akan muncul pada menu add role apabila OS yang digunakan adalah Windows Server 2008 x32. Role Hyper-V di-*install* pada komputer host.

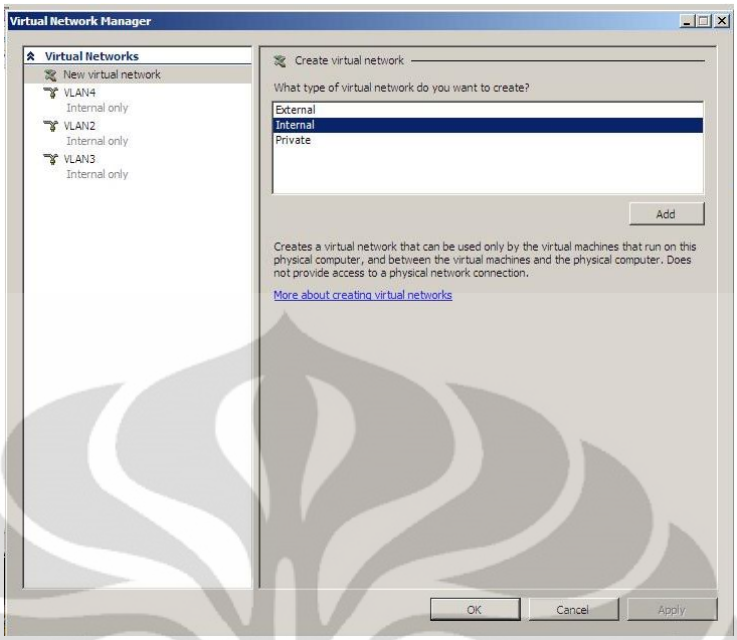
Step 3: Membuat Virtual *Network*

Pembuatan *virtual network* dilakukan pada menu Hyper-V *Manager*, dimana Gambar 4.1 menunjukkan tampilan *wizard* pembuatan *virtual network*. Seperti tampak pada gambar dalam membuat *Virtual network* akan diminta untuk menentukan nama dari *virtual network* yang akan dibentuk dalam prakteknya ada tiga buah *virtual network* yang dibuat dengan nama VLAN 2 untuk *public network* yaitu *network* yang memungkin koneksi jaringan. VLAN 3 untuk *heartbeat network* yaitu *private network* untuk menghubungkan *nodes* 1 dan *nodes* 2 (selain *domain controller*). *Private Network* ini digunakan oleh *cluster* sehingga tiap *nodes* dapat berkomunikasi satu sama lain dan memverifikasi bahwa tiap *node* dalam kondisi *up* dan *running*. VLAN 4 untuk iSCSI *network* yaitu *private network* yang menghubungkan tiap *nodes* dalam *cluster* (termasuk *domain controller*) ke *storage*, sehingga *storage* dapat diakses oleh tiap anggota *cluster*.



Gambar 4.1. Tampilan awal *wizard* Virtual *Network Manager* untuk membuat virtual *network*

Selain menentukan nama dari *virtual network* yang akan dibuat pada tahap ini juga diminta menentukan jenis koneksi yang akan digunakan. Seperti yang dapat dilihat pada Gambar 4.2 jenis koneksi yang digunakan pada ketiga VLAN di atas adalah koneksi internal. Tipe koneksi jenis ini hanya dapat digunakan oleh *virtual machine* yang berada dalam sebuah komputer fisik dan antara *virtual machine* dengan komputer fisik tersebut. Karena pada implementasinya memang tidak dibutuhkan koneksi keluar dan hanya butuh koneksi antar *virtual machine* saja maka jenis koneksi internal inilah yang dipilih. Setelah nama dan jenis koneksi telah ditentukan maka proses pembuatan *virtual network* selesai. Proses pembuatan *virtual network* ini diulang sebanyak tiga kali karena ada tiga buah *virtual network* yang dibutuhkan dalam implementasi *virtual computer cluster* ini.



Gambar 4.2 Tampilan *wizard* Virtual *Network Manager* saat menentukan jenis koneksi.

Step 4: Membangun dan Mengkonfigurasi *Virtual Machine*

Pada tahap ini dilakukan pembuatan *virtual machine* dan konfigurasi dari *virtual machine* itu sendiri. Ada empat *virtual machine* yang dibuat dalam implementasi sistem ini. Tiga *virtual machine* merupakan *cluster node* dan satu *virtual machine* digunakan sebagai *storage*. Spesifikasi masing-masing *virtual machine* yang berperan sebagai *node cluster* yaitu memiliki *virtual hard disk* sebesar 40 GB dan Virtual RAM sebesar 784 MB. Untuk *virtual machine* yang digunakan sebagai *storage* pada dasarnya memiliki spesifikasi yang sama, hanya saja ada tambahan satu buah *virtual hard disk* IDE sebesar 40 GB dan virtual har*disk* SCSI 4 GB. Untuk *network adapter* dari masing-masing *virtual machine* jumlahnya disesuaikan dengan peran masing-masing *virtual machine* tersebut. Proses pembuatannya hampir sama dengan pembuatan virtual *network* hanya saja pada saat menu New diklik yang dipilih adalah *Virtual Machine*, pembuatannya pun cukup mengikuti petunjuk yang ada pada *wizard* lalu saat diminta menentukan nama, besar *hard disk* dan besar memori diisi dengan mengikuti spesifikasi di atas. Berikut ini detail *virtual machine* yang dibuat:

- *Domain Controller*: dc01

  Virtual machine ini setelah dibuat kemudian di *install* Windows Server 2008 x64 sebagai sistem operasinya. Memiliki sebuah *virtual hard disk* IDE *fixed size* sebesar 40 GB dan dua buah *network adapter* yang terhubung dengan VLAN 2 dan VLAN 4. RAM dari *virtual machine* ini sebesar 784 MB

  Pengaturan Jaringan:

  Public NIC: VLAN 2

  - IP Addr: 10.1.1.10
  - Mask: 255.255.255.0
  - DNS: 10.1.1.10

  iSCSI NIC: VLAN 4

  - IP Addr: 192.168.2.6
  - Mask: 255.255.255.0

- *Cluster Nodes*:

  Virtual machine untuk *cluster nodes* ini setelah dibuat kemudian di *install* Windows Server 2008 x64 sebagai sistem operasinya. *Virtual machine* ini memiliki sebuah *virtual hard disk* IDE *fixed size* sebesar 40 GB sama seperti *domain controller* dan tiga buah *network adapter* yang terhubung dengan VLAN 2, VLAN 3 dan VLAN 4. RAM dari *virtual machine* ini sebesar 784 MB. Maksud dari *cluster nodes* di atas adalah *nodes-nodes* yang perannya bukan sebagai *domain controller*. *Virtual machine* kedua dinamakan *Node*01 dan *virtual machine* ketiga dinamakan *Node*02. Berikut ini konfigurasi jaringan dari masing-masing *virtual machine* tersebut.

  *Node*01

  Public Network: VLAN 2

  - IP Address: 10.1.1.20
  - Mask: 255.255.255.0
  - DNS: 10.1.1.10

  Heartbeat Network: VLAN 3

- IP Address: 192.168.1.4
- Mask: 255.255.255.0

iSCSI *Network*: VLAN 4

- IP Address: 192.168.2.4
- Mask: 255.255.255.0

*Node*02

Public Network: VLAN 2

- IP Address: 10.1.1.21
- Mask: 255.255.255.0
- DNS: 10.1.1.10

Heartbeat Network: VLAN 3

- IP Address: 192.168.1.5
- Mask: 255.255.255.0

iSCSI *Network*: VLAN 4

- IP Address: 192.168.2.5
- Mask: 255.255.255.0

- iSCSI *Targe*

Virtual machine untuk *storage* ini dinamakan iSCSI *Target*. *Virtual machine* ini setelah dibuat kemudian di *install* Windows Server 2008 x64 sebagai sistem operasinya. *Virtual machine* ini memiliki sebuah *virtual hard disk* IDE *fixed size* sebesar 40 GB, ditambah sebuah SCSI *hard disk* sebesar 4GB dan sebuah IDE *hard disk* sebesar 40 GB dan dua buah *network adapter* yang terhubung dengan VLAN 2 dan VLAN 4. RAM dari *virtual machine* ini sebesar 784 MB.

Pengaturan Jaringan:

Public Network: VLAN 2

- IP Address: 10.1.1.22
- Mask: 255.255.255.0
- DNS: 10.1.1.10

iSCSI *Network*: VLAN 4

- IP Address: 192.168.2.2
- Mask: 255.255.255.0

Pada saat mengkonfigurasi *network* pastikan untuk VLAN 3 dan VLAN 4 *Client for Microsoft Networks* dalam kondisi *disable*, DNS *registration* dalam kondisi *disable* dan NetBIOS dalam kondisi *disable*. Pastikan urutan konfigurasinya dimana *public* harus menjadi yang pertama. Cara konfigurasi *network* di atas seperti cara mengatur IP pada umumnya yaitu pada bagian *network and sharing center*. Pastikan pula SCSI *hard disk* telah di format dan ditentukan *drive letter*-nya, sehingga dapat dikenali oleh anggota *cluster*.

Step 5: Konfigurasi *Domain Controller*

Domain controller (DC) adalah sebuah server atau *nodes* pada *cluster* yang menanggapi *security authentication requests (logging in, checking permissions,* dan lain-lain) yang berada dalam Windows Server *domain*. Dengan terdaftarnya *user* dalam suatu *domain* maka *user* aan diberikan akses ke beberapa sumber daya komputer hanya dengan menggunakan sebuah *username* dan kombinasi *password*. Agar dapat memanfaatkan fitur-fitur untuk *clustering* dalam Windows Server 2008 *user* diharuskan *log in* menggunakan *domain user account*. Selain itu tiap *nodes* dalam *cluster* harus berada dalam *domain* yang sama.

Pertama-tama dalam mengkonfigurasi *domain controller* perlu dilakukan instalasi *Active Directory Domain Service*. Fitur ini dibutuhkan untuk keperluan pembuatan *domain user account, domain* untuk cluster sehingga tiap bagian *cluster* baik *domain controller* maupun *nodes cluster* bisa berkomunikasi dan terhubung dalam satu lingkungan *domain* yang sama, dimana nama *domain* yang dibuat adalah *clust.Local*. *Domain user account* merupakan *account* yang digunakan untuk *log on* ke dalam sistem *cluster*.

Pada saat instalasi *Active Directory Domain Service,* ter-*install* pula DNS Server sebagai modul tambahan. Maka tahap selanjutnya adalah konfigurasi *Reverse Lookup Zone*, pada tahap ini dibuat zona baru pada *reverse lookup zone*,

network yang digunakan adalah 10.1.1.0. Selanjutnya lakukan penambahan host, pada tahap inilah *nodes-nodes* yang akan menjadi anggota *cluster* akan dihubungkan dalam satu *domain* yang sama. Setelah berhasil maka *nodes-nodes* tersebut telah terdaftar dalam *domain* yang sama dan akan muncul pada *DNS Server Console* baik pada *Forward Lookup Zone* maupun pada *Reverse Lookup Zone*.

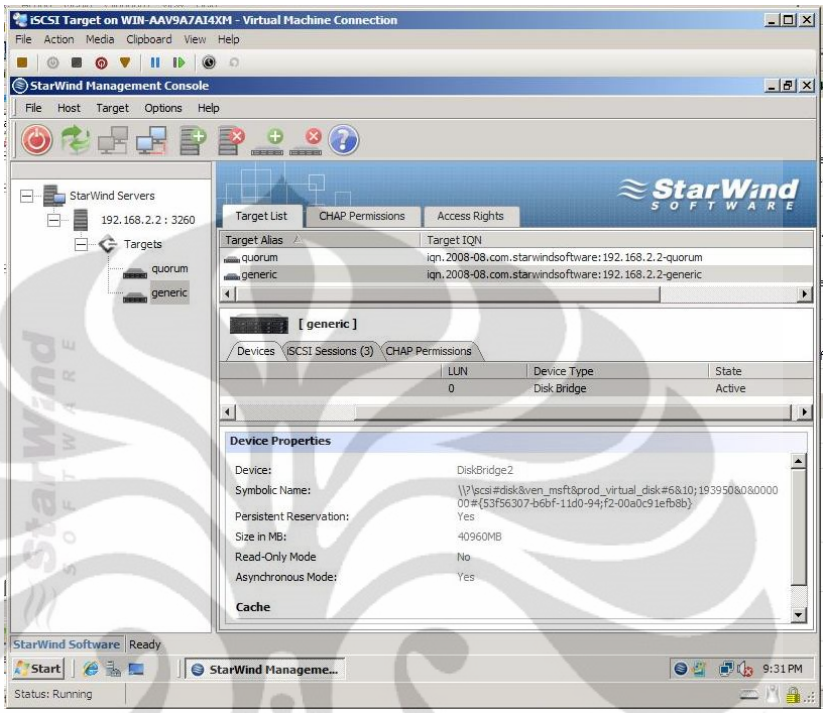
User Account dibuat melalui *Active Directory Users and Computer management console*. Pembuatan *account* dilakukan sebanyak dua kali yaitu untuk node01 dan node02. Setelah kedua *account* tersebut telah dibuat maka masukkan kedua group itu kedalam *Administrators group* dan *Domain Admins group*. Setelah berhasil maka proses konfigurasi *domain controller* ini telah berhasil dan fitur *Failover Cluster* sudah dapat digunakan.

