PERBANDINGAN KINERJA PENDEKATAN VIRTUALISASI

Rio Rasian[1] dan Petrus Mursanto[2]

[1]Magister Teknologi Informasi, Universitas Indonesia, Jl. Salemba Raya 4, Jakarta, 12000, Indonesia
[2] Enterprise Computing Lab – Fakultas Ilmu Komputer, Universitas Indonesia, Kampus Baru UI, Depok, 16424, Indonesia

E-mail: rio@gmail.com

Abstrak

Kajian terhadap beberapa pendekatan virtualisasi telah dilakukan dan dilaporkan dalam *paper* ini. Pendekatan virtualisasi yang dikaji adalah *full virtualization*, *hardware-assisted virtualization, paravirtualization,* dan *operating system-level virtualization*. Evaluasi dilakukan melalui pengukuran kinerja masing-masing pendekatan dengan beban kerja tertentu saat dijalankan menggunakan virtualisasi dan saat virtualisasi tidak digunakan. Hasil eksperimen menunjukkan bahwa *operating system-level virtualization* lebih efisien dalam hal penggunaan sumber daya komputer dibanding pendekatan lainnya. Hasil evaluasi menunjukkan bahwa pendekatan virtualisasi seharusnya ikut menjadi pertimbangan dalam memilih solusi virtualisasi.

Kata Kunci: *virtualization, paravirtualization, operating system level*

Abstract

Study of some of the virtualization approach has been carried out and reported in this paper. Virtualization approaches that were examined are full virtualization, hardware-assisted virtualization, paravirtualization, and operating system-level virtualization. Evaluation is done by measuring the performance of each approach with a particular work load when run using virtualization and virtualization when not in use. The experimental results show that the operating system-level virtualization is more efficient in the use of computer resources than other approaches. Evaluation results show that the virtualization approach should help to be a consideration in choosing a virtualization solution.

Keywords: *virtualization, paravirtualization, operating system level*

1. Pendahuluan

Virtualisasi, dalam dunia teknologi informasi bisa berarti banyak hal. Secara umum virtualisasi adalah teknik untuk menyembunyikan karakter fisik suatu sumber daya komputer dari cara yang digunakan oleh sistem lain, aplikasi atau pengguna untuk berinteraksi dengan sumber daya tersebut [1]. Akan tetapi, pada penelitian ini yang dimaksud dengan virtualisasi adalah apa yang biasa disebut sebagai *platform virtualization*. Menurut Ramanathan & Bruening, "*platform virtuali-zation can be defined as the creation of a logically partitioned computing system that runs on top of an actual platform*" [2].

Saat ini, penggunaan virtualisasi diprediksi akan terus berkembang dengan cepat seiring dengan tuntutan global akan penghematan energi dan kebutuhan tertentu dari suatu organisasi. Salah satu faktor utama penggunaan virtualisasi saat ini adalah konsolidasi *server*. Dengan melakukan konsolidasi *server*, beberapa beban kerja dapat disatukan dalam sebuah komputer sehingga lebih menghemat penggunaan energi dan ruang [3]. Hal ini dikarenakan virtualisasi memungkinkan beberapa sistem operasi untuk berjalan secara bersamaan di dalam beberapa komputer atau mesin *virtual* pada satu komputer fisik [1], sehingga pada akhirnya dapat mengoptimalkan penggunaan sumber daya komputer yang umumnya hanya terpakai antara 10-15% [3][4].

Tuntutan di atas telah memicu berbagai pihak untuk mengeluarkan berbagai macam solusi perangkat lunak untuk virtualisasi dengan pendekatan yang berbeda-beda. Ada tiga pendekatan virtualisasi, yaitu *full virtualization, paravirtualization,* dan *hardware-assisted virtualization* [4]. Namun menurut Jones ada dua pendekatan virtualisasi lainnya, yaitu *hardware emulation* dan *operating system-level virtualization* [5]. Bahkan saat ini sebagian *vendor*

sudah ada yang menyediakan suatu solusi virtualisasi yang memiliki lebih dari satu pendekatan.

Perangkat lunak atau solusi yang melakukan virtualisasi bisa disebut sebagai *Virtual Machine Monitor* (VMM) maupun *hypervisor* [6]. Dalam *paper* ini keduanya digunakan secara bergantian.

Banyaknya pendekatan yang digunakan untuk melakukan virtualisasi akhirnya menimbulkan pertanyaan baru dari sebagian kalangan. Seberapa besar perbedaan kinerja yang dihasilkan oleh masing-masing pendekatan?.

Pertanyaan di atas muncul karena pendekatan-pendekatan tersebut menentukan bagaimana suatu solusi virtualisasi akan bekerja. Dengan menggunakan virtualisasi, beban kerja komputer kini tidak sebatas pada menjalankan sistem operasi, tapi juga menjalankan perangkat lunak virtualisasi itu sendiri. Pendekatan-pendekatan tersebut akan menjadi salah satu faktor penentu seberapa efisien suatu solusi virtualisasi dalam melakukan tugasnya, karena suatu solusi yang efisien dalam bekerja memungkinkan lebih banyak sistem operasi yang berjalan pada satu komputer. Hal ini memotivasi peneliti untuk menjawab pertanyaaan akan berapa besar perbedaan kinerja yang dihasilkan oleh masing-masing pendekatan melalui serangkaian pengujian empiris.

Pengujian yang dilakukan diharapkan dapat memberikan data atau masukan baru dalam melihat berbagai macam kinerja yang dihasilkan oleh beberapa pendekatan virtualisasi. Pada akhirnya hasil penelitian diharapkan dapat menjadi salah satu bahan pertimbangan dalam pemilihan suatu solusi virtualisasi.

Penelitian lain yang juga mencoba mencari perbedaan kinerja yang dihasilkan oleh lebih dari satu pendekatan atau teknik virtualisasi adalah penelitian yang dilakukan oleh Adams & Agesen [7]. Perbedaannya, penelitian ini melibatkan empat pendekatan, sedangkan penelitian Adams & Agesen melibatkan dua pendekatan (*full virtualization* dan *hardware-assisted virtualization*). Dengan melibatkan empat pendekatan, hasil kajian diharapkan bisa menunjukkan perbedaan kinerja dengan pendekatan lainnya, yaitu *paravirtualization* dan *operating system-level virtualization*.

