PETIR
JURNAL
PENGKAJIAN DAN PENERAPAN
TEKNIK INFORMATIKA

VOLUME 8 - NOMOR 1 | MARET 2015 | ISSN 1978-9262

ISSN 1978-9262
771978 926272

SEKOLAH TINGGI TEKNIK - PLN (STT-PLN)					
PETIR	VOL. 8	NO. 1	HAL. 1 - 132	JAKARTA, MARET 2015	ISSN 1978-9262

IMPLEMENTASI DAN PENGUJIAN KINERJA ORACLE 10g REAL APLICATION CLUSTER (RAC) PADA SISTEM OPERASI SUN SOLARIS 10

Gatot Budi Santoso & Yanuar Indra Wirawan
Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Trisakti
Jl Kiai tapa No 1 Grogol, jakarta Barat 11410
e-mail : bulish@gmail.com, gbs@trisakti.ac.id

Abstract

During the implementation of database system, needs of getting fast, accurate and high-availability data, has become major clauses for the implementation of "high-availability" system. Oracle 10g RAC is a clustered database solution that requires a two or more node hardware configuration capable of working together under a clustered operating system. A clustered hardware solution is managed by cluster management software, usually provided by the hardware vendor. The operating system is also responsible for providing access to the shared disk subsystems. Parameters that used for the performance testing and analyzing of Oracle 10g Real Application Cluster are the cpu idle, free memory, SGA, buffer cache, and shared pool. As result for the performance testing and analyzing, it can be concluded that Oracle 10g Real Application Cluster was really good for the implementation of "high-availability system" concept. But it all depends to the analysis for its system.

Keywords: *RAC, cluster, instance, shared disk subsystems.*

Abstrak

Di dalam implementasi sistem database, kebutuhan akan data yang cepat, akurat dan dengan tingkat ketersediaan yang tinggi menjadi sebuah persyaratan utama untuk penerapan konsep "high availability system". Oracle 10g Real Application Cluster (RAC) adalah sebuah solusi database ter-cluster yang memerlukan dua atau lebih konfigurasi node hardware dan dapat bekerja sama di bawah sebuah sistem operasi yang ter-cluster. Sebuah solusi hardware diatur oleh software manajemen cluster, yang biasanya diberikan oleh vendor hardware tersebut. Sistem operasi juga bertanggung jawab dalam memberikan akses terhadap media penyimpanan yang dipakai bersama-sama. Terdapat beberapa yang digunakan dalam pengujian dan analisa kinerja Oracle 10g Real Application Cluster ini yaitu penggunaan cpu idle, free memory, SGA, buffer cache, dan shared pool. Sebagai hasil dari pengujian dan analisa kinerja, dapat disimpulkan bahwa pada dasarnya Oracle 10g Real Application Cluster sangat baik untuk penerapan konsep "high-availability system". Akan tetapi itu semua sangat bergantung pada baik buruknya analisis kinerja sistem itu sendiri.

Kata kunci *: RAC, cluster, instance, shared disk subsystems*

1. Pendahuluan

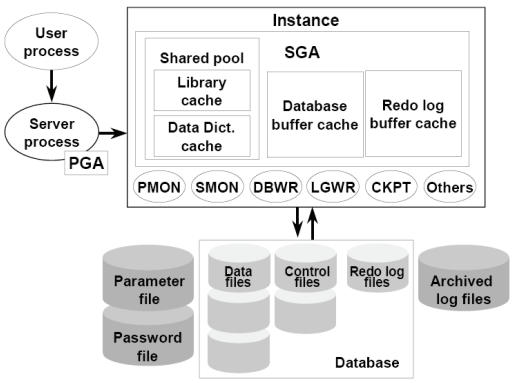
Efektifitas, efisiensi, dan produktivitas sumber daya manusia merupakan tantangan utama yang dihadapi oleh suatu organisasi. Suatu perusahaan terkadang membutuhkan suatu sistem komputasi dengan tingkat ketersediaan yang tinggi atau yang lebih dikenal dengan sebutan *"high-availability system"*. Apabila sistem komputasi ini mengalami kegagalan dan membutuhkan waktu dalam hitungan jam atau hari untuk bisa kembali beroperasi,maka perusahaan tersebut akan mengalami kerugian dengan jumlah besar. Sistem komputasi tersebut tidak dapat bertoleransi terhadap kegagalan walaupun hanya membutuhkan waktu 1 (satu) detik saja untuk kembali beroperasi. Terlebih lagi di dalam implementasi sistem database, kebutuhan akan data yang cepat, akurat dan

dengan tingkat ketersediaan yang tinggi menjadi sebuah persyaratan utama untuk penerapan konsep *"high availability system"*.

