PETIR

JURNAL
PENGKAJIAN DAN PENERAPAN
TEKNIK INFORMATIKA

VOLUME 8 - NOMOR 2 | SEPTEMBER 2015 | ISSN 1978-9262

ISSN 1978-9262	SEKOLAH TINGGI TEKNIK - PLN (STT-PLN)					
771978 926272	PETIR	VOL. 8	NO. 2	HAL. 133 - 239	JAKARTA, SEPTEMBER 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 redo log buffer, persentase efisiensi instant, aktifitas library cache,dan buffer 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*

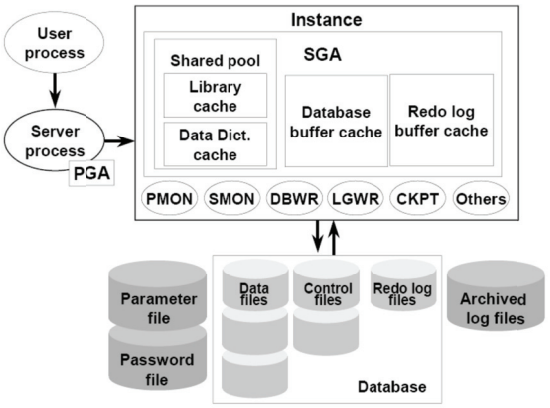
1. Pendahuluan

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"*.

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*.

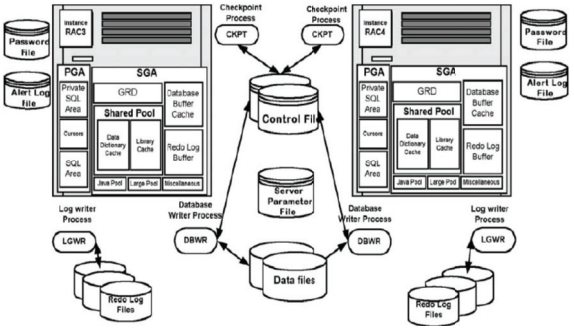


Gambar 1. Arsitektur *Oracle Instance*

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.

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*.

Di dalam lingkungan RAC, hampir semua *database files* di-*shared* antar berbagai *instance*. Tetapi ada *file-file* tertentu yang tidak di-*shared*, setiap *instance* memiliki *file-file* tersebut masing-masing [Vallath 2003].



Gambar 2. *Database files* di dalam konfigurasi RAC.

Seperti yang terlihat pada gambar 2, *database files* yang di-*shared* adalah *control files*, *data files* dan *server parameter file*. *Redo log files*, *password file* dan *alert log file* untuk masing-masing *instance* digunakan sendiri tanpa di-*shared*. *Password file* dan *alert log file* disimpan di *node*-nya masing-masing, sedangkan *redo log files* disimpan di *shared storage*. Khusus untuk *archived redo log files* hanya ditulis oleh satu *instance*. Penyimpanan *archived redo log files* bisa di *storage* lokal atau *shared*. Tetapi, untuk kemudahan dalam pelaksanaan operasi *recovery* dan *backup*, *archived redo log files* sebaiknya disimpan di *shared storage*.

2. Metodologi Penelitian

2.1 Analisis Kebutuhan

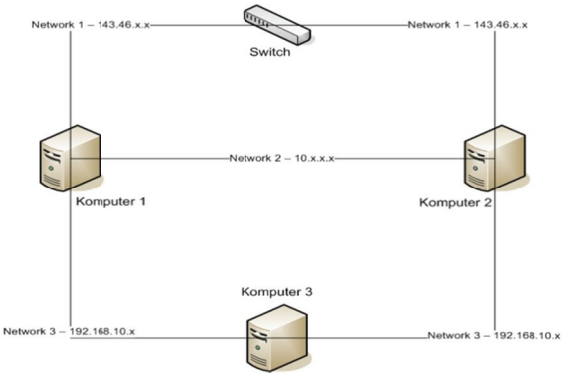
Oracle 10g Real Application Cluster (RAC) dalam implementasinya memiliki persyaratan utama yang harus dipenuhi. Persyaratan ini meliputi aspek perangkat keras, perangkat jaringan, sistem operasi, dan media penyimpanan..

