PENERAPAN *ORACLE RAC* DAN *ORACLE DATA GUARD* DALAM MENINGKATKAN KETERSEDIAAN LAYANAN INFRASTRUKTUR DATABASE PADA PERUSAHAAN YANG BERGERAK DALAM BIDANG *MONITORING* PENGGUNAAN FREKUENSI RADIO

WANDI MARTA SUPANGKAT
41512110110

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS MERCU BUANA
JAKARTA
2016

PENERAPAN *ORACLE RAC* DAN *ORACLE DATA GUARD* DALAM MENINGKATKAN KETERSEDIAAN LAYANAN INFRASTRUKTUR DATABASE PADA PERUSAHAAN YANG BERGERAK DALAM BIDANG *MONITORING* PENGGUNAAN FREKUENSI RADIO

Laporan Tugas Akhir

Diajukan Untuk Melengkapi Persyaratan
Menyelesaikan Gelar Sarjana Komputer

Disusun oleh :
WANDI MARTA SUPANGKAT
41512110110

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS MERCU BUANA
JAKARTA
2016

LEMBAR PERNYATAAN

Yang bertanda tangan dibawah ini:

NIM	: 41512110110
Nama	: WANDI MARTA SUPANGKAT
Judul Skripsi	: PENERAPAN *ORACLE RAC* DAN *ORACLE DATA GUARD* DALAM MENINGKATKAN KETERSEDIAAN LAYANAN INFRASTRUKTUR DATABASE PADA PERUSAHAAN YANG BERGERAK DALAM BIDANG *MONITORING* PENGGUNAAN FREKUENSI RADIO

Menyatakan bahwa skripsi tersebut diatas adalah hasil karya saya sendiri dan bukan plagiat. Apabila ternyata ditemukan didalam laporan skripsi saya terdapat unsur plagiat, maka saya siap untuk mendapatkan sanksi akademik yang terkait dengan hal tersebut.

Jakarta, 31 Mei 2016

Wandi Marta Supangkat

LEMBAR PERSETUJUAN

NIM	: 41512110110
Nama	: WANDI MARTA SUPANGKAT
Judul Skripsi	: PENERAPAN *ORACLE RAC* DAN *ORACLE DATA GUARD* DALAM MENINGKATKAN KETERSEDIAAN LAYANAN INFRASTRUKTUR DATABASE PADA PERUSAHAAN YANG BERGERAK DALAM BIDANG *MONITORING* PENGGUNAAN FREKUENSI RADIO

SKRIPSI INI TELAH DIPERIKSA DAN DISETUJUI

JAKARTA, 10 Juni 2016

Afiyati , S.SI, MT
Dosen Pembimbing

Yaya Sudarya Triana, M.Kom, Ph.D
Kaprodi Informatika

Desi Ramayanti, S.Kom, MT
Koordinator Tugas Akhir

PERSEMBAHAN

Bismillaahirrahmaanirrahiim,
Bacalah dengan menyebut nama Tuhanmu
Dia telah menciptakan manusia dari segumpal darah Bacalah, dan Tuhanmulah yang maha mulia
Yang mengajar manusia dengan pena,
Dia mengajarkan manusia apa yang tidak diketahuinya (QS. Al-'Alaq: 1-5)

Dan seandainya pohon-pohon di bumi menjadi pena dan laut (menjadi tinta), ditambahkan kepadanya tujuh laut (lagi) sesudah (kering) nya, niscaya tidak akan habis-habisnya (dituliskan) kalimat Allah. Sesungguhnya Allah Maha Perkasa lagi Maha Bijaksana.(QS. 31:27)

Alhamdulllahirabbil'alamin…. Alhamdulllahirabbil 'alamin….
Alhamdulllahirabbil alamin….
Akhirnya aku sampai ke titik ini,
sepercik nikmat yang Engkau hadiahkan padaku ya Rabb
Tak henti-hentinya aku mengucap syukur pada Mu ya Rabb
Serta shalawat dan salam kepada junjungan alam Rasulullah Muhammad SAW dan para sahabat yang mulia
Semoga karya mungil ini menjadi amal shaleh bagiku dan menjadi kebanggaan bagi keluargaku tercinta
Ku persembahkan karya mungil ini…
Kepada ibunda Wagiati dan ayahanda Una Iskandar (alm) tercinta
tanpa tangan-tangan dan hati yang tulus ibunda dan ayahanda niscaya belum tentu aku akan seperti sekarang ini,
dan untuk tiga bidadariku tersayang, istriku Siti Maryuni dan anak-anakku Alika Zahra Ghasani dan Qonita Qorri Aina,
terima kasih atas doa, pengertian dan dukungannya
sungguh kalian telah menghiasi hari-hariku…

Alhamdulllahirabbil 'alamin….

KATA PENGANTAR

Bismillahirohmanirrohim,

Puji syukur penulis panjatkan kehadirat Allah Subhanahu Wa Ta"ala, berkat limpahan dan rahmat-Nya penyusun mampu, sehingga penulis dapat menyelesaikan sehingga penulis dapat menyelesaikan laporan tugas akhir ini yang merupakan salah satu persyaratan untuk menyelesaikan program studi strata satu (S1) pada Program Studi Informatika Universitas Mercu Buana. Shalawat serta salam semoga selalu tercurahkan kepada Rasulullah Muhammad Shallallahu „alaihi Wa Sallam.

Dengan selesainya laporan tugas akhir ini tidak terlepas dari bantuan banyak pihak yang telah membantu dan memberikan masukan-masukan kepada penulis sehingga dapat menyelesaikan laporan tugas akhir ini. Oleh karena itu penulis mengucapkan terimakasih kepada semua pihak yang telah banyak membantu penulis dalam pelaksanaan maupun penyusunan laporan ini, antara lain:

1. Ibu Afiyati, S.SI, MT selaku pembimbing akademis yang telah banyak memberikan dukungan, bimbingan serta arahan.
2. Ibu Desi Ramayanti, S.Kom, MT selaku koordinator tugas akhir Program Studi Informatika Universitas Mercu Buana.
3. Bapak Yaya Sudarya Triana, M.Kom, Ph.D selaku Kepala Program Studi Informatika Universitas Mercu Buana.
4. Ibunda Wagiati dan ayahanda Una Iskandar (Alm) tercinta, terima kasih yang tak terhingga atas doa, semangat, kasih sayang, pengorbanan, dan ketulusannya dalam membimbing penulis. Semoga Allah SWT senantiasa melimpahkan rahmat dan ridho-Nya kepada ayahanda dan ibunda.
5. Istriku Siti Maryuni dan anak-anakku Alika Zahra Ghasani dan Qonita Qorri Aina tersayang, terima kasih yang sedalam-dalamnya atas doa, semangat, dukungan dan pengertiannya selama ayah menempuh kuliah sampai dengan terselesaikannya laporan tugas akhir ini

6. Adik-adikku Ajeng Hartini Dewi dan Rinawati terima kasih atas dukungan moril maupun materil selama penulis menempuh kuliah sampai dengan terselesaikannya laporan tugas akhir ini.
7. Sahabat-sahabatku yang tidak dapat disebutkan satu per satu, terima kasih atas segalanya semoga Allah SWT membalas amal kebaikan kalian semua.

Penulis menyadari bahwa masih terdapat banyak kekurangan pada laporan tugas akhir ini, baik dari materi maupun teknik penyajiannya. Oleh karena itu, dengan segala kerendahan hati, penulis mohon maaf atas segala kekurangan dan keterbatasan dalam penulisan laporan tugas akhir ini.

Akhir kata, semoga laporan tugas akhir ini dapat memberikan manfaat khususnya bagi penulis, pembaca dan pihak-pihak yang berkepentingan.

Jakarta, 31 Mei 2016

Wandi Marta Supangkat

ABSTRAKSI

Ketersediaan layanan menjadi prioritas utama dari perusahaan agar integritas perusahaan senantiasa dapat terjaga, karena dengan menurunnya ketersedian layanan mengarah kepada menurunnya pendapatan dan kepercayaan dari pengguna pada sistem atau aplikasi yang digunakan. Redundansi disetiap komponen pembentuk infrastruktur teknologi informasi menjadi opsi yang dipilih agar layanan senantiasa dapat terjaga. Dari sisi RDBMS, *Oracle RAC* menawarkan dua fitur yang sangat sangat menarik bagi pelaku bisnis yang menjalankan *mission critical application* yang mengutamakan ketersediaan layanan *database*, yaitu *high availability* dan *load balancing*. *Load balancing* memungkinkan dibaginya beban kerja pada semua anggota *cluster*. *High availability* memungkinkan pengguna untuk tetap bisa melakukan aktivitas walaupun terjadi kerusakan fisik ataupun non-fisik pada server anggota *cluster* yang tengah terhubung, dengan catatan masih ada server anggota *cluster* yang masih berjalan dengan normal dan berdasar jenis aktivitas yang tengah dilakukan. Setelah dipilihnya solusi yang mempunyai kapabilitas *high availability* dan *load balancing*, selanjutnya dipilih metode migrasi *database* menggunakan *Oracle Data Guard* yang termasuk kedalam kategori metode *minimum downtime* yang mana merupakan metode *low cost* sangat dicari oleh para pelaku bisnis. Dalam mengembangkan penerapan solusi *high availability* digunakan metode pengembangan SDLC (*System Development Life Cycle*) model *prototyping* yang terbagi dalam fase : analisa kebutuhan, pengembangan *prototype*, pengujian dan *feedback* dari pengguna dan tahap implementasi dari produk. Dengan diterapkannya solusi *high availability*, ketersediaan layanan pada perusahaan senantiasa dapat terjaga yang berarti tetap terjaganya pula tingkat pendapatan perusahaan dan tingkat kepercayaan dari seluruh pengguna.

Kata Kunci: Ketersediaan layanan, *High Availability, Load Balancing, Oracle RAC,* Migrasi dengan *Minimum Downtime.*

ABSTRACT

Availability of services is a top priority of the company to the integrity of the company can always be maintained, due to the decreasing availability of services leads to decreased revenue and the trust of users on the system or application used. Redundancy each forming part of the information technology infrastructure into the option selected so that services can always be maintained. From the RDBMS, Oracle RAC offers two features that are very attractive for businesses that run mission critical application that prioritizes the availability of database services, namely high availability and load balancing. Load balancing allows the sharing of the workload on all cluster members. High availability allows users to remain able to perform activities despite damage to physical or non-physical server cluster members being connected, with records still exist cluster member server is still running normally and by type of activity being performed. After choosing a solution that has the capability of high availability and load balancing, database migration method further been using Oracle Data Guard are included in the category of minimum downtime method which is a low cost method are highly sought after by the business. In developing the application of the solution high availability use development methods SDLC (System Development Life Cycle) model of prototyping is divided into phases: analysis of requirements, development of prototype, user testing and feedback and final product release. With the implementation of high availability solutions, service availability on the company can always be maintained, which means the level remained subdued corporate earnings and the level of trust of all users.

Keywords : The availability of services, High Availability, Load Balancing, Oracle RAC, Minimum Downtime Migration.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DEFINISI

Istilah	Pengertian
SDLC (Systems Development Life Cycle)	Adalah model konseptual yang digunakan dalam manajemen proyek yang menggambarkan tahap- tahap yang terlibat dalam suatu proyek pengembangan sistem informasi dari studi kelayakan awal melalui pemeliharaan aplikasi selesai
OLTP (Online Transaction Processing)	Adalah salah satu tipe *database* yang mempunyai karakteristik memproses transaksi (*insert*, *update*, *delete*) secara langsung dalam rentang waktu yang singkat dan dalam jumlah yang besar.
DSS (Decission Support System)	Adalah suatu tipe *database* yang digunakan untuk proses analisa data dan selanjutnya hasilnya digunakan oleh pengguna sebagai acuan dalam pengambilan keputusan.
Batch Workload	Suatu tipe *database* yang merupakan hasil gabungan (hybrid) dari *database* tipe OLTP dan *database* tipe DSS. *Database batch workload* mempunyai karakteristik dengan banyaknya transaksi yang sama yang berulang kali, dan juga proses yang memerlukan komputasi yang berat.
Data Loss	Hilangnya data akibat kesalahan user, kerusakan perangkat ataupun kesalahan prosedur.
Down Time	Terhentinya aksesibilitas suatu *service*/layanan yang disebabkan oleh suatu kejadian yang terencana ataupun tidak terencana.
Planned Down Time	Terhentinya aksesibilitas suatu *service*/layanan secara terencana, contohnya adalah pergantian berkala *spare-part server*, penerapan *patch* pada *database* server.
Unplanned Down Time	Terhentinya aksesibilitas suatu *service*/layanan secara tidak terencana, contohnya adalah *server crash*, bencana alam.

Instance	adalah seperangkat struktur memory yang mengelola *database file*. Sebuah *instance* terdiri dari *shared memory area* yang disebut dengan SGA (*System Global Area*) dan satu set *background proses*. Sebuah *instance* terhubung hanya pada satu *database*.
Database	Dalam sebuah *Oracle Database server*, sebuah *database* adalah sekumpulan *file* yang tersimpan pada media penyimpanan, contohnya pada *disk*.
Listener	Suatu proses dalam *Oracle Database server* yang mempunyai fungsi sebagai penerima permintaan koneksi ke dalam database, setelah permintaan terautentikasi, selanjutnya *listener* akan mengarahkan koneksi user pada sebuah *server process*.
Local Listener	Proses *listener* dalam konfigurasi *Oracle RAC*, dimana *local listener* ini adalah *listener* yang berjalan pada *localhost*.
Remote Listener	Proses *listener* dalam konfigurasi *Oracle RAC*, dimana remote *listener* ini adalah *listener* yang berjalan pada *remote host* anggota dari konfigurasi *Oracle RAC*.
Redo Log	*File output* tempat ditulisnya semua aktivitas yang terjadi pada sebuah *database*. Diperlukan saat *proses recovery database* dan oleh proses sinkronisasi *Data Guard*.
Response Time	Waktu yang dibutuhkan oleh suatu sistem/aplikasi untuk memberikan *output* atas *request* yang diajukan oleh pengguna.
Single Point of Failure	Adalah potensi kegagalan fungsi pada suatu komponen/bagian yang dapat menyebabkan kegagalan fungsi sistem secara keseluruhan.
High Availability	Adalah karakteristik suatu sistem yang menjamin suatu level operational *performance* yang telah disetujui yang melebihi dari level operasional *performance* pada umumnya

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Salah satu faktor yang menentukan suatu perusahaan sebagai pelaku bisnis dapat melewati persaingan bisnis yang semakin ketat adalah dengan senantiasa terjaganya integritas perusahaan melalui salah satunya yaitu tersedianya layanan bisnis kapanpun pelanggan membutuhkannya. Perusahaan membutuhkan dukungan infrastruktur teknologi informasi (TI) yang dapat menyediakan kinerja yang handal yang memungkinkan layanan perusahaan senantiasa dapat diakses oleh pelanggan kapan pun tanpa ada batasan waktu dan pelanggan senantiasa mendapat *response time* yang memuaskan.

Dalam rangka memenuhi ketersediaan layanan bagi setiap pelanggan tanpa mengenal batasan waktu, perlu dilakukan perombakan yang cukup besar pada infrastruktur TI yang telah ada. Dalam proses perombakan ini termasuk didalamnya adalah proses pemindahan data dari infrastruktur yang telah ada ke infrastruktur yang baru. Sejalan dengan visi perusahaan untuk menyediakan layanan yang senantiasa dapat diakses oleh pelanggan, maka muncul tantangan baru, yaitu bagaimana caranya memindahkan data dari infrastruktur lama ke infrastruktur baru tanpa mengganggu kenyamanan user dalam mengakses layanan perusahaan. Dalam prakteknya, pemindahan data dengan *zero downtime* membutuhkan usaha dan dana yang sangat besar, dibutuhkan redudansi dari semua komponen infrastruktur TI.

Dalam perkembangannya dunia TI telah menawarkan teknologi yang mempunyai kemampuan untuk menjawab kebutuhan-kebutuhan para pelaku bisnis di atas. Dalam menjawab kebutuhan pelaku bisnis yang menginginkan akses yang tidak terbatas oleh waktu dan *response time* yang memuaskan, pada umumnya dibangun suatu infrastruktur yang mempunyai redundansi pada setiap komponennya. Dengan adanya redundasi menjamin bahwa pada setiap komponen infrastruktur tidak menjadi potensi penyebab *single point of failure*, dalam artian gagalnya fungsi pada suatu komponen akan

menyebabkan kegagalan fungsi pada keseluruhan infrastruktur atau sistem yang ada. Selain daripada itu, dengan adanya redundasi maka beban kerja pun dapat terdistribusi, dan umumnya kemampuan ini membutuhkan perangkat ataupun aplikasi tambahan yang dapat mendeteksi serta mengatur beban.

Teknologi yang dapat dimanfaatkan untuk menjawab kebutuhan pelaku bisnis akan ketersediaan layanan perusahaan pada wilayah *server*, adalah teknologi *cluster computer*, yaitu teknologi yang memungkinkan beberapa komputer bekerja sama bersama-sama sehingga membentuk sistem yang lebih besar dan user hanya melihat sistem ini sebagai satu komputer saja. teknologi *cluster computer* pada umumnya memanfaatkan *high speed dedicated network* untuk menghubungkan antar *server* dalam satu lingkungan *cluster*. Dalam implementasinya, *cluster computer* mengenal 2 pendekatan arsitektur yang dominan dipergunakan, yaitu *shared-nothing architecture* dan *shared-disk architecture*. Meskipun pendekatan *shared-nothing architecture* memerlukan biaya yang relatif lebih kecil bila dibandingkan dengan pendekatan *shared-disk architecture*, namun pendekatan *shared-nothing architecture* lebih umum diterapkan (relatif terbatas) pada tipe-tipe aplikasi *data warehouse* ataupun tipe-tipe aplikasi dimana perbandingan proses baca lebih tinggi dari pada proses penulisan (Mullins, 2003). Sedangkan pada pada pendekatan *shared-disk architecture*, dapat diterapkan pada tipe-tipe aplikasi *hybrid*, dimana aplikasi relatif berimbang antara proses baca dan proses penulisan, ataupun proses penulisan data sedikit lebih dominan dibanding proses pembacaan data.

Berkaitan dengan tujuan penerapan teknologi oleh pelaku bisnis, yaitu ketersediaan layanan yang tidak terbatas oleh waktu dan *response time* yang memuaskan, penerapan teknologi *cluster computer* berbasis *shared-disk architecture* dinilai masih mempunyai faktor kemungkinan penyebab *single point of failure*, yaitu pada komponen *network* dan komponen *storage*. Komponen *storage* sebagai media penyimpanan data sering kali tidak mempunyai tingkat redundansi yang memuaskan, redundasi yang dimaksudkan adalah perlindungan akan kegagalan fungsi pada tingkat

controller ataupun *box storage*, karena pada tingkat *disk*, redundansi telah terlindungi oleh mekanisme RAID. Sehingga pelaku bisnis masih merasa masih perlu untuk meningkatkan tingkat *confidence* akan infrastruktur yang ada, yaitu dengan cara penerapan redundansi yang lebih tinggi pada tingkat *controller* ataupun *box storage*. Begitu pula dengan komponen *network*, bagian ini pun perlu untuk diterapkan redundansinya. Dengan demikian maka semua komponen pembentuk *cluster computer* mempunyai tingkat redundansi, dan tipe *cluster computer* seperti ini masuk kedalam tipe *high availability cluster*.

1.2. Rumusan Permasalahan

Berdasarkan uraian dan penjelasan latar belakang di atas dengan adanya potensi *single point of failure* pada infrastruktur yang ada dimana akan mengakibatkan terhentinya ketersediaan layanan bagi para pengguna, maka diidentifikasikan rumusan masalah berdasarkan tema dan judul yang telah ditetapkan. Berikut ini adalah rumusan masalah dalam skripsi ini :

1. Bagaimana penerapan infrastruktur yang dapat menyediakan layanan yang mendekati klasifikasi *high availability infrastructure*.
2. Bagaimana cara pemanfaatan semua *resource computer/server* yang ada, sehingga nantinya pengembangan yang akan dilakukan menjadi efektif dari sisi *resource consumption*.
3. Serta bagaimana proses migrasi dapat dilakukan dari infrastuktur yang telah ada dan tengah di akses oleh user ke infrastruktur yang baru dengan waktu *downtime* yang seminimal mungkin.

1.3. Tujuan & Manfaat Penelitian

1.3.1 Tujuan Penelitian

Berdasarkan permasalahan yang diteliti, maka tujuan dari penulisan tugas akhir ini adalah :

1. Mengimplementasikan *Oracle RAC* untuk meningkatkan ketersediaan layanan infrastruktur database dan memaksimalkan penggunaan resos pada semua *server* yang tersedia.

2. Mengimplementasikan *Oracle Data Guard* untuk meminimalisir terjadinya *downtime* pada saat pemindahan *database* dari infrastruktur lama ke infrastruktur baru.

1.3.2 Manfaat Penelitian

Manfaat yang ingin dicapai pada penelitian ini adalah:

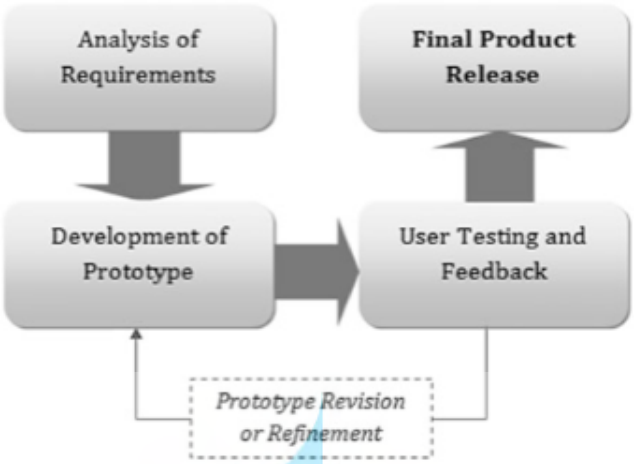
Para pengguna dapat mengkases data yang diinginkan setiap saat tanpa ada batasan waktu dan senantiasa mendapat *response time* yang memuaskan, serta dalam proses migrasi dari infrastruktur yang telah ada ke infrastruktur yang baru tidak menyebabkan *data loss* maupun tidak dapat diaksesnya *system* atau aplikasi oleh pengguna secara berlarut-larut.

1.4. Metodologi Penelitian

Berdasarkan referensi definisi sejumlah model pengembangan sistem yang ada, dalam penelitian ini digunakan metode pengembangan sistem SDLC (*System Development Life Cycle*) model *prototyping*. Menurut Isaias dan Issa (2014), sebuah model *prototyping* terdiri dari empat tahapan yang berbeda, tahap pertama yaitu pengumpulan kebutuhan dan *requirement* pengguna untuk selanjutnya dilakukan identifikasi dan analisa, tahap kedua berdasar dari hasil identifikasi dan analisa selanjutnya dilakukan pengembangan *prototype* dari *product* yang akan diimplementasikan, tahap selanjutnya pengguna melakukan pengujian sehingga dapat memberikan *feedback* secara langsung dan merasakan pengalaman sebagai mana nantinya menggunakan *product* dilingkungan *production*. Jika dalam tahapan ini ditemukan perlu adanya perubahan dan perbaikan maka dilakukan penyempurnaan pada *prototype* untuk selanjutnya dilakukan pengujian kembali. Proses ini diulang sampai dicapai *prototype* yang memenuhi kebutuhan pengguna dan tidak diperlukan lagi penyempurnaan. Tahap selanjutnya adalah *final product release* yang siap untuk diimplementasikan (Carr dan Verner, 1996).

SDLC dijadikan metode yang digunakan sebagai acuan (secara keseluruhan atau secara garis besar) pada proses pengembangan dan perancangan suatu sistem, mengingat bahwa setiap sistem memiliki

kebutuhan yang berbeda dan memiliki permasalahan yang unik, sehingga membutuhkan solusi yang adaptif dari metode SDLC.



