BAB 4
ANALISIS DATA

Pada bab ini dilakukan analisis terhadap hasil pengukuran yang telah diperoleh. Selanjutnya dilakukan perhitungan untuk mengetahui skalabilitas *server* virtual dan penggunaan sumber daya perangkat keras.

4.1 Skalabilitas *Server* Virtual

Pengukuran dilakukan terhadap waktu akses aplikasi untuk menghitung parameter-parameter skalabilitas *overhead*, linearitas, dan isolasi kinerja.

4.1.1 Overhead

Perbandingan yang menarik dapat dilakukan antara *server* virtual dan *server* tradisional pada perangkat lunak dan perangkat keras yang sama. Berdasarkan konfigurasi perangkat keras yang terdapat pada gambar 3.1. diperoleh hasil yang dapat dilihat pada tabel 4.1. Pengukuran dilakukan ketika aplikasi yang sama dijalankan pada *server* tradisional dan sebuah *server* virtual dengan fungsi masing-masing sebagai *Active Directory Server, Exchange Server* dan *Database Server*.

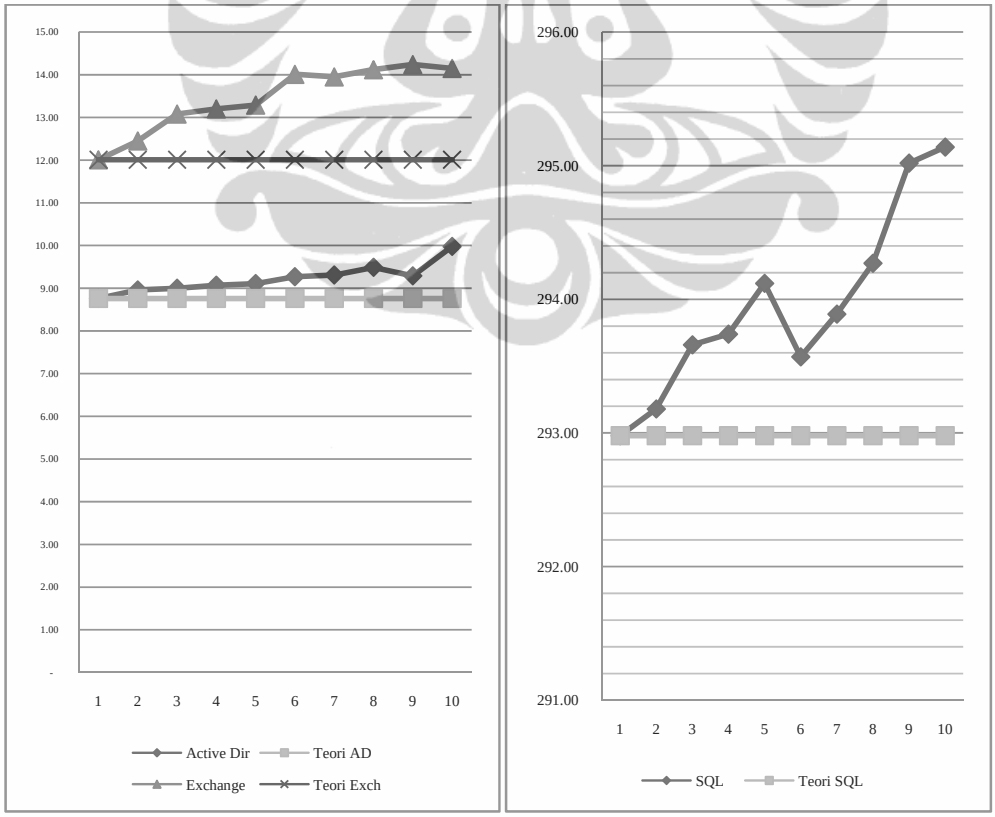
Tabel 4.1 Evaluasi *Overhead* dengan skenario pembebanan pada *Server*

Skenario Pembebanan	Jumlah Proses	Tradisional (sec)	Virtual	
			VM aktif	Waktu (sec)
Copy File 62 MB	1	08:65	1	08:76
			2	08:96
			3	09:00
			4	09:07
			5	09:11
			6	09:27
			7	09:31
			8	09:49
			9	09:29
			10	09:98
Send/Receive Email dengan *attachment* 9 MB	1	0:11:68	1	12:01
			2	12:45
			3	13:08
			4	13:20
			5	13:29
			6	14:01
			7	13:95
			8	14:12
			9	14:24
			10	14:15

Tabel 4.2 Evaluasi *Overhead* dengan skenario pembebanan *Server*

Skenario Pembebanan	Jumlah Proses	Tradisional (sec)	Virtual	
			Jumlah VM aktif	Waktu (sec)
Backup Database 2 GB	1	292:38	1	292:98
			2	293:18
			3	293:66
			4	293:74
			5	294:12
			6	293:57
			7	293:89
			8	294:27
			9	295:02
			10	295:00

Pada bagian percobaan ini, pada satu waktu baik *server* tradisional dan *server* virtual hanya ada satu proses yang berjalan. Namun pada *server* virtual jumlah mesin virtual *non-aplikasi* yang berjalan bersamaan secara bertahap ditingkatkan jumlahnya. Berdasarkan data tabel 4.1 dan tabel 4.2 diperoleh hasil pengukuran pada gambar 4.1



Gambar 4.1 Grafik Pengukuran Overhead pada AD Server, Exchange Server dan SQL Server

Berdasarkan hasil pengukuran diperoleh nilai-nilai untuk masing-masing *server* sesuai dengan peran masing-masing :

- AD *Server* : T_a = 08:65

  Dengan menggunakan rumus 3.1 diperoleh nilai *Overhead*

  $O_v = T_{av} - T_a$

  = 08:76 – 08:65

  = 00:11 = 110 *ms*

  Dengan menggunakan rumus 3.2 diperoleh nilai *Overhead* virtualisasi

  $O_{vn} = T_{anv} - T_a$, untuk n = 10

  = 09:98 – 08:65

  = 01:33 *second* = 1330 ms

  Degradasi kinerja *server* setelah sepuluh mesin virtual dijalankan :

  $= \frac{Ov}{Ovn}\ x\ 100\% = \frac{110}{1330}\ x\ 100\% = 8.27\%$

- Exchange *Server* : T_a = 11:68

  Dengan menggunakan rumus 3.1 diperoleh nilai *Overhead*

  $O_v = T_{av} - T_a$

  = 12:01– 11:68

  = 00:32 = 320 *ms*

  Dengan menggunakan rumus 3.2 diperoleh nilai *Overhead* virtualisasi

  $O_{vn} = T_{anv} - T_a$, untuk n = 10

  = 14:15 – 11:68

  = 01:47 *second* = 1,470 ms

  Degradasi kinerja *server* setelah sepuluh mesin virtual dijalankan :

  $= \frac{Ov}{Ovn}\ x\ 100\% = \frac{320}{1470}\ x\ 100\% = 21.77\%$

- SQL *Server* : T_a = 292:38

  Dengan menggunakan rumus 3.1 diperoleh nilai *Overhead*

  $O_v = T_{av} - T_a$

  = 292:98 – 292:38

  = 00:60 = 600 *ms*

  Dengan menggunakan rumus 3.2 diperoleh nilai *Overhead* virtualisasi

  $O_{vn} = T_{anv} - T_a$ untuk n = 10

O_{vn} = 295:14 – 292:38

= 1:76 second = 1,760 ms

Degradasi kinerja *server* setelah sepuluh mesin virtual dijalankan :

$$= \frac{Ov}{Ovn} \, x \, 100\% = \frac{600}{1760} \, x \, 100\% = 34.09\%$$

Nilai teori *overhead* berkorelasi dengan nilai ketika hanya ada satu mesin virtual yang dijalankan [2]. Berdasarkan observasi ketika hanya ada satu mesin virtual yang dijalankan *overhead* yang ada setelah dibandingkan dengan eksekusi pada host OS dapat diabaikan. Ketika jumlah mesin virtual yang dijalankan meningkat, pada aplikasi database dan aplikasi email terlihat adanya peningkatan nilai. Sedangkan untuk AD *Server* cenderung bersifat linier mendekati skalabilitas optimal dengan bentuk kurva yang mendekati kurva teoritikal.

