MATERI VIRTUALISASI

▪ KONSEP TEKNOLOGI VIRTUALISASI

Infrastruktur IT merupakan aset strategis dan dasar yang penting bagi perangkat lunak untuk dapat memberikan layanan dan aplikasi user yang dibutuhkan oleh perusahaan dalam rangka membentuk *People-Ready Business*. Perkembangan teknologi baru yang serba cepat berdampak pada *data center* dan infrastruktur *desktop* yang menjadi sangat kompleks, tidak fleksibel, dan sulit untuk disesuaikan dengan kebutuhan biaya yang semakin tinggi dan secara relatif pasti namun tanpa mempertimbangkan perubahan kebutuhan bisnis. Oleh karena itu dibutuhkan suatu metode untuk menjawab tantangan ini, sehingga infrastruktur IT menjadi aset strategis yang mampu menjadi sarana informasi dan hubungan antara mitra kerja dengan *customer* yang diperlukan untuk mencapai sukses.

Seiring pertumbuhan bisnisnya, perusahaan dihadapkan pada kebutuhan infrastruktur teknologi informasi yang makin besar. Hal tersebut tentu akan berimbas kepada naiknya kebutuhan energi, tambahan sumber daya manusia, dan secara keseluruhan akan menaikkan *total cost ownership* (TCO) yang harus dikeluarkan.

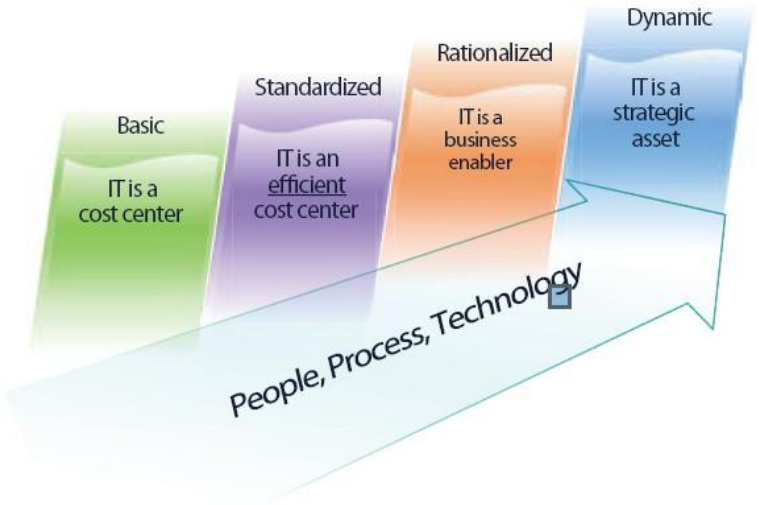
Teknologi virtualisasi menawarkan ekspansi infrastruktur teknologi informasi ke level yang lebih tinggi tanpa harus dihadapkan kepada ongkos investasi yang berlipat. Hal tersebut memungkinkan karena virtualisasi menyederhanakan cara memanfaatkan sumber daya komputer sehingga dapat digunakan untuk berbagai kebutuhan. Sebelum ada virtualisasi, setiap aplikasi bisnis harus berjalan dengan *server* masing-masing sehingga dikenal ada *server* CRM (*customer relationship management*), ERP (*enterprises resource planning*), dan sejenisnya.

Setiap penambahan aplikasi akan diikuti dengan penambahan mesin baru dan kebutuhan ruangan baru. Dengan virtualisasi, penambahan ruang bisa dikatakan tak diperlukan karena sebuah *server* dapat digunakan bersama-sama untuk banyak aplikasi. Penambahan mesin pun, jika diperlukan karena keterbatasan kapasitas, menggunakan sistem modular sehingga lebih cepat dan efisien. Biaya untuk membayar lisensi perangkat lunak juga akan lebih kecil dengan jumlah *server* yang lebih sedikit.

Salah satu model optimalisasi infrastruktur IT membedakan dalam empat tingkatan *infrastructure maturity* dengan karakteristiknya masing-masing [4]:

1. ***Basic***: "*IT fights fires*"
   Uncoordinated desktop, infrastruktur manual, biaya pengaturan dan pemeliharaan yang tinggi.
2. ***Standardized***: "*IT is gaining control*"
   Pengaturan infrastruktur IT dengan beberapa sistem otomasi.
3. ***Rationalized***: "*IT enables business success*"
   Pengaturan dan konsolidasi infrastruktur IT.
4. ***Dynamic***: "*IT is a strategic asset*"
   Pengaturan yang *fully automated*, penggunaan sumber daya yang dinamis.

Pada saat ini sebahagian besar organisasi berada pada tahap *Basic*, dimana IT masih dipandang sebagai *cost center*. Bagi organisasi-organisasi yang mengadopsi teknologi-teknologi standar dan mengimplementasikannya, IT dapat menjadi *cost center* yang efisien. Namun bagi sebahagian besar organisasi yang benar-benar ingin mengubah pandangan IT sebagai *cost center* tanpa memperdulikan efisiensinya, maka perlu dilakukan rasionalisasi IT sehingga mampu menjadi bagian yang mendukung kelancaran bisnis. Pada akhirnya diharapkan perkembangan IT ke arah yang dinamis sehingga mampu menjadi aset strategis yang dapat memberikan keuntungan kompetitif.