Gambar 1-1 Metode Pengembangan SDLC Model *Prototyping*

1.5. Ruang Lingkup & Batasan Penelitian

Beberapa batasan dan lingkup permasalahan dalam penelitian ini adalah:

1. Skripsi ini hanya membatasi pembahasan penerapan *high availability cluster* pada sektor *server* RDBMS.
2. Penerapan fitur *load balancing* dan *high availability* menggunakan perangkat lunak *Oracle RAC*.
3. Mekanisme *load balancing* yang digunakan adalah mekanisme *server side load balancing.*
4. Mekanisme *high availability* yang digunakan adalah mekanisme *fail-over.*
5. Penerapan migrasi dengan *minimum downtime* menggunakan fitur *Oracle Data Guard* dari RDBMS *Oracle*.
6. Digunakan lingkungan penelitian buatan sebagai *prototype* dengan melakukan simulasi untuk mendapatkan beberapa data dari lingkungan *virtual*. Lingkungan *virtual* ini terfokus pada bagian RDBMS *server* yang telah terkonfigurasi sama dengan konfigurasi yang telah terimplementasi pada lingkungan produksi yang telah dibangun. Lingkungan penelitian buatan mempergunakan *Oracle Virtualbox* versi 5.0.16 sebagai aplikasi virtualisasi, Sistem Operasi menggunakan

Oracle Enterprise Linux versi 6.5, RDBMS menggunakan *Oracle* RDBMS versi 11.2.0.4, Aplikasi *Cluster* menggunakan *Oracle Grid Infrastructure* versi 11.2.0.4, dan 2 buah *external hard disk*.

7. Analisa dilakukan di perusahaan yang bergerak dalam bidang monitoring penggunaan frekuensi radio.

1.6. Sistematika Penulisan Laporan

Laporan hasil penelitian ini disusun menurut sistematika sebagai berikut:

1.6.1 Pendahuluan

Membahas Latar Belakang Masalah, Identifikasi Masalah, Batasan Masalah, Tujuan Penelitian, Metodelogi Penelitian serta Sistematika Penulisan.

1.6.2 Landasan Teori

Memaparkan teori-teori yang didapat dari sumber-sumber yang relevan untuk digunakan sebagai panduan dalam penelitian serta penyusunan Skripsi.

1.6.3 Analisis Sistem

Menjelaskan tentang gambaran sistem serta deskripsi dari hasil analisis sistem yang akan dijadikan sebagai petunjuk untuk perancangan pada tahapan berikutnya.

1.6.4 Perancangan Sistem

Berisi tentang Perancangan Sistem, Perancangan Arsitektural dan Perancangan Prosedural.

1.6.5 Implementasi Dan Testing

Menjelaskan mengenai kebutuhan *hardware*, *software* serta mengenai arsitektur dan proses penerapan/implementasi.

1.6.6 Penutup

Mengemukakan kesimpulan yang diambil dari hasil penelitian dan penulisan Skripsi ini, serta saran-saran untuk pengembangan selanjutnya, agar dapat dilakukan perbaikan-perbaikan di masa yang akan datang.

BAB 2
LANDASAN TEORI

2.1. *Server*

Server adalah sebuah perangkat komputer ataupun sebuah program komputer yang menyediakan fungsi atau layanan kepada program lain ataupun perangkat lain, yang disebut sebagai *client*. Arsitektur ini disebut sebagai arsitektur *client-server.*

2.1.1 *Blade Server*

Blade server adalah salah satu bentuk fisik suatu *computer server* selain bentuk lainnya seperti *rackmounting server* dan *microserver*. *Blade server* adalah *computer server* dalam bentuk modular yang didesain sedemikian rupa yang bertujuan untuk optimalisasi konsumsi ruang dan energi. Jika pada umumnya sebuah *rackmount server* dilengkapi dengan sekurang-kurangnya sebuah *power cord* dan *network card*, lain halnya pada *blade server* komponen-komponen tersebut dihilangkan demi optimalisasi penggunaan ruang, minimalisasi penggunaan energi dan beberapa pertimbangan lainnya, tanpa mengurangi fungsinya sebagai sebuah komputer. Pada prakteknya sebuah *blade server* adalah sebuah modul atau *card* yang berisi *processor*, *memory*, dan *integrated network controller*, dan sebagai optional ditanamkan juga *fibre channel host bus adapter* (HBA) dan beberapa *input/output port* lainnya (Dell, n.d.). Dan pada umumnya sebuah *blade server* dialokasikan untuk satu buah aplikasi.

Sebuah *blade enclosure/chassis* dapat menampung beberapa *blade server*. *Blade enclose/chassis* menyediakan *power*, *cooling system*, *networking* dan beberapa *interconnect* dan *management*. Sebuah *blade enclose/chassis* yang didalamnya terdapat beberapa *blade server*, disebut sebagai sebuah *blade system*. Berikut adalah salah satu contoh *blade system* :

Gambar 2-1 Contoh *Blade System*

Contoh untuk *rackmount server* adalah sebagai berikut :

Gambar 2-2 Contoh *Rackmount Server*

2.2. Jaringan Komputer

Jaringan Komputer adalah sekelompok komputer otonom yang saling berhubungan antara satu dengan lainnya menggunakan protokol komunikasi melalui media komunikasi sehingga dapat saling berbagi data, informasi, program – program, penggunaan bersama perangkat keras seperti *printer*, *hard disk*, dan sebagainya. Selain itu jaringan komputer bisa diartikan sebagai kumpulan sejumlah terminal komunikasi yang berada diberbagai lokasi yang terdiri dari lebih satu komputer yang saling berhubungan (Pamungkas, 2011).

2.2.1 Definisi jaringan komputer

Jaringan komputer dapat didefinisikan sebagai sebuah sistem yang terdiri atas komputer beserta perangkat jaringan lainnya. Informasi dan data bergerak melalui media transmisi sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar dokumen dan data, berbagi sumber daya (*hardware* dan *software*) dan mengkases informasi (*web browsing*).

Dalam jaringan komputer terdapat dua komponen utama jaringan, yaitu *Node* dan *Node Link* merupakan media transmisi data (kabel dan nirkabel), sedangkan *Node* adalah unit perangkat keras yang digabungkan dalam jaringan komputer seperti (*PC, Switch, Router*).

Dalam mempelajari macam-macam jaringan komputer terdapat dua klasifikasi yang sangat penting yaitu teknologi transmisi dan jarak. Secara garis besar, terdapat dua jenis teknologi transmisi yaitu jaringan *broadcast* dan jaringan *point-to-point*.

Jaringan *broadcast* memiliki saluran komunikasi tunggal yang dipakai bersama-sama oleh semua mesin yang ada pada jaringan. Pesan- pesan berukuran kecil, disebut paket, yang dikirimkan oleh suatu mesin akan diterima oleh mesin-mesin lainnya. *Field* alamat pada sebuah paket berisi keterangan tentang kepada siapa paket tersebut ditujukan. Saat menerima paket, mesin akan mengecek field alamat. Bila paket tersebut ditujukan untuk dirinya, maka mesin akan memproses paket itu, bila paket ditujukan untuk mesin lainnya, mesin tersebut akan mengabaikannya.

Jaringan *point-to-point* terdiri dari beberapa koneksi pasangan individu dari mesin-mesin. Untuk mengirim paket dari sumber ke suatu tujuan, sebuah paket pada jaringan jenis ini mungkin harus melalui satu atau lebih mesin-mesin perantara. Seringkali harus melalui banyak *route* yang mungkin berbeda jaraknya. Karena itu algoritma *route* memegang peranan penting pada jaringan *point-to-point*.

Pada umumnya jaringan yang lebih kecil dan terlokalisasi secara geografis cendurung memakai *broadcasting*, sedangkan jaringan yang lebih besar menggunakan *point-to-point*.

Untuk mengirim paket dari sumber ke suatu tujuan, sebuah paket pada jaringan jenis ini mungkin harus melalui satu atau lebih mesin-mesin perantara. Seringkali harus melalui banyak rute yang mungkin berbeda jaraknya. Karena itu algoritma *route* memegang peranan penting pada jaringan point-to-point (Pamungkas, 2011).

Tabel 2-1 Klasifikasi jaringan menurut jaraknya

Jarak antar prosessor	Prosesor di tempat yang sama	Jenis jaringan
0.1 m	Papan rangkaian	Data flow machine: komputer-komputer paralel, memiliki beberapa unit fungsi yang semuanya bekerja untuk program yang sama
1 m	Sistem	Multicomputer, sistem yang berkomunikasi dengan cara mengirim pesan-pesannya melalui bus* pendek dan sangat cepat.
10 m	Ruangan	Local Area Network (LAN)
100 m	Gedung	Local Area Network (LAN)
1 km	Kampus	Local Area Network (LAN)
10 km	Kota	Metropolitan area Network (MAN)
100 km	Negara	Wide Area Network (WAN)
1.000 km	Benua	Wide Area Network (WAN)
10.000 km	Planet	Internet

*Jalan data elektrik yang mana bit dikirimkan dalam CPU, antar CPU dan komponen-komponen lain di mainboard.

Dari tabel di atas terlihat pada bagian paling atas adalah *data flow machine*, komputer-komputer yang sangat paralel yang memiliki beberapa unit fungsi yang semuanya bekerja untuk program yang sama. Kemudian multikomputer, sistem yang berkomunikasi dengan cara mengirim pesan – pesannya melalui bus pendek dan sangat cepat. Setelah kelas multicomputer adalah jaringan sejati, komputer-komputer yang bekomunikasi dengan cara bertukar data/pesan melalui kabel yang lebih panjang. Jaringan seperti ini dapat dibagi menjadi *local area network* (LAN), *metropolitan area network* (MAN), dan *wide area network* (WAN). Akhirnya, koneksi antara dua jaringan atau lebih disebut *internetwork*. *Internet* merupakan salah satu contoh yang terkenal dari suatu *internetwork*.

2.2.2 Manfaat Jaringan

Secara umum, jaringan mempunyai beberapa manfaat yang lebih dibandingkan dengan komputer yang berdiri sendiri dan dunia usaha telah

pula mengakui bahwa akses ke teknologi informasi modern selalu memiliki keunggulan kompetitif dibandingkan pesaing yang terbatas dalam bidang teknologi.

Jaringan memungkinkan manajemen sumber daya lebih efisien. Misalnya, banyak pengguna dapat saling berbagi printer tunggal dengan kualitas tinggi, dibandingkan memakai *printer* kualitas rendah di masing-masing meja kerja. Selain itu, lisensi perangkat lunak jaringan dapat lebih murah dibandingkan lisensi *stand-alone* terpisah untuk jumlah pengguna sama.

Jaringan membantu mempertahankan informasi agar tetap handal. Sistem penyimpanan data terpusat yang dikelola dengan baik memungkinkan banyak pengguna mengaskses data dari berbagai lokasi yang berbeda, dan membatasi akses ke data sewaktu sedang diproses.

Jaringan membantu mempercepat proses berbagi data (*data sharing*). Transfer data pada jaringan selalu lebih cepat dibandingkan sarana berbagi data lainnya yang bukan jaringan.

Jaringan memungkinkan kelompok kerja berkomunikasi dengan lebih efisien. Surat dan penyampaian pesan elektronik merupakan substansi sebagian besar sistem jaringan, disamping sistem penjadwalan, pemantauan proyek, konferensi online dan groupware, dimana semuanya membantu team bekerja lebih produktif.

2.3. Sistem Operasi Linux

Linux adalah *unix-like operating system* dan merupakan sistem operasi yang paling mendekati kepada *POSIX-compliant* yang dikembangkan secara *free* dan *open source*.

2.3.1 Sejarah Linux

MINIX, sebuah sistem bertipe Unix yang ditujukan untuk penggunaan akademis dirilis oleh Andrew S. Tanenbaum pada tahun 1987. Kode sumber MINIX 1.0 tercantum dalam bukunya *Operating Systems: Design and Implementation*. Walaupun dapat secara mudah didapatkan, modifikasi dan pendistribusian ulang tidak diperbolehkan pada saat itu.

Hak cipta dari kode sumbernya termasuk ke dalam hak cipta dari bukunya yang dipublikasikan oleh Prentice Hall. Sebagai tambahan, disain versi 16-bit dari MINIX kemudian tidak secara baik diadaptasikan kepada versi 32-bit dari arsitektur Intel 386 yang murah dan populer yang digunakan secara luas di komputer pribadi.

Tahun 1991, Torvalds mulai bekerja untuk membuat versi non-komersial pengganti MINIX sewaktu ia belajar di Universitas Helsinki. Hasil karyanya itu yang kemudian akan menjadi kernel Linux (Torvalds, 2013).

Sekarang ini Linux telah digunakan di berbagai domain, dari *embedded system* sampai superkomputer, dan telah terpercaya dalam instalasi *web server* dengan aplikasi LAMP-nya yang populer. Pengembangan *kernel* Linux masih dilanjutkan oleh Torvalds, sementara Stallman mengepalai Yayasan Perangkat Lunak Bebas yang mendukung pengembangan komponen GNU (GNU sendiri singkatan dari *GNU's Not Unix!)*. Selain itu, banyak individu dan perusahaan yang mengembangkan komponen non-GNU. Komunitas Linux menggabungkan dan mendistribusikan *kernel*, komponen GNU dan non-GNU dengan perangkat lunak manajemen paket dalam bentuk distribusi Linux.

2.3.2 User Interface

User interface di operating sistem Linux, yang dikenal juga sebagai *shell*, dapat berupa sebuah *command line interface* (CLI), bisa berupa *graphical user interface* (GUI), ataupun berupa control terhadap *hardware* yang telah diasosiasikan. Untuk pengguna *desktop*, *mode default*nya adalah *mode* GUI. GUI *shell user interface* pada pengguna *desktop* biasanya telah satu paket dengan *desktop environment* seperti *K Desktop Environment* (KDE), GNOME dan lain-lainnya. Walaupun *default mode*nya adalah GUI *shell user interface* namun pengguna *desktop* tetap dapat mengakses *mode* CLI melalui terminal *emulator window* ataupun melalui *virtual console*. *Shell* CLI adalah *text based user interfaces*, yang mempergunakan teks sebagai masukan dan keluarannya. *Shell* yang umum digunakan pada operating sistem Linux adalah *Bourne-Again Shell* (bash).

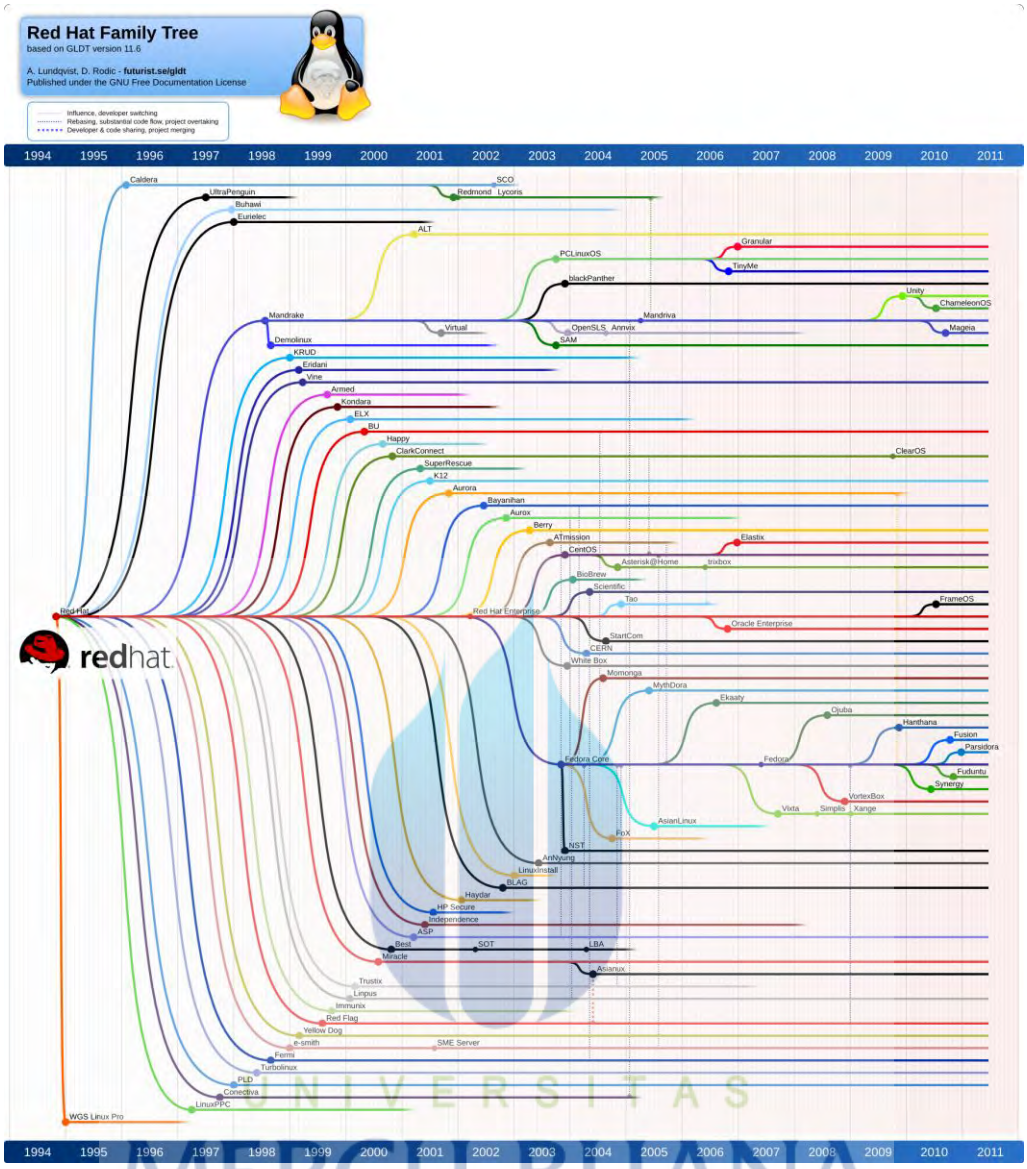
2.3.3 Distro Linux

Terdapat banyak distribusi Linux (lebih dikenali sebagai distro) yang dibuat oleh individu, grup, atau lembaga lain. Masing-masing disertakan dengan program sistem dan program aplikasi tambahan, di samping menyertakan suatu program yang memasang keseluruhan sistem di komputer (*installer program*).

Inti di setiap distribusi Linux adalah *kernel*, koleksi program dari proyek GNU (atau proyek lain), cangkang (*shell*), dan tools utilitas seperti pustaka (*libraries*), kompilator, dan penyunting (*editor*). Beberapa sistem juga menyertakan aplikasi dan utilitas yang bukan-GNU. Utilitas tersebut dapat dipisahkan dan sistem ala UNIX masih tersedia. Beberapa contoh adalah aplikasi dan utilitas dari BSD dan sistem grafik-X adalah *X-Window System*. *X-Window System* ini menyediakan antarmuka grafis (GUI) yang umum untuk Linux.

Contoh-contoh distro Linux :

- *Ubuntu* dan derivatifnya : *Sabily* (*Ubuntu Muslim Edition*), *Kubuntu*, *Xubuntu*, *Edubuntu*, *GoBuntu*, *Gnewsense*, *ubuntuCE*
- *Red Hat* dan derivatifnya : *Oracle Enterprise*, *CentOS*, *Mandrake*
- *OpenSUSE*
- *Fedora*
- *BackTrack*
- *Mandriva*
- *Slackware*
- *Debian*
- *PCLinuxOS*
- *Knoppix*
- *Xandros*
- *Sabayon*
- *CentOS*
- *ClearOS*
- *Chrome OS*
- *Gentoo Linux*



Gambar 2-3 Distro *Red Hat Linux*

2.3.4 *Oracle Enterprise Linux*

Oracle Linux sebelumnya dikenal sebagai *Oracle Enterprise Linux*, merupakan salah satu derivatif distro dari *Red Hat Enterprise Linux* (RHEL). Dipackage dan didistribusikan secara gratis oleh *Oracle Corporation*, tersedia dibawah lisensi GNU *General Public Lisence* (GPL) semenjak akhir tahun 2006. *Oracle Linux* dapat diunduh dari web site milik *Oracle* yaitu, *Oracle E-Delivery* ataupun dari web site mirror-mirrornya. Penggunaan dan pendistribusian Oracle Linux ini adalah gratis. *Commercial technical support* tersedia dari *Oracle Linux Support Program* milik *Oracle*, yang mendukung *Oracle Linux*, *RedHat* ataupun

CentOS (tanpa diharuskan menginstall ulang operating sistem yang telah terinstall). Pada tahun 2013, *Oracle Linux* telah memiliki lebih dari 11,000 pelanggan yang meregistrasikan dirinya di *Oracle Linux Support Program*.

2.3.4.1 *RHEL Compatibility*

Oracle Corporation mendistribusikan *Oracle Linux* dengan dua alternatif *kernel*, yaitu :

- *Red Hat Compatible Kernel* (RHCK), *kernel Oracle Linux* yang sama dengan *kernel Red Hat Enterprise Linux.*
- *Unbreakable Enterprise Kernel* (UEK), *kernel Oracle Linux* yang secara garis besarnya sama dengan *kernel Red Hat Enterprise* Linux, namun disertai dengan beberapa pengembangan dari *Oracle Corporation* yang berkaitan dengan *Oracle* RDBMS, seperti : OLTP, *InfiniBand, SSD disk access, NUMA-optimizations, Reliable Datagram Sockets* (RDS), *async I/O*, OCFS2 dan *networking*

Oracle Corporation menyatakan bahwa *kernel Unbreakable Enterprise Kernel* mempunyai kompatibilitas dengan RHEL, sehingga aplikasi-aplikasi *middleware* Oracle yang mempunyai sertifikasi berjalan di atas kernel RHEL dapat berjalan dengan tanpa hambatan di *kernel Unbreakable Enterprise Kernel Oracle Linux*. Namun untuk user yang menginginkan kompabilitas yang benar-benar sesuai dengan *Red Hat* atau untuk pengguna yang membutuhkan *package kernel* yang spesifik, maka kernel *Red Hat Compatible Kernel* menawarkan tingkat kompabilitas 100% dengan *kernel Red Hat Enterprise Linux*.

2.3.4.2 *Hardware Compatibility*

Hardware yang telah tersertifikasi dengan *Oracle Linux* ini antara lain hardware dari : IBM, *Hewlett-Packard*, *Dell*, *Lenovo*, and *Cisco*. Dan juga *Oracle Linux* tersedia pada *Amazon EC2* sebagai Amazon *Machine Image*, dan pada *Microsoft Windows* Azure sebagai VM *Image*.

2.3.4.3 *Version and Related Products*

Versi terakhir dari *Oracle Linux* adalah versi 7.2, dan saat ini *Oracle Linux* dipergunakan sebagai *Default Operating System* untuk *appliance-aplliance* sebagai berikut :

- *Oracle Exadata*
- *Oracle Exalogic*
- *Oracle Big Data Appliance*
- *Oracle Exalytics*
- *Oracle Database Appliance*

Gambar 2-4 Logo Distro *Oracle Linux*

2.4. *Cluster Computer*

Menurut Krzyzanowski (April 2007), c*luster computer* adalah sekumpulan komputer (umumnya *server* dalam satu jaringan) independen yang beroperasi serta bekerja secara erat dan terlihat oleh klien jaringan seolah-olah komputer-komputer tersebut adalah satu buah unit komputer. Proses menghubungkan beberapa komputer agar dapat bekerja seperti itu dinamakan dengan *Clustering*. Komponen cluster biasanya saling terhubung dengan cepat melalui sebuah interkoneksi yang sangat cepat, atau bisa juga melalui jaringan lokal (LAN).