Penelitian serupa juga pernah dilakukan oleh Haris Fauzi [8]. Dalam tesisnya, Haris mencari tahu sejauh mana atau seberapa baik kinerja saat menggunakan virtualisasi jika dibandingkan dengan implementasi *native* dengan menggunakan satu pendekatan virtualisasi.

2. Metodologi

Metode yang digunakan dalam penelitian ini menggunakan tahapan-tahapan seperti diperkenalkan oleh Meier dkk. [9]. Terdapat sepuluh tahapan pada penelitian ini. Tahap pertama mengidentifikasi pertanyaan riset. Dalam hal ini menentukan apa yang menjadi pertanyaan utama dalam penelitian penelitian ini.Tahap kedua melakukan studi pustaka. Mencari dan memahami literatur mengenai virtualisasi, terutama mengenai pandekatan-pendekatan dan teknik-teknik dalam virtualisasi. Tahap ketiga mengidentifikasi lingkungan pengujian. Mencari informasi terkait hal-hal yang digunakan dalam pengujian pada penelitian ini. Ini meliputi perangkat keras, perangkat lunak, dan *tools* khusus lainnya.

Tahap keempat mengidentifikasi kriteria penerimaan kinerja. Menentukan kriteria kinerja sehingga suatu solusi virtualisasi dapat diterima dan layak digunakan. Tahap kelima membuat perencanaan dan perancangan pengujian. Membuat rencana dan rancangan dari skenario pengujian. Ini juga meliputi data yang digunakan untuk pengujian dan metrik yang digunakan. Tahap keenam melakukan konfigurasi lingkungan pengujian. Menyiapkan lingkungan yang digunakan untuk pengujian. Beberapa diantaranya adalah topologi jaringan, konfigurasi sistem operasi, dan konfigurasi *tools* untuk pengujian. Tahap ketujuh melakukan implementasi rancangan pengujian. Melakukan pembuatan *scripts* untuk melakukan pengujian sesuai dengan apa yang sudah direncanakan dan dirancang. Tahap kedelapan melakukan pengujian pada penelitian. Tahapan ini berisi aktivitas yang dilakukan pada pengujian. Tahap kesembilan melakukan analisis, laporan, dan uji ulang. Mengumpulkan data-data hasil pengujian dan melakukan analisis. Uji ulang dilakukan jika ada hasil yang dirasa kurang sesuai dengan yang diharapkan. Tahap kesepuluh menarik kesimpulan dari penelitian. Menarik kesimpulan dari hasil analisis yang dilakukan pada tahap kesembilan.

Pengujian yang dilakukan memiliki batasan-batasan. Pertama pengujian dilakukan terhadap solusi virtualisasi yang berjalan pada arsitektur x86/x86-641. Kemudian pengujian dilakukan pada solusi-solusi yang bisa didapat atau dicoba dengan gratis. Ketiga pengujian dilakukan pada solusi yang dianggap layak untuk digunakan dalam komputer *server* operasional. Terakhir pengujian tidak menyertakan virtualisasi dengan pendekatan *hardware emulation*, karena akan sangat lambat sehingga tidak cocok dengan batasan atau butir sebelumnya [5].

TABEL I.
SPESIFIKASI KOMPUTER *SERVER*

Merk	HP ProLiant ML350 G5
CPU	1 x Intel Xeon E5130 (Dual-Core, 2.0 GHz)
	1 x Intel Xeon E5140 (Dual-Core, 2.33 GHz)
RAM	4 GB
Disk	2 x 146 GB (SAS, 15K RPM, RAID-1)
NIC	2 x Gigabit Ethernet

Perangkat keras yang digunakan dalam pengujian pada penelitian ini terbagi menjadi tiga yaitu, dua *server* dan satu *client*. Tabel I menyajikan spesifikasi dari komputer *server* yang dipakai dalam eksperimen. Kedua komputer *server* dalam tabel II merupakan dua *core* yang berarti memiliki dua CPU dalam satu *packaging*. Pada pengujian yang dilakukan di dalam penelitian ini, sistem operasi dan perangkat lunak virtualisasi dijalankan tanpa fitur *multiprocessing* (sering disebut sebagai *symetric multiprocessing* atau SMP) karena dua alasan. Pertama memudahkan dalam membandingkan masing-masing solusi virtualisasi dengan pada saat *native* tanpa memperhitungkan seberapa baik masing-masing perangkat lunak virtualisasi atau sistem operasi tersebut memanfaatkan fitur SMP. Kedua Solusi virtualisasi seperti *VirtualServer* tidak dapat memanfaatkan SMP pada mesin *virtual*.

Sama halnya dengan RAM, meskipun tersedia 4 GB masing-masing komputer dibatasi menjadi 1 GB. Ini artinya pada saat pengujian dengan komputer *native* sistem operasi diberikan 1 GB, sedangkan pada saat menggunakan virtualisasi masing-masing mesin *virtual* juga diberikan jumlah RAM yang sama yakni 1 GB.

TABEL II
PEMBATASAN SPESIFIKASI KOMPUTER

	Tanpa virtualisasi (*native*, 1 komputer)	Tanpa virtualisasi (*native*, 2 komputer)	Denganvirtualisasi	
			Sistem operasi *host*	Sistem operasi guest
CPU	1	@1	1	1
RAM	2 GB	@1 GB	~	1 GB
Disk	~	~	~	1 GB

Di mana ~ merupakan sebesar sisa yang tersedia.

Selain komputer *server* pengujian ini juga melibatkan komputer yang berperan sebagai *client*. Komputer ini digunakan untuk melakukan *request* kepada aplikasi *Web* pada *Webserver*. Spesifikasi komputer yang digunakan terlihat di tabel III.

TABEL III
SPESIFIKASI KOMPUTER *CLIENT*

Merk	IBM/Lenovo ThinkPad X60
CPU	1 x Intel Core Duo T2300 (Dual-Core, 1.66 GHz)
RAM	1 GB
Disk	60 GB
NIC	1 x Gigabit Ethernet

Perangkat lunak yang digunakan dalam pengujian ini terbagi empat, yaitu sistem operasi, aplikasi atau program yang diujikan, aplikasi atau program penguji, dan perangkat lunak virtualisasi itu sendiri.