Ada 3 (tiga) istilah kunci yang sering dipergunakan di dalam model *relational database*: *relation*, *attribute*, dan *domain* [Alapati 2005]. Sebuah *relation* adalah sebuah tabel dengan kolom dan baris. Kolom-kolom yang ada di dalam tabel disebut *attribute*, dan *domain* adalah nilai-nilai yang diperbolehkan untuk diisikan ke dalam *attribute*. Struktur dasar data dari model *relational* adalah tabel, dimana informasi suatu entitas tertentu direpresentasikan di dalam kolom dan baris (disebut juga *tuple*). Dan *"relation"* di dalam *"relational database"* mengacu kepada berbagai tabel di dalam *database*; sebuah *relation* adalah sebuah himpunan yang terdiri dari *tuple-tuple*. Kolom-kolom tersebut menurunkan berbagai *attribute* dari entitas, dan baris-baris tersebut adalah *instance* dari entitas yang direpresentasikan oleh *relation*.

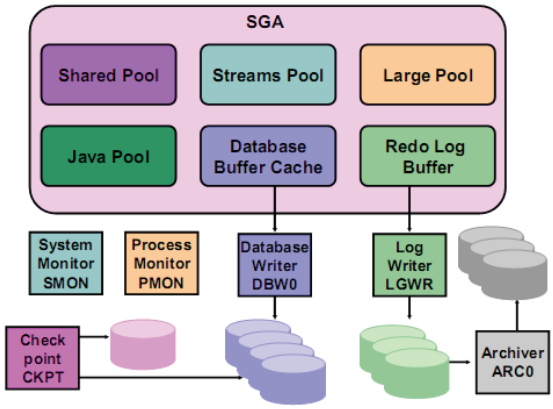
Oracle Database 10g adalah sekian banyak contoh dari teknologi *Relational Database Management System* (RDBMS), walaupun *Oracle* lebih suka menyebutnya sebagai *Object Relational Database Management System* (ORDBMS) dikarenakan telah menerapkan konsep penggabungan dari model desain *object-oriented* dengan model *relational* [Alapati 2005].

Oracle 10g Database Server memiliki arsitektur yang terdiri dari beberapa komponen utama, pertama struktur fisik merupakan struktur yang berbentuk file-file yang ada di dalam sistem operasi seperti *control, data, redo log, archived redo log, parameter, password*. Kedua struktur memory terdiri dari 2 (dua) area memory, yaitu *System Global Area* (SGA) dan *Program Global Area* (PGA). Dan ketiga masing-masing struktur proses yang terdiri dari user, server dan background



Gambar 1. Arsitektur *Oracle Instance*

Oracle Instance terdiri dari struktur *memory System Global Area* (SGA) dan proses-proses *background* yang digunakan untuk mengatur database [Alapati 2005]. Sebuah *instance* dapat berjalan dan digunakan hanya untuk satu database. Bagan arsitektur *Oracle Instance* dapat dilihat pada gambar 2 berikut.