Karena menggunakan lebih dari satu buah *server*, maka manajemen dan perawatan sebuah cluster jauh lebih rumit dibandingkan dengan manajemen *server mainframe* tunggal yang memiliki skalabilitas tinggi (semacam IBM AS/400), meski lebih murah. Menurut Vugt (2014), pada umumnya komponen-komponen pembentuk suatu *cluster computer* adalah sebagai berikut :

- *Shared storage*

  Adalah media tempat peyimpanan data ataupun konfigurasi yang diperlukan diakses bersama oleh seluruh anggota *cluster*.
- *Different networks* (*public* dan *private network*)

  Pemisahan *network* antara *network* sebagai jalur data antara *cluster* dan pengguna dan *private network* yang berguna sebagai jalur komunikasi antar anggota *cluster*.
- *Bonded network devices*

  Teknologi yang memungkinkan NIC (*Network Card Interface*) dikenali sebagai satu NIC saja, bertujuan untuk redundansi bilamana satu NIC bermasalah masih ada NIC cadangan yang masih berfungsi sebagai penerima/pengirim *packet*.
- *Multipathing*

  Pengenalan LUN (*Logical Unit Number*) dari *SAN storage* oleh anggota-anggota dalam *cluster* melalui path yang berbeda, tujuannya adalah untuk meningkatkan level redundansi.
- *Fencing/STONITH devices*

  STONITH (*shoot the other node in the head*) adalah mekanisme dalam *cluster computer* untuk menjaga konfigurasi *cluster*, yaitu dikeluarkan secara sementara anggota *cluster* bilamana terjadi *split brain*. *Split brain* ini terjadi ketika terpisahnya anggota *cluster computer* dan setiap anggota *cluster computer* "berpikir" bahwa dirinya adalah anggota terakhir dari *cluster computer*.

2.4.1 Tipe *Cluster Computer*

Menurut Vugt (2014), secara umum *cluster computer* dapat digolongkan kedalam tiga tipe yang berbeda.

Ketiga tipe *cluster computer* itu adalah sebagai berikut :

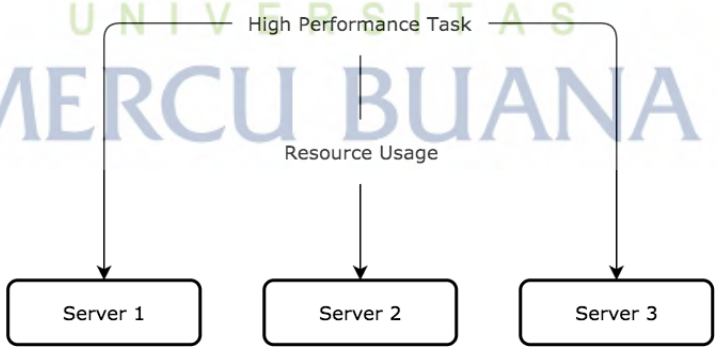
- *High Performance*, beberapa komputer yang bekerja sama dalam mengerjakan satu atau lebih aplikasi yang membutuhkan resos yang banyak.

- *Load Balancing*, dimana sebuah *server* sebagai penerima request dari pengguna, dan kemudian mendistribusikan *request* dari pengguna tersebut ke beberapa *server*.
- *High Availability*, beberapa komputer yang bekerja sama untuk memastikan downtime yang terjadi pada suatu sistem dapat ditekan seminimal mungkin.

2.4.1.1 *High Performance Cluster*

High performance cluster banyak digunakan pada environment dengan beban kerja yang membutuhkan kemampuan komputasi yang tinggi, karena beban kerja yang tidak dapat terselesaikan jika hanya diproses pada satu komputer saja, sehingga dibutuhkan beberapa komputer/server untuk mengerjakannya agar proses/beban kerja yang berjalan dapat terselesaikan pada waktu yang tepat.

Salah satu contoh pendekatan penerapan *high performance cluster* adalah *Single System Image*. Dengan pendekatan ini beberapa *server* diperlakukan sebagai satu kesatuan, dimana alokasi resos di atur sedemikian rupa oleh *software cluster*. *High performance cluster* biasanya hanya digunakan pada *environtment* tertentu, dan penggunaannya tidak sebanyak jenis *high availability cluster*.



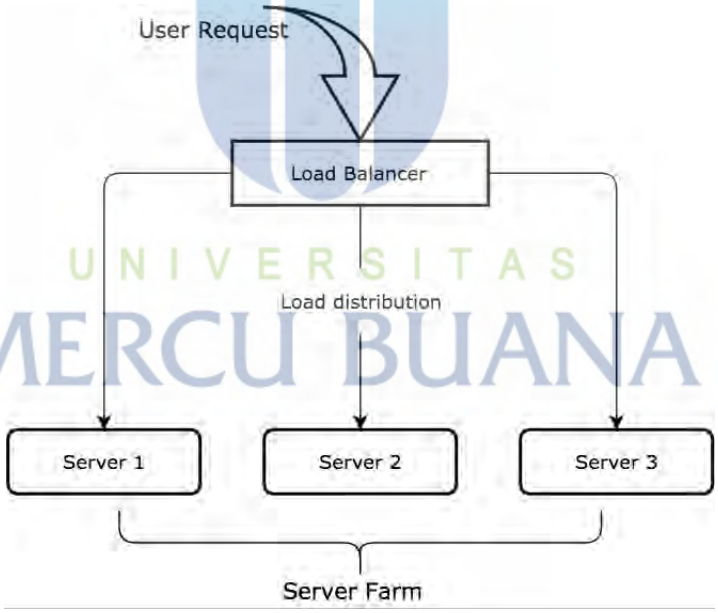
Gambar 2-5 *Overview High Performance Cluster*

2.4.1.2 *Load Balancing Cluster*

Load balancing cluster pada umumnya digunakan pada lingkungan dengan permintaan yang tinggi, contohnya halaman web yang mempunyai *traffic* tinggi yang banyak dikunjungi. Kegunaan dari *load balancing*

cluster ini adalah untuk mendistribusikan pekerjaan/task kepada *server* yang memiliki ketersediaan resos. *Load balancing* selintas memiliki kesamaan dengan *high performance computer*, namun ada perbedaan yang mendasar antara keduanya, yaitu, jika *high performance cluster* mengerjakan satu pekerjaan/task secara bersama-sama, sedangkan pada load balancing cluster lebih mengarah kepada pendistribusian beban kerja, untuk mendapatkan efisiensi dalam pengerjaan pekerjaan/task.

Sebuah *load balancing cluster* terdiri dari 2 komponen utama, yaitu : *load balancer* dan beberapa *server* dibelakangnya. Setiap *request* dari pengguna akan diterima oleh *load balancer* dan seterusnya akan didistribusikan kepada salah satu *server* yang tersedia. Pendistribusian ini dilakukan oleh load balancer setelah melakukan monitoring pada semua *server* anggota cluster, *server* mana yang lebih siap untuk menerima dan memproses request dari user.

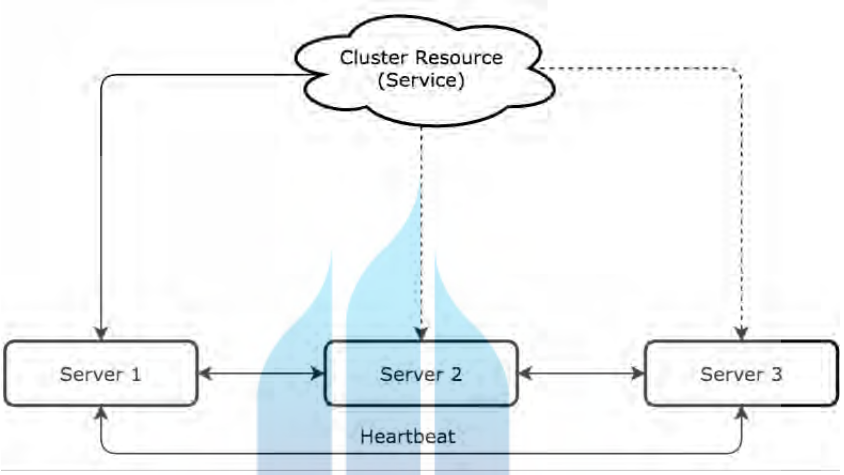


Gambar 2-6 *Overview Load Balance Cluster*

2.4.1.3 *High Availability Cluster*

Goal dari sebuah *high availability cluster* adalah memastikan selalu maksimalnya ketersediaan service dari sebuah *critical resource*. Pada setiap *server* anggota *cluster di install cluster software*, dimana tugas

software cluster ini akan memonitor ketersediaan setiap *server* dan *service cluster* yang berjalan di atasnya. Jika sebuah *server* ataupun *service* yang berjalan mengalami masalah (misalnya *down*), maka *software high availability cluster* akan mengetahuinya dan memastikan bahwa *service* tersebut dapat dijalankan ulang pada *server* anggota *cluster* lainnya, sehingga *service* yang sebelumnya mengalami gangguan dapat sesegera mungkin kembali melayani *request*/permintaan user.



Gambar 2-7 *Overview High Availability Cluster*

Penting untuk dipahami bahwa kegunaan *high availability cluster* adalah untuk memaksimalkan ketersediaan resos, *high availability cluster* tidak dapat menjamin ketersediaan resos yang tanpa adanya interupsi. *High availability cluster* mempunyai mekanisme pendeteksian, sehingga jika terjadi kegagalan fungsi pada salah satu anggota *cluster*, *service* yang menyediakan layanan kepada pengguna akan segera dialihkan kepada anggota *cluster* lainnya, dalam proses pengalihan *service* ini lah terjadi interupsi layanan kepada pengguna. Pada skripsi ini *cluster computer* yang diterapkan adalah tipe *cluster computer* perpaduan *high availability* dan *load balancing* menggunakan *Oracle RAC.*

2.5. *Storage Area Network*

Menurut *Storage Networking Industry Association*, SAN adalah *high-performance network* yang tujuan utamanya untuk memungkinkan perangkat

penyimpanan (*storage devices*) dapat berinteraksi dengan perangkat komputer ataupun berinteraksi dengan perangkat penyimpanan lainnya (Barker dan Massiglia, 2002).

2.5.1 Komponen SAN

Menurut *vmware* (*n.d.*), sebuah *storage area network* terbentuk dari beberapa layer yang juga terdiri dari beberapa komponen yaitu :

i. Host Component

Host component adalah *server* itu sendiri dan komponen yang memungkinkan *server* terhubung secara fisik dengan *storage*. komponen-komponen pada bagian ini adalah :

- HBA (*Host Bus Adapter*), terletak dalam fisik *server* yang mempunyai tugas merubah sinyal digital menjadi sinyal optik.
- HBA *Driver*, *driver* yang bertugas sebagai penghubung komunikasi antara *operating system* pada dan HBA.

ii. Fabric Component

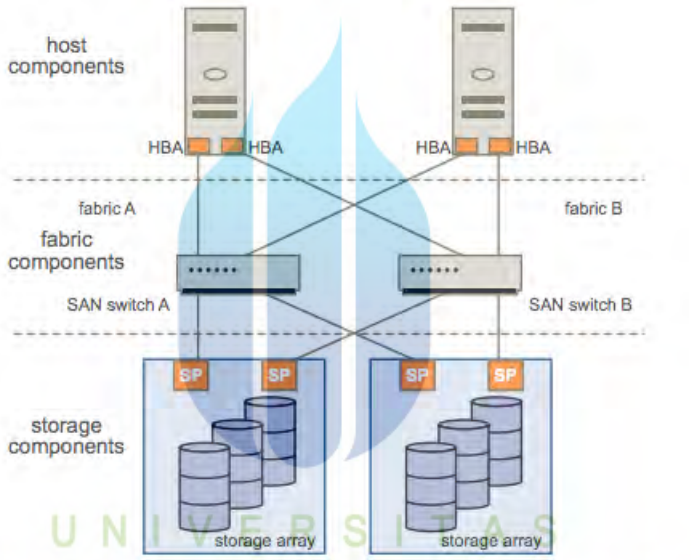
Semua *server* yang terhubung ke perangkat *storage* melalui SAN *fabric*. Komponen pada bagian *fabric* ini adalah :

- *SAN Switch*, adalah perangkat *network* yang menghubungkan antara *server*, perangkat *storage* dan perangkat *switch* lainnya.
- *Cables*, kabel yang menghubungkan semua *fabric* komponen biasanya berupa kabel *fiber optic* namun saat ini dalam perkembangannya banyak SAN storage yang telah mendukung kabel ethernet dengan menggunakan *protocol* iSCSI.
- *Comunication protocol*, pada umumnya *protocol* yang digunakan untuk berkomunikasi adalah *fibre channel protocol disingkat menjadi FCP*. FCP dikembangkan

untuk memungkinkan *transfer* data antar 2 serial port I/O dengan kecepatan yang tinggi.

iii. Storage Component

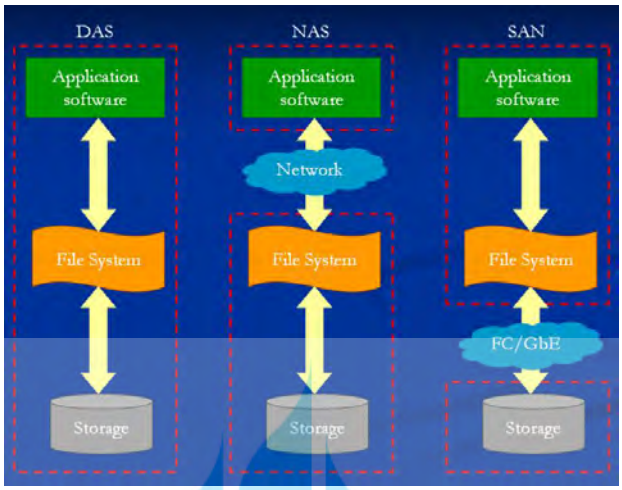
- Komponen pada perangkat *storage* adalah *storage array*. *Storage array* ini termasuk juga *storage prosessor* (SP). SP berkomunikasi dengan *disk array* dan bertugas juga sebagai penyedia fungsi RAID/LUN.
- *Storage device*, pada sebuah *SAN storage*, data tersimpan pada disk array atau tape ataupun kombinasi keduanya.



Gambar 2-8 *Overview SAN Component*

Berbeda dengan NAS (N*etwork Attached storage*) yang menyediakan akses pada level file system dengan menggunakan *protocol CIFS* dan NIFS dan pengiriman data dari storage ke perangkat komputer menggunakan kabel *ethernet* dan *protocol TCP/IP*, *SAN Storage* menyediakan akses pada level *block* mempergunakan *protocol* SCSI dan pengiriman data dari *storage* ke perangkat komputer menggunakan media *fibre channel*, dan saat ini SAN *storage* pun mendukung penggunaan media kabel *ethernet* dengan menggunakan *protocol* iSCSI.

Dalam penerapannya, NAS lebih ditujukan pada penggunaan *file sharing* yang sederhana, sedangkan untuk SAN ditujukan untuk kebutuhan-kebutuhan yang mensyaratkan *performance* i/o yang tinggi.



Gambar 2-9 Perbedaan Cara Kerja DAS, NAS dan SAN

2.6. Virtualisasi

Menurut *Oracle Corporation* (*n.d.*), virtualisasi adalah pembuatan suatu bentuk atau versi *virtual* dari perangkat keras komputer, *operating system*, perangkat penyimpanan, ataupun perangkat jaringan.

Virtualisasi perangkat keras adalah pembuatan suatu mesin *virtual* yang bekerja layaknya sebuah mesin komputer lengkap sistem operasinya. Pada virtualisasi perangkat keras terbagi atas 2 bagian yaitu *host* dan *guest*. Istilah *host* mengacu kepada mesin tempat mesin *virtual* berjalan, sedangkan *guest* adalah mesing *virtual* itu sendiri yang bejalan di atas mesin host.

2.6.1 Jenis Virtualisasi Perangkat Keras

Pada virtualisasi perangkat keras terdapat 3 jenis virtualisasi, yaitu :

- *Full virtualization* (virtualisasi penuh), adalah pembuatan *environment* virtualisasi yang hampir menyerupai perangkat keras aslinya, dan dapat menjalankan setiap perintah tanpa adanya perubahan pada saat eksekusi.
- *Partial virtualization*, adalah type virtualisasi yang hanya membuat simulasi dari sebagian perangkat keras, sehingga

beberapa perintah perlu dimodifikasi untuk dapat berjalan dalam lingkungan *virtual* ini.

- *Para-virtualization*, pada type virtualisasi ini tidak dilakukan simulasi pada perangkat keras, namun aplikasi pada mesin guest dapat tetap berjalan seolah dalam domain tersendiri. Perintah dari guest akan dimodifikasi untuk dapat berjalan tipe virtulasasi ini.

Environment testing dan *prototype* pada laporan skripsi ini dibuat pada lingkungan virtualisasi jenis *full virtualization* dengan menggunakan aplikasi *Oracle Virtualbox* versi 5.0.16.

2.7. DNS

Domain name system atau yang biasa disingkat dengan DNS merupakan sebuah sistem yang berfungsi menterjemahkan alamat IP ke nama domain atau sebaliknya, dari nama domain ke alamat IP (Jeftovic, 2014). Jadi, host komputer mengirimkan *queries* berupa nama komputer dan domain name *server* yang kemudian dipetakan ke dalam alamat IP oleh DNS.

Sebagai contoh, ketika diketikkan sebuah alamat suatu *website* misalkan : detik.com, maka DNS akan menterjemahkannya ke dalam alamat IP : 203.190.242.69 agar dapat dimengerti oleh komputer.

DNS biasanya digunakan pada aplikasi yang terhubung pada internet seperti *web browser* maupun pada sebuah layanan email. Selain itu, DNS juga dapat di terapkan pada *private network* maupun *intranet*.

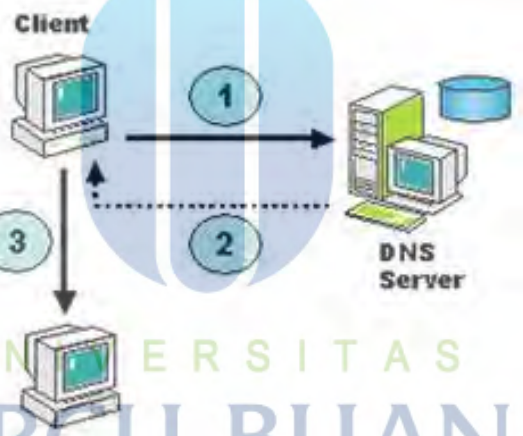
Berikut kelebihan yang dimiliki oleh DNS : dengan menggunakan DNS, pengguna tidak perlu lagi menghafalkan alamat IP dari sebuah komputer maupun situs pada jaringan *internet*. Cukup menghafalkan *host name* atau nama domainnya saja. Bisa jadi alamat IP pada sebuah komputer bisa berubah, tetapi host name (nama komputer) tidak dapat berubah. Maka dari itu, DNS cenderung konsisten.

2.7.1 Cara Kerja DNS

DNS menggunakan relasi *client–server* untuk resolusi nama. Pada saat *client* mencari satu *host*, maka ia akan mengirimkan *query* ke *server* DNS.

Query adalah satu permintaan untuk resolusi nama yang dikirimkan ke *DNS server*.

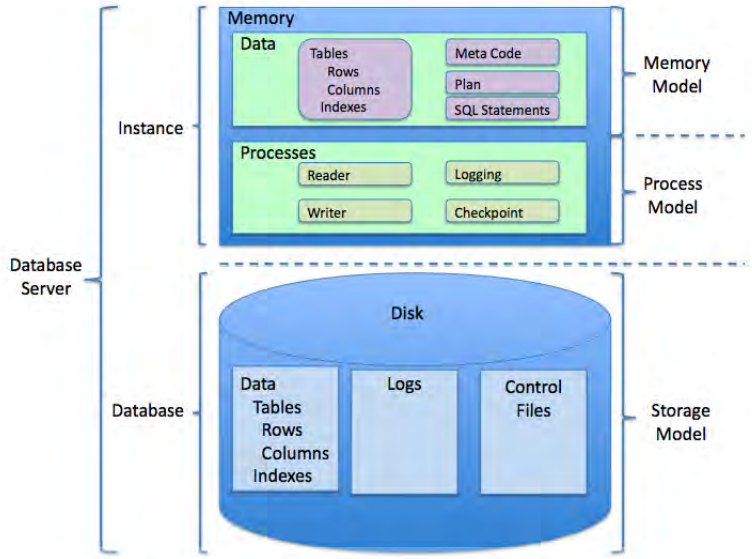
1. Pada komputer *client*, sebuah program aplikasi misalnya http, meminta pemetaan *IP Address* (*forward lookup query*). Sebuah program aplikasi pada host yang mengakses *domain system* disebut sebagai *resolver*, *resolver* menghubungi *DNS server*, yang biasa disebut *name server*.
2. *Name server* meng-cek ke *local database*, jika ditemukan, *name server* mengembalikan *IP Address* ke *resolver* jika tidak ditemukan akan meneruskan *query* tersebut ke *name server root server*.
3. Terakhir barulah si client bisa secara langsung menghubungi sebuah *website* / *server* yang diminta dengan menggunakan *IP Address* yang diberikan oleh *DNS server*.



Gambar 2-10 Cara Kerja DNS

2.8. RDBMS

RDBMS adalah kependekan dari *Relational Database Management System*. RDBMS adalah program yang melayani sistem basis data yang entitas utamanya terdiri dari *tabel-tabel* yang mempunyai relasi dari satu *tabel* ke *tabel* yang lain. Istilah *Relational Database* ditemukan oleh E. F. Codd, seorang peneliti di laboratorium IBM San Jose pada tahun 1970, dan pertama kali dikemukakan pada seminar papernya dengan judul "*A Relational Model of Data for Large Shared Data Banks*".



Gambar 2-11 Struktur Umum Sebuah RDBMS

Pada tahun 1974, IBM mulai mengembangkan *System* R, sebuah *prototype RDBMS*, namun versi komersial RDBMS pertama yang ada dipasaran adalah *Oracle* pada tahun 1974 oleh sebuah perusahaan yang bernama *Relational* yang saat ini telah berganti nama menjadi *Oracle Corporation*. Selain *Oracle Database*, beberapa contoh dari RDBMS lainnya yaitu : *DB2*, *SAP Sybase ASE*, *Informix* dan *SQL Server*.

2.9. RDBMS *Oracle*

2.9.1 Sejarah *Oracle Database*

Oracle RDBMS yang dikenal sebagai *Oracle Database* atau hanya *Oracle* saja, adalah sebuah produk dari *Oracle Corporation*. Larry Elison dan dua rekannya, Bob Miner dan Ed Oates, memulai sebuah perusahaan konsultan yang bernama *Software Development laboratories* (SDL) pada tahun 1977. Pada perusahaan SDL inilah *Oracle Database* pertama kali dikembangkan.

2.9.2 Edisi *Oracle Database*

Saat ini tersedia beberapa Edisi *Oracle Database*, dari edisi yang paling prestisius sampai dengan edisi tidak berbayar yang cukup sederhana, namun satu keunggulan dari *Oracle Database*, walaupun

pengguna hanya menggunakan *Oracle Database* edisi tidak berbayar, namun fungsionalitas *Oracle Database* yang mumpuni tetap dapat digunakan. Edisi-edisi *Oracle Database* adalah sebagai berikut :

- *Oracle Enterprise Edition*

  Edisi *Oracle Database* paling prestisius yang penggunaannya ditujukan untuk perusahaan kelas *enterprise*.
- *Oracle Standard Edition*

  Edisi *Oracle Database* dengan dukungan penggunaan hingga 4 *prosessor*, termasuk juga dukungan untuk penggunaan *Real Application Cluster*.
- *Oracle Standard Edition One*

  Edisi Standar *Oracle Database* dengan dukungan penggunaan hingga 2 *prosessor* dengan harga pasaran dibawah *Oracle Database Edisi Standar*.
- *Oracle Personal Edition*

  Edisi *Oracle Database* untuk penggunaan pribadi, kompatible dengan semua teknologi *Oracle Database* kecuali *Real Application Cluster*.
- *Oracle Express Edition (Oracle XE)*

  Edisi *Oracle Database* versi gratis/tidak berbayar namun penggunaannya terbatas pada 1 *prosessor*, 1 GB RAM dan 11 GB data.

