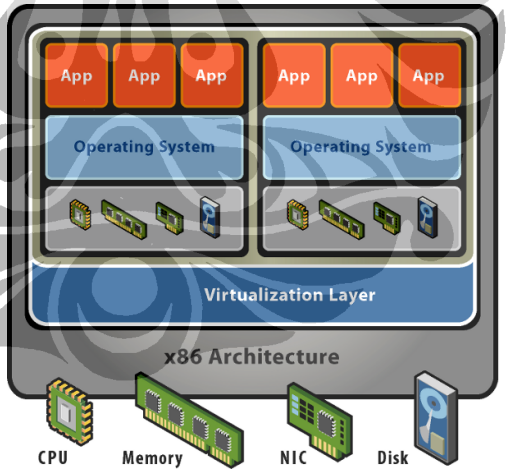
BAB 2 LANDASAN TEORI

2.1 Arsitektur Virtualisasi

Pada virtualisasi *platform* x86/x86-64, perangkat lunak/lapisan virtualisasi ditambahkan di antara perangkat keras dan sistem operasi *guest* seperti yang terlihat pada Gambar 2. Perangkat lunak virtualisasi ini memungkinkan beberapa sistem operasi untuk berjalan secara bersamaan di dalam beberapa komputer/mesin virtual pada satu komputer fisik, yang secara dinamis memisahkan dan membagi-bagi sumber daya fisik seperti CPU, RAM, *disk* dan perangkat keras lainnya (VMware, 2007).



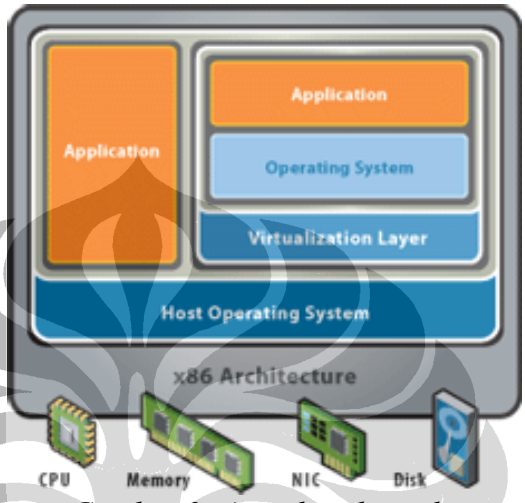
Gambar 2: Lapisan virtualisasi.

(VMware, 2007; telah diolah kembali)

Pada sistem komputer x86/x86-64, perangkat lunak virtualisasi dapat menggunakan arsitektur *hosted* maupun *bare-metal* (VMware, 2009). Pada arsitektur *hosted* perangkat lunak virtualisasi dipasang dan berjalan di atas suatu

Perbandingan kinerja ..., Rio Rasian A., Fasilkom UI, 2009

sistem operasi (sistem operasi *host*) seperti yang terlihat pada Gambar 3, sedangkan pada arsitektur *bare-metal* perangkat lunak virtualisasi berjalan langsung di atas perangkat keras, seperti yang ditunjukkan oleh Gambar 2.



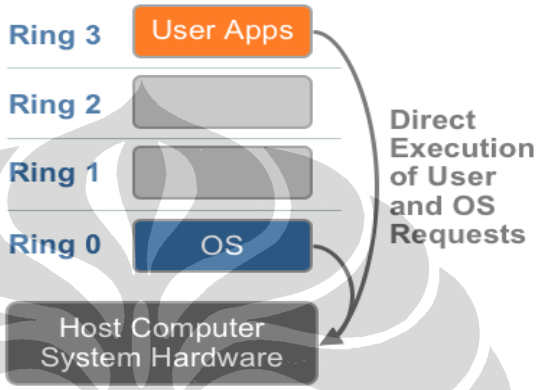
Gambar 3: Arsitektur hosted.