Solusi Virtualisasi: Terdapat delapan daftar perangkat lunak virtualisasi yang digunakan dalam pengujian. Perangkat lunak tersebut adalah Microsoft Hyper-V, Linux-V*server*, OpenVZ, Sun xVM *Virtual*Box, Microsoft *VirtualServer*, VMware ESX/ESXi, VMware *Server*, dan Xen.

Microsoft Hyper-V: merupakan solusi virtualisasi dari Microsoft yang tersedia bersama dengan sistem operasi Windows *Server* 2008. Microsoft juga memberikan secara cuma-cuma Microsoft Hyper-V *Server* 2008 yang sebagian orang menyebutnya sebagai varian dari Windows *Server* 2008 "*Core*" dengan Hyper-V sebagai satu-satunya *role* yang tersedia. Hyper-V merupakan perangkat lunak virtualisasi bertipe *bare-metal* yang memerlukan CPU x86-64 dan teknologi Intel VT-x atau AMD-V (hanya mendukung pendekatan *hardware-assisted virtualization*). Hyper-V mendukung sistem operasi *desktop*/*server* sejak Windows 2000 hingga Windows *Server* 2008 dan beberapa distribusi GNU/Linux.

Linux-V*server*: merupakan salah satu solusi virtualiasi yang memberikan kemampuan *OS-level virtualization* pada *kernel* Linux. Linux-V*Server* didistribusikan sebagai perangkat lunak bebas dan dikembangkan oleh komunitas. Linux-V*Server* dapat berjalan pada kebanyakan arsitektur CPU yang didukung oleh *kernel* Linux, terutama x86 dan x86-64.

OpenVZ: merupakan salah satu solusi *virtual*iasi yang memberikan kemampuan *OS-level virtualization* pada *kernel* Linux. OpenVZ didistribusikan sebagai perangkat lunak bebas dan dikembangkan oleh komunitas. Pengembangan OpenVZ juga didukung oleh perusahaan komersil yaitu Parallels dan menjadi basis dari salah satu produk mereka, Parallels Virtuozzo.

Sun xVM *Virtual*Box: merupakan perangkat lunak virtualisasi dari Sun Microsystem dengan tipe *hosted*. *Virtual*Box merupakan perangkat lunak bebas, tetapi versi *proprietary* (gratis) dari Sun memberikan beberapa fitur tambahan seperti *Remote Desktop Protocol*, USB, iSCSI, dan lain-lain. *Virtual*Box dapat berjalan pada CPU dengan arsitektur x86 atau x86-64 dan sistem operasi Windows, GNU/Linux, Mac OS X, atau Solaris sebagai *host*. *Virtual*Box dapat menggunakan pendekatan *full virtualization* maupun *hardware-assisted virtualization*, sedangkan *paravirtualization* direncanakan di masa mendatang.

Microsoft *VirtualServer*: Sebelum Hyper-V, *VirtualServer* adalah solusi virtualisasi untuk *server* satu-satunya dari Microsoft. *VirtualServer* merupakan perangkat lunak virtualisasi bertipe *hosted* dan bisa melakukan virtualisasi dengan pendekatan *full virtualization* atau *hardware-assisted virtualization.* Saat ini *VirtualServer* sudah bisa didapatkan secara gratis. *VirtualServer* dapat berjalan pada CPU dengan arsitektur x86 atau x86-64, tetapi hanya bisa menjalankan sistem operasi atau mesin *virtual* x86. *VirtualServer* mendukung sistem operasi sejak Windows XP hingga Windows *Server* 2008 sebagai *host*, sedangkan untuk *guestVirtualServer* mendukung sistem operasi *server* sejak Windows NT *Server* 4.0 hingga Windows *Server* 2003. Meskipun begitu Microsoft tidak menyarankan Windows XP atau Windows Vista sebagai system operasi *host* pada *server* produksi.

VMware ESX/ESXi: merupakan salah satu perangkat lunak virtualisasi dari VMware yang bertipe *bare-metal*. Apa yang membedakan ESX dengan ESXi adalah arsitektur dan menajemen operasi. Walaupun inti dari kedua perangkat lunak virtualisasi sama dan tidak bergantung pada sistem operasi tertentu untuk manajemen, tetapi ESX memerlukan sistem operasi GNU/Linux untuk melakukan manajemen. ESXi bisa dikatakan sebagai ESX versi minimal tanpa fitur-fitur tambahan tertentu yang bisa didapat dengan gratis. Dalam penelitian ini yang digunakan adalah ESXi. ESX/ESXi berjalan pada arsitektur CPU x86 atau x86-64. ESX/ESXi dapat melakukan pendekatan virtualisasi *full virtualization*, *paravirtualization*, dan *hardware-assisted virtualization* (hanya pada sistem operasi 64-bit).

VMware *Server*: sebelumnya bernama VMware GSX *Server* dan merupakan produk utama dari VMware. VMware memberikan VMware *Server* secara cuma-cuma dengan harapan menjadi titik mula pengguna menuju VMware ESX. VMware *Server* bertipe *hosted* dan mendukung CPU dengan arsitektur x86 atau x86-64 dengan sistem sistem operasi GNU/Linux dan Windows sebagai *host*. VMware *Server* dapat melakukan pendekatan virtualisasi *full virtualization, paravirtualization*, dan *hardware-assisted virtualization*.

Xen: merupakan perangkat lunak virtualisasi yang awalnya dikembangkan di Universitas Cambridge dan saat ini dikembangkan oleh komunitas sebagai perangkat lunak bebas. Selain itu dalam pengembangannya Xen juga didukung oleh banyak perusahaan TI terkemuka di dunia seperti Citrix, IBM, Intel, Hewlett-Packard, Novell, Red Hat, Sun Microsystems, dan Oracle. Xen dapat berjalan pada arsitektur CPU x86/x86-64 dan menjalankan sistem operasi di dalam mesin *virtual* dengan arsitektur yang sama. Pendekatan utama Xen adalah *paravirtualization*, tetapi sejak versi 3.0 Xen juga mendukung *hardware-assisted virtualization*. Xen merupakan perangkat lunak virtualisasi dengan tipe *bare-metal*.