Step 6: Konfigurasi *Storage Area Network*

Konfigurasi *Storage Area Network* dilakukan dengan cara meng-*install* Starwind iSCSI *Target* Software. Starwind Software ini di *install* pada *Virtual Machine* iSCSI *Target*. Konfigurasi yang dilakukan antara lain menambahkan *host* dan *target*. Host dari iSCSI *target* ini di-*set* pada ip 192.168.2.2 pada portal 3260.

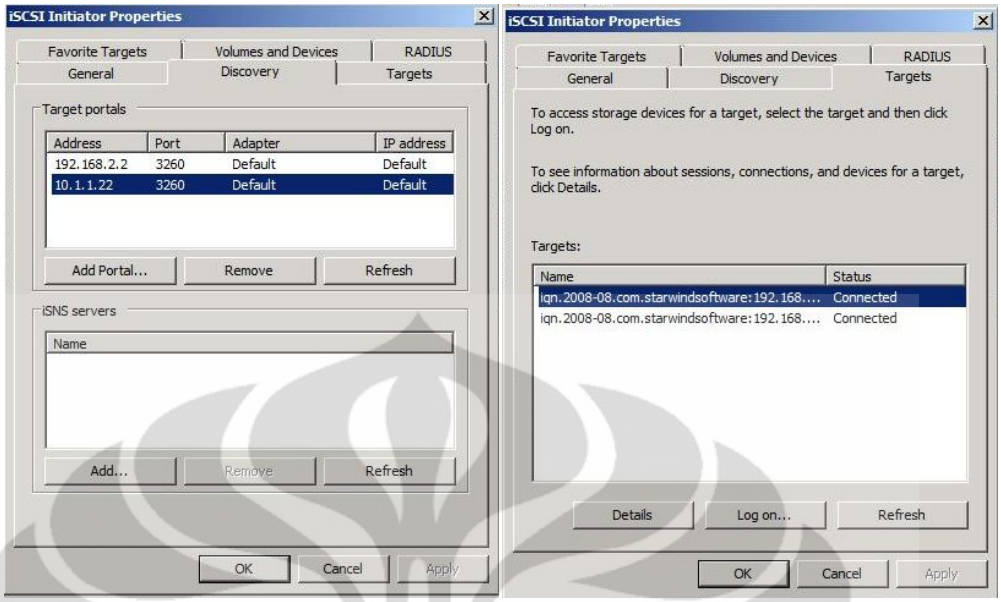
Pada tahap sebelumnya yaitu pembuatan *virtual machine*, pada saat pembuatan *virtual machine* untuk iSCSI *target* ada dua *disk* tambahan yang ditambahkan pada VM, yaitu IDE *hard disk* dan SCSI *hard disk*. Kedua *disk* tambahan inilah yang dijadikan *target* pada *host*. Pada implementasi ini nama *target* pertama yaitu SCSI *hard disk* yang berukuran 4 GB yang *target alias*-nya diset sebagai *quorum*, *target name* haruslah unik sehingga dapat diidentifikasi oleh iSCSI *initiator* pada saat dikirimkan lewat IP, oleh karena itu *target name* yang digunakan adalah *target name* yang dibuatkan secara otomatis oleh *wizard* tersebut. Pada *wizard disk parameter* pastikan *Asynchronous Mode* dan *Allow multiple concurrent iSCSI connection (clustering)* di-*checklist*. Pada implementasi ini *target alias* untuk *target* kedua adalah *generic* dan *virtual disk* yang digunakan sebagai *target* adalah IDE *hard disk* yang berukuran 40 GB. Gambar 4.3

menunjukkan tampilan starwind setelah rangkaian proses di atas berhasil dilakukan dapat dilihat pada *console tree* sudah berhasil me-*list target*.



Gambar 4.3 Tampilan *Management Console* Starwind yang menunjukkan *target* list *storage*.

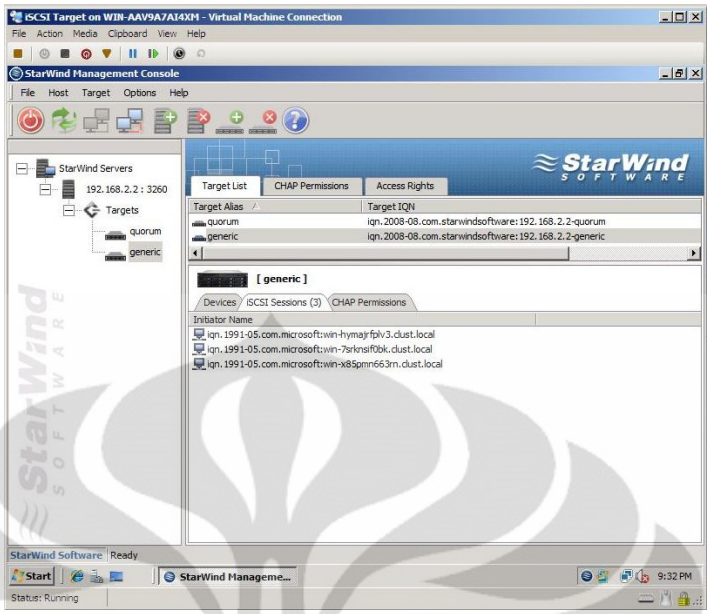
Setelah melakukan konfigurasi pada *storage* yang dilakukan selanjutnya adalah menghubungkan *storage* dengan *nodes*. Untuk tahap ini digunakan fitur iSCSI initiator dari Windows Server 2008. *Target storage* dan *initiator* dihubungkan dengan menambahkan portal pada bagian intiator-nya ip dan portal yang ditambahkan adalah 192.168.2.2 dengan portal 3260. Hasil konfigurasi dapat dilihat pada *tab target* dimana akan muncul *target name* yang telah terdeteksi. Konfigurasi lain yang dilakukan adalah pengaturan koneksi otomatis sehingga *nodes* dan *storage* akan terhubung secara otomatis setiap *cluster nodes* dijalankan. Setelah berhasil maka akan muncul tampilan seperti pada Gambar 4.4 dan pada *tab volume and devices* akan muncul tampilan seperti Gambar 4.5. Dari Gambar 4.5 tampak bahwa *disk target* yang telah dikonfigurasi pada *storage* telah terdaftar dan berhasil terhubung dengan initiator.



Gambar 4.4 Tampilan pada *tab Discovery* dan *tab Target*s pada saat koneksi berhasil

Gambar 4.5 Tampilan *tab Volume and Devices* pada saat koneksi berhasil

Setelah semua berhasil dilakukan maka akan muncul tampilan seperti pada Gambar 4.6 pada *management console*. Gambar 4.6 ini menunjukan iSCSI *initiator* yang terhubung dengan iSCSI *target*. Tampak pada gambar bahwa ada tiga *initiator* yang terhubung ke *target* karena ada tiga *nodes* dalam *cluster*.



Gambar 4.6. Tampilan Management Console Starwind yang menunjukkan jumlah initiator yang terkoneksi.

Step 7: Instalasi Fitur Failover *Cluster*

Pada tahap ini yang dilakukan adalah instalasi fitur *Failover Cluster*. Sama halnya dengan cara instalasi Hyper-V role, instalasi fitur Failover *Cluster* juga dapat dilakukan pada menu server *manager* atau menu *initial configuration task*. Fitur *Failover cluster* ini di-*install* hanya pada *domain controller nodes*.

Step 8: Membangun dan Mengkonfigurasi *Cluster*

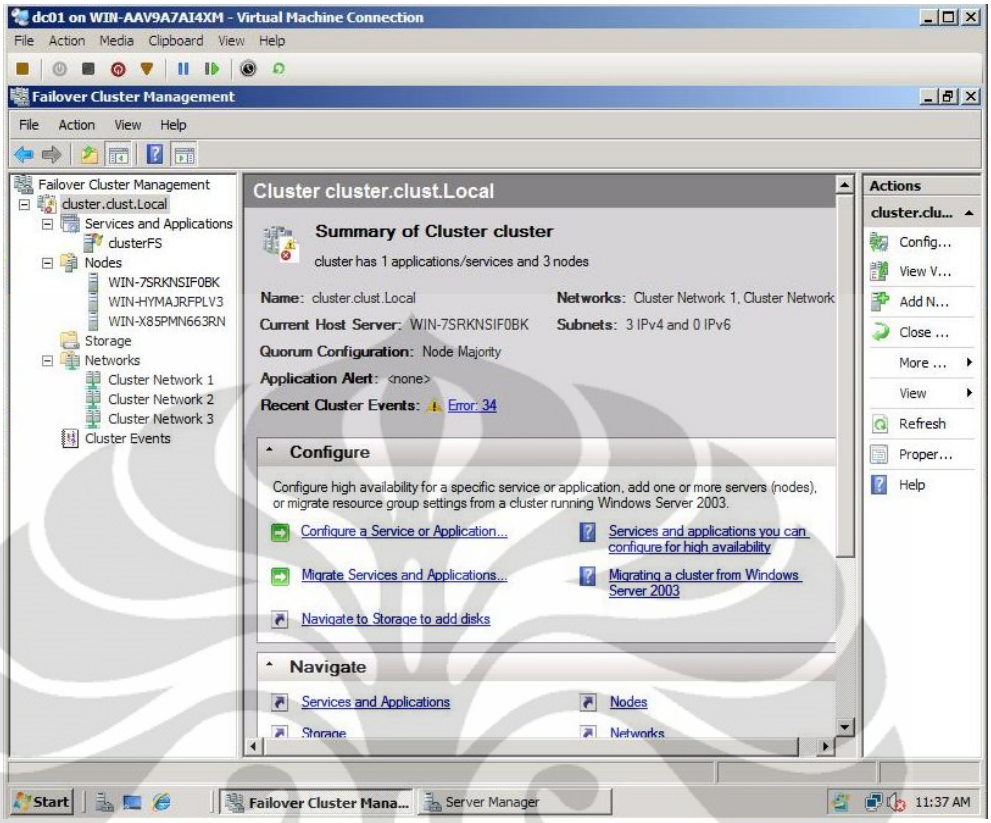
Saat semua komponen *cluster* telah tersedia melalui beberapa proses instalasi dan konfigurasi pada tahap sebelumnya, maka tahap selanjutnya adalah membangun dan mengkonfigurasi *cluster* itu sendiri. Tahap ini dilakukan menggunakan fitur *Failover Cluster*. Fitur *Failover Cluster* ini dapat digunakan hanya jika semua *node* telah berada di *domain* yang sama dan *log in* menggunakan *domain user account* yang telah terdaftar pada *domain controller*. Pembuatan *domain* dan *domain user account* telah dijelaskan pada step awal, sehingga dapat diasumsikan step tersebut telah berhasil dan semua fitur *Failover Cluster* sudah dapat digunakan.