Gambar 2. Arsitektur Oracle Instance

Struktur *memory* di dalam *instance* terdiri dari sebuah *System Global Area* (SGA). SGA terbagi menjadi 3 (tiga) komponen utama dan 3 (tiga) komponen *optional* [Alapati 2005]. Komponen-komponen tersebut adalah:

1. *Shared Pool*. Komponen *memory* ini digunakan untuk menyimpan *statement* SQL dan *data definition* yang terakhir digunakan. *Shared Pool* terbagi menjadi 2 (dua):
   a. *Library Cache*. *Memory* ini digunakan untuk menyimpan informasi mengenai SQL dan PL/SQL yang terakhir digunakan. *Library cache* terbagi lagi menjadi 2 (dua), yaitu *Shared SQL area* dan *Shared PL/SQL area*.
   b. *Data Dictionary Cache*. *Memory* ini digunakan untuk menyimpan *database object definition* yang terakhir digunakan.
2. *Database Buffer Cache*. Komponen *memory* ini bertugas sebagai tempat penyimpan sementara *data block* yang diambil dari *data files*.
3. *Redo Log Buffer Cache*. Komponen *memory* ini digunakan untuk menyimpan semua aksi dalam perubahan yang dilakukan terhadap *data block*. Data-data ini meliputi data sebelum dan sesudah diadakannya perubahan terhadap *data block*.

4. *Large Pool*. Komponen *memory* ini merupakan komponen yang bersifat *optional*. Komponen tersebut digunakan untuk operasi *backup* dan *restore*, I/O *server process*, dan *session memory* untuk *shared server*.
5. *Java Pool*. Komponen *memory* ini adalah komponen yang bersifat *optional* dan digunakan untuk semua *session* terhubung yang menggunakan kode *Java*.
6. *Streams Pool*. Komponen *memory* ini digunakan oleh *Oracle Stream* yang juga merupakan komponen bersifat *optional*..

Real Application Cluster (RAC) dapat diartikan sebagai penambahan dari konfigurasi *single-instance* yang biasa [Vallath 2003]. Dalam konsepnya, hal ini benar adanya karena RAC adalah sebuah komposisi dari beberapa *instance* Oracle. Tetapi ada banyak perbedaan di dalam manajemen komponen, proses *background* tambahan, *file-file* tambahan, dan penggunaan *resource* bersama oleh *instance-instance* terkait, belum lagi perbedaan komponen yang digunakan di dalam sistem operasi untuk mendukung lingkungan *hardware* yang ter-*cluster*. Perbedaan utama antara sebuah *database* dan *instance* sangat terlihat di dalam konfigurasi RAC [Vallath 2006]. Sebagai contoh, apabila di dalam konfigurasi *database single-instance*, nama *instance* dan *database* akan diidentifikasikan dengan nama yang sama. Tetapi di dalam konfigurasi RAC, nama tersebut akan diidentifikasikan dengan nama yang berbeda.

2. Metodologi Penelitian

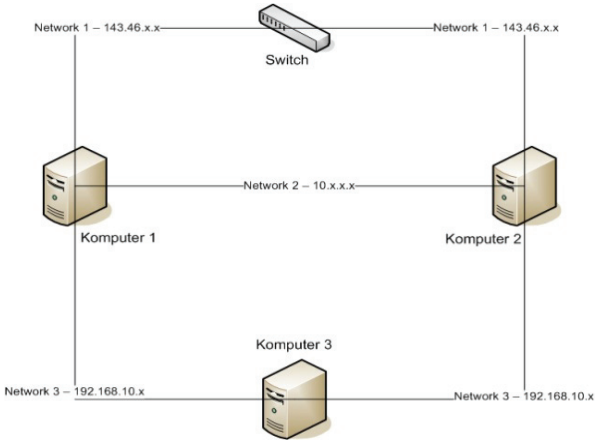
2.1 Analisis Kebutuhan

Oracle 10g Real Application Cluster (RAC) dalam implementasinya memiliki persyaratan utama yang harus dipenuhi. Persyaratan ini meliputi aspek perangkat keras, perangkat jaringan, sistem operasi, dan media penyimpanan..

Sebuah sistem RAC dalam implementasinya membutuhkan paling sedikit 2 (dua) buah komputer yang bertindak sebagai *node-node* dari RAC itu sendiri. Dan komputer-komputer tersebut harus mempunyai spesifikasi yang identik, agar sistem RAC yang berjalan di atasnya dapat berjalan dengan seimbang [Vallath 2003]. Komputer-komputer yang dijadikan *node-node* RAC itu sendiri juga harus memenuhi persyaratan minimal sebagai berikut [Whalen 2005] yaitu RAM dengan ukuran 512 MB, *Swap space* berukuran 1 GB (atau dua kali ukuran RAM), ruang sebesar 400 MB pada direktori /tmp dan terakhir ruang *disk* hingga mencapai 4 GB untuk instalasi perangkat lunak *Oracle*, tergantung pada tipe instalasi.