TABEL IV
SISTEM OPERASI YANG DIGUNAKAN

Nama	OS *host*	OS *guest*
Native	Debian GNU/Linux 5.0 (x86)	None (no *virtualization*)
Hyper-V	None (*bare-metal*, Windows *Server* 2008 as root partititon, x86-64)	Debian GNU/Linux 5.0 (x86)
Linux-V*Server*	Debian GNU/Linux 5.0 (x86-64)	Debian GNU/Linux 5.0 (x86)
OpenVZ	Debian GNU/Linux 5.0 (x86-64)	Debian GNU/Linux 5.0 (x86)
Sun xVM *Virtual*Box	Debian GNU/Linux 5.0 (x86-64)	Debian GNU/Linux 5.0 (x86)
VirtualServer R2	Windows *Server* 2008 (x86-64)	Debian GNU/Linux 5.0 (x86)
VMWareESXi	None	Debian GNU/Linux 5.0 (x86)
VMWare *Server*	Debian GNU/Linux 5.0 (x86-64)	Debian GNU/Linux 5.0 (x86)
Xen	None (*bare-metal*, Debian 5.0 as *domain 0*, x86-64)	Debian GNU/Linux 5.0 (x86)

Sistem Operasi: Ada dua sistem operasi yang terlibat dalam pengujian ini, pertama GNU/Linux dan Microsoft Windows *Server* 2008. Distribusi yang digunakan untuk GNU/Linux itu sendiri adalah Debian 5.0 untuk *server* dan Ubuntu 8.10 untuk *client*. Untuk solusi virtualisasi yang berasal dari Microsoft digunakan Windows *Server* 2008 (x86-64) sebagai sistem operasi *host*. Untuk solusi virtualisasi lainnya, meskipun tersedia untuk sistem GNU/Linux dan Windows, namun yang digunakan sebagai sistem operasi *host* sistem adalah GNU/Linux (x86-64). Untuk sistem operasi *guest* semuanya disamakan menggunakan sistem GNU/Linux (x86). Untuk lebih detail dapat dilihat pada tabel IV.

Beberapa perangkat lunak virtualisasi tertentu yang bertipe *bare-metal* memerlukan suatu mesin *virtual* khusus dengan sistem operasi tertentu yang digunakan untuk mengendalikan perangkat lunak virtualisasi yang bersangkutan.

Dalam pengujian ini, untuk Xen, mesin *virtual* khusus (dalam istilah Xen disebut sebagai domain 0) itu sendiri terpasang Debian 5.0 (x85-64). Untuk Microsoft Hyper-V sistem operasi yang terpasang pada mesin *virtual* khusus (Root Partition dalam istilah Hyper-V) tersebut sudah pasti untuk saat ini adalah Windows *Server* 2008. Untuk VMware ESX/ESXi sendiri, karena bisa dikendalikan melalui jaringan menggunakan aplikasi *thick client* maka tidak diperlukan mesin *virtual* tambahan.

Aplikasi atau Program: Pada skenario pengujian terdapat tiga skenario yang membutuhkan program atau aplikasi tertentu agar skenario tersebut bisa dijalankan. Meskipun begitu, untuk memenuhi kebutuhan masing-masing skenario tersebut tidak hanya sebatas pada tiga program. Masing-masing program yang digunakan membutuhkan program lainnya agar bisa digunakan sebagai alat penguji, tetapi yang menjadi tugas utama dari masing-masing skenario pengujian adalah WordPress, Bzip2, dan Gcc.

WordPress: merupakan suatu aplikasi *Web* untuk penerbitan (*publishing*) yang populer digunakan sebagai aplikasi untuk *blogging*. WordPress adalah aplikasi bebas dan *open source*. WordPress sebagian besar kodenya dibuat menggunakan bahasa pemrograman PHP dan MySQL sebagai basis datanya. Ini berarti WordPress membutuhkan sebuah *Webserver* dengan *interpreter* PHP dan *databaseserver* MySQL terpasang agar bisa digunakan.

Bzip2: merupakan algoritma atau program untuk melakukan kompresi data dengan menggunakan algoritma Burrows-Wheeler dan Huffman. Bzip2 merupakan algoritma dan program yang berlisensi bebas.

Gcc: merupakan singkatan dari GNU Compiler Collection yang mana merupakan sistem *compiler* yang dibuat oleh GNU Project dan dapat mendukung berbagai macam bahasa pemrograman. Pada kebanyakan sistem operasi UNIX-like, khususnya GNU/Linux, gcc sering digunakan sebagai *compiler* standar.

Alat Pengujian: Alat pengujian terbagi menjadi dua, yakni alat untuk melakukan simulasi beban dan alat untuk mendapatkan data dari metrik. Alat untuk melakukan simulasi beban hanya diperlukan untuk skenario Aplikasi *Web*, sedangkan alat untuk mendapatkan data metrik diperlukan pada semua skenario.

Httperf: merupakan program untuk mengukur kinerja dari *Web server* yang dibuat oleh David Mosberger dari HP Labs. Httperf menyediakan fitur yang fleksibel dalam pembuatan beban kerja pada *Web* atau HTTP *server*. Httperf dapat mensimulasikan berbagai macam beban kerja sesuai dengan parameter yang diberikan kepadanya. Program ini diperlukan untuk menyimulasikan beban kerja pada skenario.

Aplikasi *Web*: Versi yang digunakan adalah sama seperti yang tersedia pada repositori Ubuntu 8.10.

Sar: merupakan bagian dari paket program sysstat digunakan untuk mengambil informasi mengenai aktivitas sistem. Sar dapat digunakan untuk mendapatkan kecepatan transfer perangkat I/O seperti jaringan atau *disk*, tingkat penggunaan RAM atau CPU, dan sebagainya. Untuk solusi virtualisasi yang bukan berbentuk *bare-metal* dan berjalan pada sistem operasi GNU/Linux program ini dapat digunakan untuk melihat tingkat penggunaan sumber daya CPU. Selain itu program ini juga digunakan pada saat *native*. Versi yang digunakan adalah sama seperti yang tersedia pada repositori Debian 5.0. Pengambilan metrik tingkat penggunaan sumber daya CPU menggunakan program ini dilakukan dengan cara mengurangi 100% (mewakili seluruh sumber daya CPU yang tersedia) dikurangi oleh hasil rata-rata pada kolom *% idle* (persentase CPU dalam keadaan *idle*).