Hal pertama yang dilakukan untuk membangun dan mengkonfigurasi *cluster* adalah melakukan tes validasi terhadap *node-node cluster*. Ketiga *node cluster* tersebut adalah:

- WIN-7SRKNSIF0BK yaitu VM yang berperan sebagai *node* 1
- WIN-HYMAJRFPLV3 yaitu VM yang berperan sebagai *node* 2
- WIN-X85PMN663RN yaitu VM yang berperan sebagai *domain controller*

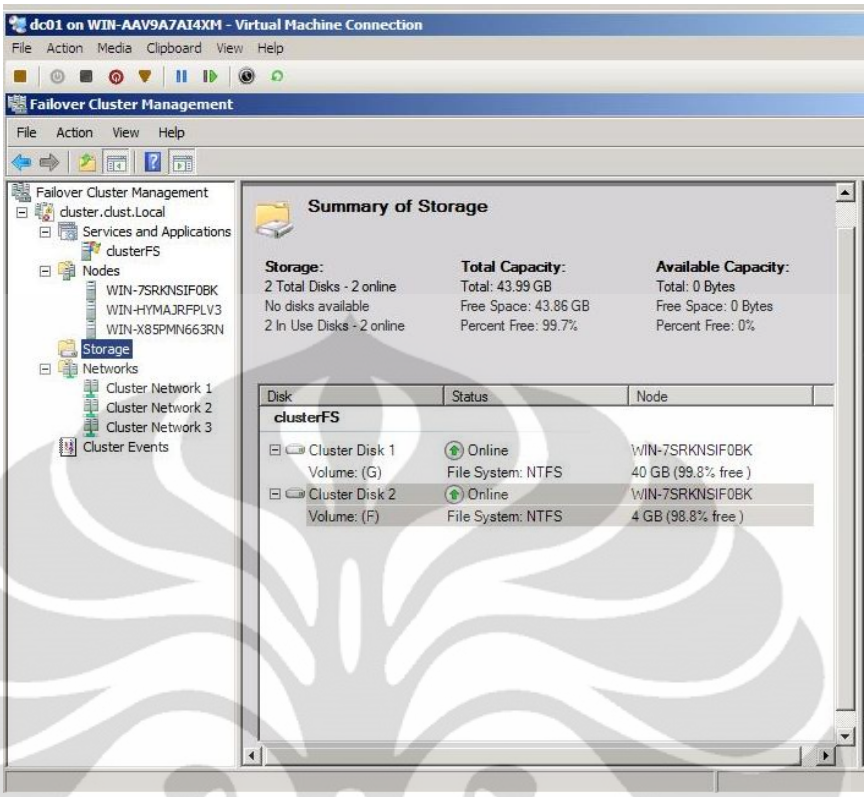
Pembuatan *cluster* dimulai dengan mendaftarkan komputer-komputer yang akan menjadi anggota *cluster*, yaitu ketiga komputer diatas. Kemudian akan diminta menentukan nama dari *cluster* yang akan dibuat untuk sistem *virtual computer cluster* ini adalah *cluster.clust.Local* dimana *clust.Local* adalah *domain* dari *cluster* ini. Pada tahap ini diminta pula untuk menentukan ip dari *cluster* ini pada implementasi ini IP yang digunakan adalah 10.1.1.11 dengan *network* 10.1.1.0. Proses pembuatan *cluster* ini diakhiri dengan melakukan tes validasi ulang untuk men-*test cluster*. Saat proses selesai maka *cluster*.clust.local telah terbentuk.

Proses terakhir dari tahap pembuatan dan konfigurasi *cluster* adalah instalasi aplikasi. Pada implementasinya *service and application* yang di-*install* adalah *File Server*. Setelah rangkaian proses ini telah selesai maka akan tampak tampilan seperti pada Gambar 4.7.



Gambar 4.7 Tampilan Failover *Cluster* Management Setelah *Cluster* dibuat.

Pada Gambar 4.7 Dapat dilihat bahwa pada *cluster cluster*.clust.local terdapat aplikasi *cluster*FS atau File Server yang sedang berjalan atau bekerja pada salah satu *node* anggota *cluster*. Dari gambar tersebut diketahui bahwa *current host server* saat ini adalah WIN-7SRKNSIF0BK, maka disanalah *service* dan aplikasi File Server berjalan. Kemudian dapat dilihat pada bagian *nodes* pada *console tree* terdapat tiga buah *nodes* yang terdaftar sebagai anggota *cluster*. Kemudian pada bagian *Network*s dapat dilihat bahwa *cluster* ini terhubung dengan tiga buah *cluster network* yang berbeda.



Gambar 4.8 *Storage* dari *Cluster cluster*.clust.Local

Dari Gambar 4.8 di atas dapat dilihat ada dua buah *disk* yang digunakan sebagai *storage*. Dimana kedua *disk* tersebut merupakan iSCSI *Target* yang sebelumnya telah dikonfigurasi. Kedua *disk* tersebut memiliki ukuran berbeda yaitu 40 GB dan 4 GB. Dari tampilan *Failover Cluster Management* ini tampak bahwa step 1 sampai dengan step 8 telah berhasil.

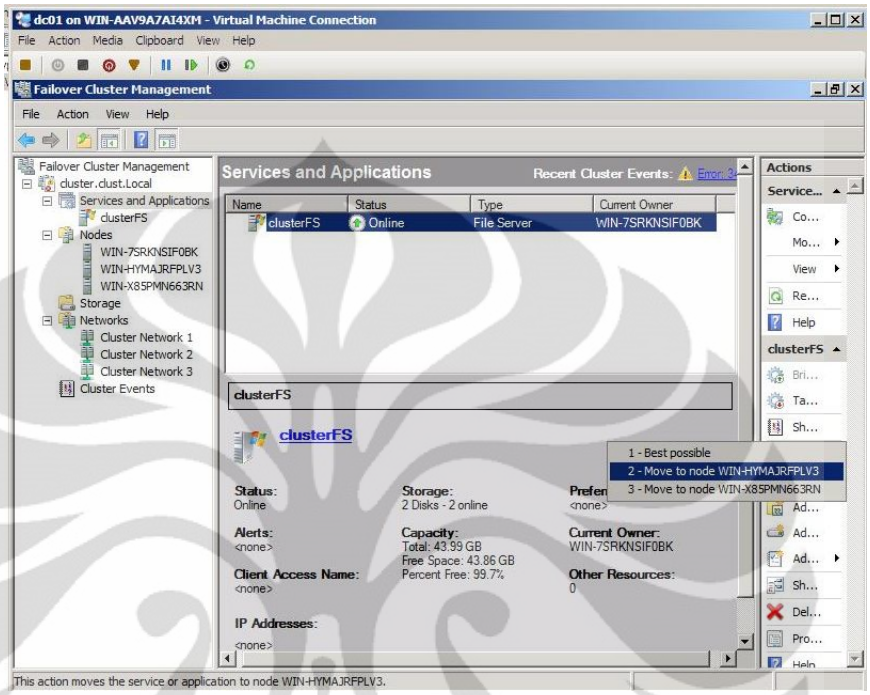
Step 9: Melakukan *Cluster* Test

Seperti yang telah dijelaskan pada tahap perancangan, ada dua jenis *test* yang akan dilakukan untuk mengetahui apakah *cluster* yang dibuat telah bekerja atau belum. Kedua test tersebut adalah test untuk planned failover dan test untuk unplanned failover.

- **Test planned failover**

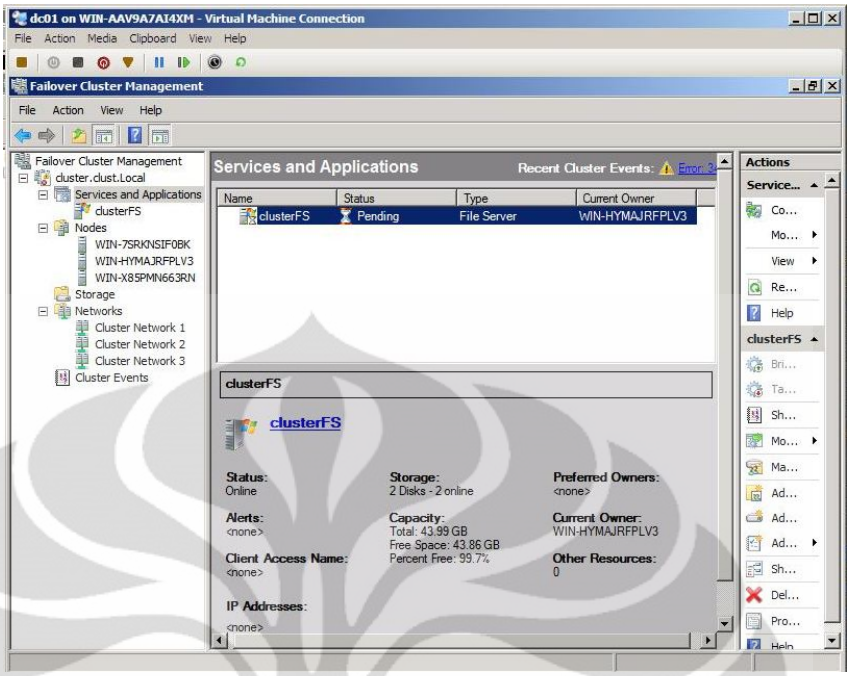
Pada dasarnya yang dilakukan pada planned failover test adalah memindahkan aplikasi yang sedang dijalankan ke *node* lain secara terencana atau ditentukan secara manual tidak otomatis. Gambar 4.9 menunjukan aplikasi File Server

sedang bekerja pada *node* WIN-7SRKNSF0BK. Untuk men-test apakah planned failover bekerja maka aplikasi ini kemudian dipindahkan ke *node* WIN-HYMAJRFPLV3.

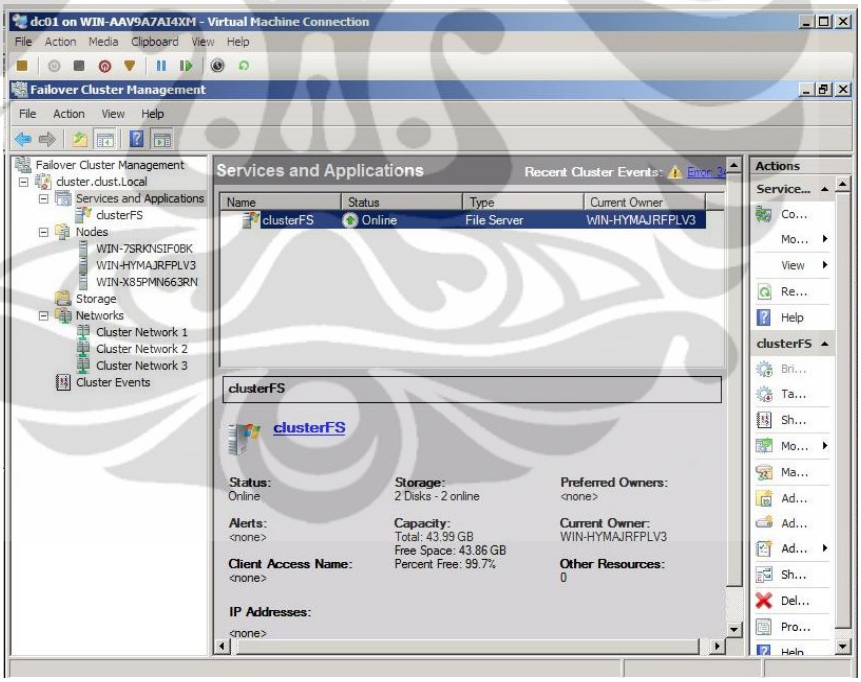


Gambar 4.9 Proses Awal Planned Failover Test.

Verifikasi mengenai berhasil tidaknya perpindahan *node* aplikasi tersebut dilakukan dengan menginpeksi detail *node* tersebut. Gambar 4.10 Menunjukkan proses perpindahan dari *node* WIN-7SRKNSF0BK ke *node* WIN-HYMAJRFPLV3. Tampak pada Gambar 4.10 status perpindahannya masih dalam keadaan pending.



Gambar 4.10 Proses perpindahan pada *planned failover test*.

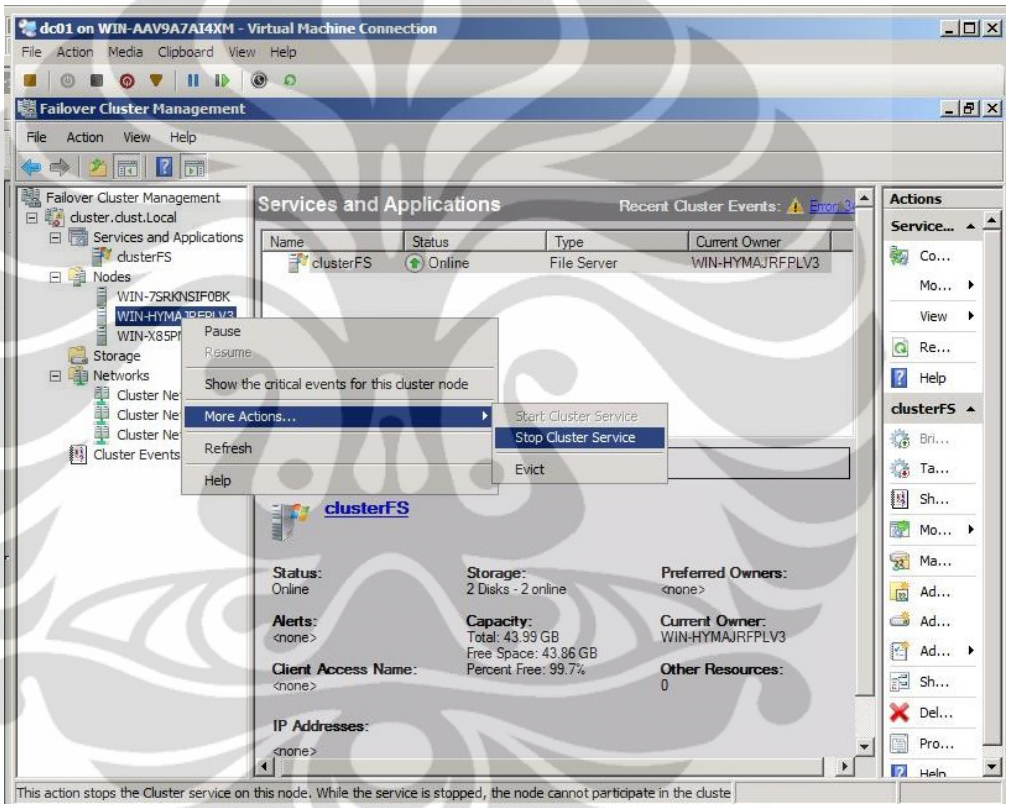


Gambar 4.11 Proses perpindahan berhasil pada *planned failover test*.