4.1.2 Linearitas

Berbeda dengan pengukuran *overhead*, untuk linearitas dilakukan penjadwalan jumlah mesin virtual yang mengeksekusi aplikasi yang sama. Untuk linearitas normal diasumsikan waktu eksekusi meningkat secara linear sebanding dengan jumlah mesin virtual yang dijalankan bersama-sama. Seluruh hasil yang ditampilkan merupakan waktu eksekusi aplikasi yang dijalankan pada semua mesin virtual yang aktif. Pada percobaan ini pengukuran untuk Exchange *server* tidak dilakukan karena keterbatasan jumlah *workstation* dimana dalam satu *workstation* yang sama tidak dapat di-*instal* lebih dari satu email *client* exchange.

Tabel 4.3 Evaluasi Linearitas dengan skenario pembebanan pada *Server*

Skenario Pembebanan	Jumlah Proses	Tradisional (sec)	Virtual	
			Jumlah VM aktif	Waktu (sec)
Copy File 62 MB	1	08:65	1	08:76
	2	10:89	2	13:88
	3	19:55	3	19:42
	4	29:15	4	24:40
	5	31:08	5	30:22
	6	34:67	6	36:29
	7	39:80	7	42:14
	8	44:65	8	46:82
	9	51:90	9	51:58
	10	55:50	10	56:80

Tabel 4.4 Evaluasi Linearitas dengan skenario pembebanan pada *Server*

Skenario Pembebanan	Jumlah Proses	Tradisional (sec)	Virtual	
			Jumlah VM aktif	Waktu (sec)
Backup database 2 GB	1	292:38	1	292:98
	2	530:25	2	533:53
	3	634:65	3	635:31
	4	799:80	4	810:73
	5	923:46	5	994:26
	6	1,122:85	6	1,120:04
	7	1,265:34	7	1,267:56
	8	1,400:85	8	1,402:78
	9	1,458:77	9	1,550:30
	10	1,578:89	10	1,697:82

Jika virtualisasi adalah konstan (tidak terikat pada jumlah mesin virtual), maka maksimum eksekusi aplikasi harus menjadi fungsi *affine* dari jumlah mesin virtual yang menjalankan aplikasi [2]. Berdasarkan hasil pengukuran diperoleh nilai-nilai untuk masing-masing *server* sesuai dengan peran masing-masing :

- AD *server* virtual, untuk n = 10

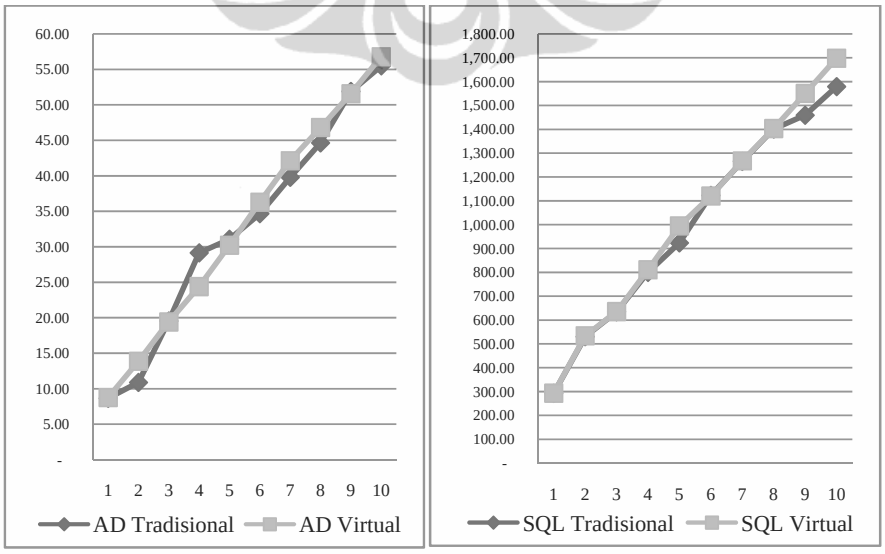
  Dengan menggunakan persamaan 3.3 :

  $t_{max} = O_v + t \times n$, dimana : O_v = 110 ms

  t = 08:76 sec = 8.760 ms

  maka $t_{max} = 110 + 8.760 \times 10$

  = 87.710 ms



Gambar 4.2 Grafik Pengukuran Linearitas pada AD Server dan SQL Server

- SQL *server* virtual, untuk n = 10

  Dengan menggunakan persamaan 3.3 :

  $t_{max} = O_v + t \times n$, dimana : O_v = 600 ms

  t = 292:98 sec = 292.980 ms

  maka $t_{max} = 600 + 292.980 \times 10$

  = 2.930.400 ms

Waktu akses aplikasi maksimum merupakan *affine function* terhadap jumlah mesin virtualnya [2]. *Affine function* adalah fungsi linier ditambah dengan translasinya.

Berdasarkan hasil perhitungan dapat dilihat bahwa :

- AD *server* virtual

  Fungsi Linier : f(x) = m(x) + b,

  dimana nilai b adalah konstanta waktu aplikasi yang dijalankan pada sebuah mesin virtual maka untuk :

  x = 1, y = 8,870 8,870 = m(1) + 110

  m = 8,760

  x = 2, y = 17,630 17,740 = m(2) + 110

  m = 8,760

Hasil perhitungan waktu akses aplikasi maksimum dapat dilihat pada tabel 4.5.

Tabel 4.5 Perhitungan Waktu Akses Aplikasi Maksimum

Jumlah VM Aktif	AD Virtual (ms)	SQL Virtual (ms)
1	8,870.00	293,580.00
2	17,630.00	586,560.00
3	26,390.00	879,540.00
4	35,150.00	1,172,520.00
5	43,910.00	1,465,500.00
6	52,670.00	1,758,480.00
7	61,430.00	2,051,460.00
8	70,190.00	2,344,440.00
9	78,950.00	2,637,420.00
10	87,710.00	2,930,400.00