Gambar 2.1 Tingkatan Optimalisasi Infrastruktur IT [4]

2.1 Komponen-komponen Jaringan

Jaringan komputer dibentuk dengan menghubungkan beberapa komputer sehingga dapat saling berkomunikasi dan melakukan pertukaran data. Jaringan ini memiliki beberapa elemen dasar yang meliputi komponen perangkat keras dan perangkat lunak, yaitu :

1. **Komponen Fisik :**
   a) Personal Computer (PC)

      Tipe personal komputer yang digunakan di dalam jaringan akan sangat menentukan unjuk kerja dari jaringan tersebut. Pada jaringan tipe *Client-Server*, komputer yang difungsikan sebagai *server* mutlak harus memiliki unjuk kerja yang lebih tinggi dibandingkan komputer-komputer lain sebagai *workstation*-nya

   b) Network Interface Card (NIC)

      Network Interface Cards (NIC) dibutuhkan oleh setiap mesin yang terhubung ke jaringan. Perangkat ini memungkinkan sinyal dari jaringan ditransmisikan ke mesin melalui kabel, infra merah, atau gelombang radio. NIC dipasang pada setiap komputer yang terhubung ke jaringan. Terdapat beberapa macam kartu jaringan berdasarkan jenis kartu, jenis protokol dan tipe kabel yang didukungnya.

   c) Kabel

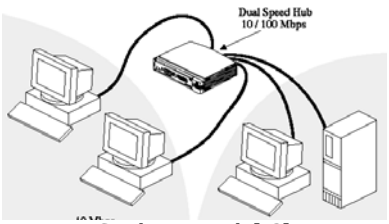
      Kabel merupakan komponen penting dalam jaringan karena media ini yang digunakan oleh data agar dapat mengalir pada jaringan. Terdapat empat jenis kabel yang digunakan untuk pengiriman data yaitu Thin Ethernet, Thick Ethernet, Twisted Pair dan Fiber Optik.

   d) Hub

      Hub, atau sering juga disebut konsentrator, adalah komponen dalam jaringan yang menghubungkan beberapa komputer sekaligus menjadi satu jaringan. Perangkat ini menyatukan kabel-kabel jaringan dari tiap workstation, *server*, atau perangkat lainnya.

   e) Bridge dan Switch

      Bridge dan *Switch* adalah perangkat komunikasi data yang beroperasi secara prinsip pada lapisan kedua di model referensi OSI. Secara umum sering disebut sebagai perangkat lapisan *data link* (*data link layer devices*).



Gambar 2.2 Hub [13]

f) Router

Router bekerja dengan cara yang mirip dengan *switch* dan *bridge*. Perbedaannya, *router* menyaring lalu lintas data dengan menggunakan protokol tertentu. Sebuah IP *router* bisa membagi jaringan menjadi beberapa subnet sehingga hanya lalu lintas yang ditujukan untuk IP *address* tertentu yang bisa mengalir dari satu segmen ke segmen lain. *Router* menghubungkan jaringan komputer dengan jaringan komputer yang lain

a) **Komponen Perangkat lunak,** berupa Sistem Operasi Jaringan, Network Adapter Driver, dan Protokol Jaringan. Sistem operasi jaringan diperlukan untuk mengelola suatu jaringan, dibedakan menjadi dua berdasarkan tipe jaringannya, yaitu sistem operasi *client-server* dan sistem operasi jaringan *peer* to *peer*.

2.2 Pengertian Virtualisasi

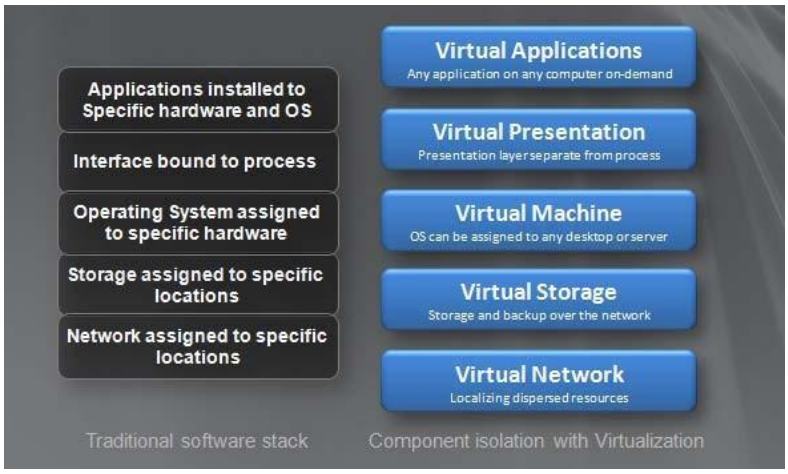
Pengertian Virtualisasi dalam lingkungan IT secara esensial melakukan isolasi terhadap satu sumber daya komputasi dengan yang lainnya. Dengan memisahkan layer-layer yang berbeda dalam *logic stack* dimungkinkan fleksibilitas yang lebih tinggi karena tidak diperlukan lagi konfigurasi tiap-tiap elemen untuk dapat bekerja bersama-sama.

Salah satu jalan yang paling baik untuk memahami virtualisasi adalah dengan melihat ke mesin virtual. Pada mesin virtual, sistem operasi dan aplikasi dikemas bersama-sama untuk selanjutnya dilakukan *hosted* pada *server* fisikal yang menjalankan sistem operasi *host* atau *virtual layer. Virtual layer* adalah

Gambar 2.3 Router [13]

sebuah layer perangkat lunak tipis yang menyediakan *basic interface* dengan perangkat keras. Konsep terpenting untuk memudahkan pemahaman adalah bahwa mesin virtual (OS+Aplikasi) dioperasikan secara independen dari OS pada *server* fisikal seakan-akan berada pada *discrete hardware*-nya sendiri. Hal ini memungkinkan beberapa mesin virtual dijalankan pada sebuah *server* fisikal. Untuk memperoleh gambaran mengenai *hardware/software traditional stack* dan *logic stack virtual* dapat dilihat pada gambar 2.4.