Saat ini *Oracle Corporation* telah mengembangkan *Oracle database* sampai dengan versi 12c.

2.9.3 Arsitektur *Oracle Database*

Menurut *Oracle Corporation* (2016), sebuah *Oracle Database server* terdiri dari sebuah *database* dan sekurang-kurangnya sebuah *database instance*, yang umumnya hanya disebut sebuah *instance*. Karena sebuah *instance* dan sebuah *database* saling terhubung tanpa bisa dipisahkan, maka kata *Oracle Database* digunakan untuk merujuk kepada keduanya, yaitu *instance* dan *database*.

Pengetian dari *database* dan *instance* adalah sebagai berikut :

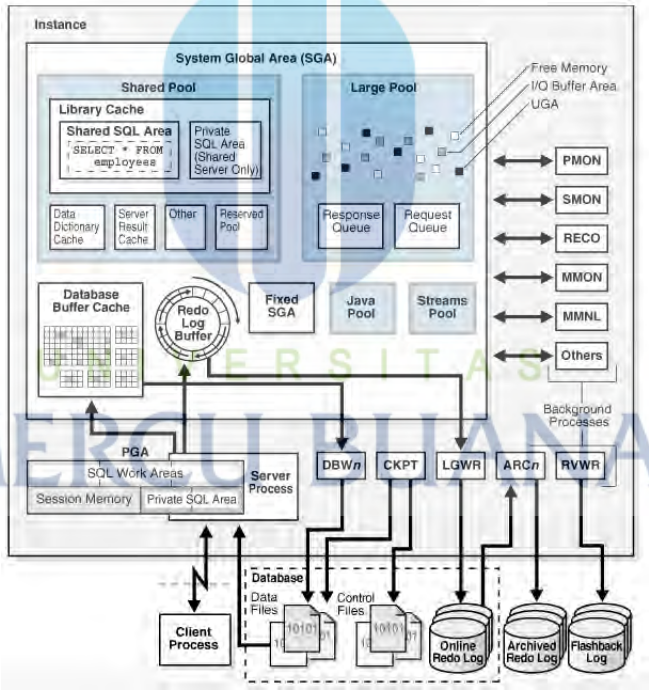
- *Database* :

  Dalam sebuah *Oracle Database server*, sebuah *database* adalah sekumpulan file yang tersimpan pada media penyimpanan, contohnya pada disk.

- *Instance* :

  Sebuah *instance* adalah seperangkat struktur *memory* yang mengelola *database file*. Sebuah instance terdiri dari *shared memory area* yang disebut dengan SGA (*System Global Area*) dan satu set *background proses*.

Gambar di bawah menunjukkan sebuah *database* dan *instance*. Setiap pengguna yang terhubung pada *instance* terkait dengan sebuah *server process*. *Setiap server process* mempunyai *private session memory* tersendiri yang dikenal sebagai *Program Global Area* (PGA).



Gambar 2-12 *Oracle Instance* dan *Database*

2.9.4 *Oracle RAC* (*Real Application Cluster*)

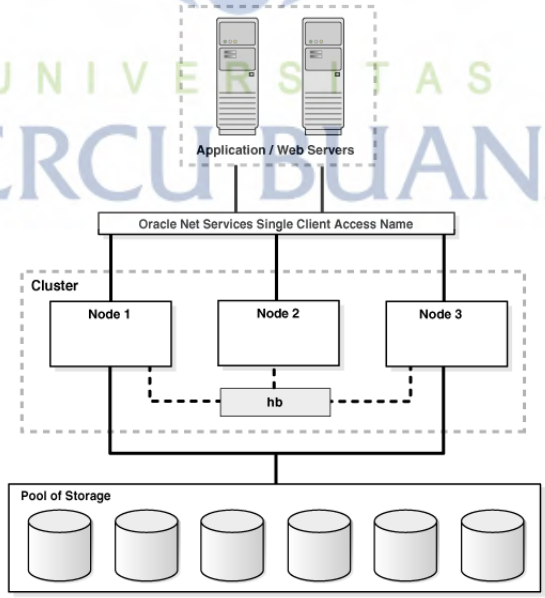
Berkaitan dengan topik skripsi ini yang membahas mengenai penerapan *cluster computer*, *Oracle Corporation* melalui *Oracle Database* telah mengembangkan *fitur cluster* yaitu *Oracle RAC*. Sebelum versi *Oracle 9i*, *Oracle Database* mempunyai *Oracle Parallel Server* (OPS),

namun karena beberapa kekurangannya, salah satunya karena sulitnya mengkonfigur, fitur OPS ini menjadi kurang terkenal dan kurang banyak dimanfaatkan oleh pengguna. Kemudian pada *Oracle Database* versi 9i diperkenalkan fitur *Oracle Real Application Cluster*. Fitur ini mempunyai banyak kelebihan dan kemudahan dibandingkan dengan *Oracle Parallel Server*, terbukti dengan semakin banyaknya pengguna yang memanfaatkan fitur *Oracle Real Application Server* ini.

2.9.5 Arsitektur *Oracle RAC*

Menurut Mike Ault dan Madhu Tumma (2004), secara garis besar, arsitektur *Oracle RAC* terdiri dari komponen-komponen sebagai berikut :

- *Server* atau *node* atau *host*
- Interkoneksi dan protokolnya
- *Oracle Clusterware*
- *Oracle Instance* dan *cache fusion*
- *Shared storage*
- *Clustered file system* (rekomendasi *Oracle* menggunakan ASM)
- *Network service*
- *Workload Management Services - Virtual IP configuration*



Gambar 2-13 *Oracle RAC* Arstitektur

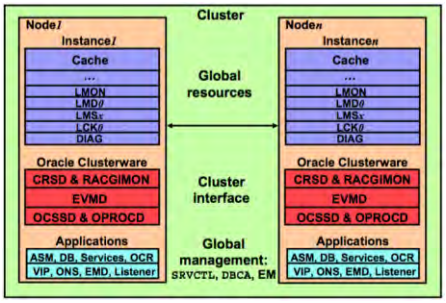
2.9.6 *Oracle RAC Clusterware*

Oracle RAC memiliki seperangkat *background process* yang bertugas untuk menjamin kelangsungan operational sebuah *environment Oracle RAC*. *Background proses* yang dimiliki oleh *Oracle Database* dengan konfigurasi RAC jauh lebih komplek bila dibandingkan dengan *Background proses* sebuah *Oracle Database* dengan konfigurasi *single instance*, karena dibutuhkan mekanisme kerjasama antar node/server dalam lingkungan RAC dan pekerjaan-pekerjaan lainnya yang tidak terdapat pada konfigurasi *Oracle Database single instance*. *Background proses* yang bertugas untuk menjamin keberlangsungan keselararasan (koherensi) kerjasama antar instance dalam *Oracle Database RAC* disebut sebagai *global resources*, yaitu sebagai berikut :

- LMON*: Global Enqueue Service Monitor*
- LMD0*: Global Enqueue Service Daemon*
- LMSx*: Global Cache Service Processes*
- LCK0*: Lock process*
- DIAG*: Diagnosability process*

Sedangkan pada level *cluster*, beberapa proses bertugas sebagai *cluster interface* dan juga bertanggung jawab dalam menyediakan layanan *high availability*. Proses-proses tersebut adalah :

- CRSD dan RACGIMON: Sebagai *engine* operasional *high-availability*.
- OCSSD: Sebagai penyedia akses ke node anggota *cluster* dan akses ke sekumpulan *service*.
- EVMD: Sebagai jembatan antara pendeteksian *event* dan reaksinya.
- OPROCD: Proses yang berfungsi sebagai monitor terhadap *cluster*.



Gambar 2-14 *Oracle Clusterware Configuration*

2.9.7 Manfaat *Oracle RAC*

Dengan teknologi *cluster*nya *Oracle RAC* menawarkan beberapa manfaat bagi pengguna baik itu pengguna dengan tipe aplikasi OLTP maupun pengguna dengan aplikasinya yang bertipe DSS, manfaat-manfaat tersebut yaitu :

- *High availability*, bila salah satu node/anggota *cluster down*, maka node lainnya akan siap untuk menerima dan melanjutkan proses dengan beberapa catatan tertentu.
- *Speedup* (meningkatkan *response time*), untuk tipe aplikasi tertentu, contohnya DSS (*Decision Support System*), dengan pemanfaatan fitur *parallel query*, proses yang berjalan dapat ditingkatkan *performance*nya sehingga menghasilkan *response time* yang lebih baik.
- *Scaleup* (meningkatkan jumlah transaksi), manfaat ini khususnya ditujukan pada tipe aplikasi OLTP, dimana transaksi-transaksi pada aplikasi OLTP cenderung bersamaan dalam jumlah yang besar namun dalam waktu yang singkat, dengan penggunaan *Oracle RAC* ini aplikasi bertipe OLTP mendapatkan manfaatnya sehingga dapat meningkatkan jumlah/volume transaksi.
- *Load balancing*, dengan menggunakan *Oracle RAC* maka dimungkinkan untuk melakukan distribusi beban kerja pada semua anggota *cluster*.

Workload	Speedup	Scaleup
OLTP and Internet	No	Yes
DSS with parallel query	Yes	Yes
Batch (mixed)	Possible	Yes

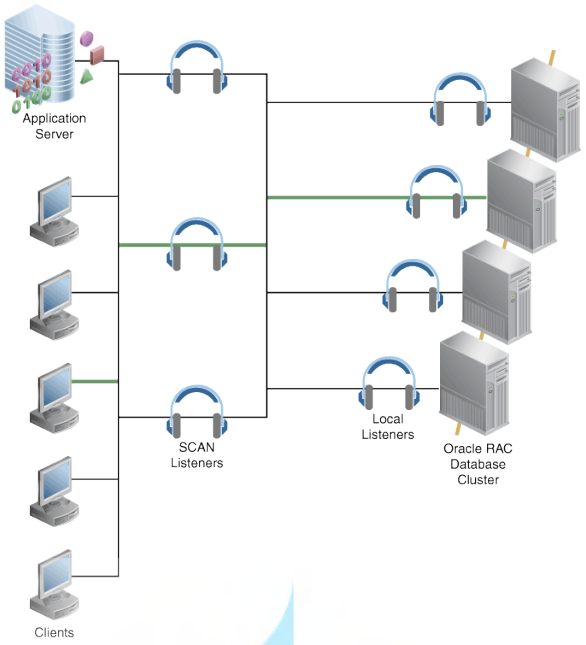
Gambar 2-15 *Oracle RAC speedup/scaleup* dan *workload*

Pada skripsi ini akan dibahas penerapan fitur *Oracle RAC*, yaitu *load balancing* dan *high availability*, dan migrasi data menggunakan *Oracle Data Guard*

2.9.8 *Scan Listener*

SCAN *(Single Client Access Name)* adalah sebuah fitur dalam lingkungan *Oracle RAC Database* yang menyediakan akses tunggal bagi pengguna/client untuk dapat melakukan koneksi kedalam *database*. SCAN ini dapat dikatakan juga sebagai alias bagi sebuah *Oracle RAC database,* keuntungan menggunaka*n SCAN listener* ini adalah konfigurasi SCAN tidak akan mengalami perubahan walaupun telah dilakukan perubahan konfigurasi pada sisi *database*nya, selain juga selayaknya fungsi translasi pada DNS, SCAN juga memudahkan pengguna untuk mengakses *database* karena hanya perlu mengingat nama SCAN-nya saja. *SCAN listener* ini pertama kali diperkenalkan pada *Oracle RAC Database* versi 11.2, dengan konfigurasi umum sebuah SCAN terdiri dari 3 *ip address* yang didaftarkan ke DNS *server*. Contoh *connection string* dengan menggunakan SCAN : *sqlplus system/s3st3m@rac-scan.db.local:1521/racdb*

Cara kerja *SCAN listener* adalah sebagai berikut : pada saat pengguna akan melakukan koneksi kedalam *database*, akan dilakukan *resolve* dari nama scan menjadi *IP address* oleh *DNS server*. Pengguna kemudian melakukan koneksi ke salah satu *IP adrress* dari *SCAN listener* yang telah diberikan oleh *DNS server*, setelah terhubung dengan *SCAN listener*, kemudian *SCAN listener* akan meminta LBA *(Load Balance Advisor)* untuk memberikan rekomendasi instance mana yang siap menerima koneksi, respon dari LBA kemudian diteruskan oleh *SCAN listener* ke pengguna, dan sebagai hasil akhir dari proses koneksi ini, pengguna akan melakukan koneksi kedalam *database* menggunakan *IP address Local Listener* yang diberikan oleh *SCAN listener*.



Gambar 2-16 *Overview SCAN Listener*

2.9.9 *High Availability dan Load Balancing*

Workload management pada Oracle RAC terdiri dari 2 komponen utama, yaitu :

1. *Failover*, yaitu jika koneksi kepada salah satu *instance* dimana *instance* tersebut mengalami kegagalan/masalah, maka *Oracle RAC Database* akan melakukan *failover* koneksi kepada *instance* lainnya yang masih berjalan dengan normal.
2. *Load balancing*, pendistribusian beban kerja pada *Oracle RAC Database*.

Mekanisme *failover* dilakukan pada proses pengguna yang karena satu dan lain hal instance database mengalami kegagaln fungsi, sedangkan mekanisme *load balancing* dilakukan pada 2 jenis aktivitas, yaitu pada saat pengguna akan melakukan koneksi dan pada saat aktivitas pengguna tengah berjalan pada *Oracle RAC database*.

Fitur *failover* dan *load balancing* ini keduanya dapat dikonfigurasi pada sisi *server* maupun pada sisi *client*. Untuk *best* practise-nya, fitur *failover* direkomendasikan untuk selalu melakukan konfigur di sisi *server*, dengan pertimbangan *server* mempunyai fungsionalitas lebih baik bila dibandingkan dengan *client*, dan perubahan pada sisi *server* lebih

memberikan dampak menyeluruh dibanding bila harus melakukan perubahan konfigurasi pada sisi *client* satu per satu.

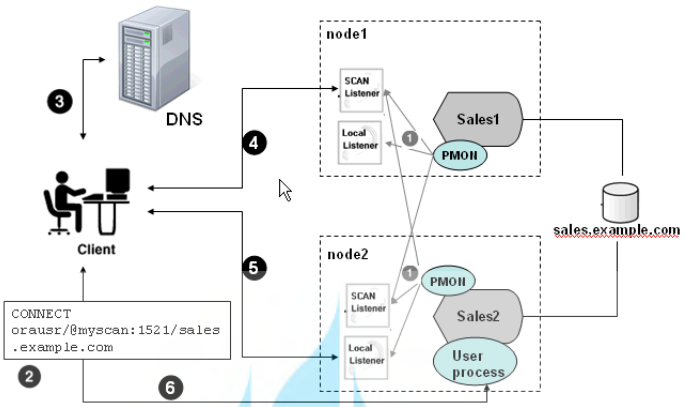
2.9.9.1 *Load Balancing*

Dalam suatu *environment Oracle RAC database* fitur *load balancing* mempunyai peran yang krusial yaitu melakukan pendistribusian koneksi dari *client* ke *instance-instance* anggota *cluster*. Dalam proses pedistribusian koneksi ini, *scan listener* dibantu oleh LBA *(Load Balance Advisor)*, untuk menentukan *instance* mana yang lebih siap dalam untuk menerima koneksi dari client/pengguna.

Ada dua tipe *load balancing* :

1. *Connect Time Load Balancing* (CLB), tipe *load balancing* yang melakukan pendistribusian koneksi pada saat *client*/pengguna akan melakukan koneksi ke *database*. *Connect time load balancing* ini dapat dikonfigur dari sisi *server* ataupun dari *client*. Terdapat 2 tipe CLB, yaitu :
    - *Short*, ditujukan untuk tipe aplikasi OLTP dengan waktu session yang benar-benar singkat, contohnya adalah aplikasi trading.
    - *Long*, ditujukan untuk tipe aplikasi dimana pengguna yang terhubung kedalam database dalam waktu yang cukup lama, bisa dalam hitungan menit ataupun jam.
2. *Run Time Load Balancing* (RTLB), adalah tipe *load balancing* dimana pengguna telah terhubung ke database. tipe RTLB dibagi ke dalam 2 goal, yaitu :
    - *Service_time*, yang ditujukan untuk tipe aplikasi OLTP.
    - *Throughput*, yang ditujukan untuk tipe aplikasi dengan proses *batch prosessing*, yaitu aplikasi yang merupakan gabungan antara OLTP dan DSS *workload*.

Pada penerapannya *run time load balancing* bekerja sama dengan aplikasi dengan cara mengirimkan informasi berupa *perfomance statistics* yang sebelumnya dikumpulkan pada periode waktu tertentu oleh LBA. Secara default aplikasi yang berbasis JDBC dapat memanfaatkan fitur ini (*Oracle Corporation, n.d.*).



Gambar 2-17 *Oracle RAC Load Balancing Overview*

Dari gambar di atas langkah-langkah dilakukannya load balancing pada *connect time load balancing* sebagai berikut :

1. *Background process* PMON pada setiap *instance* melakukan *dynamic registration database services* dengan *default local listener* dan dengan setiap *SCAN listener*, dimana *SCAN listener* ini tercatat sebagai *REMOTE_LISTENER* pada database parameter.
2. Pengguna/client melakukan koneksi ke *database* dengan menggunakan SCAN name : *myscan:1521/sales.example.com*
3. Mekanisme *resolver* pada komputer pengguna/*client* akan melakukan proses permintaan *resolve* ke *DNS server*.
4. Dari *DNS server* pengguna mendapatkan *IP address* untuk *SCAN listener* yang berada pada *instance SALES1*, setelah terhubung kemudian *SCAN listener* dengan bantuan LBA mendapat informasi beban kerja antara i*nstance SALES1* dan *SALES2*. Dan didapatkan informasi bahwa *SALES2*

tengah berada pada kondisi beban kerja yang lebih minim dibandingkan dengan *instance SALES1*.

5. Dari *SCAN listener*, pengguna mendapatkan *IP address* dari local listener pada SALES2.
6. Selanjutnya pengguna melakukan koneksi ke database.

2.9.9.2 *Failover*

Fitur high availability pada *Oracle RAC database* yang diakomodasi oleh TAF (*Transparent Application Failover*), memungkinkan pengguna/*client* untuk dapat tetap terhubung kepada *database* (*instance* yang masih tersedia) setelah terjadi kegagalan pada *instance database* dimana pengguna sedang terhubung (*Oracle Corporation, n.d.*).

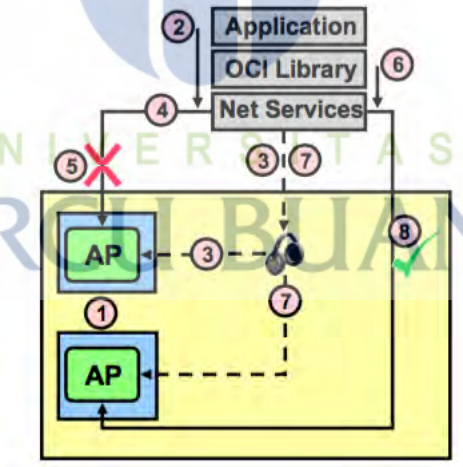
TAF menyediakan pilihan *failover* untuk level transaksi (*select*) ataupun *level session* :

1. *Select failover*, dengan opsi ini, *Oracle RAC database* melalui komponen *Oracle Net* melakukan monitoring pada *statement select*, dilakukan penelusuran berapa banyak jumlah *row data* yang telah dikirim ke sisi *client*/pengguna sebagai respon dari *statement select* dari pengguna. Pada saat pengirim data, bila mana terjadi kegagalan pada *instance database*, maka *Oracle Net* akan melakukan *failover* koneksi ke instance yang masih berjalan dengan normal, lalu melanjutkan *statement select* dari pengguna dengan posisi *cursor* terakhir. Pendekatan *select failover* ini umumnya diterapkan pada tipe aplikasi yang banyak melakukan *query*, misalnya : pembuatan laporan.
2. *Session failover*, dengan opsi ini, *Oracle RAC database* melalui komponen *Oracle Net* melakukan *failover* koneksi ke *instance* yang masih berjalan dengan normal, namun yang di*failover* hanya *session*nya saja, sedangkan aktivitas yang sedang dilakukan (misalnya *select*) akan terhenti. Pendekatan *select failover* ini umumnya diterapkan pada

tipe aplikasi OLTP, dimana transaksi hanya berjalan dalam periode yang sangat singkat.

Dalam melakukan *failover* transaksi, TAF menawarkan 2 pilihan, yaitu :

- *Basic failover*, pada opsi ini koneksi dialihkan pada *instance* yang masih berjalan dengan normal (*backup node*) hanya setelah terputus dari koneksi *primary*. Pada pendekatan ini memerlukan *overhead* yang minim, namun dampaknya pada pengguna adalah saat terjadi peralihan koneksi, pengguna akan mengalami *delay*.
- *Preconnect failover*, pada opsi ini saat pengguna melakukan koneksi ke *database*, secara simultan akan dibuat koneksi keseluruh *instance* anggota *Oracle RAC database*. Pada pendekatan ini menawarkan proses *failover*/peralihan koneksi yang lebih cepat karena backup koneksi telah tersedia/terbentuk sebelumnya, namun mekanisme ini memerlukan tambahan *resource*.



Gambar 2-18 *Oracle RAC TAF Overview*

Dari gambar di atas langkah-langkah dilakukannya *load balancing* pada *connect time load balancing* sebagai berikut :

1. *Startup database service* pada setiap *node* anggota *cluster*.
2. *Client* melakukan inisiasi koneksi ke *database*.

3. *Client* melakukan telah terhubung ke *database*.
4. *Client* melakukan aktivitasnya (*select*, *insert*, *delete*).
5. Karena suatu dan lain hal, koneksi ke *database* terputus.
6. Aplikasi yang digunakan pengguna/*client* akan menerima pesan error pada saat akan melakukan aktivitas ke database.
7. OCI *driver* pada fungsi *failover* akan mengarahkan koneksi dari pengguna yang telah terputus dari instance ke *instance* yang masih berjalan dengan normal.
8. *Client* melanjutkan aktivitasnya.

2.9.10 *Oracle Data Guard*

Oracle Data Guard adalah salah satu solusi dari *Oracle Corporation* untuk senantiasa menjaga *high availability*, *data protection*, dan *disaster recovery* untuk *enterprise data*. *Oracle Data Guard* menyediakan seperangkat layanan yang komprehensif dalam membuat, menjaga dan memonitor satu ataupun beberapa *standby database* yang ditujukan untuk tetap menjaga kelangsungan operational *Oracle database* dari kegagalan/kerusakan akibat bencana ataupun terjadinya data yang corrupt. *Standby database* adalah *copy* dari *primary*/*production database*. Jika *production database* mengalami kegagalan/kerusakan yang disebabkan oleh *downtime* yang terencana ataupun *downtime* yang tidak terencana, *Oracle Data Guard* dapat melakukan pertukaran *role*, dari *standby database* menjadi *primary database*, sehingga *downtime* yang terjadi dapat diminimalisir dan dampak positip yang dapat dirasakan oleh pengguna adalah tetap dapat diaksesnya layanan dengan interupsi yang minimal (*Oracle Corporation, n.d.*).