Sistem RAC tidak dapat diinstalasikan di semua sistem operasi. Khususnya untuk *Oracle 10g* RAC *release 2*, hanya beberapa sistem operasi saja yang dapat menjalankannya. Beberapa sistem operasi yang didukung oleh *Oracle 10g* RAC *release 2* adalah [Whalen 2005] yaitu pertama *Red Hat Enterprise Linux 3 update 4* dan *Red Hat Enterprise Linux 4* atau versi yang lebih baru, kedua *Suse Linux Enterprise Server 9.0* atau versi yang lebih baru, ketiga *Solaris 10* atau versi yang lebih baru, dan keempat *Windows Server 2000* atau versi yang lebih baru. Sistem RAC membutuhkan sebuah mekanisme media penyimpanan data yang dapat dipergunakan oleh seluruh *node* RAC secara bersama-sama. Media penyimpanan ini disebut *shared storage* [Whalen 2005]. Implementasi *shared storage* dapat menggunakan konsep *Network Attached Storage* (NAS) ataupun *Storage Area Network* (SAN). Dan sebagai persyaratan utama, ada 3 (tiga) partisi *disk* yang harus disediakan dengan fungsi-fungsi sebagai media penyimpanan untuk file-file OCR, voting disk dan data base [Vallath 2006]:

Jaringan yang digunakan penulis menggunakan *protocol* TCP/IP. Kelas alamat IP yang digunakan adalah kelas C. Untuk keperluan implementasi sistem, penulis menggunakan 3 (tiga) *network* ID. Masing-masing *network* ini tidak terhubung satu sama lain untuk meminimaliskan gangguan yang mungkin terjadi. Desain jaringan yang digunakan penulis dapat dilihat pada gambar 3 berikut.



Gambar 3. Desain Jaringan Implementasi

Pada gambar 3 di atas jelas terlihat bahwa *network* pertama menggunakan *network* ID 143.46.x.x, *network* kedua menggunakan *network* ID 10.x.x.x dan *network* ketiga menggunakan *network* ID 192.168.10.x. Pada gambar 3 juga dapat dilihat bahwa penulis menggunakan 3 (tiga) buah komputer. Komputer 1 dan 2 merupakan *node-node* dari RAC, dan komputer 3 digunakan untuk menjadi *shared storage*.

Spesifikasi dari komputer-komputer tersebut ialah pertama komputer 1 dan 2 adalah komputer komputer rakitan yang identik dengan spesifikasi masing-masing (Prosesor : AMD Athlon XP 1500+, *Memory* : 768 MB, *Harddisk* : 15 GB dan kartu jaringan : 3 buah). dan kedua komputer 3 merupakan sebuah komputer rakitan yang memiliki spesifikasi:(Prosesor : AMD Duron 1000, *Memory* : 512 MB, *Harddisk* : 40 GB dan Kartu jaringan : 2 buah). Untuk menghubungkan komputer-komputer yang ada, penulis menggunakan kabel UTP *Category* 5. Penulis juga menggunakan sebuah *switch* 8 *port*, sehingga komputer-komputer yang ada di *network* pertama dapat terhubung ke *client*.

Dalam implementasi sistem RAC, banyak alternatif pilihan perangkat lunak yang dapat digunakan. Untuk kebutuhan mendasar, penulis harus memilih sistem operasi yang akan digunakan, karena pemilihan perangkat lunak RAC bergantung pada *platform* sistem operasi yang mendukungnya. Dengan analisis yang telah dilakukan, akhirnya penulis memilih *Solaris* 10 *update* 11/06 sebagai sistem operasi yang digunakan untuk komputer *node-node* RAC yaitu komputer 1 dan 2 dengan dasar pertimbangan diantaranya adalah sistem operasi ini bersifat gratis, pengoperasiannya relatif lebih mudah dan buku manual sangat lengkap dapat diunduh dari situs pengembangnya (www.sun.com).