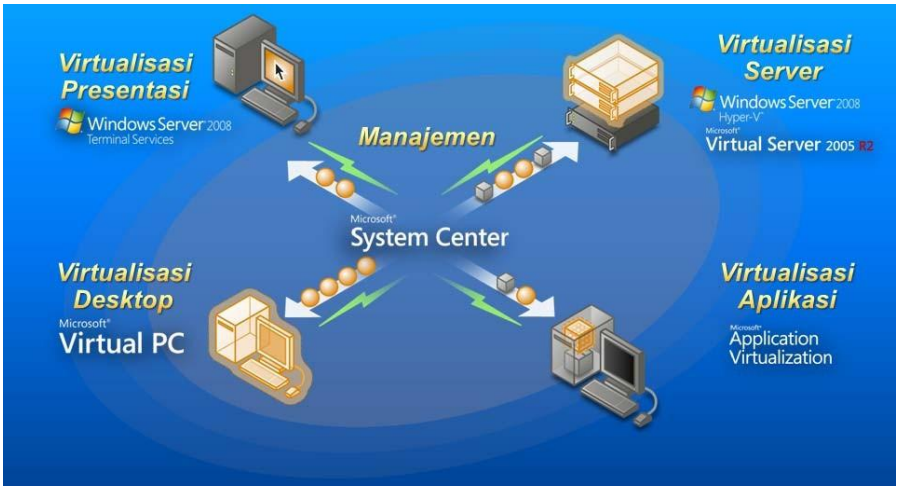
Pada *traditional software stack*, semua elemen dikumpulkan bersama sehingga membutuhkan konfigurasi yang tepat agar setiap komponennya bisa bekerja sama antara satu dengan yang lainnya. Sedangkan dalam *virtual stack*, setiap elemen secara logikal terisolasi dan berdiri sendiri, tidak terikat antara satu dengan yang lainnya. Dasar pemikiran mesin virtual ini makin jelas ketika dilakukan pengamatan terhadap beban kerja yang umumnya hanya menggunakan beberapa bagian dari keseluruhan kemampuan perangkat keras. Dengan menyesuaikan beban kerja yang saling melengkapi dalam kaitannya dengan *processing* dan penggunaan *memory*, jumlah *server* fisikal yang dibutuhkan untuk mendukung operasional bisnis dapat dikurangi. Secara tipikal ratio penggunaan *server* hanya berkisar 15% dimana 85% dari kapasitas *server* tidak dimanfaatkan dengan maksimal. Dengan meningkatkan ratio penggunaan menjadi 60% berarti terjadi pengurangan sebanyak empat kali dari kebutuhan *space*, perangkat keras, *electrical cost powering* dan pendinginan dari kebutuhan *server*. Hal ini disebut *server consolidation*



Gambar 2.4 Traditional dan Virtual Stack [5]

Konsep virtualisasi dapat diaplikasikan pada *enterprise storage, networks*, aplikasi, dan *desktop*. Virtualisasi merupakan *toolset* lengkap yang digunakan untuk mengurangi *server* fisikal. Penggunaan masing-masing virtualisasi dijelaskan sebagai berikut :

1. Virtualisasi *Server* memisahkan OS yang secara logikal diisolasi dari *host server*-nya. Hal ini akan meningkatkan penggunaan sumber daya (perangkat keras, *utilities, space*).
2. Virtualisasi *Desktop* dapat melakukan *host desktop* pada mesin virtual dalam *data center*, memungkinkan tiap *end user* menentukan akses melalui sebuah *remote graphics protocol*. Sebagai pilihan, virtualisasi *desktop* dimungkinkan juga dibuat OS *environment* yang terpisah dalam *desktop user*, dimana beberapa OS dan aplikasi-aplikasi yang berkaitan dapat berjalan secara simultan pada desktop user.
3. Virtualisasi Aplikasi dilakukan dengan mengisolasi aplikasi yang berjalan pada sistem operasi yang sama sehingga membantu untuk meniadakan konflik yang potensial terjadi dan memungkinkan provisioning dengan cepat. Sebagai contoh, sebuah aplikasi yang mungkin biasanya melakukan update ke registry, dapat melakukan update ke sebuah virtual registry. Dengan demikian sistem mampu mendapatkan kebutuhan aplikasi tanpa mengganggu aplikasi lainnya. Aplikasi-aplikasi tidak di install dengan cara tradisional, sehingga aplikasi-aplikasi tersebut dapat di-*setup* dan di-*uninstall* lebih cepat dibandingkan



Gambar 2.5 Strategi virtualisasi yang ditawarkan oleh Microsoft bagi aset virtual dan fisikal [4]

setup dan *uninstall* dengan prosedur yang biasa, termasuk opsi *custom* yang dapat di konfigurasi secara manual.

4. Virtualisasi Presentasi memungkinkan *user remote* untuk mengakses aplikasi dan sistem operasi yang di-*host* dari lokasi *remote*. Model yang umum digunakan adalah mengakses data perusahaan dari rumah atau selama dalam perjalanan dinas. Sistem ini memungkinkan *user remote* untuk melakukan manipulasi data, *log-in* kedalam aplikasi pada PC *desktop*, dan menggunakan sumber daya lain yang mungkin tidak tersedia. Virtualisasi presentasi telah menambahkan keuntungan dari aplikasi *resource-intensive* untuk digunakan melalui komputer *portable* atau komputer lainnya yang mungkin tidak kompatibel, bahkan yang berjalan menggunakan sistem operasi lain.

Pembangunan infrastruktur dengan perencanaan virtualisasi yang baik akan memberikan dampak bagi penurunan biaya dan tingkat pelayanan yang lebih tinggi.

2.3 Virtualisasi *Server*

Teknologi virtualisasi *server* memungkinkan beberapa sistem operasi *server* berjalan di satu mesin fisik yang sama. Tujuan utama penggunaan teknologi ini adalah fungsi infrastruktur yang dapat diandalkan dan memungkinkan penggunaan yang maksimal dari sebuah mesin *server*. Kebutuhan akan penggunaan infrastruktur yang maksimal diperlukan karena biasanya dalam skala *enterprise*, satu *server* didedikasikan hanya untuk satu peran saja. Dengan demikian sering kali terjadi sebuah *server* penggunaannya hanya 10%. Hal ini tentu saja tidak efektif, terutama apabila investasi yang telah dikeluarkan untuk membeli mesin tersebut cukup besar.

Pilihan yang dapat diambil untuk mengatasi masalah ini adalah menggabungkan beberapa peran dalam satu mesin *server*, misalnya menggabungkan DHCP *server*, DNS *server*, *File server* dan *Print server* dalam satu mesin. Namun jika terlalu banyak peran yang dipasang di satu *server* justru berpotensi menimbulkan *bottleneck*. Selain itu menggabungkan beberapa peran dalam satu *server* terkadang membutuhkan dibukanya beberapa port yang tentunya hal ini berpotensi menimbulkan serangan.