Perfmon.exe: merupakan program untuk sistem operasi Windows yang berfungsi kurang-lebih sama seperti sar. Untuk solusi virtualisasi berjenis *hosted* pada sistem operasi Windows, program ini bisa digunakan secara langsung, karena *by default* tingkat penggunaan sumber daya CPU ditampilkan oleh program ini.

Xentop & esxtop: merupakan program yang serupa dengan program top yang tersedia pada sistem operasi UNIX, yaitu untuk menampilkan informasi sistem secara ringkas. Xentop dan esxtop digunakan untuk melihat tingkat penggunaan sumber daya CPU pada Xen dan

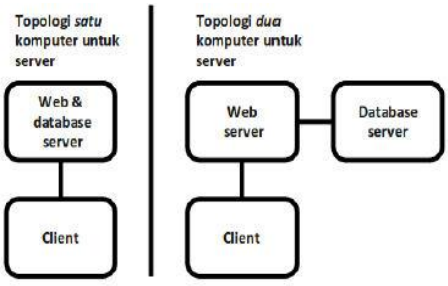
VMware ESX/ESXi karena keduanya merupakan solusi virtualisasi bertipe *bare-metal*.

Time: program yang digunakan untuk menjalankan program lainnya. Setelah selesai program ini akan menampilkan informasi waktu mengenai program yang berjalan tersebut. Apabila suatu proses dijalankan dengan menggunakan program time, maka setelah proses tersebut selesai program time akan melaporkan berapa lama proses tersebut berjalan (*real elapsed time*). Selain *itu by default* program *time* juga akan menampilkan berapa lama CPU bekerja pada *system* atau *user mode* selama proses yang bersangkutan berjalan.

Jaringan: Dalam pengujian ada dua bagian utama, yakni pengujian tanpa virtualisasi (*native*) dan dengan virtualisasi. Pada pengujian tanpa virtualisasi ada dua macam topologi jaringan. Topologi jaringan yang dibahas di sini hanya digunakan pada pengujian yang melibatkan aplikasi *Web*, karena skenario pengujian lainnya tidak melibatkan jaringan. Lebih detail dapat dilihat pada gambar 1.

Pengujian yang melibatkan aplikasi *Web* membutuhkan dua buah *server* (program *server*, bukan komputer *server*). Masing-masing *server* bisa dipasang pada satu komputer atau pada dua komputer yang berbeda. Untuk konfigurasi satu komputer maka topologi jaringan hanya berupa *point-to-point* antara dua komputer, komputer *server* dan komputer *client*.

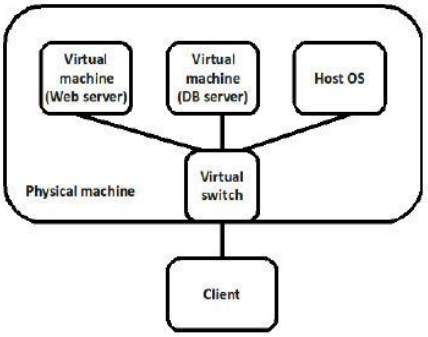
Pada konfigurasi dua komputer, *Web server,* dan *database server* masing-masing terpasang pada dua komputer dan saling terhubung secara *point-to-point*. Komputer *client* juga terhubung langsung secara *point-to-point* dengan komputer *Web server*. Ini artinya komputer *Web server* minimal harus memiliki dua buah NIC (*network interface controller*), untuk *database server* dan untuk *client*.



Gambar 1. Topologi jaringan tanpa virtualisasi.

Untuk pengujian pada saat menggunakan virtualisasi, pada dasarnya topologi yang digunakan berupa satu atau lebih mesin *virtual* yang terhubung pada satu *bridge/switch.* Walaupun mesin *virtual* terhubung dengan salah satu *port* dari *switch virtual*, tetapi karena *port* fisik komputer digunakan sebagai salahsatu *port* dari *switch virtual* tersebut, maka ini digunakan sebagai jalan menuju "dunia luar". *Port* fisik inilah yang terhubung dengan komputer *client.*

Pada skenario kompresi dan kompilasi hanya satu mesin *virtual* yang terlibat, sehingga hanya satu mesin *virtual* yang dibuat dan berjalan. Jika dibandingkan dengan gambar 2, pada skenario kedua dan ketiga hanya ada satu gambar mesin *virtual*.



Gambar 2. Topologi jaringan dengan virtualisasi.

Kriteria penerimaan kinerja: Meier dkk. mengatakan bahwa umumnya ada tiga hal yang menjadi sumber perhatian dalam suatu uji kinerja, yakni *Response time*, *Throughput*, dan *Resource utilization* [9].

Response time mengacu pada seberapa cepat suatu sistem/aplikasi dapat merespon suatu aksi. Misalnya, halaman *web* harus bisa tampil kurang dari lima detik. Sementara *Throughput* menunjukkan banyaknya tugas yang bisa dilakukan dalam satu satuan waktu. Misalnya, *web server* harus bisa melayani minimal 10 pengunjung dalam satu detik. Kemudian *Resource utilization* menunjukkan tingkat penggunaan sumber daya perangkat keras tertentu. Misalnya, pada saat jam sibuk *web server* tidak boleh memakan CPU *time* lebih dari 20%. Perencanaan dan perancangan uji kinerja menentukan skenario yang diinginkan, menentukan variasi yang digunakan, menentukan data untuk pengujian, dan menentukan metrik yang diambil.

Skenario: Pengujian yang dilakukan di sini terbagi menjadi tiga skenario, dan melayani halaman *web* dinamis, kompresi, dan kompilasi. Ketiga skenario tersebut dijalankan pada saat *native* dan pada masing-masing solusi virtualisasi. Ini termasuk setiap pendekatan yang disediakan oleh solusi tersebut, karena solusi tertentu menyediakan lebih dari satu pendekatan virtualisasi.