- SQL *server* virtual

  Fungsi Linier : f(x) = m(x) + b,

  dimana nilai b adalah konstanta waktu aplikasi yang dijalankan pada sebuah mesin virtual maka untuk :

  x = 1, y = 293,580 293,580 = m(1) + 600

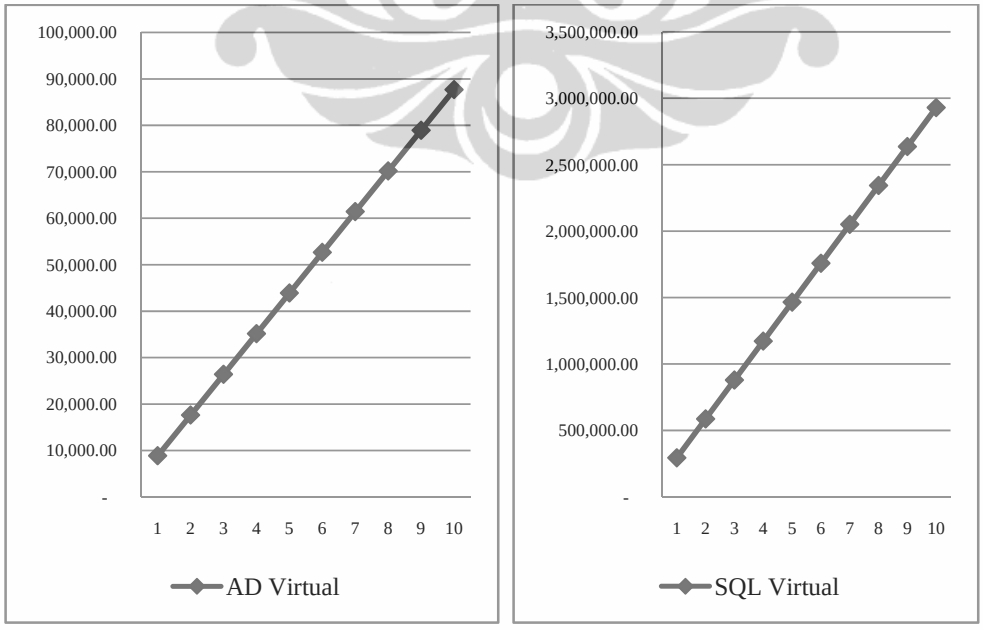
  m = 292,980

  x = 2, y = 586,560 586,560 = m(2) + 600

  m = 292,980

4.1.3 Isolasi Kinerja

Pengukuran isolasi kinerja dilakukan dengan menjalankan dua mesin virtual secara bersamaan. Pada VM1 dilakukan eksekusi sebuah aplikasi dan dua *copy* aplikasi yang sama pada VM2. Pada kasus *scheduling by process* seluruh mesin virtual seharusnya dapat selesai dalam waktu yang bersamaan. Sedangkan pada kasus *scheduling by virtual machine*, sebelum aplikasi yang dijalankan pada VM1 berakhir maka setengah dari sumber daya mesin dialokasikan untuk setiap mesin virtual. Setelah aplikasi yang dijalankan pada VM1 berakhir, setengah dari sumber daya mesin dialokasikan kepada ke setiap mesin virtual. Aplikasi pada VM1 selesai sebelum eksekusi VM2 berakhir.



Gambar 4.3 Grafik waktu eksekusi aplikasi maksimum pada AD Server dan SQL Server

Tabel 4.6 Evaluasi isolasi kinerja dengan skenario pembebanan pada *Server*

Pengambilan ke-	Skenario	AD Virtual	SQL Virtual
1	1 aplikasi pada VM1	16:83	558:43
2		17:01	532:76
3		16:05	550:99
4		16:70	539:87
5		17:10	544:88
6		16:99	542:98
7		16:88	549:89
8		17:22	550:75
9		17:46	552:90
10		16:54	551:76
Rata-rata		16:88	547:52

Hasil pengukuran dapat dilihat pada tabel 4.6 dan 4.7

- AD *Server* : T_a = 08:76

  Dengan menggunakan rumus 3.4 diperoleh nilai isolasi kinerja

  $T_{a1} = 2T_a$

  = 2(08:76)

  = 17:52 second

  $T_{a2} = 3T_a$

  = 3(08:76)

  = 26:28 second

Tabel 4.7 Evaluasi isolasi kinerja dengan skenario pembebanan pada *Server*

Pengambilan ke-	Skenario	AD Virtual (sec)	SQL Virtua (sec)l
1	2 copy aplikasi pada VM2	26:08	875:86
2		25:98	875:34
3		26:50	876:09
4		26:03	876:23
5		25:76	876:01
6		25:99	877:03
7		26:34	877:34
8		26:05	876:99
9		26:00	877:17
10		26:09	877:13
Rata-rata		26:08	876:52

- SQL *Server* : T_a = 292:98

  Dengan menggunakan rumus 3.4 diperoleh nilai isolasi kinerja

  $T_{a1} = 2T_a$

  = 2(292:98)

  = 585:96 second

  $T_{a2} = 3T_a$

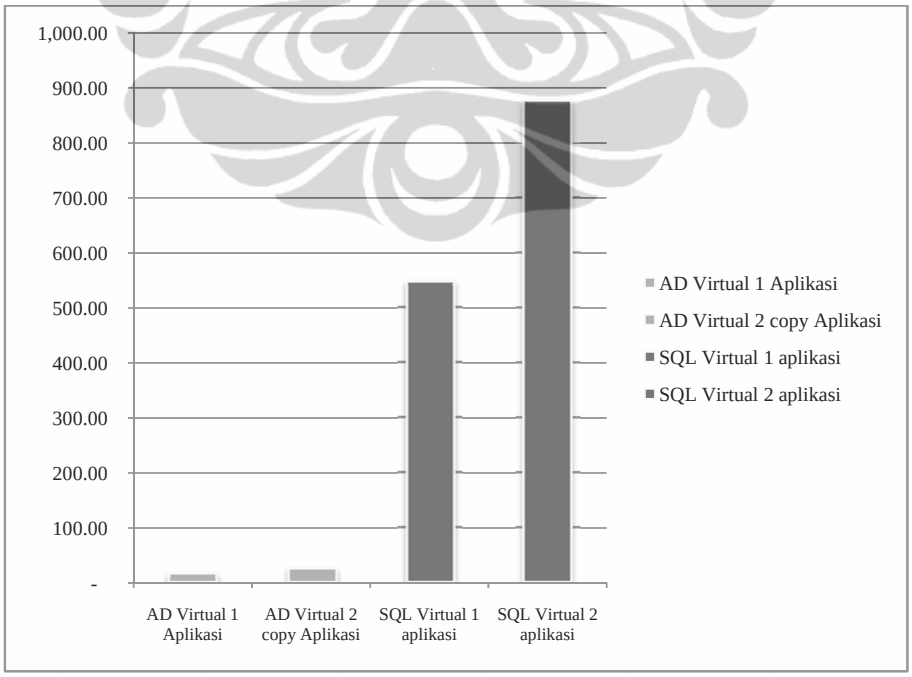
  = 3(292:98)

  = 878:94 second

Berdasarkan hasil perhitungan dapat dilihat bahwa isolasi kinerja pada mesin virtual mendapatkan nilai yang mendekati hasil berdasarkan pengambilan data. Hal ini menunjukkan bahwa isolasi kinerja berlaku diantara mesin virtual yang aktif pada suatu saat.

4.1.4 Analisis Data 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



Gambar 4.4 Isolasi kinerja mesin virtual

dijalankan dengan beban yang sama. Disamping itu dilakukan pula penghitungan untuk mengetahui 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.