Bila kita menggabungkan beberapa peran dalam satu mesin, kita juga akan menemui kendala saat harus melakukan proses *update*. Misalnya kita ingin melakukan *update* pada suatu *service* yang ternyata membutuhkan proses *restart server*. Hal ini tentu akan mengorbankan *service* lain yang sedang berjalan.

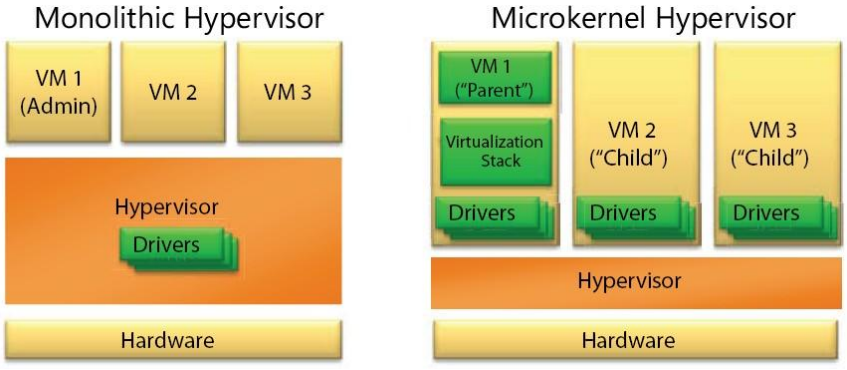
Konsep virtualisasi memungkinkan beberapa *server* berjalan diatas satu mesin. Hal ini menurunkan *space* yang dibutuhkan oleh *server* dan memaksimalkan penggunaan *server*. Setiap peran dapat berjalan di sebuah lingkungan virtual yang terisolasi sehingga relatif lebih aman dan mudah untuk diatur. Bila salah satu *server* down, maka administrator cukup mematikan *server* tersebut dan menyalakan cadangannya. Semudah melakukan aktivitas *copy* dan *paste*. Implementasi virtualisasi dapat menekan *down time* secara drastis. Salah satu konsep virtualisasi seperti yang ditawarkan oleh *Microsoft Windows Server 2008* memperkenalkan *Hypervisor* yang mengadopsi konsep *microkernelized* dan langsung terintegrasi dengan peran *server*. Implementasi dari *microkernelized* memungkinkan sebuah *instance* mesin virtual berperan sebagai partisi *host* dan *instance* yang lain sebagai partisi *guest*. Semua *instance* mesin virtual akan berjalan diatas *hypervisor*.

2.4 Arsitektur Virtual Hypervisor

Virtualisasi merupakan solusi yang telah diadaptasi secara luas. Sekitar 75 persen dari seluruh organisasi yang telah menggunakan atau minimal mengevaluasi teknologi virtualisasi dapat melihat dan memperoleh manfaatnya untuk konsolidasi *server*, manajemen tersentralisasi, pengurangan biaya sehubungan dengan pengurangan perangkat keras, penggunaan listrik, dan kebutuhan pendinginan ruangan *server*. Implementasi teknologi virtual yang dirasakan memberikan banyak manfaat menyebabkan perusahaan-perusahaan ingin mendapatkan manfaat lain dari virtualisasi yang mampu menjawab tantangan beban kerja yang tinggi. Dalam hal ini diharapkan solusi virtualisasi yang lebih *powerful* dan fleksibel namun dapat terintegrasi dengan lebih baik terhadap *management tools* yang mereka miliki. Penggunaan *server* 64-bit, *multi-processor*, dan *multi-core* mampu memenuhi kebutuhan akan mesin virtual sehingga dapat diperoleh manfaat yang lebih baik dari perangkat keras *server* dengan skalabilitas tinggi.

Hypervisor merupakan sebuah layer berisi kode di bagian atas perangkat keras dengan *attack surface* yang dibuat seminim mungkin. Terdapat dua jenis hypervisor yaitu :

1. *Monolithic Hypervisor* adalah sebuah layer yang secara relative lebih tebal diantara sistem operasi *guest* dan perangkat keras. *Monolithic hypervisor* membawa *driver* perangkat kerasnya sendiri. Model ini masih menggunakan banyak kode antara sumber daya perangkat keras dan mesin virtual, karena monitor mesin virtual mengemulasikan perangkat keras untuk mesin virtualnya. Hypervisor mengontrol akses *guest* ke prosesor, *memory*, *input*/*output*, dan mengisolasi antara *guest* yang satu dengan yang lain. Oleh karena layer *monolithic hypervisor* masih relatif besar dan membawa *multiple drivers* mengakibatkan *attack surface*-nya masih signifikan. Contoh *monolithic hypervisor* adalah *Server* ESX VMware.
2. *Microkernel Hypervisor* merupakan sebuah layer tipis diantara *guest* dan perangkat keras. Satu-satunya layer yang berada diantara sebuah sistem operasi *guest* dan *perangkat keras* adalah *hypervisor streamline* dengan fungsi partisi sederhana. Hypervisor tidak memiliki *driver device third-party*. Untuk meningkatkan daya guna, hypervisor telah memiliki arsitektur yang lebih aman dengan *surface attack* yang kecil. *Driver* yang dibutuhkan untuk perangkat keras dalam berbagi sumber daya terletak pada sistem operasi *host*, yang menyediakan akses ke *driver-driver* yang telah dibuat untuk Windows.. Dengan adanya penggunaan *driver* dari masing-masing *guest* maka ukuran *trusted computing base* (TCB) menjadi berkurang karena *guest* tidak melakukan *routing* melalui *host partition drivers*.



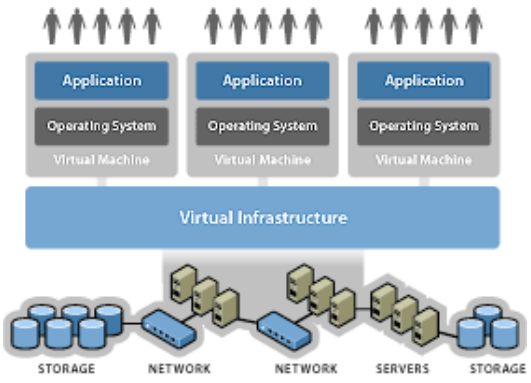
Gambar 2.6 Arsitektur Hypervisor [8]

Dari dua jenis hypervisor di atas, pendekatan *microkernel hypervisor* dianggap lebih baik karena OEM (*Original Equipment Manufacturer*) tidak perlu membuat driver khusus untuk *hypervisor*, lebih banyak perangkat keras yang bisa digunakan, dan mengurangi kemungkinan perbedaan kinerja sistem yang ketika divirtualisasi. Prosesor-prosesor modern telah mengantisipasi virtualisasi yang memungkinkan *layer hypervisor* menjadi lebih tipis.