(VMware, 2009; telah diolah kembali)

2.2 Tantangan Virtualisasi

Sistem operasi pada arsitektur x86/x86-64 dirancang untuk berjalan langsung di atas perangkat keras. Seperti yang ditunjukkan Gambar 4, arsitektur x86/x86-64 menyediakan empat tingkat *privileges,* yang dikenal sebagai Ring 0, 1, 2 dan 3 oleh sistem operasi dan aplikasi, untuk mengendalikan akses kepada perangkat keras komputer. Aplikasi normalnya berjalan pada Ring 3, sedangkan sistem operasi karena perlu mengakses memori dan perangkat keras secara langsung maka ia harus mengeksekusi instruksi-instruksi *privileged* pada Ring 0. Melakukan virtualisasi pada arsitektur x86/x86-64 mengharuskan agar perangkat lunak virtualisasi ditempatkan di bawah sistem operasi (yang mengharapkan

untuk berada pada Ring 0) untuk membuat mesin virtual dan mengendalikannya. Lebih jauh lagi, beberapa instruksi sensitif tidak dapat secara efektif divirtualisasikan karena mereka memiliki arti berbeda ketika mereka tidak dieksekusi pada Ring 0 (VMware, 2007).



Gambar 4: Privilege level tanpa virtualisasi.

(VMware, 2007)

2.3 Pendekatan Virtualisasi

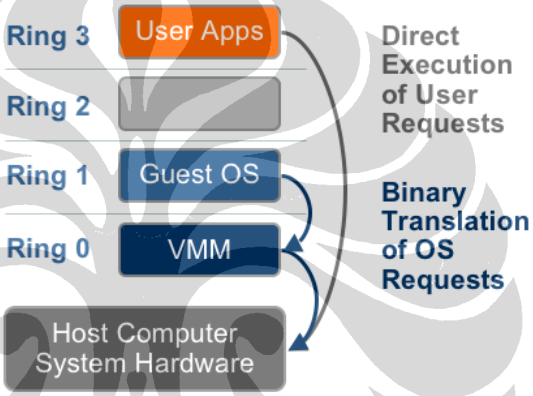
Seperti yang telah disebutkan pada BAB 1, kini ada beberapa pendekatan/teknik virtualisasi, dan sesuai dengan batasan-batasan yang disebutkan pada sub-bab 1.5 maka pendekatan-pendekatan yang dilibatkan dalam tesis ini adalah: *full virtualization, hardware-assisted virtualization, paravirtualization,* dan *operating system-level virtualizaton.*

2.3.1 Full Virtualization dengan Binary Translation

Pendekatan ini, seperti yang ditunjukkan Gambar 5, menterjemahkan kode dari *kernel* sistem operasi *guest* untuk mengganti intruksi-instruksi yang tidak

Perbandingan kinerja ..., Rio Rasian A., Fasilkom UI, 2009

dapat divirtualisasikan dengan instruksi-instruksi yang dimaksudkan untuk perangkat keras virtual. Sementara itu, kode/instruksi dari tingkat *user* dapat langsung dieksekusi pada CPU untuk proses virtualisasi yang cepat. Perangkat lunak virtualisasi menyediakan setiap mesin virtual dengan layanan-layanan yang diberikan sistem komputer fisik, seperti *virtual BIOS, virtual devices,* dan *virtualized memory management* (VMware, 2007).



Gambar 5: Pendekatan full virtualization dengan binary translation.

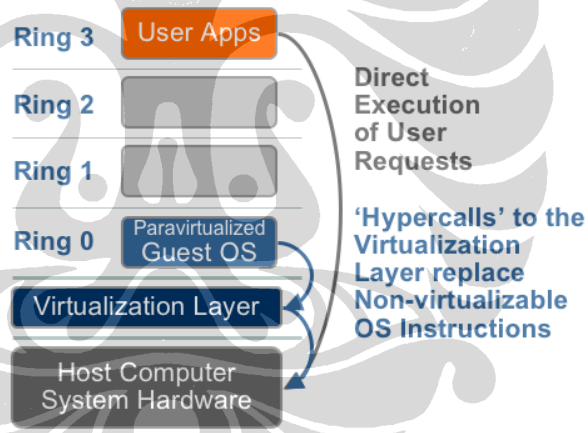
(VMware, 2007)

Kombinasi *binary translation* dan eksekusi langsung ini memberikan *full virtualization* karena sistem operasi *guest* diabstraksikan penuh dari perangkat keras oleh perangkat lunak virtualisasi. Sistem operasi *guest* tidak menyadari kalau ia sedang berjalan di dalam mesin virtual dan tidak perlu dimodifikasi. *Full virtualization* memberikan isolasi dan kemanan untuk mesin virtual, dan menyederhanakan migrasi dan *portability* karena sistem operasi *guest* yang sama dapat divirtualisasikan atau berjalan pada perangkat keras *native* (VMware, 2007).