2.9.10.1 *Data Guard Configuration*

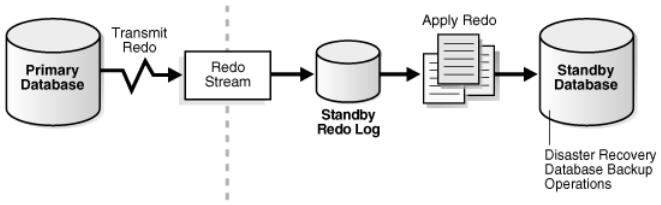
Konfigurasi *Oracle Data Guard* terdiri dari sebuah *primary database* dan satu ataupun beberapa *standby database* dalam satu lokasi yang sama ataupun terpisah secara geografis dengan jarak yang terhitung jauh jaraknya, bahkan menurut *Oracle Corporation* dalam dokumennya yang dimuat pada *http://www.oracle.com/technetwork/database/features/availability/*

dataguardremotemirroring-086151.html, konfigurasi *Data Guard* tidak terbatas oleh jarak. Secara teori sebuah konfigurasi *Data Guard* dapat melayani sebuah *primary database* dan 30 *standby database*. Sebuah konfigurasi *Data Guard* terdiri dari :

- *Primary database* dalam konfigurasi *Oracle Data Guard* adalah sebuah *database* baik itu *single instance database* ataupun *multi instance database* (RAC) yang menjadi *target back-end database* dari sistem yang ada.
- *Standby database* adalah *database* yang senantiasa terjaga konsistensi transaksinya dengan *primary database*.

Sebuah *standby database* dapat berupa :

1. *Physical standby database*, *copy* dari *primary database* yang berada dalam fisik *server* yang berbeda, dengan struktur *logical standby database* yang identik dengan struktur *logical primary database*. proses sinkronisasi transaksi dengan mengirimkan *redolog* dari *primary database* ke *standby database*. Tipe *standby database* inilah yang nanti akan dimanfaatkan dalam migrasi *database* dengan metode *minimum downtime*.
2. *Logical standby database*, *copy* dari *primary database* yang berada dalam fisik yang sama dengan *primary database* dengan struktur *logical* yang umumnya berbeda dengan struktur *logical primary database*. proses sinkronisasi transaksi dengan mengirimkan *sql statement* dari *primary database* ke *standby database*.
3. *Snapshot standby database*, adalah sebuah *physical standby database* yang dikonversikan sebagai sebuah *database* yang siap diakses oleh pengguna/user. Selama *standby database* dirubah menjadi *snapshot standby database*, tidak terjadi sinkronisasi sampai dengan *standby database* dikonversikan kembali menjadi *physical standby database*



Gambar 2-19 *Typical Oracle Data Guard Configuration*

2.9.10.2 *Data Guard Services*

Oracle Data Guard dalam melakukan fungsinya terdiri dari 3 *service*, yaitu :

1. *Redo transport services*, mengontrol pengiriman *redo data* dari *primary database production* ke *standby database*, melakukan kontrol terhadap terjadinya *gap archivelog* antara *primary database* dan *standby database* yang biasanya diakibatkan oleh kegagalan fungsi *network*.
2. *Apply services*, bertanggung jawab untuk menuliskan *redo data* yang diterima dari *primary database* ke *standby database* untuk menjaga konsistensi antara *primary database* dan *standby database*.
3. *Role transitions*, pada *Oracle Data Guard* yang terdiri dari *database-database* yang berperan sebagai *primary* atau *standby*. Peran (*role*) ini dapat dirubah antara *switchover* ataupun *failover*, dimana :
   - *Switchover*, pertukaran peran (*role*) antara *primary* dan *standby*. *Switchover* pada umumnya terjadi pada *planned downtime*, dimana database yang asalnya berperan sebagai *primary* bertukar menjadi *standby* dan sebaliknya, *database* yang asalnya berperan sebagai *standby* bertukar menjadi *primary*, dan kedepannya dapat kembali dilakukan pertukaran kembali
   - *Failover*, berbeda dengan *switchover*, pada *failover* setelah pertukaran tidak dapat ditukar kembali karena

setelah *failover* peran *standby database* akan menjadi *primary database*, dan *original primary database* tetap sebagai *primary database*. Pada *failover* biasanya terjadi pada *unplanned downtime*. Pada metode migrasi dengan metode *minimum downtime* dipakai metode *failover* dengan *prosedur* yang telah diuji pada lingkungan *prototype* sehingga dapat dipastikan tidak terjadi *data loss*.

2.10. *High Availability*

High availability adalah karakteristik suatu sistem yang menjamin suatu level operational *performance* yang telah disetujui yang melebihi dari level operasional *performance* pada umumnya. Sebuah system yang mempunyai level operational yang mempunyai *high availability* apabila memenuhi setidaknya karakteristik-karakteristik di bawah ini :

- Menghilangkan potensi *single point of failure*, dengan cara menambahkan level redundansi pada komponen-komponen sistem sehingga kegagalan fungsi pada suatu komponen tidak menyebabkan kegagalan fungsi seluruh sistem.
- Deteksi masalah, jika terjadi masalah komponen-komponen pada sistem *high availability* dapat dengan cepat dideteksi sehingga tidak berlarut-larut yang akhirnya menyebabkan kegagalan seluruh sistem.

Terdapat beberapa kelas *high availability* yaitu sebagai berikut :

Tabel 2-2 *High Availability Classification*

Class	***Level***	***Annual Downtime***
Continuous	100%	0
Fault tolerant	99.999%	5 *minutes*
Fault resilient	99.99%	53 *minutes*
High availablity	99.9%	8.8 *hours*
Normal availablity	99-99.5%	1.8-3.6 *days*

2.11. Studi Literatur

Selain rujukan-rujukan di atas penulis juga merujuk pada beberapa jurnal yang membahas topik mengenai *Oracle RAC*. Kesimpulan-kesimpulan yang diperoleh dari jurnal-jurnal tersebut adalah sebagai berikut :

1. Jurnal menujukkan fitur *automatic failover* dan *load balancing* pada *Oracle Real Application Cluster* (RAC). Sebuah sistem *Oracle RAC* dapat melindungi dari kerusakan yang diakibatkan oleh kegagalan fungsi berupa kerusakan *hardware*, *operating system* ataupun *server crash*. Ketika sebuah *node*/*server* mengalami kerusakan/kegagalan fungsi maka koneksi akan dialihkan ke *node*/*server* yang masih berjalan dan pengguna dapat meneruskan aktivitasnya tanpa menyadari bahwa koneksinya telah dialihkan (Kadam., *et al.,* 2011). Jurnal juga menjelaskan bahwa arsitektur *Oracle RAC* adalah berbasis *shared disk architecture*.
2. Jurnal menyarankan bahwa dalam penggunaan fungsi dari *load balancing Oracle RAC* harus disertai juga sebuah fungsi *monitoring*, fungsi yang akan melakukan monitor pada jumlah *session* yang terhubung pada database pada setiap *node*/*instance*nya, fungsi monitoring ini terutama ditujukan pada jam-jam sibuk. Mekanisme monitoring ini juga diharapkan dapat melakukan pengiriman email ke personel DBA, setiap terjadi peningkatan jumlah sesssion yang mendekati jumlah maksimum session yang telah terdefinisi, maka email otomatis terkirim ke personel DBA (Chandrima., *et al.,* 2014).

BAB 3
ANALISA SISTEM

Metodologi yang akan digunakan dalam penerapan *high availability cluster* pada infrastruktur *database* menggunakan *Oracle RAC* mengacu pada metode SDLC model *prototyping* akan meliputi fase-fase sebagai berikut, yaitu :

- Analisa
- Perancangan
- *Prototyping*
- Implementasi dan pengujian

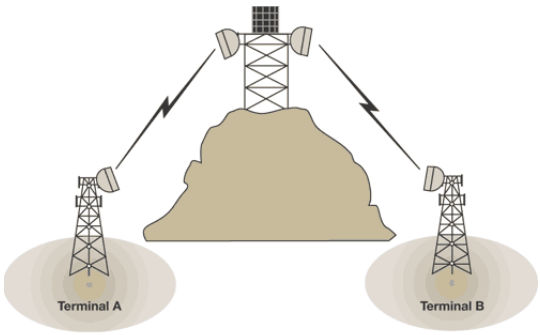
Tahap analisa akan dibahas pada bab ini, tahap perancangan, *prototyping*, implementasi dan pengujuan akan dibahas pada bab-bab berikutnya.

Sebelum memasuki tahap selanjutnya yaitu perancangan infrastruktur, terlebih dahulu dilakukan tahap analisa. Analisa terhadap infrastruktur yang tengah berjalan dilakukan agar diperoleh informasi yang akurat akan infrastruktur yang tengah berjalan. Informasi yang akurat tentang infrastruktur yang tengah berjalan ini sangatlah penting, karena dari informasi yang akurat akan infrastruktur yang berjalan, baru kemudian dapat dirancang infrastruktur yang baru yang sesuai dengan harapan pengguna.

Untuk pengumpulan data dilakukan dengan metode wawancara kepada pihak yang bertanggung jawab dalam bidang pengembangan infrastruktur.

3.1. Analisa Sistem Berjalan

Saat ini pada perusahaan telah berjalan suatu sistem pencatatan penggunaan spektrum frekuensi radio. Peralatan atau perangkat pengguna spektrum frekuensi radio terdiri dari dua macam, yaitu dinas tetap, contoh badan hukum pengguna perangkat dinas tetap adalah penyelenggara jaringan komunikasi (contohnya operator telepon selular). Perangkat lainnya yaitu dinas bergerak darat, seperti : sistem komunikasi radio konvensional atau konsesi/komrad, *repeater*, *base station*, *base-taxi*, *portable*/*mobile unit* dan *Handy Talky* (SDPPI, 2016).



Gambar 3-1 Contoh Sistem Komunikasi Radio Dinas Tetap - *Microwave Link*

Gambar 3-2 Sistem Komunikasi Radio Dinas Bergerak Darat

Setiap badan hukum atau instansi pemerintah sebelum menggunakan peralatan dan perangkat telekomunikasi yang mengunakan spektrum frekuensi radio terlebih dahulu harus memiliki Izin Stasiun Radio (ISR). Untuk saat ini proses permohonan perijinan dapat dilakukan secara online dengan mendaftar pada alamat web yang telah disediakan.

Standar waktu penyelesaian permohonan ISR baru adalah 44 hari kerja, dengan rincian sebagai berikut :

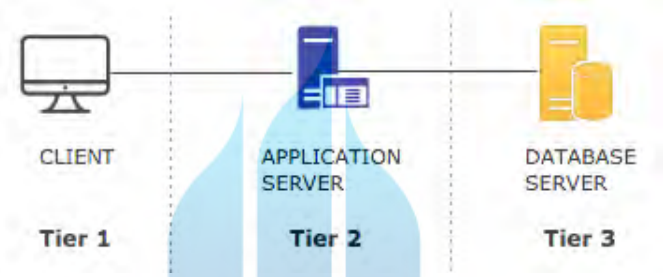
Gambar 3-3 Standard waktu penyelesaian ISR

Sistem yang telah berjalan ini melayani *internal user* dan juga *external user*. *internal user* ini adalah pada karyawan perusahaan yang berperan sebagai *ordinary user* yang mempunyai fungsi *entry data*, membuat report dan berbagai aktivitas user lainnya. Ada pula *user* yang berperan sebagai *administrator* aplikasi. Selain *internal user* yang telah disebut di atas,

terdapat pula *external user* dari berbagai badan hukum ataupun instansi pemerintah sebagai pengguna dari spektrum frekuensi radio.

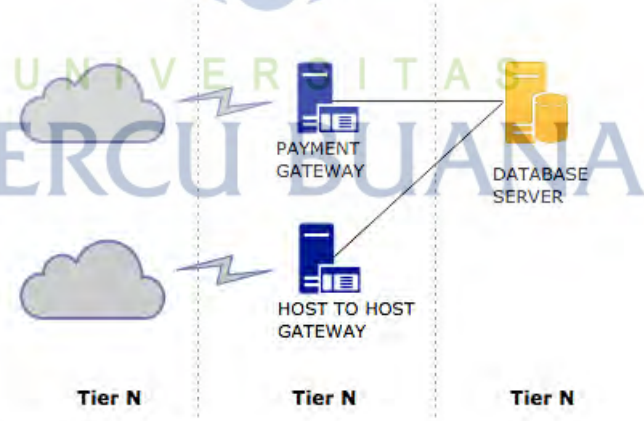
Diketahui bahwa arsitektur yang digunakan pada sistem perusahaan adalah arsitektur n-tier, yang terdiri dari pengguna akhir, aplikasi *server* dan *database server* dan tier yang terhubung dengan sistem dari instansi lain.

Pada tier 1, yaitu *client*, menggunakan *web browser* sebagai *presentation layer*. Pada tier 2 (*business logic layer*), menggunakan aplikasi berbasis c++ dan c#. Dan pada tier 3 (*data/resource layer*), menggunakan RDBMS *Oracle*. *Breakdown high-level* dari arsitektur yang digunakan pada sistem berjalan adalah sebagai berikut :



Gambar 3-4 Arsitektur *3-Tier*

Sedangkan untuk arsitektur n-tier adalah sebagi berikut :



Gambar 3-5 Arsitektur *N-Tier*

3.1.1 Perangkat Keras

Perangkat keras yang digunakan pada sistem berjalan, khususnya pada infrastruktur *database* adalah sebagai berikut :

Tabel 3-1 Perangkat Keras pada Sistem Berjalan

NO	NAMA PERANGKAT	TIPE	JUMLAH	KETERANGAN
1	*Server*	HP *Proliant* DL 580 G7	1	*Form Factor* : *Rackmount Server* *Processor* : 4 x Intel Xeon E7-4850 @2.00GHz *Memory* : 128GB *Internal Disk* : 2 x 600GB
2	SAN *Storage*	HP *StorageWorks* P4300 G2	1	8 x 450 GB

3.1.2 Perangkat Lunak

Sedangkan perangkat lunak yang digunakan pada sistem berjalan, khususnya pada infrastruktur *database* adalah sebagai berikut :

Tabel 3-2 Perangkat Lunak pada Sistem Berjalan

NO	NAMA PERANGKAT	VERSI	KETERANGAN
1	*Oracle Linux*	6.0	Sistem Operasi
2	*Oracle Database Enterprise Edition*	11.2.0.3	Perangkat lunak RDBMS

3.1.3 Ukuran Database

Data Files

Tablespace Name / File Class	Filename	File Size	Autoextensible
AAM_DATA	/u02/oradata/sims/aam_data.dbf	52,428,800	YES
ARMGMT_DATA	/u02/oradata/sims/armgmt_data.dbf	5,368,709,120	YES
BROADCAST_DATA	/u02/oradata/sims/broadcast.dbf	52,428,800	YES
CHIRPLUS_DATA	/u02/oradata/sims/chirplus_data.dbf	209,715,200	YES
CITRIX	/u02/oradata/sims/citrix.dbf	115,343,360	YES
DB_MONITORING1	/u02/oradata/sims/db_monitoring.dbf	52,428,800	NO
EBS_DATA	/u02/oradata/sims/ebs_data.dbf	209,715,200	YES
ENDI_DENDI_DATA	/u02/oradata/sims/endi_dendi.dbf	52,428,800	YES
EXTERNAL_REPORT_DATA	/u02/oradata/sims/external_data.dbf	158,012,014,592	YES
FREQPLAN_DATA	/u02/oradata/sims/freqplan.dbf	1,346,371,584	YES
FREQPLAN_DOC_DATA	/u02/oradata/sims/freqplan_doc_data.dbf	52,428,800	YES
IRVANTEMP	/u02/oradata/sims/irvtemp.dbf	209,715,200	YES
ITU_DATA	/u02/oradata/sims/itu_data.dbf	52,428,800	NO
LANDMOBILE_DATA	/u02/oradata/sims/landmobile.dbf	209,715,200	YES
MONITOR_DATA	/u02/oradata/sims/monitor.dbf	8,556,380,160	YES
MULTILINK_DATA	/u02/oradata/sims/multilink.dbf	849,346,560	YES
PROD_IAS_ORASDPM	/u02/oradata/sims/orasdpm.dbf	209,715,200	YES
PROD_IAS_TEMP	/u02/oradata/sims/ias_temp.dbf	209,715,200	YES
PROD_MDS	/u02/oradata/sims/mds.dbf	209,715,200	YES
PROD_ORABAM	/u02/oradata/sims/orabam.dbf	209,715,200	YES
PROD_SOAINFRA	/u02/oradata/sims/soainfra.dbf	734,003,200	YES
SATELLITE_DATA	/u02/oradata/sims/satellite.dbf	104,857,600	YES
SNAPSHOT	/u02/oradata/sims/snapshot.dbf	26,843,545,600	YES
SNAPSHOT	/u02/oradata/sims/snapshot02.dbf	26,843,545,600	YES
SPECTRAWEB_DATA	/u02/oradata/sims/spectraweb.dbf	13,002,342,400	YES
SPECTRA_CALCRES_DATA	/u02/oradata/sims/spectra_calcres.dbf	20,971,520	YES
SPECTRA_DATA	/u02/oradata/sims/spectra_data.dbf	81,545,658,368	YES
SPECTRA_DOCUMENTS_DATA	/u02/oradata/sims/spectra_doc.dbf	170,288,742,400	YES
SPECTRA_HISTORY_DATA	/u02/oradata/sims/spectra_history.dbf	933,232,640	YES
SPECTRA_TAKE_OVER_DATA	/u02/oradata/sims/spectra_take_over.dbf	34,676,408,320	YES
SPEED_DATA	/u02/oradata/sims/speed_data.dbf	3,145,728,000	YES
SYSAUX	/u01/app/oracle/oradata/sims/sysaux01.dbf	2,537,553,920	YES
SYSTEM	/u01/app/oracle/oradata/sims/system01.dbf	10,737,418,240	YES
TEMP	/u02/oradata/sims/temp01.dbf	3,221,225,472	YES
TEMPORARY_DATA	/u02/oradata/sims/temporary_data.dbf	34,359,721,984	YES
TEMP_REPORT	/u02/oradata/sims/temp_report.dbf	3,221,225,472	YES
UNDOTBS1	/u01/app/oracle/oradata/sims/undotbs01.dbf	30,064,771,072	YES
USERS	/u01/app/oracle/oradata/sims/users01.dbf	1,808,793,600	YES
[CONTROL FILE]	/u01/app/oracle/fast_recovery_area/sims/control02.ctl		
[CONTROL FILE]	/u01/app/oracle/oradata/sims/control01.ctl		
[ONLINE REDO LOG]	/u02/oradata/sims/redo01.log	4,194,304,000	
[ONLINE REDO LOG]	/u02/oradata/sims/redo02.log	4,194,304,000	
[ONLINE REDO LOG]	/u02/oradata/sims/redo03.log	4,194,304,000	
[ONLINE REDO LOG]	/u02/oradata/sims/redo04.log	4,194,304,000	

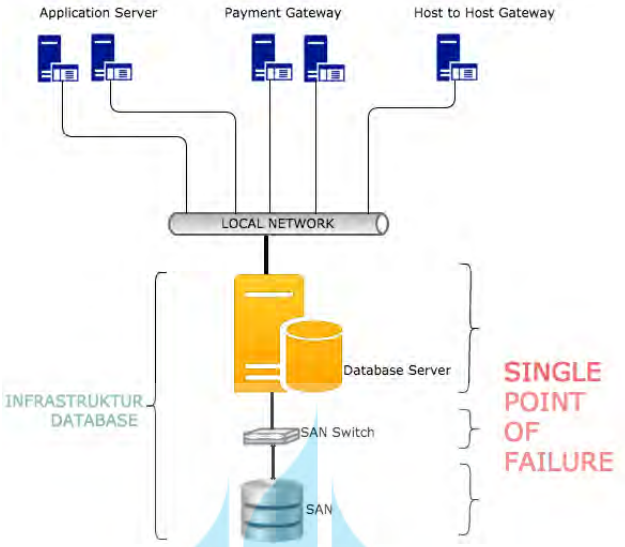
Total:		637,107,421,184	

Gambar 3-6 Ukuran *Database* Sistem Berjalan

3.2. Analisa Masalah

Dari hasil dari analisa ditemukan kelemahan yang sama pada semua bagian tier, baik itu pada *server* aplikasi maupun pada komponen-komponen pembentuk infrastruktur *database*, yaitu potensi *single point of failure*. Dengan adanya potensi *single point of failure* kerusakan ataupun kegagalan

fungsi pada salah satu bagian/komponen dapat mengakibatkan kegagalan fungsi dari seluruh sistem. Potensi *single point failure* ini disumbangkan baik bagian *back-end database (tier 3)*, bagian *application server (tier 2)*.



Gambar 3-7 Analisa Masalah *Single Point Of Failure*

Dengan adanya potensi *single point of failure*, apabila terjadi kegagalan pada suatu bagian/komponen mengakibatkan kegagalan fungsi dari seluruh sistem. Jika hal ini sampai terjadi maka keseluruhan pengguna akan merasakan dampaknya, semua transaksi yang akan tertahan, semua report tidak bisa digenerate dan diprint.

Single point of failure pada sistem ini dapat berupa, *server* yang tiba-tiba *crash*, *controller storage* yang bermasalah, *network card storage failure*. Akibat dari terjadinya *single point of failure* ini adalah semua aktivitas pengguna akan tertahan menunggu sampai dengan kerusakan tertanggulangi, dan ini adalah mimpi buruk bagi suatu bisnis, terlebih lagi bila bisnis yang mensyaratkan waktu layanan 24x7.

Selain *single point of failure* dari sisi *hardware*, terdapat pula potensi *single point of failure* dari sisi *software*. Namun penanganan terjadinya *single point of failure* pada *software* relatif lebih mudah dan *recovery system* pun jauh lebih cepat. Penanganan potensi *single point of failure* pada sisi *software* cukup dengan melakukan *restart* pada *service* aplikasi ataupun dengan melakukan restart pada *server* tempat aplikasi berada.

Saat terjadi *single point of failure* pada sisi *hardware*, adalah hal yang melegakan jika *spare part* yang bermasalah dapat dengan mudah dan cepat didapat, sehingga penggantian bisa segera dilakukan dan layanan sistem dapat kembali berjalan. Namun yang sering kali terjadi adalah *spare part* sulit didapat dan pengadaannya membutuhkan waktu yang cukup lama, bisa berhari-hari ataupun bahkan memakan waktu lebih dari satu minggu, dan hal ini sangat tidak bisa ditoleransi.

3.3. Rekomendasi

Berdasar dari hasil analisa dan hasil wawancara akan kebutuhan user, maka dapat diajukan penerapan arsitektur infrastruktur *high availability cluste*r yang mempunyai level redundansi pada setiap komponen pembentuknya, agar kegagalan fungsional dari sistem yang diakibatkan oleh potensi *single point of failure* dapat dikurangi atau bahkan dapat dihilangkan sama sekali, sehingga nantinya sistem dapat mencapai target sebagai sistem yang sudah masuk kedalam klasifikasi *high availabilty*.

Adapun klasifikasi *high availability* dari suatu sistem didefinisikan sebagai berikut :

Tabel 3-3 *High Availability Classification*

Class	*Level*	*Annual Downtime*
Continuous	100%	0
Fault tolerant	99.999%	5 minutes
Fault resilient	99.99%	53 minutes
High availablity	99.9%	8.8 hours
Normal availablity	99-99.5%	1.8-3.6 days

Kebutuhan lainnya yang diajukan oleh user yaitu berupa migrasi atau pemindahan data dari sistem yang tengah berjalan ke sistem yang baru yang diharapkan dapat dilakukan dengan *downtime* yang seminimal mungkin. Menimbang dari sisi *downtime* yang diperlukan dan dari sisi biaya yang harus dikeluarkan maka diajukan solusi migrasi dengan mempergunakan *Oracle Data Guard*. *Oracle Data Guard* dipilih karena *Oracle Data Guard*

merupakan fitur yang telah termasuk dalam versi database yang telah terpasang, dalam artian *Oracle Data Guard* dapat digunakan tanpa adanya biaya tambahan pembelian lisensi tambahan.