Gambar 4.11 menunjukkan bahwa proses perpindahan dari *node* WIN-7SRKNSF0BK ke *node* WIN-HYMAJRFPLV3 berhasil. Hal ini dapat dilihat pada detail aplikasi tersebut *current owner*-nya telah berubah dari awalnya WIN-7SRKNSF0BK (Gambar 4.9) menjadi WIN-HYMAJRFPLV3.

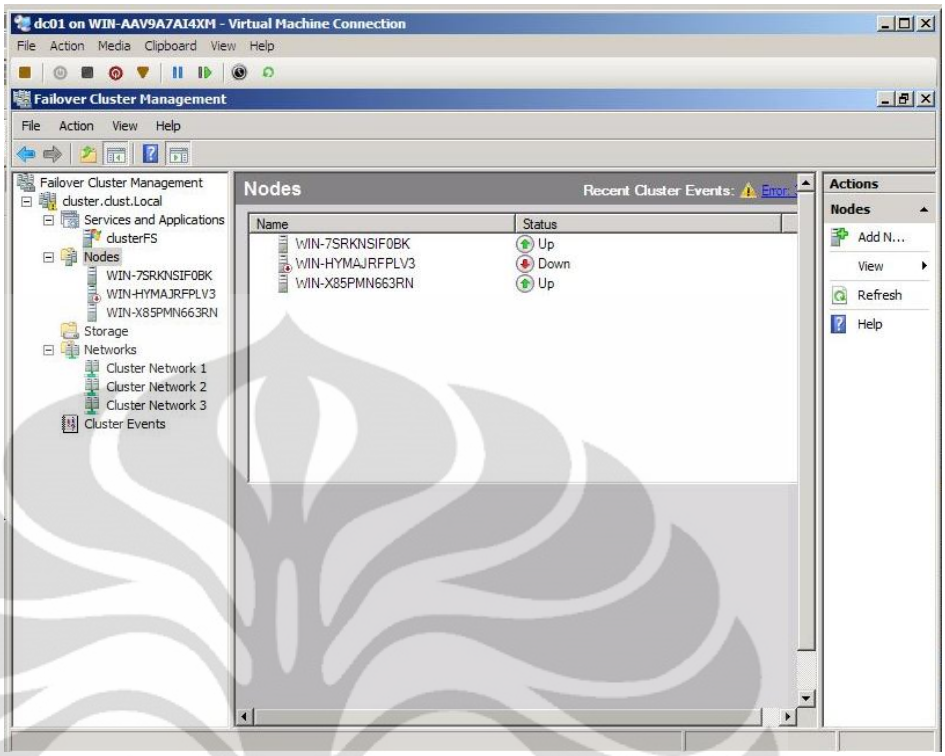
- *Test unplanned failover*

Untuk melakukan test unplanned failover, yang dilakukan adalah menghentikan *cluster service*, dengan menghentikan *cluster service* sebuah *node* sama dengan terjadi *failure* pada sebuah *node* di kondisi nyata. Tujuan dari *test* ini adalah melihat apakah *nodes* lain dalam *cluster.clust.Local* berhasil mengambil alih kerja *nodes* yang *down* secara otomatis. Gambar 4.12 menunjukkan gambaran dari tahap awal unplanned failover test ini.

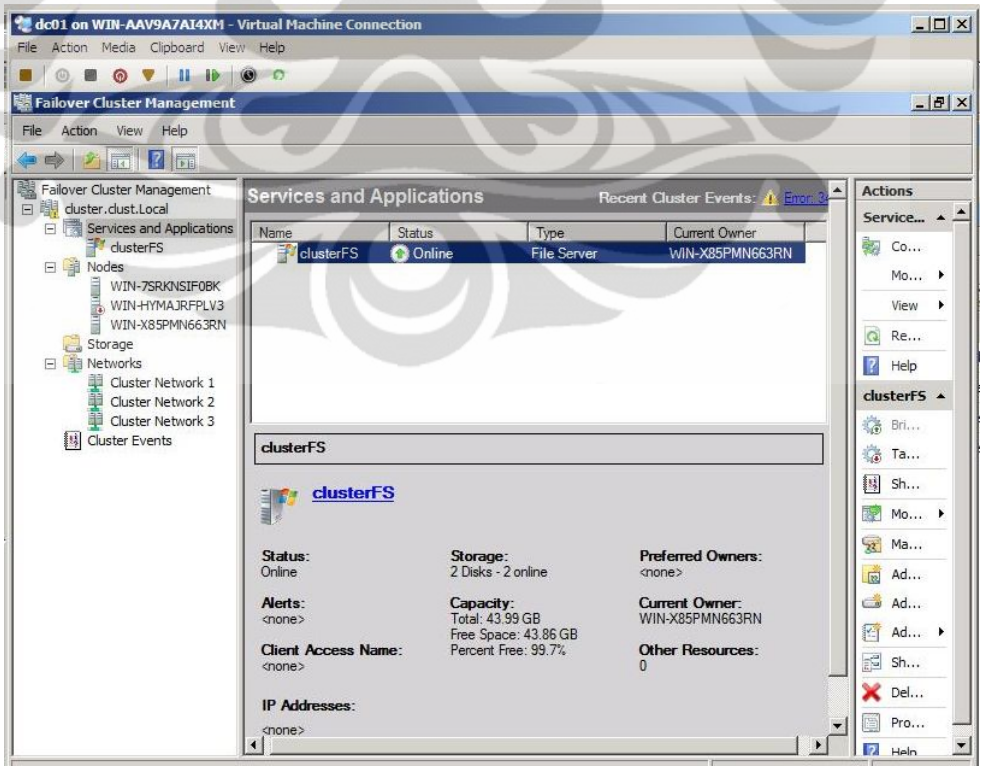


Gambar 4.12 Tampilan awal proses unplanned failover test.

Gambar 4.13 menunjukkan bahwa *node* WIN- WIN-HYMAJRFPLV3 berada pada kondisi failure. Sedangkan Gambar 4.14 menunjukkan bahwa pada saat *node* WIN-HYMAJRFPLV3 down aplikasi yang sebelumnya berada pada *node* ini berpindah secara otomatis ke *node* WIN-X85PMN663RN.



Gambar 4.13 Kondisi *nodes* WIN-HYMAJRFPLV3 saat dikeluarkan dari *cluster service*.



Gambar 4.14 Pergantian *Owner* pada *unplanned failover test.*

Step 10: Melakukan *Benchmarking* pada Sistem *Virtual Computer Cluster* dan pada sebuah *Virtual Machine*

Proses *benchmarking* ini dilakukan menggunakan *software* SiSoftware Sandra. SiSoftware Sandra di-*install* pada setiap *node cluster*. Setelah selesai program dijalankan disetiap *node* dalam dua kondisi yaitu pada saat semua *node* dalam kondisi up di *cluster* dan kondisi kedua adalah saat *cluster* memiliki satu *node* yang down. Untuk kondisi kedua *benchmark* hanya dilakukan pada *nodes* yang berada dalam kondisi up. Parameter yang diukur adalah *processor arithmetic*, *.NET arithmetic, memory bandwidth,memory latency*, dan *Cache and Memory*. Detail mengenai parameter ini telah dijelaskan pada bab sebelumnya. Proses *benchmarking* ini juga dilakukan pada sebuah *virtual machine* dengan spesifikasi yang sama hanya saja *virtual machine* ini tidak tergabung dengan sebuah *cluster*.

Step 11: Membandingkan dan Menganalisa hasil *benchmark* keduanya.

Tahap terakhir dari implementasi sistem *virtual computer cluster* ini adalah membandingkan dan menganalisa hasil *benchmark* dari sistem *virtual computer cluster* yang telah dibuat dengan hasil *benchmark* sebuah *virtual machine* yang tidak tergabung dalam *cluster*. Hasil dan analisanya akan dijelaskan pada sub bab selanjutnya.

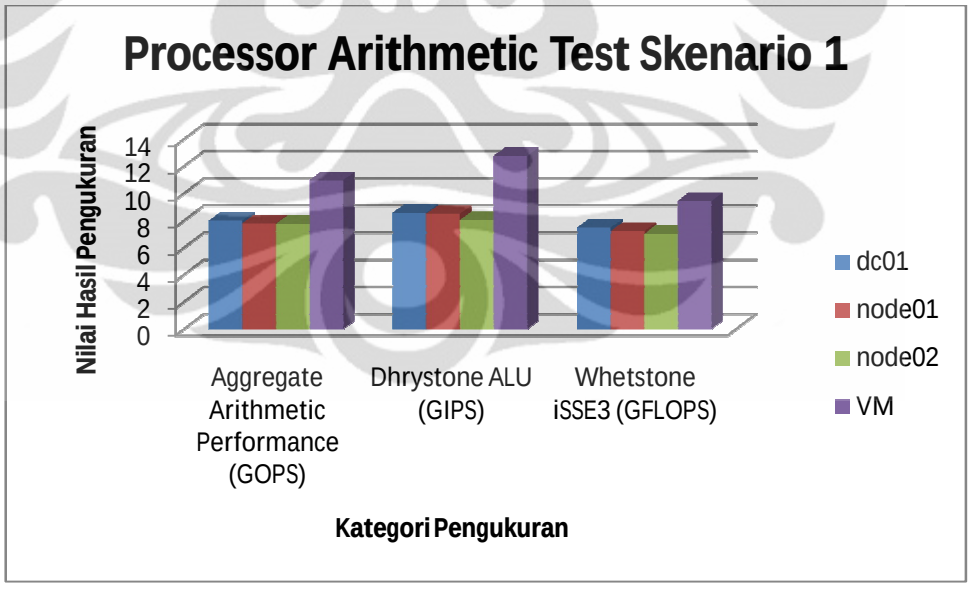
4.3 Hasil dan Analisa Sistem *Virtual Computer Cluster*

Analisa pada hasi *benchmark* dilakukan pada dua skenario berbeda. Skenario pertama adalah *Testing* dilakukan pada saat semua *nodes cluster* dalam kondisi up dan *testing* dilakukan disetiap *nodes*. Skenario kedua saat salah satu *nodes cluster* dalam kondisi *failure* dan *testing* dilakukan hanya pada *nodes* yang tidak mengalami *failure*. Pengukuran dilakukan sebanyak sepuluh kali untuk masing-masing parameter, kemudian hasilnya dirata-rata. Selanjutnya hasil kedua skenario ini dibandingkan dengan hasil pengukuran pada sebuah VM yang memiliki spesifikasi yang sama dengan *nodes cluster* hanya saja VM ini tidak tergabung dalam *cluster*. Berikut ini analisa hasil *benchmark* berdasarkan lima parameter *benchmark*.