Untuk kebutuhan *shared storage* dengan *Network File System* (NFS) yang digunakan pada komputer 3, penulis memilih untuk menggunakan sistem operasi *Red Hat Linux 9.0* dan dengan dasar pertimbangan diantaranya adalah sistem operasi ini bersifat gratis, membutuhkan *resource* perangkat keras komputer yang relatif lebih kecil dan memberikan kompatibilitas yang baik dengan perangkat keras yang digunakan. Untuk implementasi sistem RAC, penulis menggunakan perangkat lunak *Oracle Clusterware 10.2.0.2* dan *Oracle Database 10.2.0.2* untuk *Solaris* x86. Kedua perangkat lunak ini merupakan *release 2* dari *Oracle 10g*.

2.2 . Parameter Kinerja

Pengujian kinerja suatu sistem RAC melibatkan beberapa langkah terpisah yang menghasilkan suatu hasil akhir. Hasil akhir pengujian tersebut dapat berupa informasi ataupun rekomendasi bagaimana parameter-parameter hasil akhir seharusnya bernilai. Langkah-langkah pengujian kinerja yang dilakukan penulis terbagi menjadi penetapan parameter uji, pengujian dan pemonitoran parameter uji serta terakhir analisa hasil pengujian

Penetapan parameter-parameter dalam pengujian kinerja suatu sistem sangatlah penting, karena parameter-parameter tersebut akan menjadi dasar dalam melakukan analisa kinerja. Penetapan parameter pengujian kinerja mempunyai istilah "*Performance Metric*" [Whalen 2005]. Metric dalam hal ini adalah nilai-nilai yang dapat diukur sebagai takaran bagaimana baik dan buruknya kinerja suatu sistem. Ada dua macam metrik yang digunakan dalam penelitiaan ini yaitu sistem operasi dan database. Sistem operasi terdiri dari utilisasi CPU dan memori sementara database terdiri atas penggunaaan SGA, buffer cache dan share pool

Dalam melakukan pengujian kinerja RAC, penulis mempertimbangkan hal-hal apa saja yang dapat mempengaruhi metric-metric yang telah ditetapkan sebagai parameter uji. Berdasarkan teori yang telah dibahas sebelumnya, metric-metric yang telah ditetapkan, secara keseluruhan sangat dipengaruhi oleh perintah-perintah SQL (seperti "*select*", "*insert*", "*update*", dan "*commit*") dan banyaknya jumlah data. Maka untuk keperluan pengujian dan analisa penulis melakukan pengujian dengan membandingkan hasil uji sistem pada saat *idle* dan pada saat diberikan beban kerja berupa pelaksanaan perintah-perintah SQL yang mempengaruhi *metric-metric* yang telah ditetapkan..

Penulis menggunakan *tool-tool* dari sistem operasi dan *Oracle* untuk mendapatkan data-data *metric* hasil pengujian. *Tool* sistem operasi yang digunakan oleh penulis yaitu vmstat. *Tool* tersebut digunakan untuk membuat statistik dari penggunaan *cpu* dan *memory* oleh RAC. Untuk mendapatkan data-data *metric database* hasil pengujian, penulis meng-*generate Automatic Workload Repos-*

itory (AWR) dari masing-masing *instance* yaitu rac1 dan rac2.

CPU *idle* dan *free memory* menggambarkan keadaan sistem bila dilihat dari CPU *utilization* dan *memory utilization*. Hal ini menunjukkan besarnya penggunaan *resource* CPU dan *memory* dari semua proses yang berjalan di dalam sistem operasi. Analisa terhadap CPU *idle* dan *free memory* yang digunakan oleh semua proses *Oracle* sangatlah penting, karena pertama apabila CPU *idle* jatuh hingga ke angka 0% dalam kurun waktu yang lama akan menyebabkan sistem menjadi *crash* dan kedua apabila *free memory* jatuh hingga habis, akan menyebabkan CPU *idle* jatuh pula.