3.4. Batasan Sistem

Pengembangan pada sistem yang tengah berjalan yang tertulis pada skripsi ini difokuskan pada bagian infrastruktur *database* yang mempunyai kemampuan *high availability, load balancing* dengan menggunakan *Oracle RAC* dan pada saat dilakukan migrasi data dari infrastruktur *database* yang tengah berjalan ke infrastruktur *database* yang baru dilakukan dengan metode *minimun downtime* dengan mempergunakan *Oracle Data Guard*. Untuk tahap selanjutnya, perusahaan fokus pada pengembangan bagian aplikasi yang mempunyai kemampuan *high availability* dan *load balancing*.

BAB 4
PERANCANGAN

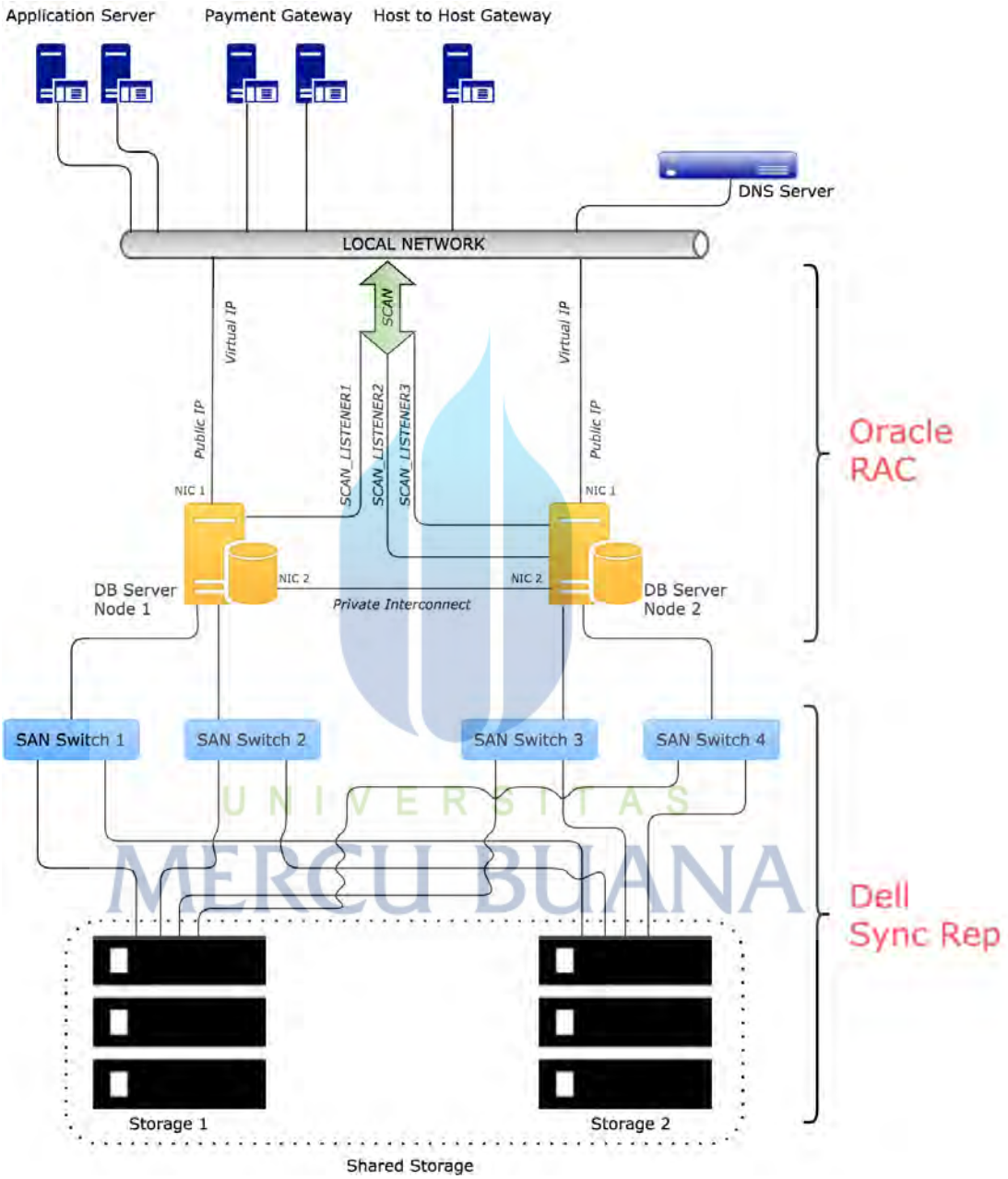
4.1. Perancangan Sistem

Pada bab ini akan dipaparkan perancangan dari sistem yang telah diterapkan pada perusahaan. Seperti sebelumnya telah dibahas dalam rumusan masalah pada Bab I, bahwasanya perusahaan mengharapkan pengembangan sistem infrastruktur yang mempunyai kemampuan *high availability* dari sisi infrastruktur, *load balancing* dari sisi penggunaan *resource server*, dan *minimun downtime* pada saat pelaksanaan migrasi data dari infrastruktur lama ke infrastruktur yang baru. *High availabilty cluster* yang akan dibangun adalah *cluster* yang terbagi atas 2 bagian, yaitu :

1. Pada bagian RDBMS dipilih *Oracle RAC* dengan pertimbangan :
   - RDBMS yang tengah berjalan menggunakan RDBMS *Oracle*, maka akan lebih efisien bila menggunakan RDBMS *Oracle* dari sisi lisensi dan pengelolaan harian RDBMS karena SDM (Sumber Daya Manusia) yang ada telah berpengalaman dalam mengelola RDBMS *Oracle*.
   - Bila menggunakan RDBMS lain maka diperlukan usaha dan waktu tambahan yang tidak sedikit untuk melakukan migrasi data, dan dari pengalaman mengatakan bahwa dalam proses migrasi data ini sering kali terjadi konversi *type data* antar RDBMS yang sering kali sulit dicari kesesuaiannya.
   - *Oracle RAC* memungkinkan *user* dapat tetap beraktivitas walaupun terjadi kegagalan fungsi pada *server* atau pun instance pada salah satu *node* tempat RDBMS *Oracle* berjalan.
2. Dari sisi *storage* dipilih solusi Sync Rep (*Synchronous Replication*) dari Dell dengan pertimbangan bahwa fitur Sync Rep dapat memberikan redundansi penyimpanan data baik dari sisi fisikal maupun dari sisi logical.

Namun sesuai dengan batasan masalah yang telah ditetapkan, maka skripsi ini hanya akan membahas bagian/komponen *high availability cluster* sisi RDBMS-nya saja.

Berikut gambaran topologi infrastruktur *high availability cluster*.



Gambar 4-1 Topologi Infrastruktur *High Availability Cluster*

Setelah dipilih fitur RAC dari *Oracle database* untuk memenuhi rancangan *high availability cluster,* selanjutnya adalah pemilihan metode

yang akan digunakan dalam memindahkan data dari infrastruktur lama ke infrastruktur baru. Dari *Oracle database* tersedia beberapa metode yang dapat dimanfaatkan dalam proses pemindahan data dari satu *database* ke *database* lainnya. Metode digunakan dengan pertimbangan dari beberapa faktor yaitu waktu yang dibutuhkan dalam melakukan migrasi, tingkat klasifikasi data (*mission critical data, vital data, sensitive data, non critical data*) dan yang terakhir dan tak kalah pentingnya cost dari sisi lisensi tool yang digunakan. Metode yang umumnya digunakan pada proses migrasi database Oracle yaitu sebagai berikut :

Tabel 4-1 Metode Migrasi pada *Oracle Database*

Metode	***Summary Activities***	***Downtime? (Y/N)***	***Impact***	***Remark***
Oracle Data Guard	*Configure Oracle Data Guard Parameter Primary & Standby Database*	N	*No Impact*	-
	Backup Primary Database	N	*No Impact*	-
	Restore to Standby Database	N	*No Impact*	-
	Activate Oracle Data Guard	N	*No Impact*	-
	Sinkronisasi Primary Database dan *Standby Database*	N	*No Impact*	-
	Failover Oracle Data Guard	Y	*Database Inaccessible*	estimasi : 10 menit
	Convert Single Instance RAC ke *Multiple Instance RAC*	Y	*Database Inaccessible*	estimasi : 15 menit
	Create Database Service	N	*No Impact*	-
Oracle Data Pump	*Export Source Database*	Y	*Database Inaccessible*	Untuk menjaga konsistensi data
	Create Tablespace di *Target Database*	N	*No Impact*	-
	Create Roles/Privileges di	N	*No Impact*	-

Metode	*Summary Activities*	*Downtime? (Y/N)*	*Impact*	*Remark*
	Target Database			
	Import Database Target	Y	*Database Inaccessible*	Berbanding lurus dengan ukuran *database,* semakin besar ukuran *database* maka semakin lama waktu yang dibutuhkan untuk proses *import.*
Oracle RMAN Backup Restore	*Backup Source Database*	N	*No Impact*	-
	Restore Backup to Target Database	Y	*Database Inaccessible*	Berbanding lurus dengan ukuran *database,* semakin besar ukuran *database* maka semakin lama waktu yang dibutuhkan untuk proses *restore.*
Oracle Golden Gate	*Configure Oracle Golden Gate* pada *Database Source*	N	*No Impact*	*Oracle Golden Gate* mensyaratkan pembelian lisensi.
	Configure Oracle Golden Gate pada *Database Target*	N	*No Impact*	
	Backup Database Source	N	*No Impact*	
	Restore to Target Database	N	*No Impact*	
	Start Cange Data Capture	N	*No Impact*	

Proses migrasi database dari infrastruktur lama ke infrastruktur baru menggunakan *Oracle Data Guard* dengan pertimbangan sebagai berikut :

1. *Oracle Data Guard* hanya memerlukan waktu *downtime* yang lebih singkat bila dibandingkan dengan *Oracle Data Pump* dan *Oracle RMAN Backup Restore.* Dengan menggunakan infrastruktur yang sama, untuk metode *Oracle Data Pump*, dibutuhkan waktu kurang lebih 1,5 jam untuk menuntaskan proses import data sebesar 100GB, dan untuk *Oracle RMAN Backup Restore*, dibutuhkan waktu kurang lebih 1 jam untuk menuntaskan proses *restore* data sebesar 100GB. Jadi dengan besaran data kurang lebih 630GB, dibutuhkan *downtime* kurang lebih 9 jam 45 menit untuk *Oracle Data Pump* menyelesaikan proses *import*, dan dibutuhkan *downtime* kurang lebih 6 jam 30 menit untuk *Oracle RMAN* menyelesaikan proses *restore*. Sedangkan bila menggunakan metode *Oracle Data Guard*, hanya dibutuhkan waktu *downtime* kurang dari 30 menit.
2. Bila dibandingkan dengan *Oracle Golden Gate, Oracle Data Guard* lebih efisien dari sisi biaya yang harus dikeluarkan. Dengan penggunaan hanya satu kali saja, yaitu pada proses migrasi data, maka sangat tidak efiesien bila harus membeli lisensi untuk penggunaan *Oracle Golden Gate*.

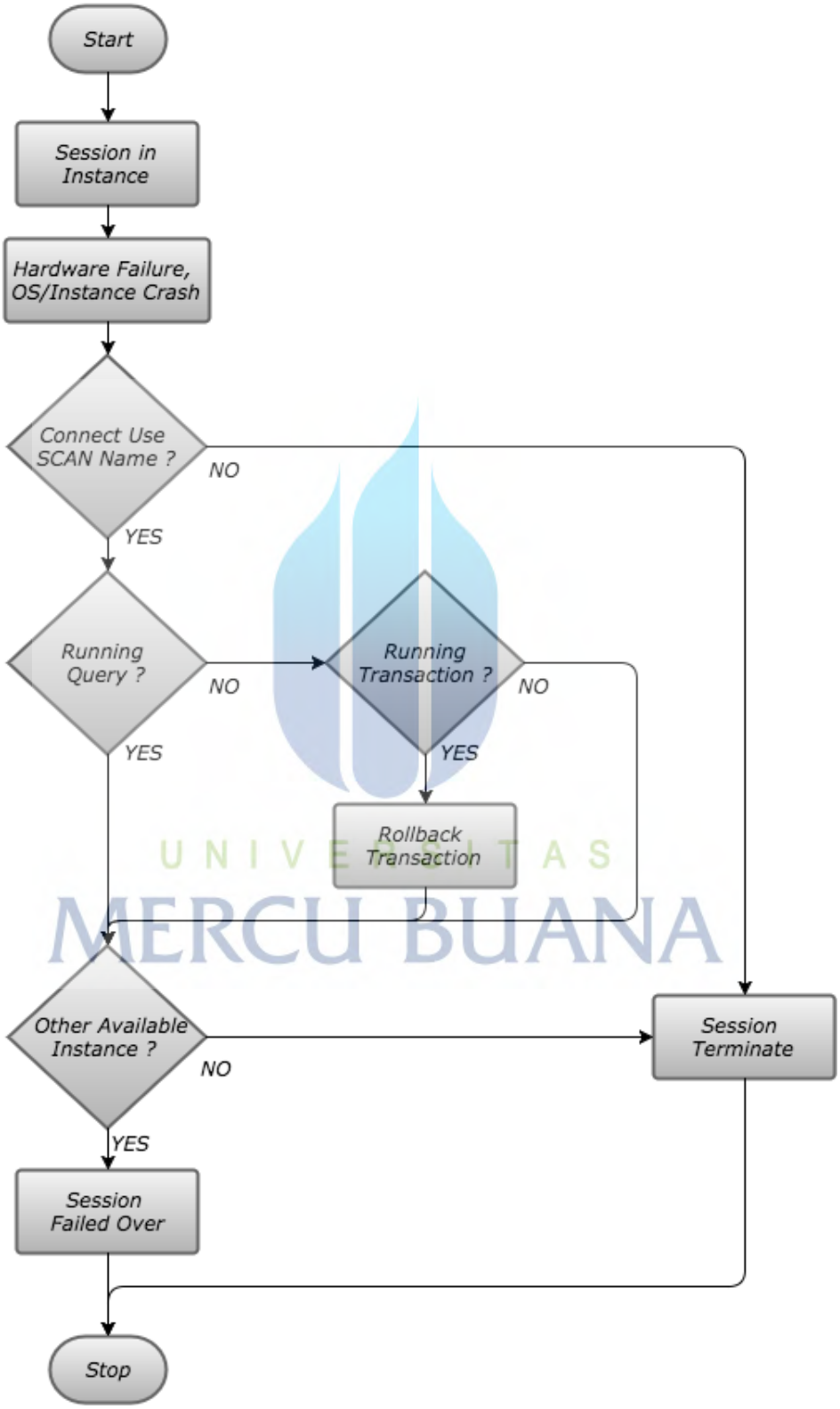
4.2. Algoritma

4.2.1 Algoritma *High Availability*

Metode yang dipergunakan pada mekanisme *high availability* pada infrastruktur *database* adalah metode *failover*, atau dalam istilah RDBMS *Oracle* disebut sebagai *Transparent Application Failover* (TAF). Dimana apabila terjadi kegagalan fungsi dari *server instance* berjalan maka *session* yang tengah terhubung pada *server* tersebut akan dipindahkan ke *server* yang masih berjalan. Terdapat batasan dalam mekanisme *failover* ini, yaitu hanya *session* yang tengah melakukan *query* yang akan dipindahkan ke instance lain, *session* yang tengah melakukan transaksi jenis DML akan dilakukan *rollback transaction* baru bisa di *failover* ke *instance* yang lain.

Untuk session yang tengah melakukan query, keunggulan dari *Transparent Application Fail Over* ini adalah sama sekali tidak diperlukan keterlibatan user, bilamana terjadi kegagalan fungsi, *session* yang tengah melakukan *query* akan terus berjalan padahal sebenarnya arah koneksi telah beralih dari *server* lama ke *server* baru. Sedangkan untuk *session* yang tengah melakukan transaksi *(insert, update, delete)*, diharuskan melakukan *rollback transaction* terlebih dahulu untuk selanjutnya *session* bisa di *failover* ke *server baru*. *Failover session* baik untuk *query* maupun transaksi dilakukan secara otomatis oleh *Oracle RAC*.

Algoritma cara kerja *High Availability Oracle RAC, Transparent Application Failover* adalah sebagai berikut :



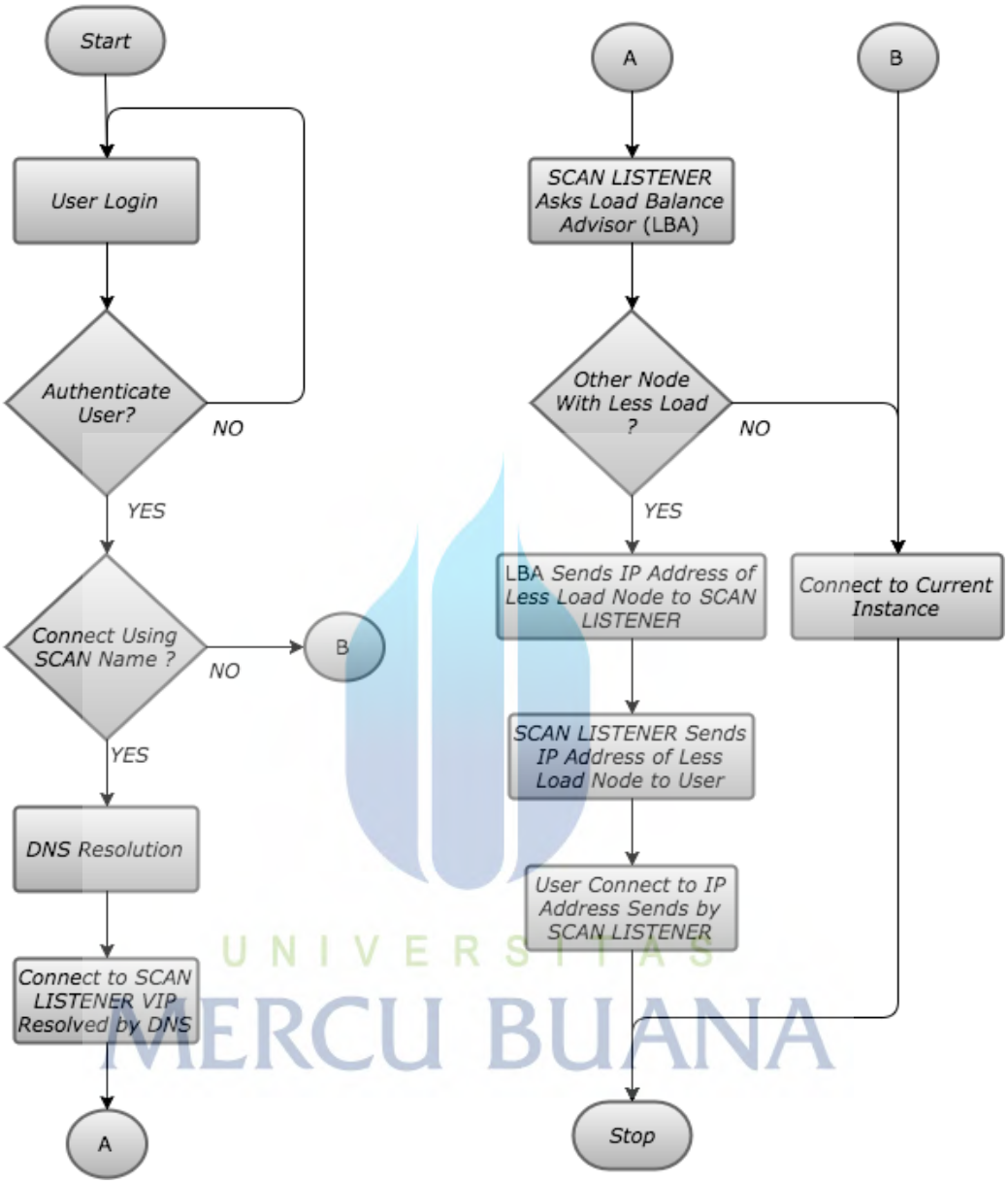
Gambar 4-2 Algoritma *Failover* pada *Oracle RAC*

4.2.2 Algoritma *Load Balance*

Pada proses *load balance*, setiap koneksi baru yang akan login ke database dengan menggunakan *SCAN Name* akan melalui proses translasi dari nama ke *IP address* yang dilakukan oleh ke *DNS server*, dari *DNS server* akan dikirimkan satu *IP address* yang merupakan VIP (*Virtual IP address*) dari *SCAN_LISTENER*. Setelah mendapat *IP address* dari DNS selanjutnya koneksi baru/*client* akan terhubung pada proses yang bernama *SCAN_LISTENERx*. *SCAN_LISTENERx* ini umumnya berjumlah 3 dan ini merupakan rekomendasi dari *Oracle*. Perlu diketahui bahwa *IP address* dari *SCAN_LISTENER* ini adalah *IP address* dari SCAN (S*ingle Client Access Name*) yang sebelumnya telah diregistrasi pada *DNS server*, maka setiap kali client melakukan koneksi ke database menggunakan SCAN maka DNS akan meresponnya dengan memberikan 1 *IP address* dari 3 *IP address* yang telah diregistrasi dengan mekanisme round robin. Langkah selanjutnya adalah *SCAN_LISTENER* akan menghubungi LBA (*Load Balance Advisor*) untuk medapatkan informasi mengenai informasi *load* pada tiap *node*nya, setelah mendapat informasi dari LBA mengenai *node* yang memiliki load paling minimum, *SCAN_LISTENER* akan meneruskan *IP address LOCAL_LISTENER* dari *node* tersebut kepada *client*. Selanjutnya *client* meresponsnya dan membuat koneksi baru ke *IP address* yang diberikan oleh *SCAN_LISTENER*.

Mekanisme *load balancing* yang diterapkan adalah mekanisme *load balance server side connect time load balance* dengan mode : *long,* dimana pada mode ini distribusi l*oad balancing* diatur pada sisi *server* dan berdasarkan banyaknya jumlah *session* pada tiap instance-nya, dan umumnya diterapkan pada aplikasi yang bertipe *hybrid*, campuran antara aplikasi OLTP yang mempunyai ciri transaksi terjadi dalam waktu yang singkat dan aplikasi DSS yang mempunyai ciri adanya job-job yang melakukan pengumpulan/*generate data* untuk keperluan report.