4.3.1 Analisa parameter *Processor* Arithmatic

Tabel 4.2. Hasil *Benchmark Processor Arithmetic* Skenario 1

Tabel Hasil *Benchmark Processor Arithmetic* Skenario 1												
	dc01			*node*01			*node*02			VM		
Uji Coba ke-	Aggregate *Arithmetic* Performance (GOPS)	Dhrystone ALU (GIPS)	Whetstone iSSE3 (GFLOPS)	Aggregate *Arithmetic* Performance (GOPS)	Dhrystone ALU (GIPS)	Whetstone iSSE3 (GFLOPS)	Aggregate *Arithmetic* Performance (GOPS)	Dhrystone ALU (GIPS)	Whetstone iSSE3 (GFLOPS)	Aggregate *Arithmetic* Performance (GOPS)	Dhrystone ALU (GIPS)	Whetstone iSSE3 (GFLOPS)
1	10,2	11,9	8,74	7,26	8,22	6,42	10,14	8,08	9,1	11	12,8	9,45
2	8,54	8,1	9,01	10,62	12,34	9,14	8,38	8,21	6,55	10,98	12,77	9,44
3	8,49	8,63	8,35	6,81	8,13	5,7	7,84	7,82	8,3	10,98	12,76	9,44
4	8,19	8,32	8,07	7,41	8,22	6,67	6,26	7,95	6,32	10,97	12,73	9,45
5	7,58	8,43	6,81	7,75	7,72	7,79	8,27	7,45	8,83	10,97	12,73	9,45
6	7,55	8,04	7,09	7,34	8,21	6,55	8,57	7,83	6,29	10,96	12,73	9,44
7	7,53	7,86	7,21	8,05	7,82	8,3	7,09	7,73	5,89	10,96	12,76	9,41
8	7,48	8,19	6,83	8,11	7,45	8,83	7,01	8,67	6,67	10,93	12,72	9,39
9	7,4	8,17	6,7	7,6	8,67	6,67	6,75	8,11	5,97	10,92	12,76	9,35
10	6,88	8,11	5,84	6,96	8,11	5,97	7,16	8,25	6,22	10,9	12,64	9,4
Rerata	7,984	8,575	7,465	7,791	8,489	7,204	7,747	8,01	7,014	10,957	12,74	9,422



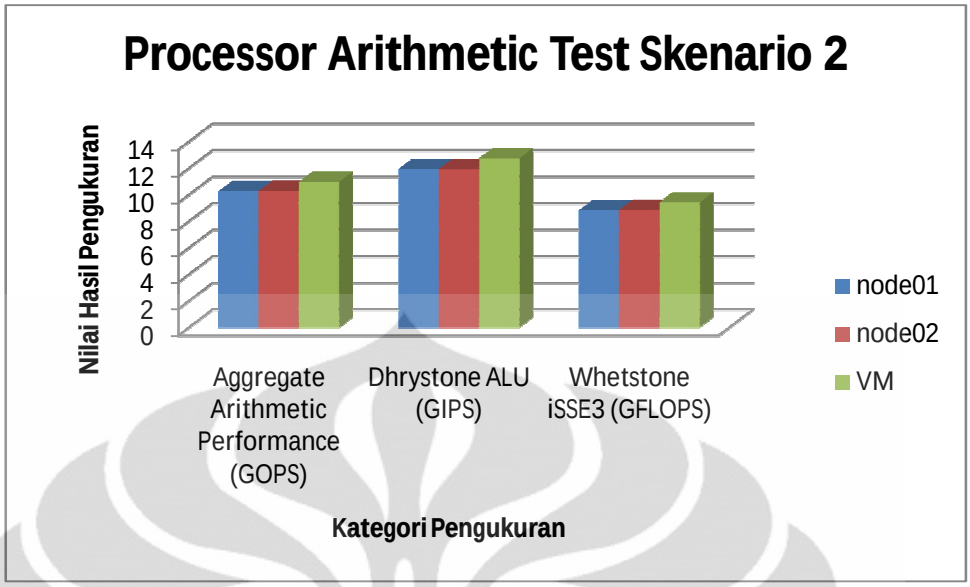
Gambar 4.15. Grafik Hasil *Benchmark Processor Arithmetic* Skenario 1

Berdasarkan hasil *benchmark* pada parameter *processor arithmetic* yang dapat dilihat pada Tabel 4.2 dan Gambar 4.15 untuk skenario 1, dapat dianalisa mengenai performa *cluster* dari parameter ini. Dari hasil yang didapat untuk

parameter *processor arithmetic* hasil *benchmark* dengan nilai lebih besar menunjukkan performa yang lebih baik, karena hasil *benchmark* ini banyaknya operasi dan intruksi per detik yang dilakukan prosesor. *Processor* memproses instruksi *arithmetic* dan *floating point*. Apabila diamati dari Tabel 4.2 dan Gambar 4.15 performa ketiga *node* dalam *cluster* tidak lebih baik dibandingkan sebuah VM dalam arti terjadi penurunan performa dalam *nodes cluster*. Ketiga *node* pada *cluster* memiliki nilai rata-rata 7,2-8,0 GOPS untuk kategori *Aggregate Arithmatic Performance*, sedangkan VM memiliki nilai sedir 11 GOPS. Dari satu kategori ini saja sudah tampak bahwa performa VM lebih baik dibanding performa *cluster* karena VM dapat mengolah lebih banyak operasi per detiknya dibanding VM pada *cluster*.

Tabel 4.3. Hasil *Benchmark Processor Arithmetic* Skenario 2

Tabel Hasil *Benchmark Processor Arithmetic* Skenario 2									
	*node*01			*node*02			VM		
Uji Coba ke-	Aggregate *Arithmetic* Performance (GOPS)	Dhrystone ALU (GIPS)	Whetstone iSSE3 (GFLOPS)	Aggregate *Arithmetic* Performance (GOPS)	Dhrystone ALU (GIPS)	Whetstone iSSE3 (GFLOPS)	Aggregate *Arithmetic* Performance (GOPS)	Dhrystone ALU (GIPS)	Whetstone iSSE3 (GFLOPS)
1	10,45	12,15	8,98	10,3	11,89	8,93	11	12,8	9,45
2	10,38	12,04	8,96	10,36	12,02	8,93	10,98	12,77	9,44
3	10,37	12,07	8,92	10,34	12,06	8,87	10,98	12,76	9,44
4	10,35	11,97	8,95	10,37	11,95	9	10,97	12,73	9,45
5	10,35	11,98	8,94	10,37	12	8,97	10,97	12,73	9,45
6	10,31	11,91	8,91	10,07	11,64	8,72	10,96	12,73	9,44
7	10,21	11,79	8,84	10,41	12,08	8,97	10,96	12,76	9,41
8	10,12	11,93	8,59	10,31	12,07	8,8	10,93	12,72	9,39
9	10,1	11,57	8,81	10,06	11,8	8,56	10,92	12,76	9,35
10	10,01	12	8,35	10,25	11,84	8,88	10,9	12,64	9,4
Rata-rata	10,265	11,941	8,825	10,284	11,935	8,863	10,957	12,74	9,422



Gambar 4.16. Grafik Hasil *Benchmark Processor Arithmetic* Skenario 2

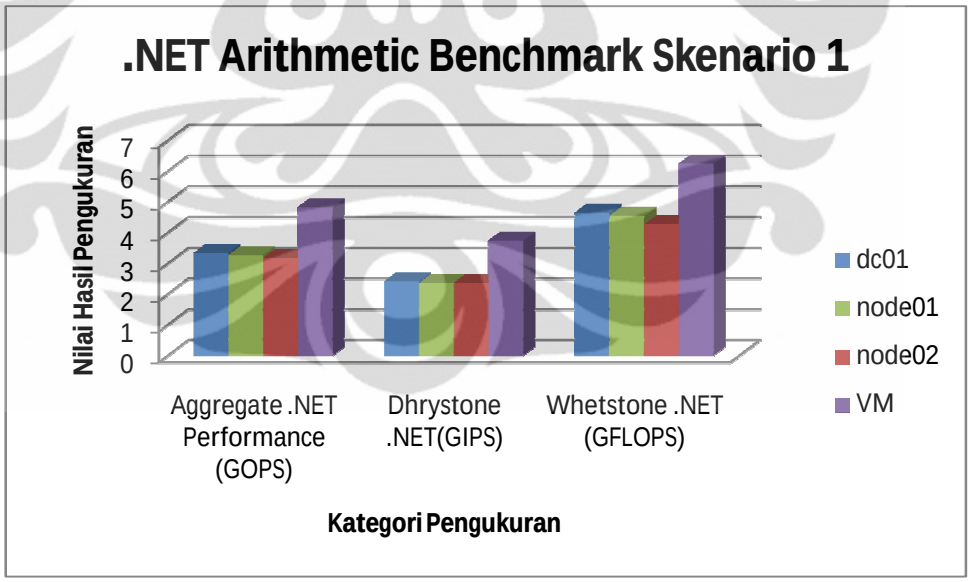
Sama halnya pada skenario 1, hasil yang didapat pada skenario kedua pun menunjukkan VM yang tidak menjadi bagian *cluster* memiliki hasil yang lebih baik dibandingkan para *nodes* di dalam *cluster*. Seperti yang dapat dilihat pada Tabel 4.3 dan Gambar 4.16 VM yang tidak di-*cluster* dapat menyelesaikan kurang lebih 11 giga operasi per detiknya sedangkan *nodes* pada *cluster* hanya menyelesaikan kurang lebih 10,3 giga operasi per detik.

4.3.2 Analisa parameter .NET *Arithmetic*

Benchmark pada parameter ini menunjukkan bagaimana *processor* memproses operasi .NET. Kategori pengukuran yang digunakan sama dengan pada parameter *processor arithmetic* yaitu *aggregate* .NET *arithmetic*, Whetstone .NET *arithmetic*, dan Dhrystone .NET *arithmetic*. Hasil *benchmark* yang lebih besar menunjukkan performa yang lebih baik karena seperti yang disebutkan sebelumnya hasil *benchmark* ini menunjukkan banyaknya operasi .NET yang dapat diproses oleh *processor* per detiknya.

Tabel 4.4 Hasil *Benchmark .NET Arithmetic* Skenario 1

Tabel Hasil *Benchmark* .NET *Arithmetic* Skenario 1												
	dc01			*node*01			*node*02			VM		
Uji Coba ke-	**Aggregate .NET Performance (GOPS)**	**Dhrystone .NET(GIPS)**	**Whetstone .NET (GFLOPS)**	**Aggregate .NET Performance (GOPS)**	**Dhrystone .NET(GIPS)**	**Whetstone .NET (GFLOPS)**	**Aggregate .NET Performance (GOPS)**	**Dhrystone .NET(GIPS)**	**Whetstone .NET (GFLOPS)**	**Aggregate .NET Performance (GOPS)**	**Dhrystone .NET(GIPS)**	**Whetstone .NET (GFLOPS)**
1	3,78	2,41	5,93	3,3	2,36	4,61	3,09	2,39	4	4,87	3,77	6,3
2	3,73	2,33	5,97	3,26	2,42	4,38	3,01	2,18	4,15	4,87	3,77	6,29
3	3,67	2,48	5,44	3,78	2,39	5,96	3,27	2,39	4,48	4,87	3,78	6,25
4	3,4	2,35	4,91	3,07	2,32	4,06	3,12	2,35	4,14	4,86	3,77	6,27
5	3,23	2,52	4,13	2,99	2,35	3,81	3	2,27	3,97	4,83	3,73	6,27
6	3,2	2,39	4,27	3,1	2,5	3,85	3,39	2,34	4,92	4,82	3,7	6,27
7	3,19	2,49	4,09	3,12	2,16	4,5	3,25	2,33	4,54	4,8	3,69	6,24
8	3,13	2,31	4,25	2,98	2,31	3,86	3,58	2,79	4,6	4,8	3,69	6,24
9	3,06	2,44	3,83	3,59	2,32	5,53	3,1	2,26	4,25	4,79	3,73	6,16
10	2,97	2,41	3,67	3,47	2,44	4,93	2,96	2,41	3,65	4,76	3,7	6,12
Rata-rata	3,336	2,413	4,649	3,266	2,357	4,549	3,177	2,371	4,27	4,827	3,733	6,241



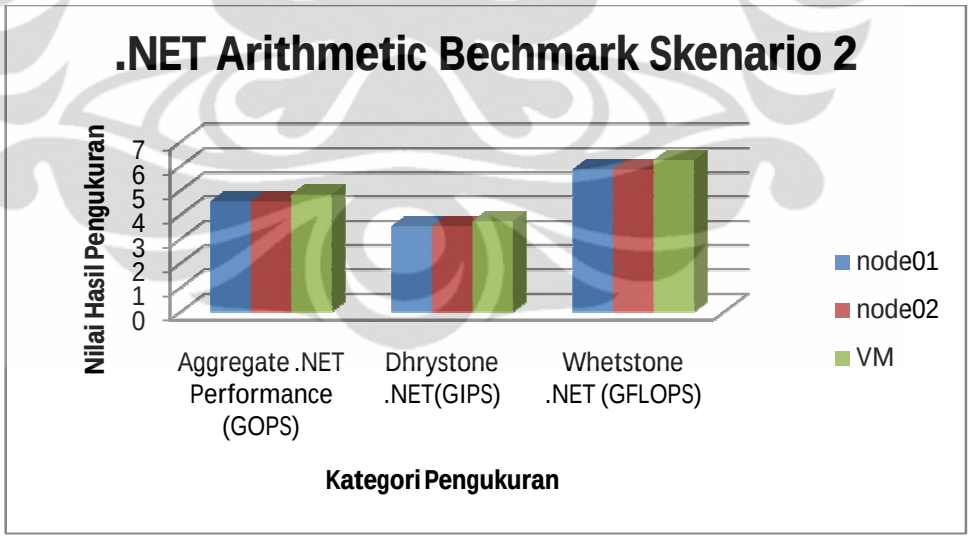
Gambar 4.17 Grafik Hasil *Benchmark* .NET *Arithmetic* Skenario 1

Dari Tabel 4.3 dan Gambar 4.17 bahwa untuk parameter .NET *Arithmetic* ini VM yang tidak menjadi anggota *cluster* menunjukkan performa eksekusi operasi .NET *Arithmetic* yang lebih baik yaitu 4,8 giga operasi per detik,

sedangkan *cluster* menunjukkan nilai rata-rata dari ketiga *nodes* kurang lebih 3,3 giga operasi per detik.