2.5 Virtualisasi Infrastruktur

Pada dasarnya virtual infrastruktur merupakan pemetaan secara dinamis dari fisikal sumber daya sesuai dengan kebutuhan bisnis. Ketika sebuah mesin virtual mewakili sumber daya fisikal dari sebuah komputer tunggal, maka virtual infrastruktur mewakili sumber daya fisikal dari keseluruhan IT *environment*, berupa sejumlah computer x86, network yang terhubung dan media penyimpanan yang tergabung menjadi sumber daya IT. Secara struktural, virtual infrastruktur terdiri dari beberapa komponen yaitu :

1. *Bare-metal hypervisors* yang memungkinkan virtualisasi secara utuh untuk setiap komputer x86.
2. Servis virtual infrastruktur yang disediakan seperti *management* sumber daya dan konsolidasi *backup* memaksimalkan sumber daya yang ada pada mesin virtual.
3. Solusi otomasi yang menyediakan kemampuan khusus untuk mengoptimalkan proses IT tertentu seperti *provisioning* dan *disaster recovery.*



Gambar 2.7 Virtual Infrastruktur [10]

Dengan menyatukan seluruh *environment software* dari infrastruktur perangkat keras yang mendasarinya, virtualisasi memungkinkan agregasi beberapa *server*, infrastruktur penyimpanan dan jaringan menjadi bagian yang saling berbagi sumber daya untuk dapat disampaikan secara dinamis, aman dan diandalkan sesuai dengan kebutuhan. Pendekatan ini memungkinkan organisasi membangun infrastruktur komputasi dengan pemanfaatan yang maksimal, ketersediaan, otomatisasi dan fleksibilitas menggunakan perencanaan yang efektif sesuai standar industri *server*.

2.6 Alokasi Sumber Daya *Server* Virtual

Teknologi Virtualisasi memungkinkan penghematan biaya perangkat keras dengan memperbaiki penggunaan sumber daya *server* karena beberapa mesin virtual menggunakan fisikal *server* yang sama. Sehubungan dengan hal tersebut dibutuhkan konfigurasi sistem operasi *host* dan sistem operasi *guest* yang mengatur penggunaan sumber daya secara efisien namun tidak mempengaruhi kinerjanya. Metode alokasi sumber daya yang dapat digunakan bervariasi bergantung kepada perangkat lunak virtualisasi yang digunakan.

2.6.1 Sistem Operasi *Host*

Secara keseluruhan sistem operasi *guest* bergantung pada sistem operasi *host* sehingga pengaturan konfigurasi sistem operasi *host* merupakan hal yang sangat penting. Sebagai contoh, sistem operasi *host* yang harus dijalankan adalah *Windows Server 2008 64-bit* untuk penggunaan *Hyper-V*, sedangkan sistem operasi *guest* bisa menggunakan sistem operasi 64-bit atau 32-bit.

Kebutuhan lain yang harus diperhatikan adalah penggunaan perangkat keras yang mendukung virtualisasi. Pengembangan perangkat lunak virtual telah menempatkan sistem operasi *guest* sedemikian rupa sehingga dapat berkomunikasi secara langsung dengan perangkat keras *server*. Pendekatan ini memberikan kinerja yang lebih baik dibandingkan sebelumnya. Saat ini dapat digunakan perangkat keras virtual baik keluaran Intel (Intel VT) maupun AMD (AMD-V).

Pada implementasi virtualisasi, salah satu bagian penting yang harus diperhatikan adalah *memory*. Untuk sistem operasi *host* direkomendasikan untuk mengalokasikan 2 GB RAM. Alokasi *memory* dapat dilakukan dengan menjum-

lahkan keseluruhan *memory* yang dibutuhkan oleh sistem operasi *guest* secara simultan, kemudian disisihkan paling sedikit 2 GB dari jumlah total *memory* yang digunakan oleh mesin-mesin virtual dengan jumlah total *memory* yang dipasang pada *server*.

2.6.2 Network Interface Card

NIC dapat menjadi salah satu penyebab terjadinya *bottlenecks* pada jaringan. Jika pada beberapa mesin virtual yang masing-masing menjalankan aplikasi jaringan secara intensif maka koneksi jaringan dapat dengan cepat mengalami saturasi. Teknologi virtual dirancang khusus agar dapat digunakan beberapa NIC pada *server*. Tiap mesin virtual diarahkan ke NIC yang berbeda. Idealnya, ada NIC untuk sistem operasi *host* dan ada pula NIC untuk masing-masing sistem operasi *guest*, namun, kadang kala tidak dapat diterapkan pada kondisi sesungguhnya.

Sebagai contoh, pada *server* yang hanya memiliki tiga *expansion slot* dan dua diantaranya telah digunakan oleh RAID *controller*, maka hanya tersisa satu expansion slot yang dapat digunakan oleh NIC. Berhubung pada *server* sendiri telah memiliki satu NIC yang terintegrasi sehingga terdapat total koneksi 2 GB sesuai dengan kebutuhan. Dengan demikian hanya dibutuhkan pemetaan terhadap setiap mesin virtual untuk menggunakan salah satu NIC berdasarkan jumlah *network traffic* yang telah diperkirakan sebelumnya. Perlu diingat bahwa distribusi mesin virtual yang dihubungkan dengan *network traffic* dan NIC tidak selalu berarti memetakan setengah dari mesin virtual ke satu NIC, dan sisanya ke NIC yang lain. Beberapa mesin virtual bisa jadi mengirim dan menerima lebih banyak *traffic* dibandingkan dengan yang lain. Hal ini dapat dijadikan pertimbangan dalam memetakan NIC ke mesin virtual.