Analisa terhadap *shared pool* merupakan prioritas utama karena apabila *shared pool* mengalami *load* ulang ke dalam *memory* terhadap perintah yang pernah dieksekusi (*cache miss*), dampak terhadap performa keseluruhan akan lebih terasa bila dibandingkan dengan *cache miss* yang dialami oleh *buffer cache*.

Dalam pelaksanaan analisa terhadap *shared pool*, penulis berkonsentrasi pada *library cache*, karena secara algoritma data yang disimpan di dalam *library cache* akan berumur lebih pendek bila dibandingkan dengan *data dictionary cache*. Sehingga apabila *tuning* terhadap *library cache* untuk mendapatkan rasio *cache hit* (kebalikan dari *cache miss*) yang bagus sudah terpenuhi, maka rasio *cache hit* dari *data dictionary cache* pasti terpenuhi juga. Apabila ukuran *shared pool* terlalu kecil, *server* harus mendedikasikan sumber daya yang ada karena terbatasnya *space* yang tersedia. Hal ini menyebabkan pengkonsumsian CPU yang lebih besar.

Oracle memberikan syarat rasio untuk *library cache* (*library hit*) yang bagus adalah dengan nilai di atas 90%. Rasio ini dapat dilihat pada AWR seperti berikut:

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.85	In-memory Sort %:	100.00
Library Hit %:	91.77	Soft Parse %:	93.62
Execute to Parse %:	-0.36	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	38.76	% Non-Parse CPU:	92.01

Shared Pool Statistics	Begin	End
Memory Usage %:	78.06	81.27
% SQL with executions>1:	53.33	76.97
% Memory for SQL w/exec>1:	90.18	92.47

Untuk pengukuran efisiensi dari *buffer cache*, penulis juga menggunakan AWR yang terlihat sebagai berikut:

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.85	In-memory Sort %:	100.00
Library Hit %:	91.77	Soft Parse %:	93.62
Execute to Parse %:	-0.36	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	38.76	% Non-Parse CPU:	92.01

Shared Pool Statistics	Begin	End
Memory Usage %:	78.06	81.27
% SQL with executions>1:	53.33	76.97
% Memory for SQL w/exec>1:	90.18	92.47

Oracle juga memberikan syarat rasio yang bagus dari *buffer cache* (*buffer hit*) harus bernilai di atas 90%.

2.3 Metode Pengujian

Seperti yang telah disebutkan sebelumnya, analisis kinerja yang dilakukan dapat berupa pengukuran terhadap *metric-metric* yang diujikan. Metode pengujian kinerja itu sendiri bisa dilakukan dengan cara membandingkan nilai-nilai *metric* pada saat RAC bekerja dan pada saat *idle*. Untuk itu penulis membuat dua skenario pengujian. Skenario pertama dilakukan pada saat RAC bekerja, dan skenario kedua dilakukan pada saat RAC dalam keadaan *idle*. Persiapan yang penulis lakukan dalam melakukan pengujian kinerja dari sistem RAC yang telah diimplementasikan ialah:

1. Membuat sebuah tabel. Tabel ini diberi nama "indra" dan memiliki dua kolom. Kolom pertama diberi nama "satu" dengan tipe data number dan berjumlah 10 karakter. Kolom kedua diberi nama "dua" dengan tipe data varchar2 dan berjumlah 10 karakter.
2. Menyiapkan perintah operasi SQL "*insert*", "*update*" dan "*select*" pada tabel yang telah dibuat. Perintah-perintah tersebut dibuat ke dalam 4 (empat) *file* ber-*extension* .sql yang akan dijalankan di sebuah komputer *client*. Isi dari *file-file* tersebut adalah:
   a. *File* insert.sql, berisikan perintah:
      "insert into indra values(1,'1');"
      "commit;"
      yang diulang sebanyak 1000 (seribu) kali.
   b. *File* update.sql, berisikan perintah:
      "update indra set satu=2;"
      "commit;"

yang juga diulang sebanyak 1000 (seribu) kali.

c. *File* select.sql, berisikan perintah:
"select * from indra;"
yang juga diulang sebanyak 1000 (seribu) kali.

d. *File* kombinasi.sql, berisikan perintah:
"insert into indra values(1,'1');"
"commit;"
"update indra set satu=2;"
"commit;"
"select * from indra;"
"commit;"
yang juga diulang sebanyak 1000 (seribu) kali.