Tabel 4.5 Hasil *Benchmark* .NET *Arithmetic* Skenario 2

Tabel Hasil *Benchmark* .NET *Arithmetic* Skenario 2									
	*node*01			*node*02			VM		
Uji Coba ke-	Aggregate .NET Performance (GOPS)	Dhrystone .NET(GIPS)	Whetstone .NET (GFLOPS)	Aggregate .NET Performance (GOPS)	Dhrystone .NET(GIPS)	Whetstone .NET (GFLOPS)	Aggregate .NET Performance (GOPS)	Dhrystone .NET(GIPS)	Whetstone .NET (GFLOPS)
1	4,6	3,57	5,92	4,52	3,5	5,84	4,87	3,77	6,3
2	4,59	3,58	5,9	4,64	3,63	5,93	4,87	3,77	6,29
3	4,57	3,56	5,87	4,56	3,56	5,85	4,87	3,78	6,25
4	4,56	3,53	5,89	4,55	3,53	5,88	4,86	3,77	6,27
5	4,56	3,51	5,91	4,55	3,54	5,86	4,83	3,73	6,27
6	4,56	3,57	5,82	4,29	3,18	5,79	4,82	3,7	6,27
7	4,53	3,51	5,85	4,57	3,52	5,91	4,8	3,69	6,24
8	4,53	3,51	5,83	4,55	3,54	5,84	4,8	3,69	6,24
9	4,51	3,48	5,84	4,57	3,56	5,86	4,79	3,73	6,16
10	4,46	3,42	5,82	4,63	3,65	5,86	4,76	3,7	6,12
Rata-rata	4,547	3,524	5,865	4,543	3,521	5,862	4,827	3,733	6,241



Gambar 4.18 Grafik Hasil *Benchmark* .NET *Arithmetic* Skenario 2

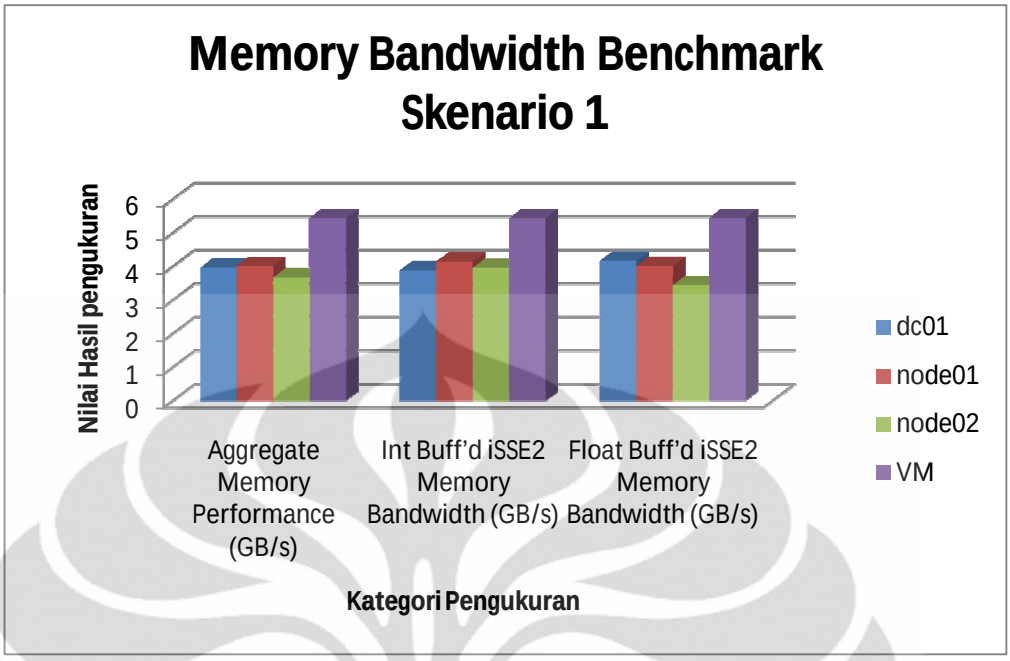
Apabila dilihat dari grafik pada Gambar 4.18 dan nilai hasil *benchmark* pada Tabel 4.5 untuk analisa skenario kedua untuk parameter .NET *arithmetic* hasilnya tidak jauh berbeda dengan hasil pada skenario satu untuk parameter yang sama. VM yang tidak di-*cluster* masih menunjukkan performa yang lebih baik

dibandingkan dua *nodes* dalam *cluster*, yang berbeda hanya pada selisih operasi yang dapat diproses tidak sebanyak pada skenario satu yaitu untuk skenario kedua selisihnya sedir 0,3 giga operasi per detiknya. Namun secara keseluruhan untuk parameter ini VM masih menunjukkan performa yang lebih baik.

4.3.3 Analisa Parameter *Benchmark* Memory Bandwidth

Tabel 4.6 Hasil *Benchmark Memory Bandwidth* Skenario 1

Tabel Hasil *Benchmark* Memory Bandwith Skenario 1												
	dc01			***node*01**			***node*02**			**VM**		
Uji Coba ke-	**Aggregate Memory Performance (GB/s)**	**Int Buff'd iSSE2 Memory Bandwidth (GB/s)**	**Float Buff'd iSSE2 Memory Bandwidth (GB/s)**	**Aggregate Memory Performance (GB/s)**	**Int Buff'd iSSE2 Memory Bandwidth (GB/s)**	**Float Buff'd iSSE2 Memory Bandwidth (GB/s)**	**Aggregate Memory Performance (GB/s)**	**Int Buff'd iSSE2 Memory Bandwidth (GB/s)**	**Float Buff'd iSSE2 Memory Bandwidth (GB/s)**	**Aggregate Memory Performance (GB/s)**	**Int Buff'd iSSE2 Memory Bandwidth (GB/s)**	**Float Buff'd iSSE2 Memory Bandwidth (GB/s)**
1	5,23	3,149	5,242	3,61	5,238	3,21	3,47	4,102	2,941	5,45	5,451	5,457
2	3,95	3,274	3,327	4,1	3,249	4,43	3,14	3,196	3,092	5,45	5,456	5,44
3	3,3	4,774	3,323	3,79	3,249	4,43	3,88	4,097	3,68	5,43	5,413	5,447
4	3,64	3,677	4,681	3,45	3,822	3,12	3,06	2,936	3,187	5,43	5,448	5,412
5	4,24	2,74	4,797	4,07	4,772	3,468	3,2	3,214	3,193	5,43	5,404	5,452
6	3,63	3,393	5,305	4,35	5,246	3,609	3,39	3,697	3,115	5,42	5,432	5,416
7	4,15	3,989	3,326	4,04	4,025	4,061	3,95	4,891	3,185	5,42	5,411	5,432
8	3,98	3,287	3,323	4,16	3,29	5,256	4,09	4,975	3,357	5,41	5,405	5,424
9	3,3	5,265	2,961	3,36	3,258	3,465	3,34	3,214	3,465	5,41	5,409	5,402
10	4,06	5,206	5,254	5,13	5,214	5,057	5,12	5,12	5,111	5,4	5,386	5,419
Rata-rata	3,948	3,8754	4,1539	4,006	4,1363	4,0106	3,664	3,9442	3,4326	5,425	5,4215	5,4301

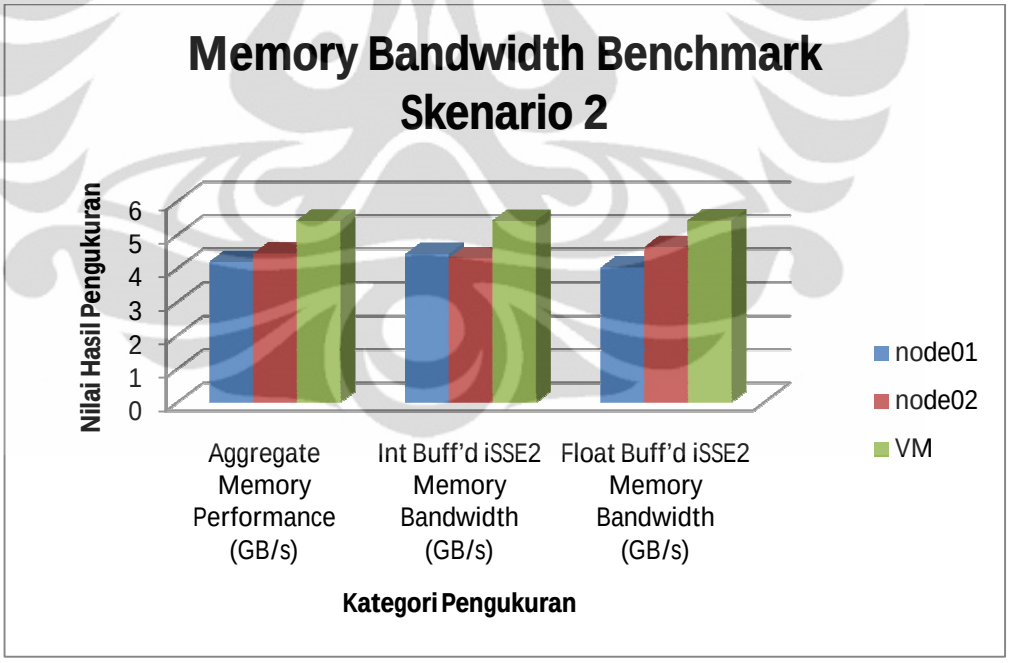


Gambar 4.19 Grafik Hasil *Benchmark Memory Bandwidth* Skenario 1

Pada proses *benchmark* terhadap parameter memory bandwidth menunjukkan kinerja dari memory subsystem, dimana untuk nilai hasil *benchmark* yang lebih besar menunjukkan performa yang lebih baik. Dari Tabel 4.6 dan Gambar 4.19 dapat dianalisa bahwa VM yang tidak di-*cluster* dinilai dari ketiga kategori perhitungan memberikan nilai hasil *benchmark* lebih tinggi dibanding ketiga *nodes* dalam *cluster*. Saat ketiga *nodes* nilai-nya berkisar antara 3,7 GB/s – 4, 1 GB/s, nilai *benchmark* dari VM berkisar pada 5,4 GB/s. Dari sini dapat ditarik kesimpulan bahwa pada skenario pertama untuk parameter memory bandwith VM yang tidak di-*cluster* menunjukkan performa yang lebih baik.

Tabel 4.7 Hasil *Benchmark Memory Bandwidth* Skenario 2

Tabel Hasil *Benchmark* Memory Bandwith Skenario 2									
	*node*01			*node*02			VM		
Uji Coba ke-	Aggregate Memory Performance (GB/s)	Int Buff'd iSSE2 Memory Bandwidth (GB/s)	Float Buff'd iSSE2 Memory Bandwidth (GB/s)	Aggregate Memory Performance (GB/s)	Int Buff'd iSSE2 Memory Bandwidth (GB/s)	Float Buff'd iSSE2 Memory Bandwidth (GB/s)	Aggregate Memory Performance (GB/s)	Int Buff'd iSSE2 Memory Bandwidth (GB/s)	Float Buff'd iSSE2 Memory Bandwidth (GB/s)
1	3.85	3.817	3.881	4.39	3.918	4.909	5.45	5.451	5.457
2	4.19	4.691	3.743	4.17	3.517	4.87	5.45	5.456	5.44
3	3.92	3.914	3.921	4.29	4.723	3.89	5.43	5.413	5.447
4	4.39	5.166	3.73	4.47	3.913	5.117	5.43	5.448	5.412
5	3.88	3.93	3.825	4.58	4.673	4.479	5.43	5.404	5.452
6	4.33	5.012	3.749	4.23	3.689	4.831	5.42	5.432	5.416
7	3.88	3.93	3.832	4.35	4.63	4.086	5.42	5.411	5.432
8	4.36	5.055	3.757	4.48	3.898	5.151	5.41	5.405	5.424
9	4.38	3.755	5.108	4.4	5.137	3.77	5.41	5.409	5.402
10	4.82	4.988	4.659	5.03	4.83	5.235	5.4	5.386	5.419
Rata-rata	4.2	7.9628	4.0205	4.439	4.2928	4.6338	5.425	5.4215	5.4301