Aplikasi Web: Skenario pertama ini dipilih karena salah satu peran yang sering ditemukan pada komputer *server* adalah melayani aplikasi

web. Aplikasi yang digunakan dalam skenario ini, seperti yang telah disebutkan sebelumnya, adalah Wordpress.

Wordpress cukup menggambarkan struktur atau bentuk dari banyak aplikasi *web* seperti situs berita dan forum diskusi, di mana akan ada daftar dari judul *entry*, konten dari *entry,* dan komentar terhadap *entry* tersebut. Versi Wordpress yang digunakan dalam pengujian ini adalah versi 2.7.1, sedangkan versi *Webserver* (Apache), *database server* (MySQL), dan PHP *interpreter* sama dengan yang tersedia pada repositori Debian 5.0 (versinya akan selalu sama).

Pada saat *native* skenario ini melibatkan maksimal dua komputer, untuk *web server* dan *database server*, yang artinya pada saat menggunakan virtualisasi juga melibatkan dua mesin *virtual*. Pengujian skenario ini saat *native* menggunakan satu komputer sekaligus untuk *web server* dan *database server* juga dilakukan di sini.Wordpress diisi dengan dua belas *entry*, dan masing-masing *entry* diisi dengan beberapa komentar. Seluruh daftar *entry* pada halaman utama dibagi menjadi dua halaman, artinya enam *entry* setiap halaman.

Simulasi kasus yang digunakan dalam skenario ini berupa seorang pengujung yang membuka *entry* terbaru dan seorang lagi melihat daftar *entry* pada halaman dua. Keduanya melakukan *request* secara bersamaan dalam satu detik. Selama simulasi dilakukan, akan dilihat tingkat penggunaan sumber daya CPU yang digunakan untuk melayani dua *request* per detik tersebut. Tingkat penggunaan sumber daya CPU yang merupakan data metrik dari skenario ini diambil minimal lima kali lalu diambil rata-ratanya. Dikarenakan beberapa perangkat lunak virtualisasi berbentuk *bare-metal*, maka cara untuk mendapatkan data metrik dalam skenario ini memerlukan cara berbeda-beda.

Tidak ada alasan khusus dalam mensimulasikan dua pengunjung dalam satu detik selain menyisakan ruang sumber daya tidak terpakai yang cukup pada saat menggunakan virtualisasi nanti. Ini dikarenakan dalam pengujian awal saat *native* konfigurasi ini pada *web server* menghabiskan sekitar sepertiga dari keseluruhan sumber daya CPU. Mengingat saat menggunakan virtualisasi akan ada *overhead* dari proses virtualisasi itu sendiri, maka konfigurasi ini memberikan cukup ruang dan juga memberikan hasil data yang cukup *visible*.

Kompresi: Pada skenario kedua ini diujikan sebuah program kompresi bzip2 dalam melakukan kompresi berkas-berkas. Apa yang menjadi data dan dikompresi dalam skenario ini adalah kode sumber *kernel* sistem operasi Linux yang besarnya mencapai 317 MB (hasil perhitungan program du -h). Versi *kernel* yang digunakan adalah sama dengan yang ada pada repositori Debian 5.0, versi 2.6.26 (versinya akan selalu sama).

Kode sumber *kernel* Linux diunduh dari repositori lalu setelah terpasang berkas arsip berada pada /usr/src/linux-source-2.6.26.tar.bz2. Berkas arsip ini diekstrak dan kemudian seluruh berkas di dalamnya kembali dikompres menggunakan program bzip2 setelah sebelumnya diarsipkan kedalam satu berkas menggunakan program tar. Pengarsipan ini dilakukan karena program bzip2 hanya bisa melakukan kompresi pada berkas tunggal. Lama dari proses kompresi adalah yang menjadi metrik dalam skenario ini, dan akan diambil minimal sebanyak tiga kali dan dihitung rata-ratanya.

Kompilasi: Dalam skenario ketiga ini kode sumber dari *kernel* Linux yang diekstrak pada skenario kompresi akan dikompilasi. Ini berarti kode sumber *kernel* Linux menjadi data dalam skenario ini. Lama dari proses kompilasi adalah yang menjadi metrik dalam skenario ini, dan akan diambil minimal sebanyak tiga kali dan dihitung rata-ratanya.

Kompilasi *kernel* Linux dilakukan menggunakan parameter "allnoconfig". Parameter ini digunakan untuk menghindari terkompilasinya modul-modul dari *kernel* yang sifatnya *optional*.

3. Hasil dan Pembahasan

Pengujian terbagi menjadi lima bagian, yakni pengujian pada saat *native* dan saat menggunakan virtualisasi dengan empat pendekatan yang berbeda. Ini berarti dari delapan solusi virtualisasi yang diujikan semua hasilnya dikelompokkan berdasarkan pendekatan yang digunakan.

Native: Pada pengujian skenario pertama dengan satu komputer, program httperf dijalankan pada komputer *client* agar melakukan *requests* kepada komputer *server* selama 10 detik. Hasil keluaran httperf yang perlu diperhatikan adalah besarnya data yang ditransfer oleh *web server* kepada httperf. Pada pengujian ini, karena skenarionya adalah dua pengunjung yang melihat dua halaman berbeda dalam satu detik maka ada dua hasil data, yaitu 12759 *bytes* dan 26238 *bytes.* Meskipun besar data yang ditransfer bukan metrik yang digunakan dalam pengujian, tetapi informasi ini bisa menunjukkan apakah pengujian-pengujian selanjutnya telah berjalan dengan benar. Apa yang menjadi metrik dalam skenario pertama adalah persentase penggunaan sumber daya CPU, sehingga program sar perlu dijalankan pada komputer *server* selama *requests* sedang berlangsung. Pada pengujian ini sendiri program

sar hanya dijalankan selama 5 kali dengan interval setiap 1 detik (sar -u 1 5). Setelah selesai program sar menampilkan rata-rata penggunaan sumber daya CPU pada komputer *server* di mana ia dijalankan. Dikarenakan perbedaan antara pengujian *native* satu komputer dengan dua komputer merupakan letak dari *database server* maka untuk konfigurasi dua komputer program sar juga dijalankan pada komputer untuk *database*. Untuk skenario kompresi atau kompilasi, salah satu skenario dijalankan sebanyak minimal tiga kali lalu diambil rata-ratanya. Setelah tugas utama selesai dijalankan program time menampilkan berapa waktu yang dibutuhkan untuk menjalankan tugas tersebut. Kedua skenario ini juga dijalankan hanya pada satu komputer karena tidak melibatkan jaringan.