Untuk pengukuran yang telah dilakukan diperoleh nilai *overhead* sebagai berikut :

- AD *Server* : O_v = 00:11 = 110 *ms*

  O_{vn} = 01:33 *second* = 1330 ms, untuk n = 10

  Degradasi kinerja *server* setelah sepuluh mesin virtual dijalankan : 8.27%
- Exchange *Server* : O_v = 00:32 = 320 *ms*

  O_{vn} = 01:47 second = 1,470 ms, untuk n = 10

  Degradasi kinerja *server* setelah sepuluh mesin virtual dijalankan : 21.77%
- SQL *Server* : O_v = 00:60 second = 600 *ms*

  O_{vn} = 01:76 second = 1.760 ms, untuk n = 10

  Degradasi kinerja *server* setelah sepuluh mesin virtual dijalankan : 34.09%

Berdasarkan hasil observasi di atas dapat dilihat ketika satu virtual mesin dijalankan maka *overhead* virtualisasi setelah dibandingkan dengan eksekusi pada *host* sistem operasi bisa diabaikan karena nilainya sangat kecil yaitu 110 ms, 320 ms dan 600 ms. Degradasi kinerja meningkat ketika jumlah mesin virtual diaktifkan bertambah pada exchange *server* dan SQL *serve*, yaitu mencapai 21.77% dan 34.09%, sedangkan pada AD *server* hanya mencapai 8.27%. Ketika jumlah mesin virtual yang aktif ditambah, terdapat perbedaan deviasi yang hampir linier pada exchange *server* dan kuadratik atau eksponensial pada SQL *server*. Sedangkan untuk AD *server* mendekati skalabilitas optimal mendekati kurva teoritikal. Hal ini disebabkan adanya dugaan bahwa *high scheduling overhead* merupakan alasan dibalik tingginya *overhead* pada exchange *server* dan SQL *server*. Pada saat mesin dalam kondisi *idle, scheduler* diaktifkan antara 4 dan 8 kali per second [2]. Ketika sebuah mesin virtual dijalankan, *host scheduler* dipanggil mendekati 400 kali per second. Sebagai konsekuensinya, *host scheduler* kehilangan *processor cycles* untuk melakukan *scheduling* mesin virtual yang *idle.*

Dalam mengukur parameter linearitas *server* diperoleh persamaan linier yang merupakan *affine function* sebagai berikut :

- AD *server* virtual

  Fungsi linier yang berlaku : y = 8,760x + 110

- SQL *server* virtual

  Fungsi Linier yang berlaku : y = 292,280x + 600

Perhitungan berdasarkan hasil pengukuran menunjukkan untuk linearitas normal waktu eksekusi meningkat sebanding dengan jumlah mesin virtual yang dijalankan pada waktu yang bersamaan. Seluruh hasil yang dipresentasikan benar merupakan waktu eksekusi yang dijalankan pada seluruh mesin virtual. Hal ini menunjukkan bahwa jumlah mesin virtual pada satu mesin fisikal yang aktif pada satu waktu mempengaruhi *respons time* dari aplikasi yang dijalankan. Kondisi ini berlaku pula pada *server* tradisional.

Dalam mengukur parameter isolasi kinerja *server* diperoleh hasil sebagai berikut :

- AD *Server* : T_a = 08:76

  $T_{a1} = 2T_a$ = 17:52 second = 1,752 ms

  $T_{a2} = 3T_a$ = 26:28 second = 26,280 ms

- SQL *Server* : T_a = 292:98

  $T_{a1} = 2T_a$ = 585:96 second = 585,960 ms

  $T_{a2} = 3T_a$ = 878:94 second = 878,940 ms

Sebuah mesin virtual melakukan isolasi terhadap fisikal mesin yang berada dibawahnya secara efisien. Ini dilakukan oleh *Virtual Machine Monitor* (VMM) yang bertanggung jawab melakukan isolasi mesin terhadap penggunaan bersama sebuah sumber daya fisikal dengan intersepsi dan emulasi terhadap perilaku instruksi-instruksi ISA (*Instruction Set of Architecture*). VMM secara eksplisit melakukan implementasi mekanisme penggunaan bersama yang dilakukan secara adil untuk memastikan adanya isolasi kinerja diantara mesin-mesin virtual. Hasil perhitungan yang dilakukan berdasarkan pengukuran menunjukkan bahwa untuk satu proses aplikasi yang dilakukan pada mesin virtual (T_{a1}) memerlukan waktu dua kali waktu eksekusi pada mesin fisikal. Sedangkan untuk waktu eksekusi aplikasi yang sama dijalankan paralel pada mesin virtual yang lain (T_{a2})

merupakan tiga kali waktu yang dibutuhkan untuk eksekusi aplikasi yang sama pada mesin fisikal. Hal ini menunjukkan bahwa pada saat yang bersamaan seluruh sumber daya mesin dialokasikan terhadap dua mesin virtual yang aktif sebanyak masing-masing VM1 50% dan VM2 50%. Oleh karena pada VM1 hanya dijalankan satu proses, sedangkan pada VM2 dijalankan dua *copy* proses, maka masing-masing proses pada VM2 hanya memperoleh alokasi sumber daya 25%. Setelah proses pada VM1 berakhir, maka VM2 memperoleh tambahan alokasi sumber daya yang berasal dari VM1. Kondisi ini menunjukkan korelasi jumlah proses yang berjalan pada mesin virtual dengan waktu akses aplikasi.

4.2 Pengukuran Penggunaan Sumber Daya pada *Server* Tradisional dan *Server* Virtual

Pengukuran parameter dilakukan pada masing-masing *server* tradisional dan *server* virtual, yaitu *Active Directory Server, Exchange Server,* dan *Database Server.* Pengukuran terhadap penggunaan sumber daya pada dua kondisi yaitu :

1. Pada saat *server* selesai *booting* dan user-user sudah *login* ke jaringan.
2. Pada saat *server* telah menjalankan aplikasi dan diakses oleh user.

4.2.1 Pengukuran parameter pada Active Directory Server

Pengukuran penggunaan sumber daya perangkat keras dilakukan dengan menggunakan *Windows Task Manager* pada *server.* Data yang diambil adalah penggunaan maksimum sumber daya dalam melakukan sebuah proses. Hasil pengukuran yang diperoleh ketika user-user telah terhubung ke jaringan namun *server* belum menjalankan program aplikasi sebagai berikut :