Gambar 4.20 Grafik Hasil *Benchmark* Memory Bandwidth Skenario 2

Dari Tabel 4.7 dan Gambar 4.20 di atas dapat dilihat bahwa pada skenario 2 untuk ketiga kategori yaitu *Aggregate Memory Performance, Integer Buffered iSSE2 Memory Bandwidth* dan *Float Buffered iSSE2 Memory Bandwidth* nilai

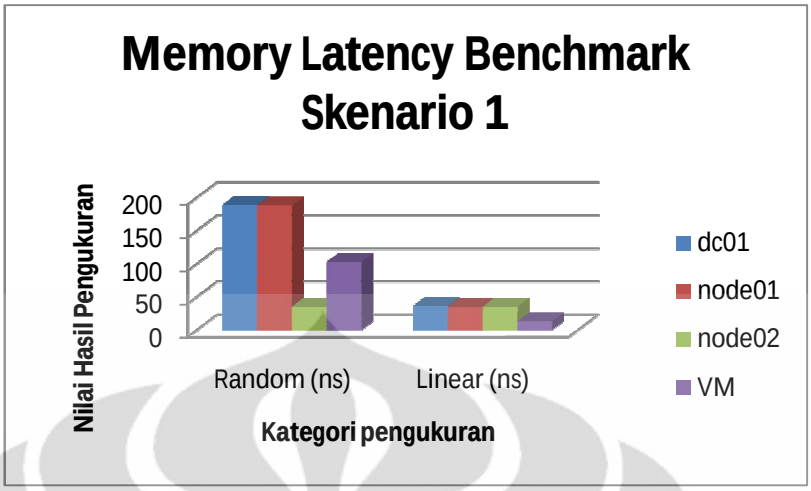
hasil pengukurun dari VM yang tidak di-*cluster* menunjukkan hasil yang lebih baik. Karena seperti yang disebutkan sebelumnya nilai yang lebih tinggi menunjukkan performa yang lebih baik. Sehingga dapat disimpulkan untuk skenario kedua ini VM yang tidak di-*cluster* menunjukkan performa yang lebih baik.

4.3.4 Analisa Parameter *Benchmark Memory Latency*

Parameter *benchmark* yang akan dianalisa selanjutnya adalah *memory latency* yaitu waktu yang dibutuhkan untuk mengakses memori. Untuk parameter ini ada dua kategori pengukuran yang dilakukan yaitu *random memory access* dan *linear memory access*. Pada dasarnya yang diukur adalah waktu respon untuk mengakses memori secara random dan secara sekuensial. Nilai yang lebih kecil menunjukkan performa yang lebih baik, karena ini berarti waktu yang dibutuhkan untuk mengakses memorinya lebih cepat.

Tabel 4.8 Hasil *Benchmark Memory Latency* Skenario 1

Tabel Hasil *Benchmark* Memory Latency Skenario 1								
	dc01		***node*01**		***node*02**		**VM**	
Uji Coba ke-	**Random (ns)**	**Linear (ns)**	**Random (ns)**	**Linear (ns)**	**Random (ns)**	**Linear (ns)**	**Random (ns)**	**Linear (ns)**
1	189.1	37.4	189.6	35.9	33.3	33.3	101.9	13
2	189.2	35.2	188.5	36.2	37.6	37.6	101.5	13
3	182.2	33.8	195.8	37.4	36.7	36.7	101.8	12.9
4	208.3	48	183.5	37.6	33.1	33.1	102.7	12.9
5	186.9	34.8	186	37.3	39	39	101.3	13.1
6	180.3	35.2	193.9	34.6	36.5	36.5	102.2	12.7
7	197.1	35.9	181.6	34.9	35.3	35.3	102.2	13.1
8	177.8	36.7	181.7	37.1	36.6	36.6	102.4	13.1
9	193.8	35	190.7	34	37.7	37.7	105.4	13.1
10	186	31	188.9	28.5	29.3	29.3	102.3	13.1
Rata-rata	**189.07**	**36.4666667**	**188.02**	**35.35**	**35.51**	**35.51**	**102.37**	**13**

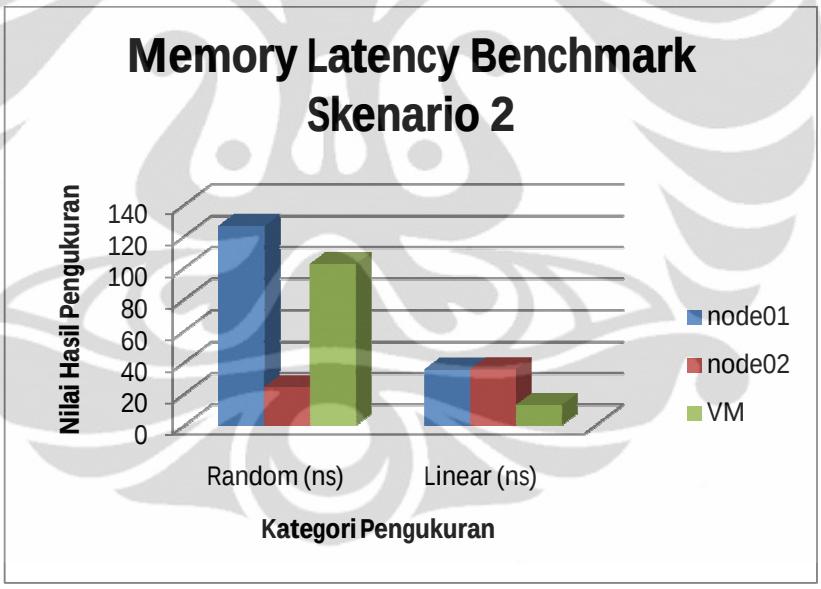


Gambar 4.21 Grafik Hasil *Benchmark Memory Latency* Skenario 1

Dari Tabel 4.8 dan Gambar 4.21 dapat dilihat bahwa dc 01 sebagai *domain controller cluster* dan *node*01 menunjukkan waktu respon yang lambat untuk kategori random, sedangkan *node*02 menunjukkan waktu respon tercepat dibandingkan kedua *nodes cluster* lainnya dan dibandingkan VM yang tidak di *cluster*. Namun apabila melihat waktu respon pada kategori linear ketiga *nodes* dalam *cluster* memiliki waktu respon lebih lambat dibandingkan dengan VM yang tidak di-*cluster*. Jadi apabila diamati secara menyeluruh untuk parameter *memory latency* skenario pertama, VM yang tidak di *cluster* memiliki performa yang lebih baik dibandingkan *cluster*.

Tabel 4.9 Hasil *Benchmark Memory Latency* Skenario 2

Tabel Hasil *Benchmark* Memory Latency Skenario 2						
	*node*01		*node*02		VM	
Uji Coba ke-	Random (ns)	Linear (ns)	Random (ns)	Linear (ns)	Random (ns)	Linear (ns)
1	124.7	35.9	24.4	33.3	101.9	13
2	134.2	36.2	24.2	37.6	101.5	13
3	141.6	37.4	23.8	36.7	101.8	12.9
4	125	37.6	24.1	33.1	102.7	12.9
5	125.1	37.3	24.5	39	101.3	13.1
6	124	34.6	24.4	36.5	102.2	12.7
7	125.2	34.9	24.3	35.3	102.2	13.1
8	133.1	37.1	23.1	36.6	102.4	13.1
9	123.4	34	24.2	37.7	105.4	13.1
10	109.8	28.5	23.7	29.3	102.3	13.1
Rata-rata	126.61	35.35	24.07	35.51	102.37	13



Gambar 4.22 Grafik Hasil *Benchmark* Memory Latency Skenario 2

Untuk skenario kedua pada kondisi random sekali lagi respon time dari *node*02 menunjukkan waktu yang paling cepat dibandingkan *node*01 dan VM yang tidak di-*cluster*. Namun suatu *cluster* bekerja dalam bentuk kesatuan dan pada skenario kedua pada kenyataannya seluruh kegiatan aplikasi dan servis dijalankan oleh *node*01 dan *node*02 hanya bertukar informasi dengan *node*01 untuk mengetahui kondisi *node*01. Jadi, jika performa *cluster* ini dianalisa

berdasarkan letak aplikasi dan servis terjadi dibandingkan dengan VM yang tidak di-*cluster*, sekali lagi VM yang tidak di-*cluster* menunjukkan waktu respon yang lebih cepat. Untuk kategori akses momori secara linear nilai yang dihasilkan lebih mudah dianalisa dimana kedua *node* dalam *cluster* sama-sama memiliki waktu respon yang lebih lambat dibandingkan VM yang tidak di-*cluster*, dan apabila ditarik kesimpulan keseluruhan untuk skenario kedua *cluster* memiliki waktu respon yang lebih lambat untuk pengaksesan memorinya dibandingkan VM yang tidak di-*cluster*.

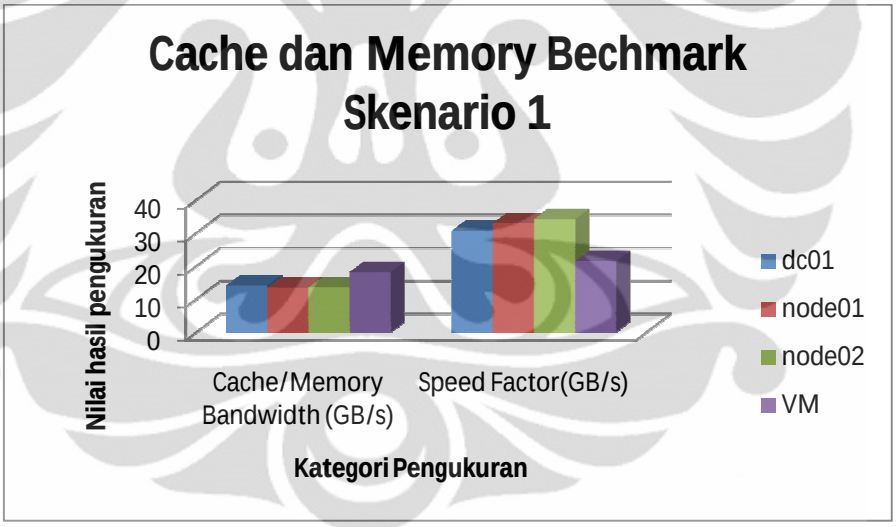
4.3.5 Analisa Parameter *Benchmark Cache* dan Memory

Untuk parameter *benchmark cache* dan memory ada dua kategori yang dilihat yaitu bandwidth *cache* dan memori dan faktor kecepatan, dimana untuk kategori pertama yaitu bandwidth *cache* dan memori nilai pengukuran yang lebih besar menunjukkan performa yang lebih baik, sedangkan untuk faktor kecepatan nilai yang lebih kecil menunjukkan performa yang lebih baik. Oleh karena itu apabila dilihat pada Gambar 4.10 dan 4.11 nilai bandwith *cache* dan meori berbanding terbalik dengan nilai faktor kecepatan.

Dilihat dari Tabel 4.10 range nilai uji coba pada ketiga *nodes* untuk kategori bandwidth *cache* dan memori berkisar antara 12,6 GB/s sampai 17,2 GB/s sedangkan sebuah VM nilai uji cobanya berkisar pada nilai 18,1 GB/s. Sehingga sudah jelas bahwa nilai uji coba VM yang tidak di-*cluster* ini menunjukkan bahwa performa VM ini lebih baik dibandingkan ketiga *nodes* anggota *cluster*, karena seperti yang telah disebutkan nilai hasil *benchmark* yang lebih tinggi menunjukkan performa yang lebih baik.