Perbandingan kinerja ..., Rio Rasian A., Fasilkom UI, 2009

2.3.2 Paravirtualization

Paravirtualization mengacu pada komunikasi antara sistem operasi *guest* dan perangkat lunak virtualisasi untuk meningkatkan kinerja dan efisiensi. *Paravirtualization* seperti yang ditunjukkan Gambar 6, melibatkan modifikasi pada *kernel* sistem operasi *guest* untuk menggantikan instruksi-instruksi yang tidak bisa divirtualisasikan dengan *hypercall* yang berkomunikasi langsung dengan perangkat lunak virtualisasi. Perangkat lunak virtualisasi juga menyediakan antarmuka *hypercall* untuk operasi *kernel* lainnya seperti *memory management, interupt handling,* dan *time keeping* (VMware, 2007).



Gambar 6: Pendekatan paravirtualization.

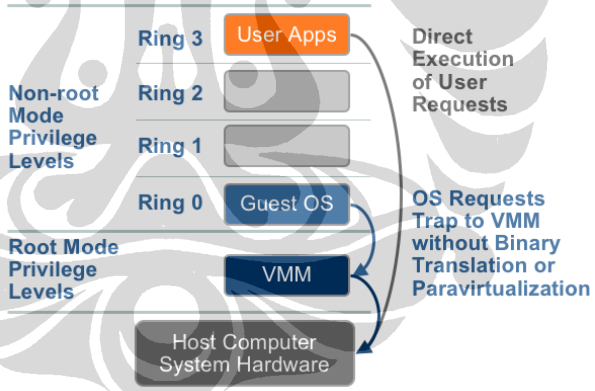
(VMware, 2007)

Karena *paravirtualization* tidak mendukung sistem operasi yang tidak dimodifikasi, maka kompatibilitasnya dan portabilitasnya menjadi buruk (VMware, 2007). Saat ini diketahui hanya empat sistem operasi yang bisa dijadikan sistem operasi *guest* dengan pendekatan *paravirtualization,* yaitu GNU/Linux, FreeBSD, NetBSD, dan OpenSolaris (Spector, 2008; OpenSolaris,

2009).

2.3.3 Hardware-assisted Virtualization

Vendor perangkat keras seperti produsen CPU dengan cepat mendukung virtualisasi dan mengembangkan fitur baru yang menyederhanakan teknik virtualisasi. Generasi pertama mencakup VT-x (Intel) dan AMD-V (AMD) yang keduanya mentargetkan instruksi-instruksi *privileged* dengan fitur mode eksekusi CPU baru yang memungkinkan perangkat lunak virtualisasi untuk berjalan pada mode baru, yaitu *root mode* yang berada di bawah Ring 0. Seperti yang ditunjukkan oleh Gambar 7, instruksi-instruksi *privileged* dan sensitif akan secara otomatis di-*trap* dan kendali diserahkan kepada perangkat lunak virtualisasi.



Gambar 7: Pendekatan hardware-assisted virtualization.

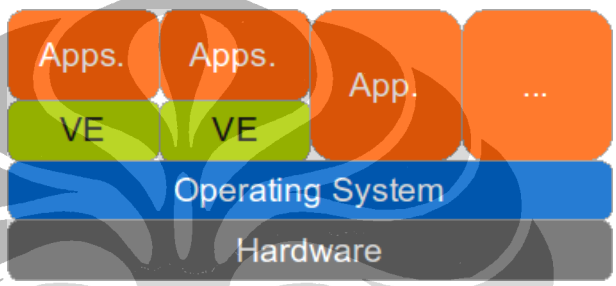
(VMware, 2007)

2.3.4 Operating system-Level Virtualization

Pendekatan *operating system-level virtualization* memiliki teknik yang berbeda dengan yang lainnya. *Operating system-level virtualization* memerlukan

Perbandingan kinerja ..., Rio Rasian A., Fasilkom UI, 2009

sebuah *kernel* sistem operasi untuk dimodifikasi agar memungkinkan dirinya untuk menyediakan beberapa lingkungan eksekusi terisolasi. Dari sudut pandang program yang berjalan di dalamnya lingkungan tersebut terlihat seperti komputer fisik yang berbeda (Kolyshkin, 2006). Ini artinya pendekatan ini tidak memungkinkan untuk menjalankan *kernel* lain dari sistem operasi lain pada saat bersamaan.



Gambar 8: Pendekatan operating system-level virtualization.