- Server Tradisional

  Memory : 11%
  CPU Usage : 0%
  Network : 0%

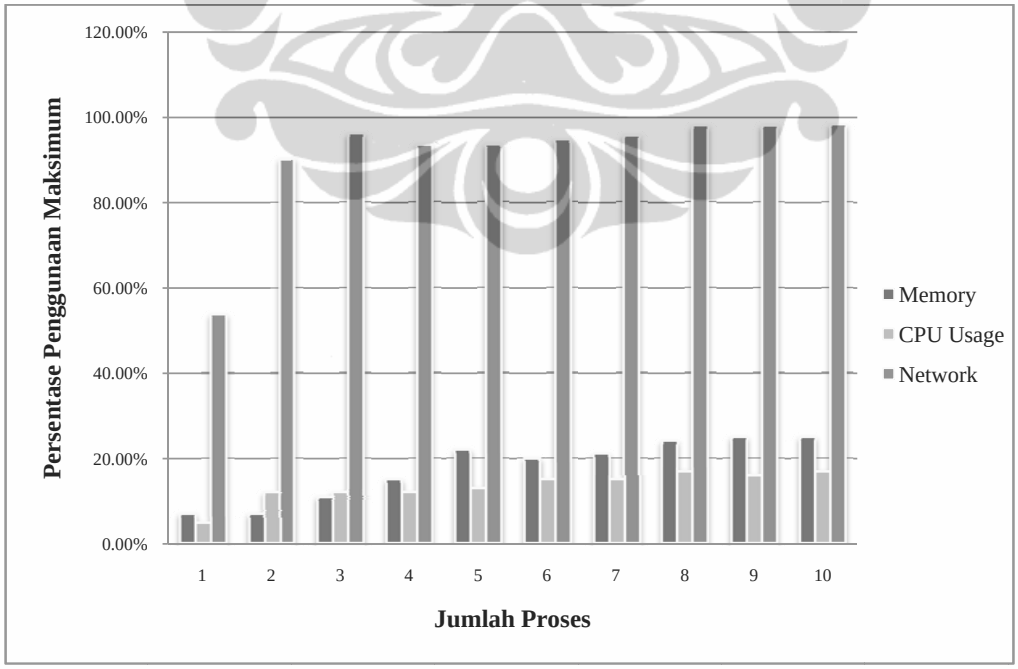
- Server Virtual

  Memory : 25%
  CPU Usage : 2%
  Network : 0%

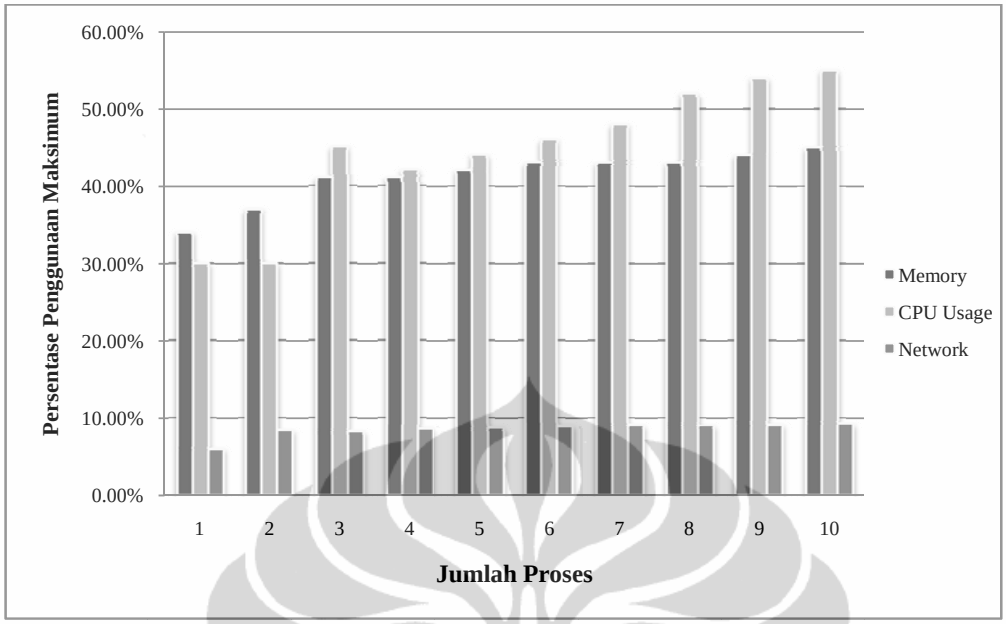
Tabel 4.8 Hasil pengukuran maksimum penggunaan perangkat keras pada Active Directory *Server*

Jumlah Proses	Server Tradisional			Server Virtual		
	Memory	CPU Usage	Network	Memory	CPU Usage	Network
1	7.00%	5.00%	53.76%	34.00%	30.00%	5.98%
2	7.00%	12.00%	90.23%	37.00%	30.00%	8.37%
3	11.00%	12.00%	96.05%	41.00%	45.00%	8.19%
4	15.00%	12.00%	93.52%	41.00%	42.00%	8.66%
5	22.00%	13.00%	93.55%	42.00%	44.00%	8.73%
6	20.00%	15.00%	94.56%	43.00%	46.00%	8.99%
7	21.00%	15.00%	95.65%	43.00%	48.00%	9.02%
8	24.00%	17.00%	97.80%	43.00%	52.00%	9.00%
9	25.00%	16.00%	98.00%	44.00%	54.00%	9.10%
10	25.00%	17.00%	98.20%	45.00%	55.00%	9.20%

Untuk pengukuran berikutnya dilakukan *copy* file sebesar 62 MB ke *server* yang dilakukan secara bertahap sebanyak lima kali pengambilan data untuk diambil nilai rata-ratanya. Tabel 4.8 menunjukkan hasil rata-rata pengambilan data yang telah dilakukan sesuai dengan jumlah *user* yang melakukan akses aplikasi pada *server* tradisional dan *server* virtual. Sedangkan grafik yang dihasilkan berdasarkan tabel di atas dapat dilihat pada gambar 4.4 dan gambar 4.5.



Gambar 4.5. Penggunaan Maksimum Sumber Daya pada Active Directory *Server* Tradisional



Gambar 4.6 Penggunaan Maksimum Sumber Daya pada Active Directory *Server* Virtual

4.2.2 Pengukuran parameter pada Exchange Server

Hasil pengukuran yang diperoleh ketika user-user telah terhubung ke jaringan namun *server* belum menjalankan program aplikasi sebagai berikut :

- Server Tradisional

  Memory : 1%

  CPU Usage : 12%

  Network : 0%

- Server Virtual

  Memory : 70%

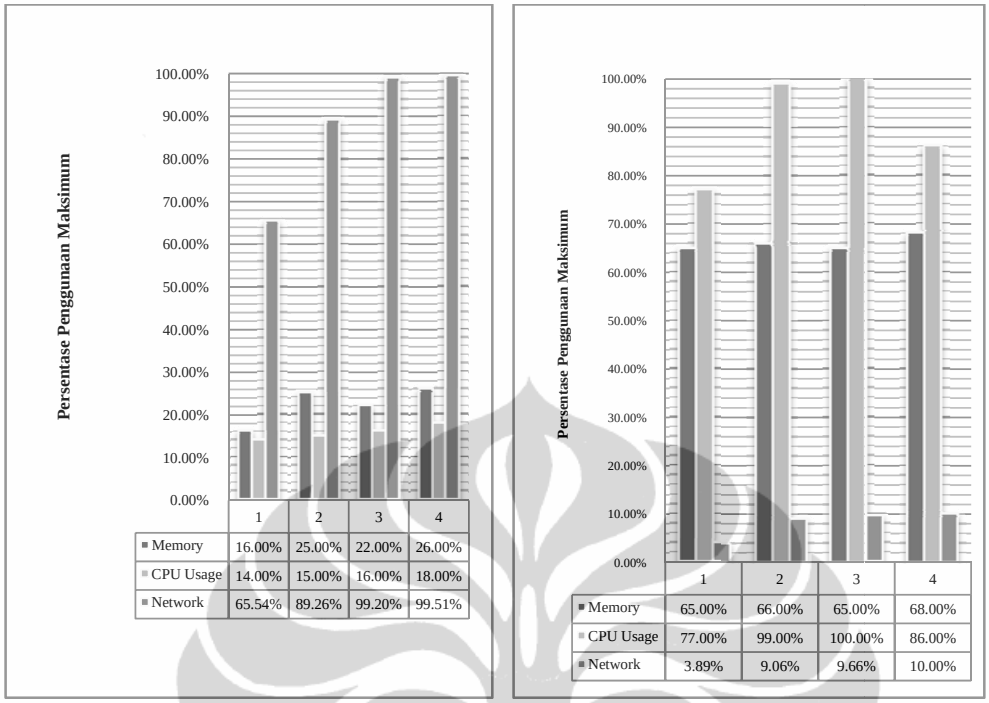
  CPU Usage : 2%

  Network : 0%

Tabel 4.9 menunjukkan hasil rata-rata pengambilan data yang telah dilakukan sesuai dengan jumlah proses aplikasi pada *server* tradisional dan *server* virtual

Tabel 4.9 Hasil pengukuran penggunaan maksimum perangkat keras pada Exchange *Server*

Jumlah User	Server Tradisional			Server Virtual		
	Memory	CPU Usage	Network	Memory	CPU Usage	Network
1	16%	14%	65.54%	65%	77%	3.89%
2	25%	15%	89.26%	66%	99%	9.06%
3	22%	16%	99.20%	65%	100%	9.66%
4	26%	18%	99.51%	68%	86%	10%



Gambar 4.7 *Send/Receive* email dengan attachment file 62 MB pada Exchange *Server*

Pengukuran penggunaan perangkat keras pada *server* Exchange dilakukan dengan *send/receive* email antar user dengan attachment file sebesar 62 MB secara bertahap dilakukan bersama sebanyak lima kali pengambilan data untuk diambil nilai rata-ratanya. Grafik yang dihasilkan berdasarkan tabel 4.9 dapat dilihat pada gambar 4.6.