Algoritma cara kerja *Load Balance* pada *Oracle RAC* adalah sebagai berikut :



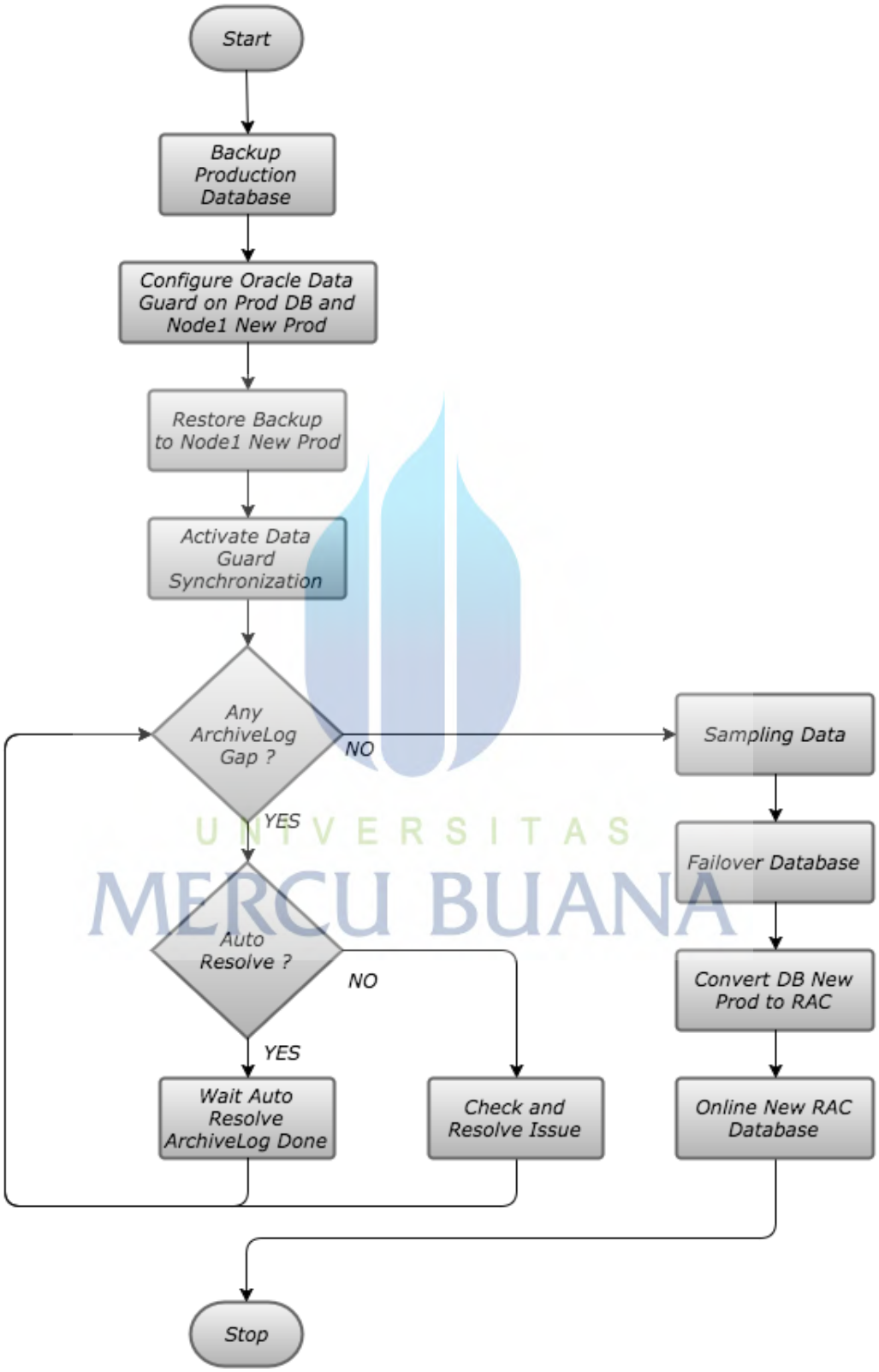
Gambar 4-3 Algoritma *Load Balance* pada *Oracle RAC*

4.2.3 Algoritma *Minimum Downtime Migration*

Seiring besarnya ukuran *database* yang akan dipindahkan/dimigrasikan, berarti semakin lama pula proses yang dibutuhkan untuk melakukan *backup* dan *restore*. Pada cara-cara traditional, *proses backup* dan *restore* membutuhkan *cost* yang cukup tinggi yaitu *cost* dari sisi *availability system*, karena pada umumnya pada

proses *restore* harus dilakukan dengan kondisi sistem *offline*. Maka semakin besar ukuran database maka semakin lama sistem *offline*. Hal ini adalah mimpi buruk bagi pelaku bisnis pada hari ini, karena *offline*nya sistem/*downtime* sama dengan kerugian baik dari sisi finansial maupun dari sisi kepercayaan pengguna. Keunggulan dari proses migrasi dengan menggunakan *Oracle Data Guard* adalah *database production* tetap dapat diakses seperti sediakala selama proses preparasi konfigur *Oracle Data Guard* yaitu proses backup dan restore database yang biasanya memakan waktu paling lama terlebih bila ukuran *database* yang akan dimigrasikan mencapai ukuran TB. Demikian pula pada proses konfigur *Oracle Data Guard*, tidak diperlukan *downtime* pada sisi *database production*. Berikut adalah algoritma yang dipergunakan pada proses migrasi dengan metode *minimum downtime* menggunakan *Oracle Data Guard*.

Berikut adalah algorima dari proses migrasi dengan metode *minimum downtime* :



Gambar 4-4 Algoritma *Minimum Downtime Migration*

BAB 5

IMPLEMENTASI DAN PENGUJIAN

5.1. Lingkungan Implementasi

5.1.1 Perangkat Keras

Tabel 5-1 Perangkat Keras Lingkungan Implementasi

NO	NAMA PERANGKAT	TIPE	JUMLAH	KETERANGAN
1	*Server*	Dell *Poweredge* M630	2	*Form Factor* : *Blade Server* *Processor* : 2 x Intel® Xeon E5-2697 v3 2.6GHz *Memory* : 8 x 16GB RDIMM *Internal Disk* : 2 x 300GB 15K RPM SAS 6Gbps 2.5inc *Embedded* NIC : • 2 x *Broadcom* 5719 *Quad port* 1GBE Mezz Card for M-*Series Blades*. • *Broadcom* 57810-k *Dual port* 10Gb KR *Blade Network Daughter Card*
2	SAN *Storage*	Dell *EqualLogic* PS6210S	2	12 x 800GB SSD 2.5” *Drives* *Useable capacity* : 3.7 TB RAID Level : 1+0
3	SAN *Switch*	Dell *Force10* MXL 10/40 GbE	4	1GbE and 10GbE *ports* 10GbE *optical* and DAC *copper twinax*

5.1.2 Perangkat Lunak

Tabel 5-2 Perangkat Lunak Lingkungan Implementasi

NO	NAMA PERANGKAT	VERSI	KETERANGAN
1	*VMWare* ESXi	6.0	Perangkat lunak virtualisasi
2	*Oracle Linux*	6.6	Operating sistem
3	*Oracle Grid Infrastructure*	11.2.0.4	Perangkat lunak *cluster*
4	*Oracle Database Enterprise Edition*	11.2.0.4	Perangkat lunak RDBMS

5.2. Proses Implementasi

Dalam proses implementasi *high availability cluster* pada infrastruktur *database* menggunakan *Oracle RAC* terdiri dari 2 bagian besar yaitu :

1. Proses instalasi *Oracle RAC software*, dan
2. Proses migrasi data.

5.2.1 Proses Instalasi *Oracle RAC*

Pada setiap instalasi *Oracle RAC software* terdiri dari proses persiapan dan proses instalasi *Oracle RAC softwar* itu sendiri.

5.2.1.1 Proses Persiapan Instalasi *Oracle RAC*

Terdapat beberapa persiapan yang harus dipenuhi agar nantinya saat proses instalasi *Oracle RAC* dapat berjalan dengan lancar. Proses persiapan ini mencakup bagian *network*, *setting parameter* pada beberapa bagian *operating system* seperti *parameter kernel*, *security policy* dan *firewall*. Persiapan ini dilakukan pada semua *node*/*host*/*server* yang nantinya akan menjadi anggota dari konfigurasi *RAC* database. Langkah-langkah preparasinya sebagai berikut :

1. *Network*

   Pada setiap *node*nya dibutuhkan 2 NIC, dan alokasi *IP address* sebanyak 3 buah. 2 *IP address* dengan *segment* yang sama, dan 1 *IP address* untuk *private interconnect*.

   Dialokasikan juga 3 *IP address public* yang dibutuhkan untuk SCAN name.

   2 *IP address* pada setiap node (jadi total 2x2=4 *IP address*) dan 3 *IP address* untuk *SCAN name* yang harus diregistrasikan di *DNS server*.

2. *Operating System*

   - *Parameter kernel :*

     net.ipv4.ip_forward = 0

     net.ipv4.conf.default.rp_filter = 1

     net.ipv4.conf.default.accept_source_route = 0

     kernel.sysrq = 0

     kernel.core_uses_pid = 1

net.ipv4.tcp_syncookies = 1

kernel.msgmnb = 65536

kernel.msgmax = 65536

vm.swappiness = 0

vm.dirty_background_ratio = 3

vm.dirty_ratio = 80

vm.dirty_expire_centisecs = 500

vm.dirty_writeback_centisecs = 100

kernel.shmmax = 68719476736

kernel.shmall = 4294967296

kernel.shmmni = 4096

kernel.sem = 250 32000 100 128

net.ipv4.ip_local_port_range = 9000 65500

net.core.rmem_default = 262144

net.core.rmem_max = 4194304

net.core.wmem_default = 262144

net.core.wmem_max = 1048576

fs.aio-max-nr = 1048576

fs.file-max = 6815744

- *Security-Enhanced Linux : Disabled*
- *Firewall : Disabled*
- *NTP Mode : Burst Mode*
- *Operating System Packages :*

  compat-libcap1, compat-libstdc++-33, cpp, gcc, gcc-c++, glibc-devel, glibc-headers, kernel-headers, libXmu, libXt, ksh, libXxf86misc, libXxf86vm, libaio-devel, libdmx, libstdc++-devel, mpfr, make, ppl, xorg-x11-utils, xorg-x11-xauth, libXv, libXxf86dga
- *Database User : oracle*
- *Database Groups : oinstall, dba*
- *Database Folders* (775 *permission type*) :

  /u01/app/oracle

/u01/app/grid

/u01/app/oraInventor

- *Limits* /etc/security/limits.conf :

  oracle soft nproc 2047

  oracle hard nproc 16384

  oracle soft nofile 1024

  oracle hard nofile 65536

  oracle soft stack 10240

  oracle hard stack 32768

  oracle soft memlock 67119104

  oracle hard memlock 67119104

- *Login parameter* /etc/pam.d/login :

  session required pam_limits.so

- *ASM Lib* :

  kmod-oracleasm-2.0.6.rh1-2.el6.x86_64.rpm

  oracleasmlib-2.0.4-1.el6.x86_64.rpm

  oracleasm-support-2.1.8-1.el6.x86_64.rpm

- *Storage*, dibutuhkan beberapa raw partition sebagai berikut :

  3x10 GB untuk alokasi OCR & *voting disk*

  4x400 GB untuk alokasi FRA

  4x500 GB untuk alokasi DATA

  Dilakukan *format* tanpa *file system* pada semua partisi, dan kemudian dilakukan pengidentifikasian oleh *software* ASM pada semua partisi.

5.2.1.2 Instalasi *Oracle RAC*

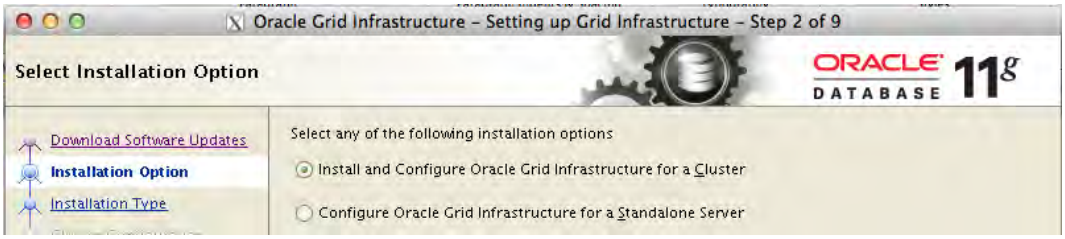
Langkah selanjutnya adalah instalasi *Oracle RAC software*. Proses instalasi ini melalui 2 tahap, yaitu :

- Instalasi *software clusterware*, yaitu komponen-komponen pendukung fungsi *cluster*.
- Instalasi *sotfware database*.

Berikut ringkasan langkah-langkahnya :

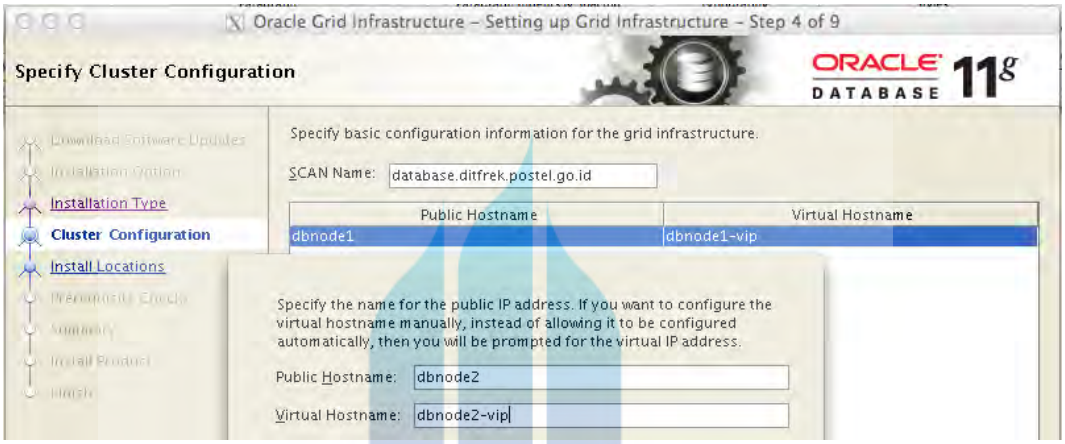
1. Instalasi *software clusterware*

- *Select Installation Option*



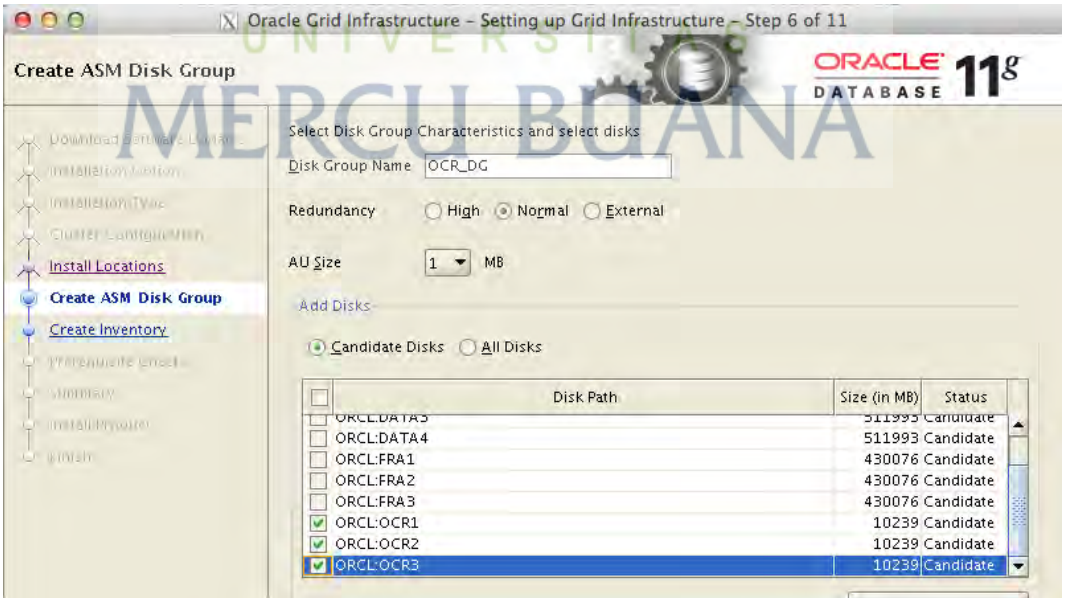
Gambar 5-1 *Select Installation Option*

- *Specify Cluster Configuration, Add Second Node*



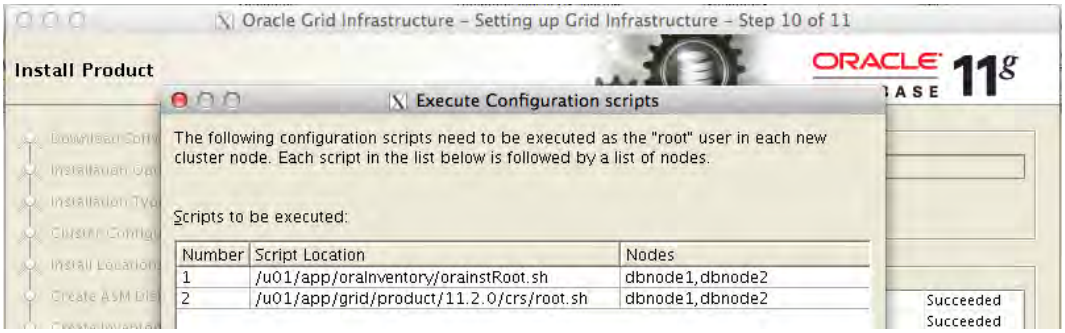
Gambar 5-2 *Add Second Node*

- *Create ASM Disk Group*



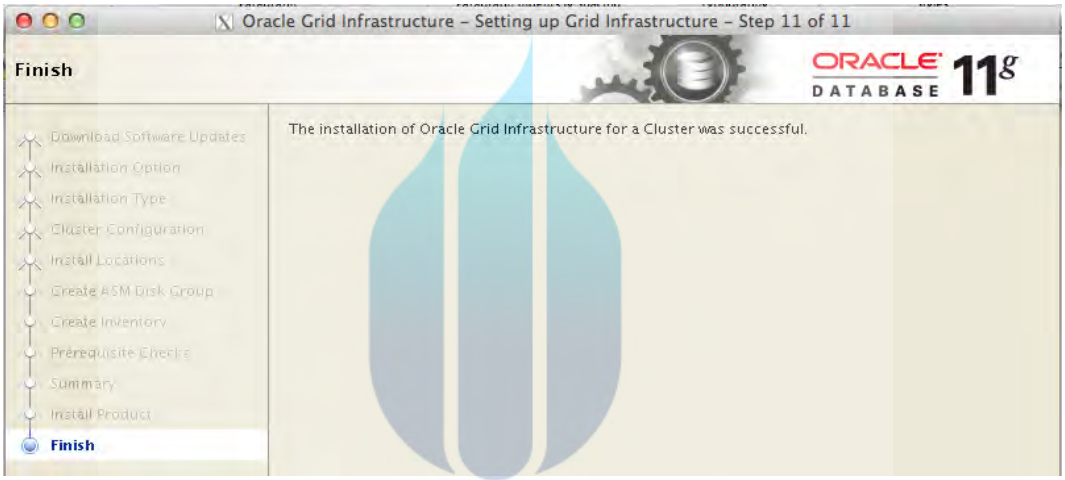
Gambar 5-3 *Create ASM Disk Group*

- *Install Product, Execute Configuration Scripts*



Gambar 5-4 *Execute Configuration Scripts*

- *Finish*



Gambar 5-5 *Finish Installation*

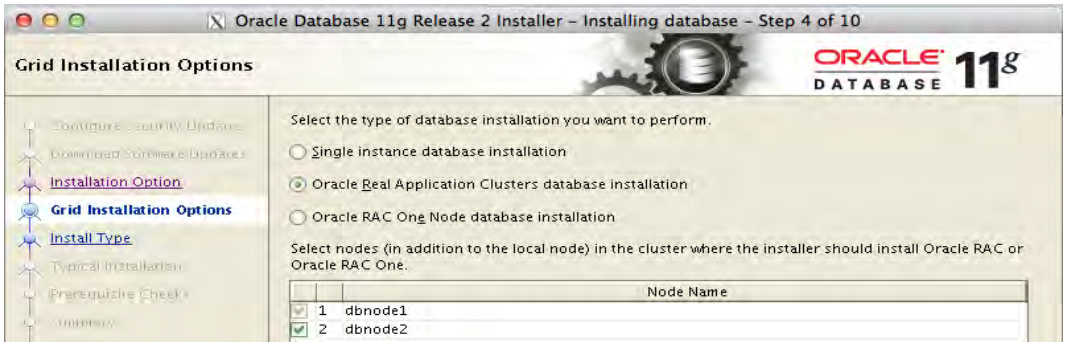
2. *Install Database Software*

Langkah selanjutnya adalah proses instalasi database software yang ringkasannya sebagai berikut :

- *Select Installation Option*

Gambar 5-6 *Select Installation Option*

- *Grid Installation Option*



Gambar 5-7 *Grid Installation Option*

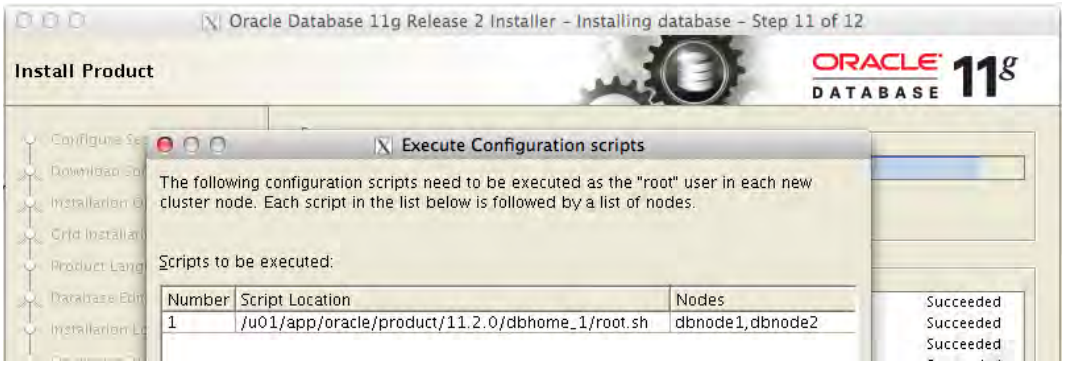
- *Select Database Edition*

Gambar 5-8 *Select Database Edition*

- *Privileged Operating System Groups*

Gambar 5-9 *Priveleged Operating System Groups*

- *Install Product, Execute Configuration Scripts*



Gambar 5-10 *Execute Configuration Scripts*

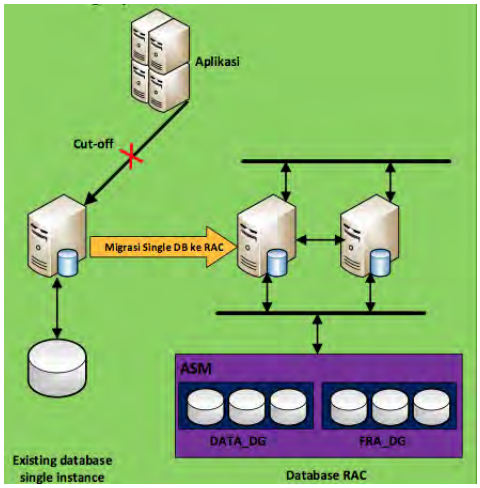
- *Finish*

Gambar 5-11 *Finish Installation*

5.2.2 Proses Migrasi

Setelah selesai proses instalasi *Oracle RAC software*, langkah selanjutnya adalah proses migrasi data dari lingkungan infrastruktur lama ke lingkungan infrastruktur yang baru. Proses migrasi ini terdiri dari :

- *Backup database*
- *Configure Oracle Data Guard*
- *Convert Single Instance RAC to Multiple Instance RAC*
- *Create Database Service*



Gambar 5-12 *Overview Migrasi Single Instance ke RAC*

Adapun rencana kerja dari aktivitas migrasi ini disusun dalam tabel sebagai berikut :

Tabel 5-3 Perangkat Lunak Lingkungan Implementasi

Environment	**Detail Kegiatan Migrasi**			
	Aktifitas	***Downtime? (Y/N)***	**Waktu Pengerjaan (menit)**	***Impact***
Single Instance	*Configure Oracle Data Guard parameter*	N	60	*No Impact*
	Backup Database	N	360	*No Impact*
	Copy Backup Database ke *Standby Node*	N	120	*No Impact*
Oracle RAC	*Configure Oracle Data Guard parameter*	N	60	*No Impact*
	Restore Database	N	420	*No Impact*
	Activate Oracle Data Guard	N	15	*No Impact*
	Sinkronisasi *Primary Database* dan *Standby Database*	N	60	*No Impact*
	Failover Oracle Data Guard	Y	10	*Database Inaccessible*
	Convert Single Instance RAC to Multiple Instance RAC	Y	15	*Database Inaccessible*
	Create Database Service	N	10	*No Impact*

5.2.2.1 *Configure Oracle Data Guard*

Pemilihan *Oracle Data Guard* sebagai instrumen pembantu dalam proses migrasi dengan dasar keunggulan yang dimiliki oleh *Oracle Data Guard*. Keunggulan dari proses migrasi menggunakan *Oracle Data Guard* adalah user tetap dapat terus meneruskan aktivitasnya tanpa terganggu baik itu pada saat dilakukannya *configure Oracle Data Guard* maupun pada saat sinkronisasi data. *Downtime* diperlukan hanya pada saat *failover* dan hanya dibutuhkan waktu kurang dari 30 menit berapa pun besarnya ukuran database yang akan dimigrasikan. Langkah-langkah dalam melakukan *configure Oracle Data Guard* adalah sebagai berikut :