Seperti yang terlihat dalam tabel V, tingkat persentase penggunaan sumber daya CPU pada konfigurasi dua komputer, komputer *database server* ternyata hanya terpakai 1%. Nilai itu sendiri adalah hasil rata-rata dari nilai yang berfluktuasi, antara 0% hingga 2%. Bisa terlihat bahwa persentase penggunaan sumber daya CPU pada pengujian dengan konfigurasi satu komputer merupakan gabungan dari konfigurasi dua komputer.

TABEL V
HASIL DATA PENGUJIAN *NATIVE*.

	Aplikasi *Web* (%CPU)	Kompresi (detik)	Kompilasi (detik)
Native (1PC)	35	57	83
Native (2PC, *Webserver*)	34	-	-
Native (2 PC, *database server*	1	-	-

Operating System-Level Virtualization: Pada pengujian *operating system-level virtualization* ada dua solusi yang terlibat, yakni Linux-V*server* dan OpenVZ. Meskipun pada pendekatan ini *guest* menggunakan *kernel* yang sama dengan *host* (x86-64), tetapi apa saja yang terdapat di dalam *guest* (program, *library,* dan lain-lain) tetap menggunakan versi*32-bit* (x86) sesuai dengan yang ditampilkan dalam tabel VI.

TABEL VI
HASIL DATA PENGUJIAN PENDEKATAN *OS-LEVEL VIRTUALIZATION*

	Aplikasi *Web* (%CPU)	Kompresi (detik)	Kompilasi (detik)
Linux V*Server*	36	58	84
OpenVZ	37	58	86

Dari tabel VI dapat terlihat bahwa dalam ketiga skenario keduanya menunjukkan kinerja yang tidak jauh berbeda, hanya terpaut 1-2 poin, bahkan dalam skenario kompresi hasilnya sama.

Paravirtualization: Pada tahun 2005 VMware mengeluarkan suatu spesifikasi antar muka untuk virtualisasi, *Virtual Machine Interface* (VMI), sebagai mekanisme komunikasi antara sistem operasi *guest* dengan perangkat lunak virtualisasi. Antarmuka ini kini memungkinkan produk dari VMware untuk melakukan virtualisasi dengan pendekatan *paravirtualization,* seperti yang dilakukan Xen.

Hasilnya, pada tabel VII menunjukkan bahwa Xen, sebagai salah satu perintis pendekatan *paravitualization* pada arsitektur x86, memiliki kinerja sedikit lebih baik dibanding solusi lainnya. Meskipun begitu, pada skenario kompresi, Xen memiliki kinerja yang sama dengan VMwareESXi, yang juga sedikit lebih baik dibanding VMware*Server*.

TABEL VII
HASIL DATA PENGUJIAN PENDEKATAN *PARAVIRTUALIZATION*

	Aplikasi *Web* (%CPU)	Kompresi (detik)	Kompilasi (detik)
VMware ESXi	42	58	107
VMware *Server*	42	61	108
Xen	38	58	92

Hardware-Assisted Virtualization: Pada pengujian ini, solusi virtualisasi yang menawarkan pendekatan *hardware-assisted virtualization* ada enam: Hyper-V, *Virtual*Box, *VirtualServer*, VMware ESXi, VMware *Server*, dan Xen. Akan tetapi VMwareESXi tidak bisa diikutsertakan. Ini dikarenakan VMware ESX/ESXi tidak mendukung sistem operasi 32-bit sebagai *guest* dengan pendekatan ini, hanya sistem operasi *guest* 64-bit yang didukung [10].

Hasil pengujian pada tabel VIII menunjukkan VMware *Server* memiliki kinerja sedikit lebih baik dibandingkan solusi lainnya. Hal yang menarik perhatian adalah kinerja *Virtual*Box pada skenario kompilasi yang memiliki selisih cukup jauh bila dibandingkan dengan VMware *Server*. Satu hal yang bisa menjadi pertimbangan adalah tidak seperti sistem GNU/Linux yang seluruh *daemon*-nya dimatikan, pada *default install* sistem operasi Windows *Server* 2008 ada beberapa *services* yang tetap berjalan karena diperlukan. *Services* ini terkadang menyebabkan tingkat penggunaan sumber daya CPU berfluktuasi antara 0% hingga 2%. Untuk pengujian ini *Virtual*Box dijalankan melalui perintah VBoxHeadless.

Full Virtualization: Pada pengujian ini, solusi virtualisasi yang menawarkan pendekatan *full virtualizaton* sebenarnya ada empat, yaitu *Virtual*Box, *VirtualServer*, Vmware ESXi, dan

VMware *Server*. Tetapi dalam pengujian ini ternyata pada saat menggunakan *VirtualServer*, *kernel* sistem operasi *guest* tidak dapat berjalan. Dalam sebuah diskusi di *mailing list,* H. Peter Anvin, salah satu pengembang *kernel* Linux mengatakan, hal ini adalah *bug* yang terdapat pada *VirtualServer* [11].

TABEL VIII
HASIL DATA PENGUJIAN PENDEKATAN *HARDWARE-ASSISTED VIRTUALIZATION*

	Aplikasi *Web* (%CPU)	Kompresi (detik)	Kompilasi (detik)
Hyper-V	44	65	133
*Virtual*Box	46	68	151
VirtualServer	44	68	118
VMware ESXi	-	-	-
VMware *Server*	42	62	113
Xen	43	63	106

Data hasil pengujian dengan pendekatan *fullvirtualization* pada tabel IX di bawah menunjukkan kinerja *Virtual*Box pada skenario Aplikasi *Web* dan kompresi memperlihatkan perbedaan yang cukup menarik perhatian jika dibandingkan dengan kedua solusi lainnya. Ini dikarenakan*Virtual*Box menunjukkan kinerja yang lebih rendah, walaupun tidak terlalu jauh, tetapi cukup menarik perhatian. Bahkan pada skenario kompilasi, *Virtual*Box menunjukkan perbedaan lebih jauh lagi yang mengundang pertanyaan dan akhirnya mendorong dilakukannya pengujian ulang. Untuk pengujian ini *Virtual*Box juga dijalankan melalui perintah VBoxHeadless.