2.6.3 Sumber Daya CPU

Sebuah sistem operasi *host* dapat melayani beberapa sistem operasi *guest* dimana sumber daya yang dialokasikan berbeda untuk setiap mesin virtual. Oleh karena itu jika ingin menambah sumber daya CPU ke mesin virtual tertentu dapat diambil beberapa sumber daya dari mesin virtual yang lain. Alokasi sumber daya dapat dibedakan berdasarkan bobot dan kapasitasnya. Pengaturan berdasarkan kapasitas memberikan kemampuan pengontrolan yang lebih besar dibandingkan

dengan berdasarkan bobot, karena pengaturan kapasitas menspesifikasikan jumlah maksimum dan minimum sumber daya yang dapat digunakan oleh mesin virtual.

Terdapat beberapa pilihan yang dapat dilakukan untuk mengatur sumber daya CPU yaitu :

1. Memilih jumlah *logical processor* yang dialokasikan bagi mesin virtual. *Logical processors* melakukan *mirroring* terhadap jumlah fisikal *cores* yang dipasang pada mesin. Contohnya, pada *server* yang memiliki empat *processor cores* hanya dipetakan satu *logical processor* ke mesin virtual meskipun mesin dapat lebih menguntungkan jika memiliki *multiple logical processors*. Pertimbangannya adalah jika *server* tersebut melakukan *host* terhadap tiga mesin virtual yang berbeda maka dialokasikan sebuah CPU *core* untuk sistem operasi *host* dan masing-masing mesin virtual.
2. Menentukan persentase sumber daya CPU bagi mesin virtual. Hal ini dilakukan pada mesin virtual yang menjalankan aplikasi CPU secara intensif sehingga bisa dipastikan bahwa mesin virtual selalu memiliki paling sedikit tingkatan minimal sumber daya CPU yang dapat digunakan olehnya.
3. Pembatasan mesin virtual yang menjaga agar tidak menggunakan sumber daya CPU secara berlebihan. Hal ini bertolak belakang dengan poin 2 di atas. Pada *server* dengan empat *processor cores* dilakukan konfigurasi pada masing-masing mesin virtual untuk dapat menggunakan hingga 100% sumber daya CPU yang telah diasosiasikan dengan satu dengan satu logik *processor*. Oleh karena terdapat empat logikal *processor* pada mesin maka pengaturan ini menggunakan 25% dari total sumber daya CPU mesin.
4. Menentukan bobot relatif penggunaan mesin virtual. Dasar pemikirannya adalah mesin virtual dengan bobot relatif lebih tinggi akan menerima lebih banyak *CPU time*, dan mesin virtual dengan bobot relatif lebih rendah akan menerima lebih sedikit *CPU time*. Namun jika tidak diperlukan perlakuan khusus bagi setiap mesin virtual maka dapat dialokasikan *CPU time* yang sama.

2.6.4 Memory

Alokasi *memory* ke mesin virtual diawali dengan memperkirakan berapa mesin virtual yang akan dijalankan secara simultan. Setelah itu dilakukan

perancangan kebutuhan sistem terhadap mesin virtual seperti halnya yang dilakukan pada mesin fisikal. Pada akhirnya, jumlahkan seluruh *memory* yang dibutuhkan lalu tambahkan 2 GB untuk sistem operasi *host.* Alokasi *memory* hanya dapat dilakukan pada mesin virtual, tidak pada sistem operasi *host* yang menggunakan *memory* tersisa setelah seluruh mesin virtual mulai bekerja. Alokasi *memory* untuk mesin virtual hanya digunakan ketika pertama kali mesin virtual dinyalakan. Sebelum mesin virtual dijalankan, alokasi *memory* untuk mesin tersebut dapat digunakan oleh sistem operasi *host*.

2.6.5 Sumber Daya Disk

Setiap mesin virtual yang dibuat dipetakan ke satu atau lebih *hard drive* virtual. Virtual *hard drive* adalah sebuah file berukuran besar yang berlaku sebagai *repository* bagi seluruh file yang dihubungkan dengan mesin virtual. Seperti halnya dengan file yang lain, virtual *hard drive* dapat ditempatkan dimana saja sepanjang tersedia kapasitas *disk* yang memadai. Bertitik tolak pada kinerja, solusi ideal adalah dengan menggunakan *independent storage array* untuk setiap virtual *hard drive*. Solusi lain adalah menyimpan virtual *hard drives* pada *Storage Area Network* (SAN).

Masalah utama yang ditimbulkan oleh kedua solusi di atas adalah biaya, sedangkan penggunaan virtualisasi selalu ditujukan untuk membantu mengurangi biaya. Jika ditinjau dari sisi perangkat lunak, tidak ada perbedaan penempatan file virtual *hard drive* pada sistem volume *server* atau jika setiap virtual *hard drive* ditempatkan pada sebuah *dedicated raid array*. Namun dapat dipastikan jika seluruh virtual *hard drive* ditempatkan di dalam *system volume* maka kinerja *server* akan menurun. Jika terdapat beberapa *server* yang berfungsi sebagai *host server* bagi mesin virtual, dapat digunakan dua teknik yang berbeda untuk menempatkan virtual *hard drives* bergantung kepada seberapa banyak harus dilakukan antisipasi sumber daya *disk* yang digunakan.

Jika mesin virtual di-*host* pada sebuah *server* yang tidak menjalankan aplikasi *disk* secara intensif maka solusi virtual *hard drive* yang digunakan adalah membuat sebuah RAID 10 array yang besar dan memanfaatkannya untuk menyimpan seluruh virtual *hard drive*. Sedangkan untuk *server* virtual yang lebih intensif menggunakan *disk* dengan perangkat keras fisikal yang sama maka dapat

dialokasikan *disk* individual ke dalam beberapa array yang lebih kecil. Dengan demikian hampir dapat dipastikan bahwa tidak ada satu pun virtual *hard drive* yang menggunakan seluruh *throughput disk* yang ada. Metode ini juga membantu menghilangkan biaya sehubungan dengan pembelian individual *dedicated array* untuk setiap mesin virtual.