4.2.3 Pengukuran parameter pada Database Server

Hasil pengukuran yang diperoleh ketika user-user telah terhubung ke jaringan namun *server* belum menjalankan program aplikasi sebagai berikut :

- Server Tradisional

  Memory : 1%

  CPU Usage : 12%

  Network : 0%

- Server Virtual

  Memory : 70%

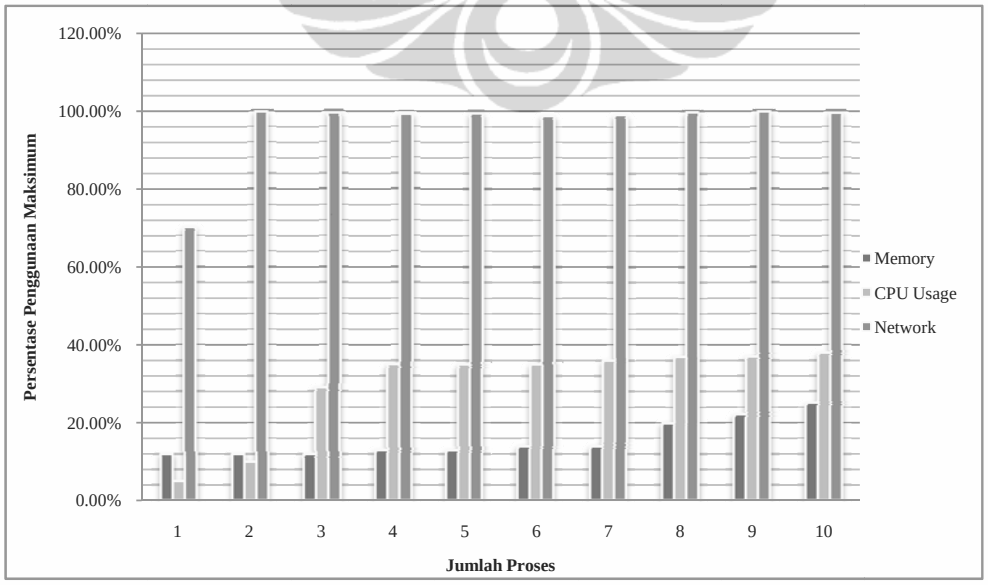
  CPU Usage : 2%

  Network : 0%

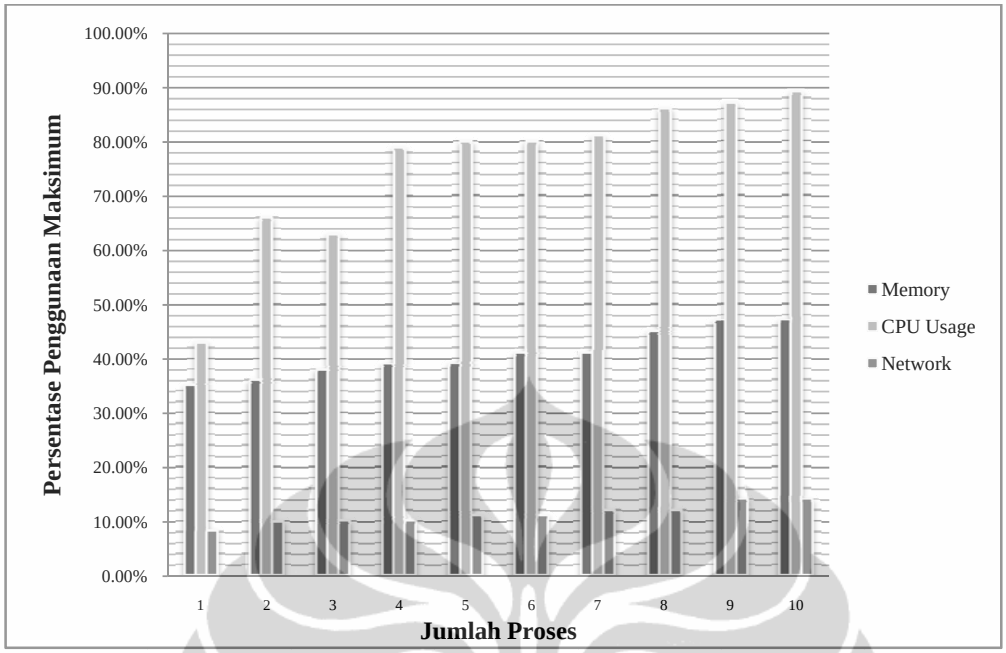
Tabel 4.10 Hasil pengukuran penggunaan maksimum perangkat keras pada SQL *Server*

Jumlah Proses	Server Tradisional			Server Virtual		
	Memory	CPU Usage	Network	Memory	CPU Usage	Network
1	12.00%	5.00%	70.05%	35.00%	43.00%	8.28%
2	12.00%	10.00%	99.84%	36.00%	66.00%	9.94%
3	12.00%	29.00%	99.51%	38.00%	63.00%	10.00%
4	13.00%	35.00%	99.15%	39.00%	79.00%	10.08%
5	13.00%	35.00%	99.40%	39.00%	80.00%	11.00%
6	14.00%	35.00%	98.80%	41.00%	80.00%	11.00%
7	14.00%	36.00%	99.02%	41.00%	81.00%	12.00%
8	20.00%	37.00%	99.68%	45.00%	86.00%	12.00%
9	22.00%	37.00%	99.77%	47.00%	87.00%	14.00%
10	25.00%	38.00%	99.55%	47.00%	89.00%	14.00%

Tabel 4.10 menunjukkan hasil rata-rata pengambilan data yang telah dilakukan sesuai dengan jumlah proses backup yang dilakukan pada *server* tradisional dan *server* virtual. Pengukuran penggunaan perangkat keras pada SQL *server* dilakukan dengan melakukan *backup* file database yang diakses dari *workstation* sebesar 2 GB bertahap dilakukan proses secara bersamaan sebanyak lima kali pengambilan data untuk diambil nilai rata-ratanya. Grafik yang dihasilkan berdasarkan tabel 4.10 dapat dilihat pada gambar 4.7 dan gambar 4.8.



Gambar 4.8 *Backup* file database sebesar 2 GB pada SQL *Server* Tradisional



Gambar 4.9 *Backup* file database sebesar 2 GB pada SQL *Server* Virtual

4.2.4 Analisis Data Hasil Pengukuran

Berdasarkan hasil pengukuran yang telah dilakukan diperoleh hasil sebagai berikut :

1. Persentase penggunaan maksimum perangkat keras *memory* pada server virtual lebih tinggi dibandingkan dengan server tradisional. Untuk pengukuran *server* virtual pada Active Directory *Server* diperoleh nilai berkisar 34% - 45%, Exchange *Server* 65% - 68%, dan SQL *server* 35% - 47%. Sedangkan pada *server* tradisional diperoleh hasil pengukuran berkisar antara 7% - 25% untuk Active Directory *server*, 16% - 26% untuk Exchange server, dan 12% - 25% untuk SQL *server*.
2. Persentase penggunaan maksimum perangkat keras CPU pada *server* virtual Active Directory berkisar 30% - 55%, 77% - 100% pada Exchange *server*, dan 43% - 89% pada SQL *server*. Sedangkan pada *server* tradisional diperoleh hasil pengukuran berkisar antara 5% - 17% untuk Active Directory *server*, 14% - 18% untuk Exchange *server*, dan 5% - 38% untuk SQL *server*.
3. Persentase penggunaan maksimum trafik jaringan pada server tradisional selalu lebih tinggi bahkan mendekati maksimum dibandingkan dengan server virtual. Pada *server* tradisional untuk Active Directory berkisar 53% - 99%, 65% - 100% pada Exchange *server*, dan 70% - 100% pada SQL *server*.