2.4 Uji Kinerja

Meier dkk. (2007) menyebutkan uji kinerja adalah suatu jenis pengujian yang dimaksudkan untuk menentukan *responsiveness*, *throughput,* kehandalan, dan/atau skalabilitas dari suatu sistem terhadap suatu beban kerja yang diberikan. Beberapa alasan dilakukannya uji kinerja adalah:

- Menilai kesiapan produksi.
- Mengevaluasi terhadap kriteria kinerja tertentu.
- Membandingkan karakteristik kinerja dari beberapa sistem.
- Mencari sumber masalah suatu kinerja.
- Membantu *tuning* pada sistem.

- Mencari tingkatan *throughput.*

Dalam melakukan uji kinerja ada beberapa aktivitas utama seperti yang telah disebutkan pada sub-bab 1.6:

- Identifikasi Lingkungan Pengujian.
- Identifikasi Kriteria Penerimaan Kinerja.
- Perencanaan dan Perancangan dari Pengujian.
- Konfigurasi Lingkungan Pengujian.
- Implementasi Rancangan Pengujian.
- Pengujian.
- Analisis, Laporan, dan Uji Ulang.

Uji kinerja biasanya termasuk di dalam salah satu dari tiga kategori berikut:

- ***Performance testing.*** Pengujian jenis ini menentukan atau melakukan validasi dari karakteristik kecepatan, skalabilitas, dan/atau stabilitas suatu sistem atau aplikasi yang diuji. Kinerja diukur dengan pencapaian dari waktu respon, *throughput,* dan tingkat penggunaan sumber daya yang memenuhi target. Ini berarti *performance testing* juga menjadi *superset* dari (sub)kategori lainnya.
- ***Load testing.*** Subkategori dari uji kinerja ini berfokus pada menentukan atau melakukan validasi dari karakteristik kinerja suatu sistem atau aplikasi yang diuji ketika diujikan dengan beban kerja yang diantisipasi pada saat produksi.
- ***Stress testing.*** Subkategori dari uji kinerja ini berfokus pada menentukan

Perbandingan kinerja ..., Rio Rasian A., Fasilkom UI, 2009

atau melakukan validasi karakteristik kinerja suatu sistem atau aplikasi yang diuji ketika diujikan melebihi kondisi yang diantisipasi pada saat produksi.

Dalam tesis ini alasan dilakukannya uji kinerja adalah *membandingkan karakteristik kinerja dari beberapa sistem.* Sistem itu sendiri adalah implementasi *native* dan beberapa solusi virtualisasi yang berjalan dengan pendekatan virtualisasi yang berbeda-beda.

Kategori uji kinerja yang digunakan pada tesis ini adalah *performance testing*. *Performance testing* yang dilakukan adalah dengan menjalankan suatu tugas dengan beban kerja tertentu yang tidak menggunakan seluruh sumber daya CPU. Pada saat beban diberikan dilihat berapa penggunaan sumber daya CPU-nya bila tugas tersebut dijalankan pada saat *native* maupun saat di dalam mesin virtual. *Performance testing* lainnya dilakukan dengan menjalankan suatu tugas baik saat *native* maupun saat di dalam mesin virtual dan melihat berapa lama tugas itu bisa diselesaikan.

2.4.1 Baseline dan Benchmarking

Dalam uji kinerja, membuat suatu *baseline* berarti menjalankan suatu pengujian untuk mendapatkan data awal kinerja yang terukur dengan tujuan untuk dievaluasi/dibandingkan dengan perbedaan kinerja dari beberapa sistem atau aplikasi dengan konfigurasi yang berbeda. Aspek penting dari *baseline* adalah semua karakteristik atau konfigurasi yang tersedia, kecuali yang memang secara sengaja divariasikan untuk perbandingan, harus tetap sama (Meier dkk., 2007).

Perbandingan kinerja ..., Rio Rasian A., Fasilkom UI, 2009

Benchmarking itu sendiri adalah membandingkan kinerja dari suatu sistem atau aplikasi terhadap suatu *baseline* yang dibuat sendiri atau terhadap standar industri yang didukung organisasi lainnya (Meier dkk., 2007).

Dalam tesis ini yang menjadi *baseline* adalah kinerja suatu sistem dan aplikasi yang diujikan ketika virtualisasi tidak digunakan, atau sering disebut sebagai *native*. Variasi terdapat pada (kinerja) beberapa solusi virtualisasi, terutama pendekatannya, yang digunakan untuk menjalankan sistem dan aplikasi yang sama.

Perbandingan kinerja ..., Rio Rasian A., Fasilkom UI, 2009