2.7 Disaster Recovery dan High Availability

Pada kondisi sekarang ini memiliki *disaster recovery* yang handal merupakan hal yang sangat esensial. Berdasarkan kemungkinan dan banyaknya peristiwa yang dapat menimbulkan kerusakan termasuk fenomena alam seperti gempa bumi, badai, banjir, dan angin puting beliung, atau yang ditimbulkan oleh manusia seperti pemadaman listrik, teroris, dan virus menyebabkan disaster recovery menjadi bagian utama strategi IT. Namun pada kenyataannya, strategi yang diterapkan memakan waktu dan biaya yang mahal. Oleh karena kendala waktu yang dibutuhkan, banyak organisasi yang tidak memiliki perencanaan disaster recovery yang melingkupi seluruh divais, data, dan aplikasi.

Berdasarkan frekuensi gangguan sistem perangkat keras dan perangkat lunak yang kerap terjadi dapat menyebabkan *downtime* yang mengganggu kelangsungan bisnis. Virtualisasi dapat menyederhanakan strategi *disaster-recovery* dengan melakukan pembagian beban kerja sehingga mampu mencegah dampak negatif terhadap aplikasi yang akan mengganggu kinerja sebagai akibat dari gangguan pada sistem.

Gambar 2.8 Solusi High Availability [4]

Untuk memastikan kelangsungan kinerja sistem, aplikasi dapat diberi perlakuan seperti halnya file data yang dapat direplikasi. Aplikasi yang telah divirtualisasi ditempatkan pada lokasi yang terpusat dan diakses pada desktop end user sesuai dengan kebutuhan sehingga tidak perlu membuat individual images secara keseluruhan. Hal ini akan mengurangi down time yang diperlukan untuk menjalankan sistem kembali.

Pada awalnya seluruh layer dari *computing environments*, termasuk perangkat keras, sistem operasi, aplikasi-aplikasi, dan *storage* bersifat *statis*, artinya masing-masing dikonfigurasi untuk saling berinteraksi dengan baik dan mendukung computing solution secara spesifik. Komponen-komponen yang diinstal pada komputer-komputer partikuler sehingga menghasilkan sistem yang kaku dan tidak mampu beradaptasi dengan baik terhadap perubahan-perubahan. Bagaimanapun *high availability* merupakan substansi vital dari strategi dunia IT yang menghendaki bisnis beroperasi dua puluh empat jam sehari, tujuh hari seminggu. Gangguan pada servis IT dapat berakibat fatal bagi kelangsungan bisnis sehingga dengan adanya virtualisasi *server* mampu membantu meyakinkan servis akan selalu ada jika dibutuhkan. Dengan virtualisasi dapat dibuat infrastruktur *server* yang dinamis dan efisien, meningkatkan kemampuan infrastruktur *server*, mengurangi disruptive events, dan meminimalkan waktu dan sumber daya yang dibutuhkan untuk mendukung infrastruktur.

Gangguan-gangguan yang mungkin timbul sehingga menyebabkan kegagalan servis sistem dapat dibedakan atas beberapa jenis, yaitu :

1. Kegagalan pada mesin *host* secara fisikal yang disebabkan oleh kerusakan *power supply*.
2. Kegagalan pada komponen mesin *host* secara fisikal yang disebabkan oleh kerusakan komponen seperti kerusakan pada *disk* I/O dan media, kerusakan kipas pendingin, dan kerusakan network I/O yang menyebabkan degradasi kinerja atau kerusakan mesin secara parsial. Secara tipikal kerusakan komponen merupakan awal dari kerusakan mesin secara keseluruhan.
3. Kegagalan virtual *server* karena kerusakan pada perangkat lunak virtual *server*-nya sendiri dapat disebabkan oleh dampak negatif disain dan

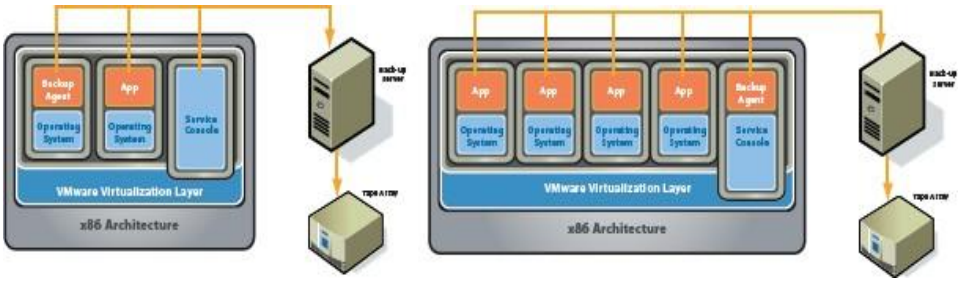
implementasi, kurangnya sumber daya (misalnya *memory*, *disk*), dan *malicious attacks* (misalnya virus).

4. Kegagalan mesin virtual. Setiap *server* virtual menjalankan beberapa mesin virtual dan setiap mesin virtual dapat rusak dan mengalami kegagalan seperti halnya mesin fisikal.
5. Kegagalan komponen mesin virtual. Setiap mesin virtual berisi beberapa komponen virtual seperti *disk* virtual dan virtual *network interface* yang dapat mengalami kegagalan dan menyebabkan sistem tidak mampu memberikan servisnya.
6. Kegagalan aplikasi yang menyebabkan kegagalan sistem.

Beberapa pertimbangan yang dapat menjadi alasan bagi penggunaan teknologi virtual *server* dalam mengimplementasikan metode *Disaster Recovery* adalah :

1. Proses *recovery* yang dapat diandalkan karena *recovery* mesin virtual dapat dilakukan tanpa memperdulikan perbedaan fisik perangkat keras, baik berbasis Intel maupun AMD. Jika dibandingkan dengan upaya *restore server* yang membutuhkan *server* yang identik (misalnya vendor, model, dan konfigurasi), kompatibilitas sistem antara perangkat keras dan OS pada *recovery site* dapat dieliminasi.
2. Kemungkinan untuk mengkonsolidasi *server* di *recovery site* dengan menempatkan beberapa mesin virtual dalam satu *server* fisik. Dalam skenario *failover*, alasan ini paling memungkinkan sebagai *backup* yang sifatnya temporer.
3. Dalam kondisi normal, user dapat memanfaatkan *server-server* pada r*ecovery site* untuk kebutuhan *testing* dan *development*.
4. Penggunaan perangkat keras menjadi maksimal dengan adanya penempatan beberapa mesin virtual dalam satu fisik mesin.