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 *release2* dari *Oracle 10g*.

Setiap komputer yang menjadi *node* RAC harus mendaftarkan alias-alias (nama *host*) seluruh anggota *node*. Alias-alias tersebut terbagi menjadi 3 (tiga), yaitu:

1. Alias untuk koneksi jaringan *public*.
2. Alias untuk koneksi jaringan *private* interkoneksi.
3. Alias untuk *Virtual IP* (VIP).
4. Alias untuk *shared storage*.

Penulis mengkonfigurasikan alias-alias tersebut dengan menambahkan *entry-entry* di dalam *file* ipnodes dan hosts pada yang ada pada direktori /etc/inet. *File* ipnodes dan hosts dibuat identik, yang isinya yaitu:

1. Komputer 1:

127.0.0.1	localhost
143.46.43.100	rac1 loghost
10.0.0.1	rac1-priv
143.46.43.104	rac1-vip
192.168.10.100	rac1-data
143.46.43.101	rac2
10.0.0.2	rac2-priv
143.46.43.105	rac2-vip
192.168.10.150	storage

2. Komputer 2:

127.0.0.1	localhost
143.46.43.101	rac2 loghost
10.0.0.2	rac2-priv
143.46.43.105	rac2-vip
192.168.10.200	rac2-data
143.46.43.100	rac1
10.0.0.1	rac1-priv
143.46.43.104	rac1-vip
192.168.10.151	storage

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 penggunaan SGA, *buffer cache* dan *share pool*. Keseluruhan parameter ini telah penulis tuangkan dalam tulisan sebelumnya (PETIR edisi maret 2015).

Dalam tulisan kali ini penulis akan mencoba mengamati pengaruh parameter data base yang lain yaitu *redo log buffer, instance efficiency percentages,* aktifitas *library cache* dan *buffer 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 *metricdatabase* hasil pengujian, penulis meng-*generateAutomatic Workload Repository* (AWR) dari masing-masing *instance* yaitu rac1 dan rac2.

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%.

Rasio yang digunakan untuk *redo log buffer* di dalam AWR adalah:

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 *redo log buffer* yang diwakilkan oleh parameter *Redo NoWait*, *Oracle* juga memberikan syarat harus bernilai lebih dari 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 TNS-Names:

```
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 melakukan 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

2.4.1 Skenario Pertama

1. Persentasi efisiensi *instance*

Persentasi efisiensi antara kedua *instance* dapat dilihat pada AWR berikut:

rac1 :

Instance Efficiency Percentages (Target 100%)		
Buffer Nowait %: 100.00 Buffer Hit %: 99.85 Library Hit %: 89.77 Execute to Parse %: -0.36 Parse CPU to Parse Elapsd %: 38.76	Redo NoWait %: 100.00 In-memory Sort %: 100.00 Soft Parse %: 93.62 Latch Hit %: 100.00 % Non-Parse CPU: 92.01	
Shared Pool Statistics	Begin	End
Memory Usage %: % SQL with executions>1: % Memory for SQL w/exec>1:	78.06 53.33 90.18	81.27 76.97 92.47

rac2 :

Instance Efficiency Percentages (Target 100%)		
Buffer Nowait %: 99.99 Buffer Hit %: 99.97 Library Hit %: 92.92 Execute to Parse %: 6.63 Parse CPU to Parse Elapsd %: 36.58	Redo NoWait %: 100.00 In-memory Sort %: 100.00 Soft Parse %: 97.02 Latch Hit %: 100.00 % Non-Parse CPU: 96.01	
Shared Pool Statistics	Begin	End
Memory Usage %: % SQL with executions>1: % Memory for SQL w/exec>1:	83.36 50.17 87.91	85.94 79.63 86.70

2. Aktifitas Librari Cache

Statistik aktivitas *Library Cache* RAC dan non-RAC dapat dilihat pada AWR berikut:

rac1 :

Library Cache Activity DB/Inst: INDRA/indra1 Snaps: 2-3						
-> "Pct Misses" should be very low						
Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Re-loads	Invali-dations
BODY	23	30.4	103	13.6	6	0
CLUSTER	44	0.0	85	2.4	2	0
INDEX	21	57.1	21	100.	9	0
SQL AREA	447	64.9	19,229	0	717	2
TABLE/ PROCEDURE	1,070	25.4	4,840	7.0 22.9	261	0
TRIGGER	8	50.0	8	50.0	0	0

Library Cache Activity (RAC) DB/Inst: INDRA/indra1 Snaps: 2-3					
Namespace	GES Lock Requests	GES Pin Requests	GES Pin Releases	GES Inval Requests	GES Invali-Dations
CLUSTER	85	0	0	0	0
INDEX	21	12	0	12	0
TABLE/ PROCEDURE	2,917	277	0	129	0

rac2 :

Library Cache Activity	DB/Inst: INDRA/indra2 Snaps: 2-3					
-> "Pct Misses" should be very low						
Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Invali-dations
BODY	529	0.8	639	1.9	8	0
CLUSTER	21	0.0	73	1.4	1	0
INDEX	21	57.1	24	87.5	9	0
SQL AREA	690	61.9	18,956	3.7	228	0
TABLE/ PROCEDURE	1,083	24.7	3,619	25.1	267	0
TRIGGER	18	38.9	105	8.6	2	0

Library Cache Activity (RAC)	DB/Inst: INDRA/indra2 Snaps: 2-3				
Namespace	GES Lock Requests	GES Pin Requests	GES Pin Releases	GES Inval Requests	GES Invali-dations
CLUSTER	73	0	0	0	0
INDEX	24	12	0	12	0
TABLE/ PROCEDURE	1,512	177	0	84	0

3. Buffer Pool

Statistik *Buffer Pool* yang didapat oleh penulis bisa dilihat pada AWR berikut:

rac1 :

Buffer Pool Statistics	DB/Inst: INDRA/indra1 Snaps: 2-3							
-> Standard block size Pools D: default, K: keep, R: recycle -> Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k								
P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Busy Waits
D	13,664	100	508,710	872	2,362	0	0	0

rac2 :

Buffer Pool Statistics	DB/Inst: INDRA/indra2 Snaps: 2-3							
-> Standard block size Pools D: default, K: keep, R: recycle -> Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k								
P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Busy Waits
D	13,664	100	1,550,288	409	16,632	0	0	177

2.4.2 Skenario Kedua

1. Persentasi efisiensi instance

Persentasi efisiensi antara kedua *instance* dapat dilihat pada AWR berikut:

rac1 :

Instance Efficiency Percentages (Target 100%)		
Buffer Nowait %: 99.00 Buffer Hit %: 99.60 Library Hit %: 88.64 Execute to Parse %: 53.67 Parse CPU to Parse Elapsd %: 36.58	Redo NoWait %: 100.00 In-memory Sort %: 100.00 Soft Parse %: 87.36 Latch Hit %: 100.00 % Non-Parse CPU: 22.78	