TABEL IX
HASIL DATA PENGUJIAN PENDEKATAN *FULL VIRTUALIZATION*

	Aplikasi *Web* (%CPU)	Kompresi (detik)	Kompilasi (detik)
*Virtual*Box	48	78	165
VirtualServer	-	-	-
VMware ESXi	42	64	98
VMware *Server*	42	63	102

4. Kesimpulan

Meskipun pengujian tidak bisa mewakili seluruh kasus atau skenario yang ada di dunia nyata, tetapi hasil pengujian dan analisis yang dilakukan dalam penelitian ini telah menunjukkan perbedaan kinerja di antara masing-masing pendekatan virtualisasi. Untuk skenario Aplikasi *Web*, persentase perbedaan rata-rata utilisasi CPU dari masing-masing pendekatan dibandingkan dengan *native* adalah *Operating system-level virtualization* 4.29% lebih tinggi, *Paravirtualization* 16.19% lebih tinggi, *Hardware-assisted virtualization* 25.14% lebih tinggi, dan *Full virtualization* 25.71% lebih tinggi.

Untuk skenario kompresi dan kompilasi, berturut-turut persentase perbedaan rata-rata waktu yang dibutuhkan dari masing-masing pendekatan dibandingkan dengan *native* adalah *Operating system-level virtualization* 1.75% dan 2.41% lebih lama, *Paravirtualization* 3.51% dan 23.29% lebih lama, *Hardware-assisted virtualization* 13.33% dan 49,64% lebih lama, dan *Full virtualization*:19.88% dan 46.59% lebih lama.

Suatu organisasi di mana tugas utamanya memberikan layanan yang sama pada banyak pihak, seperti perusahaan *webhosting,* solusi-solusi virtualisasi dengan pendekatan *operating system-level virtualization* atau bahkan *paravirtualization* akan menjadi pilihan yang menarik. Pada organisasi yang memiliki sistem informasi atau aplikasi *propietary* kuno akan melihat solusi-solusi dengan pendekatan *full virtualization* lebih menarik. Organisasi serupa juga akan tertarik dengan solusi-solusi yang menggunakan pendekatan *hardware-assisted virtualization* apabila berencana menggunakan komputer *server* yang cukup baru dan CPU-nya mendukung fitur virtualisasi.

Berdasarkan hasil pengujian dan analisis yang dilakukan, semua hipotesis benar, kecuali pernyataan yang menyatakan apabila solusi dengan pendekatan *hardware-asssited virtualization* akan memiliki kinerja lebih baik dibandingkan dengan solusi dengan pendekatan *full virtualization.* Di antara keduanya menunjukkan hasil kinerja yang bercampur dan hasil yang serupa juga ditunjukkan oleh penelitian Adams and Agesen [7].

Selama ada lebih dari satu komputer *server* yang penggunaan sumber dayanya hanya terpakai antara 10-15% seperti yang disebutkan oleh Vmware [4] dan Sun Microsystem [3], seluruh solusi dengan pendekatan apapun yang diujikan di sini menunjukkan virtualisasi bisa membuka peluang untuk konsolidasi. Ini dengan catatan pada komputer-komputer *server* tersebut masih tersedia ruang untuk konsolidasi.

Referensi

[1] VMware, Understanding Full Virtualization, Paravirtualization, and Hardware Assist, http://www.vmware.com/files/pdf/VMware_para*virtualization*.pdf, 2007, retrieved October 3, 2008.

[2] R.M. Ramanathan & F. Bruening, Virtualization –Bringing Flexibility and New Capabilities to Computing Platforms, Research & Development at Intel

Corporation, http://download.intel.com/technology/computing/archinnov/teraera/download/*Virtualization*_0604.pdf, 2004, retrieved September 3, 2007.

[3] Sun Microsystem, Sun xVM Virtualization Portfolio: Virtualizing the Dynamic Datacenter, http://www.sun.com/launch/2008-0910/Sun_xVMPortfolio_wp.pdf,2008, retrieved February 16, 2009.

[4] VMware, Virtualization—The Most Impactful Solution to the Data Center Power Crisis, http://www.vmware.com/files/pdf/Energy_Efficiency_WP.pdf, 2008, retrieved January 13, 2009.

[5] M.T. Jones, Virtual Linux, An overview of virtualization methods, architectures, and implementations, IBM developer Works, http://www.ibm.com/developerworks/linux/library/llinuxvirt, 2006, retrieved April 28, 2007.

[6] B. Armstrong, VMMs versus Hypervisors, http://blogs.msdn.com/*virtual*_pc_guy/archive/2006/07/10/661958.aspx, 2006, retrieved June 23, 2007.

[7] K. Adams & O. Agesen, A Comparison of Software and Hardware Techniques for x86 Virtualization, http://www.vmware.com/pdf/asplos235_adams. pdf, 2006, retrieved May 11, 2007.

[8] H. Fauzi, "Perbandingan Kinerja Server Melalui Virtualization Xen Pada Lingkungan Terbatas", Ph.D Thesis, Faculty of Computer Science, Universitas Indonesia, 2008.

[9] J.D. Meier, et al., Performance Testing Guidance for *Web* Applications, http://perftestingguide. codeplex.com/, 2007, retrieved March 9, 2008.

[10] J. Mattson, VMware Products and Hardware-Assisted Virtualization (VT-x/AMD-V), http://communities.vmware.com/docs/DOC-9150, 2008, retrieved January 7, 2009.

[11] C. Ehrhardt, Boot failures on Qemu due to P6_NOPS, http://lkml.org/ikml/2008/7/7/2432008, retrieved February 4, 2009.