Berdasarkan kebutuhan proteksi dan pemulihan data yang diperlukan terdapat tiga metode *backup* data mesin virtual yaitu *backup* yang dijalankan pada sebuah mesin virtual, *Service Console System Host*, dan Konsolidasi *Backup* pada *Host Server*.



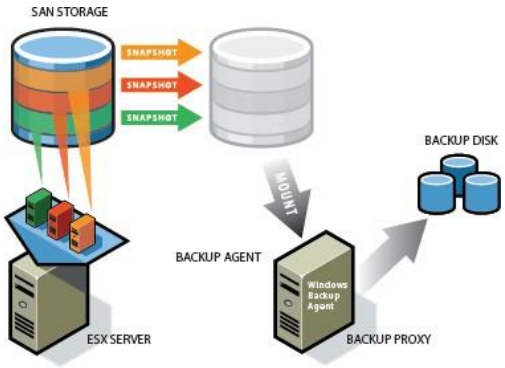
Gambar 2.9 *In-VM* atau *Console Based Backup* memberikan fleksibilitas *backup* berbasis *file* atau *image* [16]

1. *Backup* dari Mesin Virtual

   Pada metode ini *backup agent* bagi produk third-party *backup* ditempatkan di dalam mesin virtual dengan konfigurasi dan prosedur yang digunakan sama dengan proses *backup* pada mesin fisikal.

2. *Backup* melalui *Service Console System Host*

   Dengan *backup* melalui Service Console, *agent* dijalankan melalui Service Console dan melakukan *backup* seluruh *image* mesin virtual. Metode ini memberikan cara yang sederhana untuk melakukan back up full *system images* tanpa mempengaruhi aplikasi yang dijalankan pada individual mesin virtual. Proses *restore* lebih sederhana daripada pendekatan *in-virtual machine* yang hanya dapat melakukan *restore full disk images*, bukan individual files dalam mesin virtual. Metode ini lebih disarankan jika seluruh *images* mesin virtual harus secepatnya di-*restore. In-virtual machine* dan *in-Service Console backup* masing-masing dapat digunakan, sedangkan kombinasi dari keduanya berdasarkan kebutuhan.



Gambar 2.10 *Backup* berbasis image atau file menggunakan perangkat eksternal [16]

3. Konsolidasi Virtual *Backup*

   Sistem *backup* ini memungkinkan untuk melakukan *full image* dan *incremental file backup* pada mesin virtual yang sedang digunakan. Metode ini dapat pula mengurangi trafik *backup* pada LAN dengan memanfaatkan perangkat *tape* yang dipasang pada *storage network* untuk *backup* mesin virtual.

2.8 Skalabilitas Server Virtual

Pada penelitian ini secara tipikal dilakukan penghitungan *overhead* virtualisasi untuk satu mesin virtual dibandingkan dengan sistem operasi dasar. Disamping itu juga ditampilkan data yang merepresentasikan skalabilitas sistem meliputi linearitas sistem dan degradasi kinerja ketika beberapa mesin virtual dijalankan dengan beban yang sama. Skalabilitas secara khusus memiliki relevansi dengan metrik ketika membedakan sebuah sistem untuk kepentingan *hosting environments* komersial yang merupakan target utama bagi sistem virtualisasi. Aspek penting yang lain dalam melakukan pembandingan adalah seberapa jauh proteksi *environment* virtual melakukan isolasi antara satu mesin dengan mesin yang lain. Proteksi terhadap kebaikan perilaku mesin virtual dari penyimpangan perilaku merupakan fitur penting dalam sistem virtualisasi khususnya ketika digunakan untuk *hosting environment* komersial.

Tingkat isolasi kinerja dapat secara substansi dapat bervariasi berdasarkan jenis penyimpangan perilaku. Sistem virtualisasi mungkin saja melakukan isolasi terhadap dampak *CPU hog* namun tidak mampu melakukan isolasi terhadap dampak *network hog*. Terdapat beberapa jenis sistem virtualisasi yaitu full *virtualization, paravirtualization* dan *operating system level virtualization*. Pada *full virtualization*, *interface* disediakan oleh sistem virtualisasi sama dengan aktual fisikal perangkat keras. Hal ini memungkinkan *unmodified* sistem operasi biner dijalankan sebagai *guests* pada mesin virtual. Pada *paravirtualization*, target perubahan dilakukan pada *interface* perangkat keras yang dipresentasikan bagi mesin virtual untuk mengabaikan beberapa fitur yang sulit atau mahal untuk divirtualisasi. Di sini dibutuhkan modifikasi pada sistem operasi untuk disesuaikan dengan modifikasi *interface* perangkat keras. Sedangkan pada *operating system level virtualization*, *guest* mesin virtual secara aktual

menjalankan proses di dalam *general-purpose operating system* yang telah dimodifikasi untuk menyediakan nama yang berbeda ketika *guest* ditampilkan pada mesin yang berbeda pula.

Pada *operating system level virtualization*, seluruh *guest* menggunakan bersama-sama sistem operasi sebagai basis mesin. Namun sistem *operating system level virtualization* tidak mendukung kemampuan untuk menjalankan mesin virtual dengan beberapa sistem operasi yang berbeda pada mesin fisikal yang sama. Dalam banyak *environment*, ini mendukung heterogenitas perangkat lunak sebagai motivasi kunci untuk menggunakan sistem virtual. Sebagai contoh, pemeliharaan Windows XP VM dan Windows Vista VM, RedHat Linux VM dan SUSE Linux VM untuk mengurangi kebutuhan perangkat keras bagi pengujian perangkat lunak yang dijalankan pada beberapa *platform.*