Sedangkan pada *server* virtual diperoleh hasil pengukuran berkisar antara 5% - 10% untuk Active Directory *server*, 3% - 13% untuk Exchange *server*, dan 8% - 14% untuk SQL server. Berdasarkan pengamatan diperoleh tipikal proses transaksi aplikasi berjalan berdasarkan kebutuhan. Beban puncak karena penggunaan sumber daya jaringan oleh aplikasi database tidak diikuti dengan naiknya penggunaan sumber daya yang lain secara signifikan (*memory* dan penggunaan CPU). Akibatnya terjadi pemakaian sumber daya yang berlebih pada jaringan dapat memberikan dampak yang signifikan pada sistem lain yang berjalan pada mesin yang sama. Kondisi seperti ini juga memberikan pengaruh pada waktu akses aplikasi yang dijalankan.
besar.

Dari penjelasan berdasarkan pengukuran di atas dapat dilihat bahwa *server* virtual mampu menggunakan *memory* dan CPU lebih efisien. Pada server virtual terdapat proses kontrol yang bersifat dinamik terhadap jumlah alokasi memory. Sistem akan mengatur alokasi *memory* ke mesin virtual secara otomatis bergantung pada beban sistem. Teknologi ini juga menyediakan kontrol dinamik terhadap *execution rate* dan *processor assignment* yang telah dijadwalkan. *Scheduler* ini melakukan *automatic load balancing* pada sistem *multiprocessor*. Berdasarkan hasil pengukuran, terutama untuk akses aplikasi database yang membutuhkan penggunaan sumber daya yang lebih signifikan, dapat dilihat bahwa secara aktual alokasi *CPU time* bervariasi secara dinamis sesuai dengan kebutuhan. Trafik jaringan berdasarkan hasil pengukuran relatif lebih stabil sehingga kemungkinan terjadinya saturasi terhadap pemakaian jaringan lebih kecil bila dibandingkan dengan *server* tradisional. Pemakaian sumber daya yang berlebih pada jaringan dapat memberikan dampak yang signifikan pada sistem lain yang berjalan pada mesin yang sama

Penggunaan trafik jaringan yang sangat tinggi dapat dilihat pada hasil pengukuran *server* tradisional terutama pada SQL *server*. Dalam hal ini terdapat perbedaan perlakuan proses antara kedua jenis server yang melibatkan *chaching* pada server. *Caching* adalah proses dimana database secara berkala melakukan akses record di dalam memory untuk meningkatkan kecepatan akses. *Caching* hanya melakukan *speeds up retrieval* informasi atau pembacaan database

(*database reads*). Sedangkan untuk penulisan database (*database write*) yaitu update sebuah record atau pembuatan record baru masih harus dilakukan penulisan melalui *cache* ke disk. Dengan demikian manfaat yang diperoleh dari kinerja hanya berlaku untuk suatu subset tugas database.

Caching juga tidak fleksibel, lokasi penyimpanan data akan dipilih secara otomatis biasanya berdasarkan beberapa varian misalnya algoritma *most-frequently-used* atau *least-frequently-used.* User tidak dapat menentukan record-record tertentu yang penting untuk selalu di-*cached.* Secara tipikal tidak dimungkinkan untuk melakukan *cache* terhadap keseluruhan database. Sebagai tambahan, pengaturan *chache* membebani substansi *overhead.* Untuk memilih, menambahkan, ataupun menghapus record dari *chache* algoritma tersebut menggunakan *memory* dan CPU *cycles*. Ketika *buffer chache memory* diisi, beberapa bagian data ditulis ke sistem file (*logical I/O*). Setiap *logical I/O* membutuhkan interval waktu yang diukur dalam microdetik. Akhirnya ketika *buffer* sistem file juga mulai diisi maka data harus ditulis ke hard disk (pada titik dimana logical I/O menjadi fisikal I/O). Fisik I/O diukur dalam milidetik, sehingga kinerja beban adalah beberapa kali lipat lebih besar dari logika I/O.

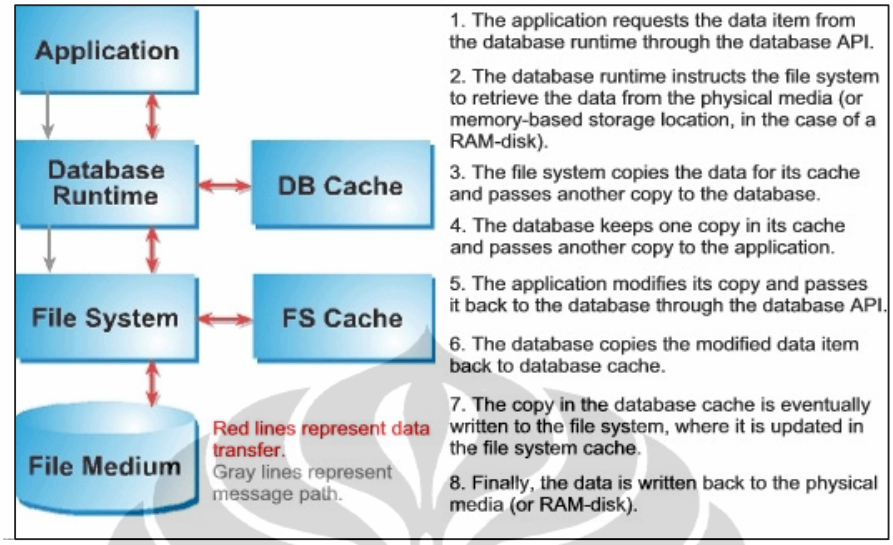
Cache dapat juga dibangun secara langsung pada peripheral. Hard disk modern sudah memiliki *memory* yang cepat dan disatukan pada hard disk. Komputer tidak secara langsung menggunakan *memory* ini namun dilakukan oleh *hard-disk controller*. Bagi komputer, *chip memory* ini adalah disk itu sendiri. Ketika komputer meminta data pada hard disk, *hard-disk controller* memeriksa ke dalam *memory* ini sebelum berpindah ke bagian mekanik hard disk yang sangat lambat bila dibandingkan dengan *memory.* Jika *hard-disk controller* menemukan data yang diminta pada *cache*, maka *hard-disk controller* akan kembali ke data yang disimpan dalam *cache* tanpa secara aktual mengakses data pada disk sehingga menghemat waktu. Sebagai gambaran, berikut ini adalah daftar sistem *caching* normal :

- **L1 cache** – Akses Memory pada kecepatan penuh mikroprosesor (10 nanoseconds, 4 kilobytes hingga 16 kilobytes in size)
- **L2 cache** – Akses Memory dari jenis SRAM (berkisar antara 20-30 nanoseconds, 128 kilobytes to 512 kilobytes in size)