Tabel 4.10 Hasil *Benchmark Cache* dan *Memory* Skenario 1

Tabel Hasil *Benchmark Cache* and Memory Skenario 1								
	dc01		*node*01		*node*02		VM	
Uji Coba ke-	*Cache*/Memory Bandwidth (GB/s)	Speed Factor (GB/s)	*Cache*/Memory Bandwidth (GB/s)	Speed Factor (GB/s)	*Cache*/Memory Bandwidth (GB/s)	Speed Factor (GB/s)	*Cache*/Memory Bandwidth (GB/s)	Speed Factor (GB/s)
1	17.275	22.3	14.35	25.7	14.56	27.9	18.144	21.2
2	17.17	22.8	14.27	24.1	13.41	32.5	18.142	21.3
3	17.224	22.4	15.05	40.2	15.52	20.8	18.139	21.5
4	13.323	31.7	12.96	39	13.09	38.3	18.135	21.6
5	13.025	29.5	12.81	39.3	12.73	38.4	18.128	21.6
6	13.11	38.8	12.7	27.9	12.85	42.5	18.124	21.4
7	12.574	30.5	12.88	39.9	13.1	32.2	18.116	21.4
8	12.646	40.9	12.92	26.8	13.1	38.9	18.109	21.4
9	12.66	39.6	12.79	39	13.21	28.6	18.102	21.5
10	13.2	28.2	13.42	27.4	13.46	41.7	18.091	21.7
Rata-rata	14.2207	30.67	13.415	32.93	13.503	34.18	18.123	21.46

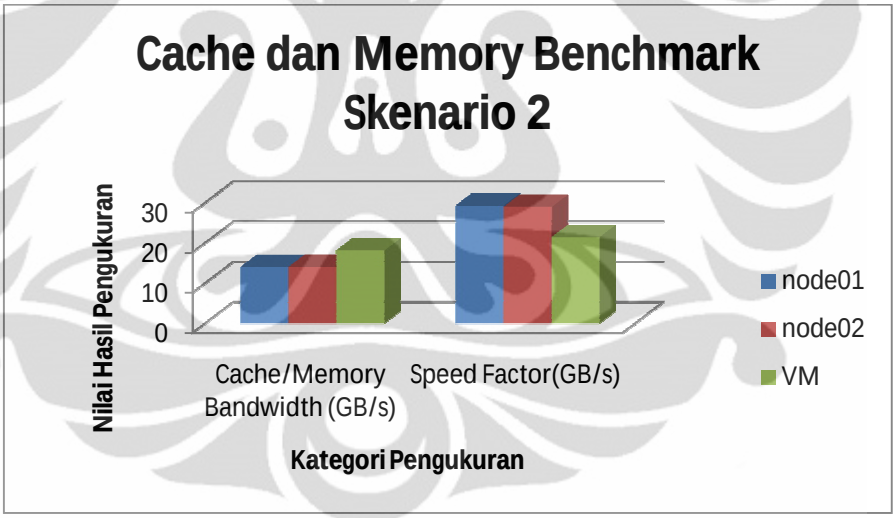


Gambar 4.23 Grafik Hasil *Benchmark Cache* dan *Memory* Skenario 1

Gambar 4.23 menunjukkan grafik hasil *benchmark cache* dan memori untuk skenario pertama. Grafik ini dibuat berdasarkan data pada Tabel 4.10 dan dari grafik ini tampak bahwa untuk kategori faktor kecepatan ketiga *nodes* pada *cluster* memiliki nilai yang lebih tinggi dibandingkan VM yang tidak di-*cluster*. Namun untuk kategori ini nilai yang lebih rendahlah yang menunjukkan performa yang lebih baik sehingga VM yang tidak di-*cluster* memiliki performa yang lebih baik.

Tabel 4.11 Hasil *Benchmark Cache* dan Memori Skenario 2

Tabel Hasil *Benchmark Cache* and Memory Skenario 2						
	*node*01		*node*02		VM	
Uji Coba ke-	*Cache*/Memory Bandwidth (GB/s)	Speed Factor (GB/s)	*Cache*/Memory Bandwidth (GB/s)	Speed Factor (GB/s)	*Cache*/Memory Bandwidth (GB/s)	Speed Factor (GB/s)
1	14.264	25.3	13.64	28.1	18.144	21.2
2	14.225	25.2	14.24	30.2	18.142	21.3
3	14.185	32.1	14.06	23.4	18.139	21.5
4	14.154	30.8	13.7	34.1	18.135	21.6
5	14.1	24.4	14.27	25.5	18.128	21.6
6	14.041	32.7	14.07	31.6	18.124	21.4
7	14.04	26.1	14.41	26.7	18.116	21.4
8	13.932	33.8	13.95	33.5	18.109	21.4
9	13.72	25.8	13.92	25.3	18.102	21.5
10	13.568	36.8	14.2	31.9	18.091	21.7
Rata-rata	14.0229	29.3	14.046	29.03	18.123	21.46



Gambar 4.24 Grafik Hasil *Benchmark Cache* dan Memory Skenario 1

Pada skenario kedua kesimpulan yang sama dapat ditarik yaitu VM yang tidak di-*cluster* menunjukkan performa yang lebih baik. Hal ini dapat dilihat dari Tabel 4.11 dan Gambar 4.24 dimana nilai kedua *nodes* anggota *cluster* memiliki nilai lebih kecil untuk kategori bandwidth *cache* dan meori dibandingkan VM yang tidak di-*cluster* sedangkan untuk kategori faktor kecepatan nilai kedua *nodes* tersebut nilainya lebih besar dibandingkan VM yang tidak di-*cluster*.

4.4 Analisa Akhir

Setelah menganalisa satu per satu untuk tiap parameter berbeda pada subbab sebelumnya, pada subbab ini yang dilakukan adalah menganalisa hasilnya secara keseluruhan. Dari hasil analisa pada sub bab sebelumnya diperoleh hasil yaitu performa VM yang tidak di-*cluster* memberikan hasil yang lebih baik dibandingkan *cluster* yang dibuat.

Apabila dianalisa lebih jauh hal ini bisa terjadi karena sistem *cluster* yang dibuat merupakan *cluster* failover, dimana pada *cluster* failover aplikasi dan servis dijalankan hanya pada sebuah *nodes*. Pada skripsi ini aplikasi dan servis untuk skenario pertama berjalan pada *nodes* dc01 sebagai *domain controller*. Berbeda dengan *cluster* load balance yang cara kerjanya adalah membagi beban secara merata pada ketiga *node*snya pada *cluster* failover yang dibuat ini *nodes* yang bekerja hanya satu sedangkan yang lain hanya bertukar informasi sehingga memiliki informasi terbaru saat terjadi kegagalan dan siap mengambil alih aplikasi dan servis yang sedang berjalan di *node* lain. Untuk itu apabila dibandingkan dengan VM yang tidak di-*cluster* performa-nya tidak menunjukkan hasil yang lebih baik.

Pada sistem yang di-*cluster* penurunan performa terjadi karena yang menjadi titik berat dari sistem ini adalah segi keamanannya. Sehingga sumber daya yang ada dialokasikan sebagian untuk kebutuhan pengamanan dalam hal ini apabila terjadi failure. Dengan begini otomatis saat dilakukan uji performa terjadi penurunan karena adanya sebagian sumber daya yang telah terpakai untuk kebutuhan keamanan.

Untuk menjalankan servis dan aplikasi pada *cluster* tiap *node* harus menjalankan beberapa aplikasi dasar suatu *cluster* sehingga alokasi memori dan prosesor-nya terpakai sebagian selain oleh servis dan aplikasi yang dijalankan oleh *cluster* itu sendiri yaitu File Server. Sedangkan pada VM yang tidak di-*cluster* apabila ingin menjalankan aplikasi file server cukup melakukan konfigurasi untuk membentuk file server tersebut dan pada saat dijalankan aplikasi yang berjalan tidak ada yang lain seperti pada *cluster*, sehingga penggunaan memori dan prosesor-nya pun lebih sedikit.

Jika dilihat dari performa tiap *node* dibandingkan VM yang tidak di-*cluster*, *cluster* memang menghasilkan performa yang tidak lebih baik, yaitu terjadi penurunan rata-rata dari seluruh parameter sebesar 41% untuk skenario pertama dan 54% untuk skenario kedua. Namun apabila dilihat dari segi *availability*-nya *cluster* akan menunjukkan hasil yang jauh lebih baik. Seperti yang telah diuji bahwa pada saat terjadi kegagalan pada sebuah *nodes*, maka *nodes* lain akan secara otomatis mengambil alih kerja *nodes* yang mengalami kegagalan dengan performa *nodes* yang hampir sama dengan *nodes* yang mengalami kegagalan. Sedangkan apabila terjadi kegagalan pada VM yang tidak di-*cluster*, walaupun VM ini memiliki performa yang lebih baik maka seluruh sistem akan down dan akan terjadi kegagalan secara menyeluruh tidak ada yang dapat mengambil alih. Sehingga aplikasi dan servis yang dijalankan tidak akan bisa diakses lagi sampai terjadi perbaikan. Jadi, untuk perbandingan dari performa tiap *node*snya *cluster* menghasilkan performa tidak lebih baik dibandingkan VM yang tidak di-*cluster* namun dari segi *availability* terkait dengan keamanan dan kehandalan sistem ini menunjukkan hasil yang lebih baik.

BAB 5
KESIMPULAN

1. Sistem *virtual computer cluster* yang dibangun bekerja sesuai dengan konsep kerjanya yaitu sebagai *failover cluster*.
2. Hasil *benchmark* sistem *cluster* skenario 1 untuk lima parameter yaitu *processor arithmetic, .NET arithmetic, memory bandwidth, memory latency,* dan *cache and memory,* masing-masing adalah 7,841 GOPS, 3,259 GOPS, 3,873 GOPS, 137,53 ns dan 35,776 ns, dan 13,713 GB/s dan 32,593 GB/s
3. Hasil *benchmark* sistem *cluster* skenario 1 untuk lima parameter yaitu *processor arithmetic, .NET arithmetic, memory bandwidth, memory latency,* dan *cache and memory,* masing-masing adalah 10,275 GOPS, 4,545 GOPS, 4,319 GOPS,75,34 ns dan 35,43 ns, dan 14,034 GB/s dan 29,165 GB/s.
4. Hasil *benchmark* VM yang tidak di *cluster* skenario 1 dan 2 untuk lima parameter yaitu *processor arithmetic, .NET arithmetic, memory bandwidth, memory latency,* dan *cache and memory,* masing-masing adalah 10,957 GOPS, 4,827 GOPS, 5,425 GOPS, 102,37 ns dan 13 ns, dan 18,123 GB/s dan 21,46 GB/s.
5. Dari hasil test performa untuk lima parameter berbeda diperoleh hasil bahwa sistem *failover virtual computer cluster* ini tidak lebih baik dibandingkan dengan kerja satu sistem yang tidak di *cluster*. Terjadi penurunan rata-rata performa sebesar 41 % untuk skenario pertama dan 54% untuk skenario kedua.
6. Penurunan performa pada sistem *cluster* terjadi akibat adanya kebutuhan dalam peningkatan kehandalan dalam hal ini terkait *availability*.
7. Konfigurasi dan kerja cluster yang dibuat menunjukkan hasil yang lebih baik dibandingkan sebuah sistem yang tidak di cluster dilihat dari segi *availability* pada saat terjadi *failure*.
8. Pertimbangan lebih lanjut diperlukan dalam membangun sebuah cluster. Tujuan dari pembuatan *cluster* tersebut harus diperjelas sehingga manfaat dari cluster tersebut dapat dirasakan.

DAFTAR ACUAN

[1] *Clustering Tutorial*. Diakses 1 Desember 2010, dari scfbio-ittd. http://www.scfbio-iitd.res.in/doc/clustering.pdf

[2] *Virtualisasi*. Diakses 1 Desember 2010, dari Wikipedia. http://id.wikipedia.org/wiki/Virtualisasi

[3] *Server Clustering*. Diakses 1 Desember 2010, dari Microsoft http://msdn.microsoft.com/enus/library/ff649250.aspx

[4] Morrison, Richard.S.2003.*Cluster Computing: Architectures, Operating Systems, Parallel Processing & Programming Languages*. GNU General Public Licence.

[5] V. Rajaravivarma. 2006. *Computer Clustering Using Home Network*. Central Connecticut State University.

[6] Corporation, Microsoft. "Hyper-V Architecture." Microsoft Corporation, 2008.

[7] *Benchmarking Hyper-V on Windows Server 2008 R2 x64*. Diakses 10 Mei 2011, dari capitalhead. http://capitalhead.com/articles/benchmarking-hyper-v-on-windows-server-2008-r2x64.aspx

[8] Qin, Xiao, Hong Jiang, Adam Manzanares, Xiaojun Ruan, Shu Yin. 2010. *Communication-Aware Load Balancing for Parallel Applications on Clusters*. IEEE

[9] *How to create a Windows Server 2008 Cluster within Hyper-V using simulated iSCSI storage*. Diakses 8 Mei 2011, dari blog-technet. http://blogs.technet.com/b/pfe-ireland/archive/2008/05/16/how-to-create-a-windows-server-2008-cluster-within-hyper-v-using-simulated-iscsi-storage.aspx

[10] Kaufmann, Russ. 2009. Hyper-V and High Availability Storage. Starwind Software Inc.

[11] *Designing a Cluster Computer*. Diakses 1 Desember 2010, dari ameslab http://www.scl.ameslab.gov/Projects/parallel_computing/cluster_design.html

[12] *Microsoft Hyper-V Server: System Requirements*. Diakses pada 3 Juni 2011, dari Microsoft. http://www.microsoft.com/hyper-v-server/en/us/system-requirements.aspx

[13] *Recast of the WEEE and RoHS Directives proposed*. Diakses 1 Desember 2010. http://ec.europa.eu/environment/waste/weee/index_en.htm

[14] *Performance Testing Guidance for Web Applications*. http://msdn.microsoft.com/en-us/library/bb924376.aspx