3. Membuat koneksi menggunakan komputer *client* dengan konfigurasi TNSNames:

```
SRVINDRA1 =
 (DESCRIPTION =
  (ADDRESS =
   (PROTOCOL = TCP)
   (HOST = 143.46.43.100)
   (PORT = 1521)
  )
  (ADDRESS =
   (PROTOCOL = TCP)
   (HOST = 143.46.43.101)
   (PORT = 1521)
  )
  (LOAD_BALANCE = yes)
  (CONNECT_DATA =
   (SERVER = DEDICATED)
   (SERVICE_NAME = srvindra1)
  )
 )
```

Lalu untuk membedakan antara sistem pada saat dilakukan operasi SQL dan pada saat *idle*, penulis melakukan penjadwalan pengujian. Penjadwalan yang dilakukan ialah satu jam pertama untuk pengujian sistem dengan menjalankan perintah SQL yang telah dibuat dan satu jam kedua untuk pengujian sistem yang berada dalam keadaan *idle*.

Untuk mengukur CPU *idle* dan *free memory*, penulis menggunakan perintah "vmstat 1 >> hasil.txt" yang dijalankan selama 2 jam di masing-masing *node*. Perintah tersebut menyalin ulang hasil yang ditampilkan ke dalam *file* hasil.txt. Sebagian kecil isi dari *file* hasil.txt adalah sebagai berikut:

```
bash-3.00# more coba.txt
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr cd cd f0 s1   in   sy   cs us sy id
 0 0 0 1657896 438844  73 181 127 0 7  0 286 15 0  0  0  371 1161  530  3
5 92
 0 0 0 1666316 429600 0  44  4  0  0  0  0  0  0  0  0  350  210  145  1  2 97
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  346  169  128  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  350  165  129  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  352  164  140  1  0 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  355  181  140  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  352  172  140  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  346  170  134  1  1 98
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  354  242  135  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  353  164  144  0  2 98
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  350  177  132  1  1 98
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  351  173  129  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  350  162  134  1  1 98
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  354  172  135  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  349  175  128  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  353  170  140  0  1 99
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  341  175  127  2  1 97
 0 0 0 1666316 429600 0   0  0  0  0  0  0  0  0  0  0  342  166  124  0  1 99
```

Sedangkan untuk pengukuran *metric database*, penulis menggunakan *Automatic Workload Repository* (AWR) yang diambil dari masing-masing *instance* dari *node-node* yang ada di dalam RAC.

2.4. Hasil Pengujian

1. cpu idle dan free memory

Skenario	CPU idle	rac1	rac2
1	maksimum	74,5 %	77,1 %
	minimum	42,6 %	47,1 %
2	maksimum	81,7%	83,1%
	minimum	31,2%	29,5%

Skenario	free memori	rac1	rac2
1	maksimum	224786 KB	234912 KB
	minimum	182568 KB	193122 KB
2	maksimum	267884KB	281346 KB
	minimum	201842KB	211624 KB

2. statistik penggunaan SGA

Skenario	SGA regions rac1	Begin Size (Bytes)	End Size (Bytes) (if different)
1	Database buffer	113,246,208	117,440,512
	Fixed size	1,218,532	
	Redo buffers	2,973,696	83,888,156
	Variable size	88,082,460	
	Sum	205,520,896	
2	Database buffer	117,440,512	
	Fixed size	1,218,532	
	Redo buffers	2,973,696	
	Variable size	83,888,156	
	Sum	205,520,896	

Skenario	SGA regions rac2	Begin Size (Bytes)	End Size (Bytes) (if different)
1	Database buffer	117,440,512	117,440,512
	Fixed size	1,218,532	
	Redo buffers	2,973,696	83,888,156
	Variable size	83,888,156	
	Sum	205,520,896	
2	Database buffer	117,440,512	121,634,816
	Fixed size	1,218,532	
	Redo buffers	2,973,696	
	Variable size	83,888,156	79,693,852
	Sum	205,520,896	