- **Main memory** - Akses Memory dari jenis RAM (berkisar antara 60 nanoseconds, 32 megabytes to 128 megabytes)
- **Hard disk** - Mekanikal, lambat (berkisar antara 12 milliseconds, 1 gigabyte hingga 10 gigabytes)
- **Internet** – Sangat lambat (antara 1 second dan 3 days, ukuran tidak terbatas)

Pada model *client/server database* seperti yang dilakukan pada percobaan mengacu kepada sistem database yang menggunakan program perangkat lunak terpisah secara *dedicated* yang disebut *database server*, diakses oleh aplikasi pada *client* melalui *Interface Interprocess Communication* (IPC) atau *Remote Procedure Call* (RPC). RPC diinisiasi oleh *client* yang mengirimkan sebuah *request message* ke remote *server* yang telah dikenali sebelumnya dalam rangka melakukan eksekusi prosedur tertentu menggunakan parameter yang disediakan. Respon dikembalikan ke *client* ketika aplikasi melanjutkan proses. Proses dari *request* dilakukan berdasarkan bobot dari setiap *request client*. Segera setelah *server* menyelesaikan seluruh proses komputasi dan selanjutnya mengirimkan hasilnya melalui jalur komunikasi sesuai dengan *client* yang dituju. Pada gilirannya menjadi tugas *client* untuk menutup jalur komunikasi dan melanjutkan proses komputasi berdasarkan informasi dari *server* yang telah tersedia. Sistem database yang memberlakukan model *client/server* menyediakan pula *remote interfaces* yang memfasilitasi akses database ditempatkan pada node lain dari jaringan. Dengan demikian implementasi sistem model *client/server database* pada LAN *server* tradisional menjadikan penggunaan jaringan menjadi sangat signifikan.

Berbeda dengan server tradisional yang menggunakan fisikal disk drive, server virtual menggunakan file yang dikenal juga sebagai *virtual hard disk* (VHD) untuk menyimpan sistem dan file user bagi sistem operasi *guest* berlokasi pada disk drive yang terhubung ke sistem host. VHD sesungguhnya hanya berupa sebuah file, namun bagi mesin virtual akan tampak seperti disk drive yang terhubung langsung. Sistem operasi *guest* dan aplikasi-aplikasinya menggunakan satu atau lebih VHD untuk penyimpanan. Aplikasi disimpan pada *server* yang dijalankan pada mesin sentral. Ketika aplikasi mulai dijalankan bisa jadi



Gambar 4.10 Data transfer pada sebuah *on-disk* sistem database [17]

bersamaan dengan aplikasi virtual dan non virtual yang lain dalam satu mesin. Setelah selesai aplikasi disimpan pada *cache* mesin server. Untuk penggunaan di masa mendatang aplikasi dijalankan mengacu kepada data yang disimpan pada *chache.* Dengan demikian pada *server* virtual keseluruhan proses dilakukan di *server*, sedangkan *client* melakukan *downloading* dari *server* ke sistem *user* sesuai kebutuhan.

Berdasarkan hasil pengukuran parameter-parameter QoS pada streaming audio dan video conference dapat dilihat bahwa *troughput* pada *server* virtual lebih baik dibandingkan dengan server tradisional. Untuk menjelaskan hal tersebut, terdapat beberapa pertimbangan aspek fisikal yang perlu diketahui dalam kaitannya dengan optimalisasi jaringan yaitu :

1. Koneksi antara *client* dan *server* dalam jaringan lokal yang dihubungkan oleh *switch* yang sama. Jika *client* dan *server* menggunakan *switch* yang berbeda maka perlu dipertimbangkan berapa banyak hop jaringan yang dibutuhkan oleh paket untuk mencapai kedua sistem tersebut.
2. Jenis NIC yang digunakan oleh kedua mesin untuk saling berinteraksi. NIC untuk sekelas *server* umumnya menawarkan kinerja yang lebih baik.
3. Ukuran paket yang ditransmisikan melalui network.

Troughput jaringan sangat bergantung kepada konfigurasi jaringan dan spesifikasi aplikasi. Pada *physical environment*, penggunaan CPU memainkan peran yang signifikan untuk mencapai *troughput* jaringan yang diharapkan. Agar dapat menghasilkan *troughput* yang lebih tinggi dibutuhkan lebih banyak sumber daya CPU. Dampak dari ketersediaan sumber daya CPU bagi *throughput* jaringan aplikasi virtual juga memainkan peran yang penting. Menjalankan server virtual membutuhkan jumlah tertentu sumber daya CPU secara tetap bergantung pada konfigurasi server. Pada hasil percobaan dapat dilihat bahwa penggunaan CPU pada server virtual lebih tinggi dibandingkan dengan server tradisional. Hal ini disebabkan fitur yang disediakan oleh teknologi virtual memungkinkan untuk melakukan alokasi sumber daya secara dinamis lebih fleksibel sesuai dengan kebutuhan aplikasi yang sedang dijalankan.

Perencanaan *troughput* jaringan bagi aplikasi virtual sama dengan perencanaan *troughput* jaringan aplikasi-aplikasi yang dijalankan pada sistem fisikal. Sama halnya dengan fisikal *platform,* seluruh kinerja sistem bergantung kepada elemen-elemen fisik jaringan, implementasi *stack* jaringan, aplikasi yang dijalankan dan trafik jaringan. Penggunaan CPU yang lebih tinggi dibutuhkan oleh transaksi jaringan dalam menjalankan aplikasi virtual.

Infrastruktur virtual memungkinkan implementasi konfigurasi jaringan secara transparan. Ketika suatu beban kerja dijalankan di dalam sebuah mesin virtual, maka beban kerja menggunakan mekanisme jaringan yang sama dengan jika dijalankan pada mesin fisikal menggunakan fisikal NIC. Maksimum *troughput* yang dapat diperoleh mesin virtual dapat dibandingkan kinerjanya dengan *troughput* pada server fisikal.

Issue lain yang mempengaruhi *troughput* jaringan melibatkan pola penggunaan jaringan. Meskipun trafik digenerasikan oleh aplikasi yang sama namun hasilnya dapat berbeda bergantung kepada *user* dan waktu aksesnya. Misalnya, untuk aplikasi CRM bisa menghasilkan *steady stream* dalam bentuk paket berukuran kecil ketika digunakan oleh seorang *sales representative*. Namun trafik bisa menjadi *bursty* dan berisi paket berukuran besar ketika digunakan untuk membuat *activity reports.* Beberapa hal yang menjadi pertimbangan pola trafik adalah :

1. Frekuensi transaksi yang menyebabkan paket menjadi berukuran besar.
2. Ukuran paket data.
3. Sensitifitas *data loss.* Misalnya, aplikasi multimedia *streaming* menggunakan UDP tetap masih dapat dimengerti oleh user walaupun terjadi *data loss*.
4. *Traffic Directiveness,* seringkali lebih banyak data yang ditransmisikan *downstream* (dari *server* ke *client*) daripada *upstream.*

Pada akhirnya implementasi *protocol stack* jaringan dalam sistem operasi dan kinerja aplikasi dalam melakukan proses transaksi jaringan seringkali mempengaruhi keseluruhan kinerja jaringan. Dengan NIC dan *switches* yang tersedia di pasaran hingga mencapai kecepatan 10 Gbps, *bottleneck* dalam melakukan proses transaksi jaringan sering kali terjadi diantara *available CPU cycles* dan sistem *memory,* dimana proses transaksi terjadi pada level aplikasi atau eksekusi sistem operasi TCP/IP *stack*. Pertimbangannya adalah ukuran *buffer* jaringan yang menentukan berapa banyak paket yang dapat di-*queued* untuk dikirim atau diterima.