Shared Pool Statistics	Begin	End
Memory Usage %: % SQL with executions>1: % Memory for SQL w/exec>1:	81.27 76.97 92.47	84.11 81.22 94.25

rac2 :

Instance Efficiency Percentages (Target 100%)		
Buffer Nowait %: 100.99 Buffer Hit %: 99.71 Library Hit %: 89.30 Execute to Parse %: 50.85 Parse CPU to Parse Elapsd %: 55.17	Redo NoWait %: 100.00 In-memory Sort %: 100.00 Soft Parse %: 86.64 Latch Hit %: 100.00 % Non-Parse CPU: 17.72	
Shared Pool Statistics	Begin	End
Memory Usage %: % SQL with executions>1: % Memory for SQL w/exec>1:	85.94 79.63 86.70	88.74 74.55 92.57

2. Aktifitas Librari Cache

Statistik aktivitas *Library Cache* RAC dan non-RAC dapat dilihat pada AWR berikut:

rac1 :

Library Cache Activity	DB/Inst: INDRA/indra1 Snaps: 3-4					
-> "Pct Misses" should be very low						
Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Invali-dations
BODY	23	30.4	103	13.6	5	0
CLUSTER	44	0.0	85	2.4	2	0
INDEX	21	57.1	21	100.0	25	0
SQL AREA	447	64.9	19,229	7.0	339	2
TABLE/ PROCEDURE	1,070	25.4	4,840	22.9	367	0

Library Cache Activity (RAC)	DB/Inst: INDRA/indra1 Snaps: 3-4				
Namespace	GES Lock Requests	GES Pin Requests	GES Pin Releases	GES Inval Requests	GES Invali-Dations
CLUSTER	37	0	0	0	0
INDEX	78	12	0	0	0
TABLE/ PROCEDURE	2,092	39	0	32	0

rac2 :

Library Cache Activity	DB/Inst: INDRA/indra2 Snaps: 3-4					
-> "Pct Misses" should be very low						
Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Invali-Dations
BODY	604	0.5	726	3.0	19	0
CLUSTER	38	0.0	90	2.2	2	0
INDEX	46	0.0	129	22.5	29	0
SQL AREA	1,239	5.7	13,549	7.3	485	0
TABLE/ PROCEDURE	1,268	2.3	5,463	20.2	482	0
TRIGGER	12	0.0	111	1.8	2	0

Library Cache Activity (RAC)	DB/Inst: INDRA/indra2 Snaps: 3-4				
Namespace	GES Lock Requests	GES Pin Requests	GES Pin Releases	GES Inval Requests	GES Invali-dations
CLUSTER	90	0	0	0	0
INDEX	129	0	0	0	0
TABLE/ PROCEDURE	2,318	55	0	44	0

3. Buffer Pool

Statistik *Buffer Pool* yang didapat oleh penulis bisa dilihat pada AWR berikut:

rac1 :

Buffer Pool Statistics DB/Inst: INDRA/indra1 Snaps: 3-4								
-> Standard block size Pools D: default, K: keep, R: recycle -> Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k								
P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Busy Waits
D	13,664	100	61,223	240	525	0	0	5

rac2 :

Buffer Pool Statistics DB/Inst: INDRA/indra2 Snaps: 3-4								
-> Standard block size Pools D: default, K: keep, R: recycle -> Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k								
P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Busy Waits
D	14,148	100	134,917	413	693	0	0	2

2.5 Analisa Hasil Pengujian

Dapat dilihat dari statistik di atas bahwa efisiensi instanse yang ditunjukan oleh *buffer hit* dan *library hit* lebih banyak dialami oleh rac2 (walaupun hanya beda sekitar 3%). Dan bila dibandingkan antara hasil pengujian pada satu jam pertama dan pengujian satu jam kedua, efisiensi instan antara kedua *instan* tanpa mengalami perubahan nilai yang signifikan.

Data statistik penggunaan *Library Cache* yang ada pada AWR menunjukan bahwa *cache miss* lebih banyak dialami oleh rac1. Hal ini menandakan bahwa perintah-perintah SQL lebih banyak dijalankan di rac1. Pada saat pengujian kedua *cache miss* tetap masih banyak dialami oleh rac1.

Sementara data *Buffer Pool Hit* menunjukkan bahwa kedua *instance* mempunyai kinerja yang bagus. Hal ini ditandai dengan nilai rasio 100 % di kedua *instance*. Keadaan ini terus berlangsung sampai pengujian kedua.

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 persentase instant yaitu buffer hit untuk *node* pertama 99,8% dan *node* kedua 99,97%. CPU *idle* minimum sebesar 99,6% untuk *node* pertama dan 99,71% 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.