3.statistik penggunaan buffer cache dan shared pool

Skenario	buffer cache	rac1	rac2
1	begin	108 M	112 M
	end	112 M	112 M
	std block size:	8K	8K
2	begin	112 M	112 M
	end	112 M	116 M
	std block size:	8K	8K

Skenario	shared pool size	rac1	rac2
1	begin	76 M	72 M
	end	72 M	72 M
	log buffer	2,904 K	2,904K
2	begin	72 M	72 M
	end	72 M	68 M
	log buffer	2,904 K	2,904K

2.5 Analisa Hasil Pengujian

Dapat dilihat dari statistik di atas bahwa penggunaan SGA secara menyeluruh berimbang antara kedua *instance*. Hal ini menandakan bahwa dalam penggunaan *memory*, kedua *instance* berjalan secara imbang. Dan bila dibandingkan antara hasil pengujian pada satu jam pertama dan pengujian satu jam kedua, penggunaan memori masih berjalan normal tanpa ada perubahan yang signifikan.

Sama halnya dengan penggunaan buffer cache dan shared pool yang terlihat di atas. Dimana penggunaan *memory* berimbang antara kedua *instance*. Secara keseluruhan penulis menyimpulkan bahwa pada saat pengujian pertama dan kedua, sistem tetap stabil. Hal ini bisa dilihat dari statistik CPU *idle*, *free memory* dan parameter-parameter *database* terutama SGA, *Buffer Cache*, *Shared Pool*, *Log Buffer* dan *Buffer Pool*.

Dan dari statistik-statistik yang ada penulis juga menyimpulkan bahwa beban kerja atau *load balance* terbagi secara seimbang. Dari sisi penggunaan sumber daya baik CPU maupun *memory* juga menunjukkan keseimbangan yang bagus. Dari hasil analisa yang telah disebutkan, penulis menyimpulkan bahwa sistem RAC yang diuji memang sangat handal dalam menerapkan konsep "*high-availability system*". Walaupun di dunia nyata, setelah implementasi selesai dan sebelum *database* digunakan, sebaiknya dilakukan *tuning* terhadap sistem dengan menggunakan metode analisa kinerja yang telah dilaksanakan di dalam penelitian ini.

3. Kesimpulan

1. Pengimplementasian RAC sangat membutuhkan ketelitian dan perencanaan yang matang, karena sangat beresiko terhadap hilangnya data.
2. Hasil pengujian CPU *idle* maksimum untuk *node* pertama 81,7% dan *node* kedua 83,1%. CPU *idle* minimum sebesar 31,2% untuk *node* pertama dan 29,5% untuk *node* kedua.
3. Hasil pengujian *free memory* maksimum untuk *node* pertama sebesar 267884 KB dan *node* kedua sebesar 281346 KB. *Free memory* minimum sebesar 201842 KB dan *node* kedua sebesar 211624 KB.
4. Hasil pengujian SGA menunjukkan nilai 205520896 di kedua *node*.
5. Nilai pengujian *buffer cache* minimum untuk *node* pertama sebesar 108 MB dan *node* kedua sebesar 112 MB. Untuk pengujian *buffer cache* maksimum sebesar 112 MB untuk *node* pertama dan 116 MB untuk *node* kedua.
6. Pengujian *shared pool* menunjukkan nilai maksimum 76 MB dan 72 MB untuk *node* kedua. Nilai minimum untuk *node* pertama sebesar 72 MB dan 68 MB untuk *node* kedua.
7. Sistem RAC yang diuji memang sangat handal dalam menerapkan konsep "*high-availability system*".

DAFTAR PUSTAKA

Alapati, Sam R. 2005. *Expert Oracle Database 10g Administration*. Apress.

Powell, Gavin. 2007. *Oracle performance Tuning for 10g R2*. Second Editon. Digital Press.

Vallath, Murali. 2003. *Real Aplication Clusters*. Digital Press.

Vallath, Murali. 2006. *Oracle 10g RAC Grid, Service & Clustering*. Digital Press.

Whalen, Edward. 2005. *Oracle Database 10g Linux Administration*. Osborne.