1. *Primary Database*
   - *Setting Parameter Database*

```
alter system set LOG_ARCHIVE_CONFIG='DG_CONFIG=(sims,simsdc)' scope=both;
alter system set LOG_ARCHIVE_DEST_1='LOCATION=USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(ALL_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=sims' scope=both;
alter system set LOG_ARCHIVE_DEST_2='SERVICE=SIMSDC async max_failure=10
max_connections=5 reopen=180 VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=simsdc' scope=both;
alter system set LOG_ARCHIVE_DEST_STATE_1=ENABLE scope=both;
alter system set LOG_ARCHIVE_DEST_STATE_2=DEFER scope=both;
alter system set LOG_ARCHIVE_FORMAT='arch_%d_%t_%r_%s.arc' scope=both;
alter system set REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE scope=both;
alter system set STANDBY_FILE_MANAGEMENT=AUTO scope=both;
alter system set FAL_CLIENT=SIMS scope=both;
alter system set FAL_SERVER=SIMSDC scope=both;
```

Gambar 5-13 *Setting Parameter Database*

   - *Setting Network*

```
SIMSDC =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 10.1.126.27)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = sims)
    )
  )
```

Gambar 5-14 *Setting Network For Replication Communication*

2. *Standby Database*
   - *Restore* Hasil Backup

```
RMAN> duplicate target database for standby nofilenamecheck;
duplicate target database for standby nofilenamecheck;
```

Gambar 5-15 *Restore Backup to Standby Database*

- *Setting Database Parameter*

```
*.log_archive_config='DG_CONFIG=(sims,simsdc)'
*.log_archive_dest_1='LOCATION=USE_DB_RECOVERY_FILE_DEST
          VALID_FOR=(ALL_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=sims'
*.log_archive_dest_2='SERVICE=SIMSDC async max_failure=10 max_connections=5
          reopen=180 VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
          DB_UNIQUE_NAME=simsdc'
*.log_archive_dest_state_1='ENABLE'
*.log_archive_dest_state_2='DEFER'
*.fal_client='SIMSDC'
*.fal_server='SIMS'
```

Gambar 5-16 *Setting Database Parameter*

- *Setting Network*

```
SIMSDC =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 10.1.126.27)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = sims)
    )
  )

SIMS =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 10.1.25.27)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = sims)
    )
  )
```

Gambar 5-17 *Setting Network For Replication Communication*

5.2.2.2 *Backup Database Production*

Setelah dilakukan proses *setting parameter Oracle Data Guard* pada sisi *primary database* (environment infrastruktur lama) dan sisi *standby database* (environment infrastruktur baru), maka langkah selanjutnya adalah mempersiapkan *backup database production*. Hasil backup nantinya akan dijadikan *base point* di *database RAC* pada infrastrutur yang baru yang selanjutnya proses sinkronisasi dari *database single instance* ke *database RAC* akan dilakukan oleh *Oracle Data Guard*, catatan : proses sinkronisasi ini berjalan selama belum diputuskannya hari H *cut-off* perpindahan dari infrastruktur lama ke infrastruktur baru.. Backup ini harus dilakukan dengan hati-hati jangan sampai mengganggu aktivitas

user, misalnya pada setelah jam kerja. Adapun *script backup* yang digunakan adalah sebagai berikut :

```
run {
allocate channel ch1 device type disk;
allocate channel ch2 device type disk;
allocate channel ch3 device type disk;
allocate channel ch4 device type disk;
sql 'alter system switch logfile';
sql 'alter system switch logfile';
sql 'alter system switch logfile';
backup as compressed backupset incremental level 0
database format '/u04/rman_backup/BKP_DB_%U.bkp'
include current controlfile tag='SIMS_DB_BACKUP';
delete force archivelog until time 'SYSDATE-8';
delete noprompt obsolete;
release channel ch1;
release channel ch2;
release channel ch3;
release channel ch4;
}
```

Gambar 5-18 *Script Backup*

5.2.2.3 *Activate Oracle Data Guard* dan Sinkronisasi

Setelah dilakukan setting konfigur *Oracle Data Guard* pada sisi *primary database* dan *standby database*, langkah selanjutnya adalah aktivasi *Oracle Data Guard* dan sinkronisasi.

- *Activate Oracle Data Guard, primary database*

```
SQL> alter system set LOG_ARCHIVE_DEST_STATE_2=ENABLE scope=both;

System altered.
```

Gambar 5-19 *Activate Oracle Data Guard in Primary Database*

- *Activate Oracle Data Guard, standby database*

```
SQL> alter database mount standby database;
Database altered.

SQL> alter database recover managed standby database disconnect from session;

Database altered.
```

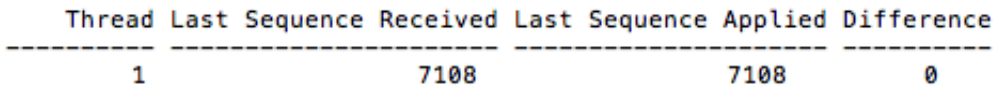
Gambar 5-20 *Activate Oracle Data Guard in Standby Database*

- Sinkronisasi *Oracle Data Guard, primary database*

```
   DEST_ID     Thread Last Sequence Generated
---------- ---------- ------------------------
         1          1                     7108
         2          1                     7108
```

Gambar 5-21 *Last Archivelog Primary Database*

- Sinkronisasi *Oracle Data Guard, standby database*



Thread	Last Sequence Received	Last Sequence Applied	Difference
1	7108	7108	0

Gambar 5-22 *Last Archivelog Primary Database*

Oracle Data Guard dalam proses migrasi ini dimanfaatkan sebagai perangkat pembantu dalam melakukan sinkronisasi data antara *database* pada infrastruktur lama dengan *database* pada infrastruktur baru. Pada tahap ini konfigurasi *Oracle Data Guard* yang disetting adalah tipe *single instance to single instance Oracle Data Guard*. Pada point tertentu yang telah disepakati antara pengguna, pihak penanggung jawab aplikasi dan pihak penanggung jawab infrastruktur, maka dilakukan proses *failover*, yang bertujuan untuk mengalihkan *target backend* database aplikasi dari database pada infrastruktur lama ke database pada infrastruktur baru. Pada tahap *failover* ini pengguna/*client* tidak dapat melakukan koneksi ke *database* sampai dengan proses konversi dan *configure service* pada *RAC database* selesai dilakukan. Adapun langkah-langkah dalam melakukan *failover* adalah sebagai berikut :

- *Finish* sinkronisasi
  alter database recover managed standby database finish;
- *Activate standby database*
  alter database activate standby database;
- *Shutdown standby database*
  shutdown immediate;
- *Startup standby database* sebagai *primary database*
  startup;

5.2.2.4 *Convert Single Instance RAC to Multiple Instance RAC*

Database yang telah dikonfigure pada infrastruktur baru masih dalam kondisi *single instance RAC database* dan perlu dilakukan konversi ke *multi instance RAC database,* maka langkah selanjutnya yaitu konversi dari *single instance RAC database* ke *multi instance RAC database* sebagai berikut :

- *Register database to cluster* :

  srvctl add database -d sims -o /u01/app/oracle/product/11.2.0/dbhome_1 -c RAC -p '+DATA_DG/SIMS/PARAMETERFILE/spfile.339.899722069' -r PRIMARY -s OPEN -a DATA_DG,FRA_DG -n sims

- *Add instance database to cluster :*

  srvctl add instance -d sims -i sims1 -n dbdev-node1

  srvctl add instance -d sims -i sims2 -n dbdev-node2

5.2.2.5 *Create Database Service*

Langkah terakhir sebelum *database* yang baru dikonfigure untuk dapat diakses oleh pengguna adalah pembuatan *database service*. Pendefinisian fungsi *failover* dan *load balance* pada *Oracle RAC* termasuk pada *service database* ini. Langkah pembuatan *database service :*

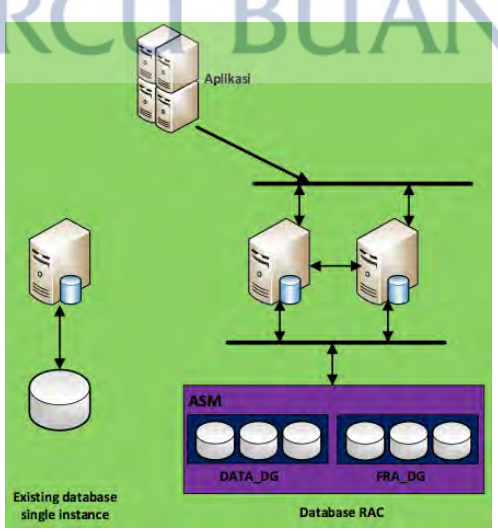
- *Create initial service (enable failover feature) :*

  srvctl add service -d sims -s simsdc.localdomain -r sims1 -a sims2 -P basic -e select

- *Modify service (enable load balance feature) :*

  srvctl modify service -d sims -s simsdc.localdomain -B SERVICE_TIME -j long

Setelah database service selesai dibuat, pengguna dapat menggunakan/mengakses *database RAC* pada infrastruktur yang baru.

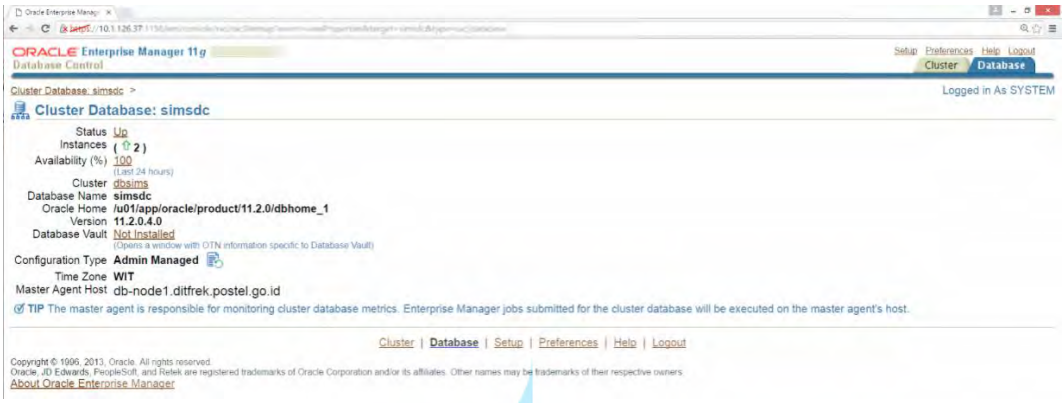


Gambar 5-23 *Overview Setelah Proses Migrasi*

5.3. Hasil Implementasi

Hasil implementasi/penerapan *high availability cluster* menggunakan *Oracle RAC* adalah sebagai berikut :

Informasi database, terlihat bahwa database dalam konfigurasi *cluster*.



Gambar 5-24 *Database Cluster Info*

Ukuran database setelah proses migrasi :

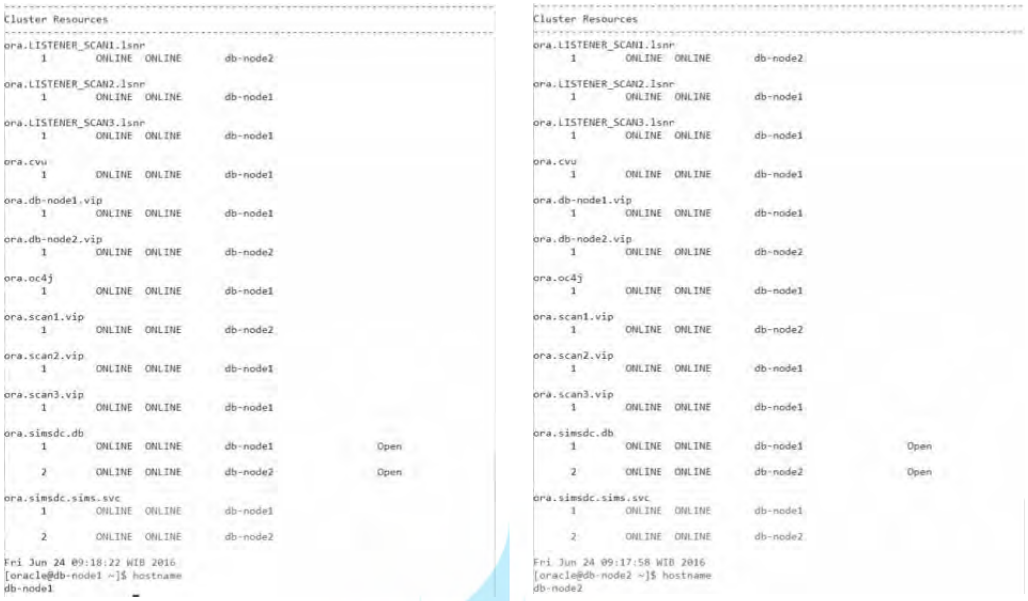
Data Files

Tablespace Name / File Class	Filename	File Size	Autoextensible
AAM_DATA	/u02/oradata/sims/aam_data.dbf	52,428,800	YES
ARMGMT_DATA	/u02/oradata/sims/armgmt_data.dbf	5,368,709,120	YES
BROADCAST_DATA	/u02/oradata/sims/broadcast.dbf	52,428,800	YES
CHIRPLUS_DATA	/u02/oradata/sims/chirplus_data.dbf	209,715,200	YES
CITRIX	/u02/oradata/sims/citrix.dbf	115,343,360	YES
DB_MONITORING1	/u02/oradata/sims/db_monitoring.dbf	52,428,800	NO
EBS_DATA	/u02/oradata/sims/ebs_data.dbf	209,715,200	YES
ENDI_DENDI_DATA	/u02/oradata/sims/endi_dendi.dbf	52,428,800	YES
EXTERNAL_REPORT_DATA	/u02/oradata/sims/external_data.dbf	488,606,614,400	YES
FREQPLAN_DATA	/u02/oradata/sims/freqplan.dbf	2,147,483,648	YES
FREQPLAN_DOC_DATA	/u02/oradata/sims/freqplan_doc_data.dbf	52,428,800	YES
IRVANTEMP	/u02/oradata/sims/irvtemp.dbf	209,715,200	YES
ITU_DATA	/u02/oradata/sims/itu_data.dbf	52,428,800	NO
LANDMOBILE_DATA	/u02/oradata/sims/landmobile.dbf	209,715,200	YES
MONITOR_DATA	/u02/oradata/sims/monitor.dbf	9,663,676,416	YES
MULTILINK_DATA	/u02/oradata/sims/multilink.dbf	849,346,560	YES
PROD_IAS_ORASDPM	/u02/oradata/sims/orasdpm.dbf	209,715,200	YES
PROD_IAS_TEMP	/u02/oradata/sims/ias_temp.dbf	209,715,200	YES
PROD_MDS	/u02/oradata/sims/mds.dbf	209,715,200	YES
PROD_ORABAM	/u02/oradata/sims/orabam.dbf	209,715,200	YES
PROD_SOAINFRA	/u02/oradata/sims/soainfra.dbf	1,610,612,736	YES
SATELLITE_DATA	/u02/oradata/sims/satellite.dbf	104,857,600	YES
SNAPSHOT	/u02/oradata/sims/snapshot.dbf	26,843,545,600	YES
SNAPSHOT	/u02/oradata/sims/snapshot02.dbf	26,843,545,600	YES
SPECTRAWEB_DATA	/u02/oradata/sims/spectraweb.dbf	21,474,836,480	YES
SPECTRA_CALCRES_DATA	/u02/oradata/sims/spectra_calcres.dbf	20,971,520	YES
SPECTRA_DATA	/u02/oradata/sims/spectra_data.dbf	81,545,658,368	YES
SPECTRA_DOCUMENTS_DATA	/u02/oradata/sims/spectra_doc.dbf	214,748,364,800	YES
SPECTRA_HISTORY_DATA	/u02/oradata/sims/spectra_history.dbf	2,147,483,648	YES
SPECTRA_TAKE_OVER_DATA	/u02/oradata/sims/spectra_take_over.dbf	42,949,672,960	YES
SPEED_DATA	/u02/oradata/sims/speed_data.dbf	3,145,728,000	YES
SYSAUX	/u01/app/oracle/oradata/sims/sysaux01.dbf	3,221,225,472	YES
SYSTEM	/u01/app/oracle/oradata/sims/system01.dbf	13,958,643,712	YES
TEMP	/u02/oradata/sims/temp01.dbf	3,221,225,472	YES
TEMPORARY_DATA	/u02/oradata/sims/temporary_data.dbf	34,359,721,984	YES
TEMP_REPORT	/u02/oradata/sims/temp_report.dbf	3,221,225,472	YES
UNDOTBS1	/u01/app/oracle/oradata/sims/undotbs01.dbf	34,359,721,984	YES
USERS	/u01/app/oracle/oradata/sims/users01.dbf	2,147,483,648	YES
[CONTROL FILE]	/u01/app/oracle/fast_recovery_area/sims/control02.ctl		
[CONTROL FILE]	/u01/app/oracle/oradata/sims/control01.ctl		
[ONLINE REDO LOG]	/u02/oradata/sims/redo01.log	4,194,304,000	
[ONLINE REDO LOG]	/u02/oradata/sims/redo02.log	4,194,304,000	
[ONLINE REDO LOG]	/u02/oradata/sims/redo03.log	4,194,304,000	
[ONLINE REDO LOG]	/u02/oradata/sims/redo04.log	4,194,304,000	

Total:		1,021,447,208,960	

Gambar 5-25 Ukuran *Database*

Informasi *cluster resource Oracle RAC database* pada infrastruktur baru :



Gambar 5-26 *Cluster Resources Status*

5.4. Pengujian

Digunakan metode pengujian *black box* untuk menguji fungsionalitas dari fitur-fitur yang akan diterapkan (Nidhra dan Dondeti, 2012). Tujuan digunakannya metode pengujian *black box* ini adalah untuk menguji apakah fitur-fitur yang diterapkan telah sesuai dengan spesifikasinya ataukan masih perlu dilakukan perbaikan, tanpa perlu melihat atau mengetahui dengan pasti bagaimana cara pembuatan dari mekanisme yang terjadi untuk mendapatkan hasil yang dimaksud. Telah dipilih beberapa skenario sebagai input bagi pengujian *black box*, dimana skenario ini adalah simulasi dari tingkah laku pengguna sistem/aplikasi nanti pada lingkungan implementasi.

5.4.1 Skenario Uji Coba

Berikut adalah daftar skenario yang akan dilakukan terhadap infrastruktur database untuk menguji fungsi-fungsi : *Load Balance* dan *Failover*. Dimana pada pengujian fungsi *Load Balance* dilakukan skenario beberapa kali *login* dengan jeda beberapa detik, dan pada pengujian fungsi *Failover* dilakukan skenario simulasi kegagalan *instance*, *operating system* atau *hardware server* pada beberapa aktivitas user, yaitu : *idle session*, *running query* dan *running transaction*.

Tabel 5-4 Skenario Uji Coba

ID UJI	NAMA UJI	FUNGSI YANG DIUJI	HASIL YANG DIHARAPKAN	SKENARIO
1	Uji *Load Balance*	Fungsi *Load Balance*	*Login* dari mesin testing akan terbagi kepada 2 *instance RAC*	1. *Login* dari mesin testing dengan n *login* dan dibatasi dengan jeda n detik 2. Dilakukan monitoring jumlah *login* terhadap 2 *Instance RAC* 3. Testing dilakukan sebanyak 30 kali, dengan jumlah *login* 10, 50, dan 200 kali *login*, dan jeda sebanyak 1 dan 2 detik
2	Uji *FailOver*	Fungsi *FailOver*	*Session* difail over dari mesin/*instance* yang mengalami kegagalan fungsi ke *instance* yang masih berjalan dengan normal	Dilakukan uji coba kepada : 1. *Session* yang sedang melakukan *query select* 2. *Session* yang tengah melakukan transaksi 3. *Idle session*

5.4.2 Hasil Uji Coba

5.4.2.1 Hasil Uji Coba fungsi *Load Balance*

Pada skenario ini akan diuji coba fungsi *Load Balance* dari *database* yang diuji. Fungsi *Load balance* dari *database* akan membagi banyaknya *login* yang masuk kedalam *database* kepada 2 *instance* yang tersedia.

Tabel 5-5 Uji *Load Balance 10 Login* Jeda 1 Detik

NAMA UJI : UJI *LOAD BALANCE* DENGAN 10 *LOGIN* DAN JEDA 1 DETIK			
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**	
		JUMLAH *LOGIN INSTANCE1*	**JUMLAH *LOGIN INSTANCE2***
1	*LOAD BALANCE*	5	5
2	*LOAD BALANCE*	4	6
3	*LOAD BALANCE*	5	5
4	*LOAD BALANCE*	6	4
5	*LOAD BALANCE*	4	6

Tabel 5-6 Uji *Load Balance 10 Login* Jeda 5 Detik

NAMA UJI : UJI *LOAD BALANCE* DENGAN 10 *LOGIN* DAN JEDA 5 DETIK			
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**	
		JUMLAH *LOGIN INSTANCE1*	**JUMLAH *LOGIN INSTANCE2***
1	*LOAD BALANCE*	5	5
2	*LOAD BALANCE*	5	5
3	*LOAD BALANCE*	5	5
4	*LOAD BALANCE*	5	5
5	*LOAD BALANCE*	5	5

Tabel 5-7 Uji *Load Balance 50 Login* Jeda 1 Detik

NAMA UJI : UJI *LOAD BALANCE* DENGAN 50 *LOGIN* DAN JEDA 1 DETIK			
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**	
		JUMLAH *LOGIN INSTANCE1*	**JUMLAH *LOGIN INSTANCE2***
1	*LOAD BALANCE*	25	25
2	*LOAD BALANCE*	25	25
3	*LOAD BALANCE*	25	25
4	*LOAD BALANCE*	25	25
5	*LOAD BALANCE*	26	24

Tabel 5-8 *Uji Load Balance 50 Login* Jeda 5 Detik

NAMA UJI : UJI *LOAD BALANCE* DENGAN 50 *LOGIN* DAN JEDA 5 DETIK			
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**	
		JUMLAH *LOGIN INSTANCE1*	**JUMLAH *LOGIN INSTANCE2***
1	*LOAD BALANCE*	25	25
2	*LOAD BALANCE*	25	25
3	*LOAD BALANCE*	25	25
4	*LOAD BALANCE*	25	25
5	*LOAD BALANCE*	25	25

Tabel 5-9 Uji *Load Balance 200 Login* Jeda 1 Detik

NAMA UJI : UJI *LOAD BALANCE* DENGAN 200 *LOGIN* DAN JEDA 1 DETIK			
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**	
		JUMLAH *LOGIN INSTANCE1*	**JUMLAH *LOGIN INSTANCE2***
1	*LOAD BALANCE*	101	99
2	*LOAD BALANCE*	99	101
3	*LOAD BALANCE*	100	100
4	*LOAD BALANCE*	100	100
5	*LOAD BALANCE*	100	100

Tabel 5-10 Uji *Load Balance 200 Login* Jeda 5 Detik

NAMA UJI : UJI *LOAD BALANCE* DENGAN 200 *LOGIN* DAN JEDA 1 DETIK			
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**	
		JUMLAH *LOGIN INSTANCE1*	**JUMLAH *LOGIN INSTANCE2***
1	*LOAD BALANCE*	100	100
2	*LOAD BALANCE*	100	100
3	*LOAD BALANCE*	100	100
4	*LOAD BALANCE*	100	100
5	*LOAD BALANCE*	100	100

Tabel 5-11 Hasil Akhir Uji *Load Balance*

ID UJI	NAMA UJI	FUNGSI YANG DIUJI	HASIL YANG DIHARAPKAN	SKENARIO	HASIL PENGUJIAN
1	Uji *Load Balance*	Fungsi *Load Balance*	Login dari mesin testing akan terbagi kepada 2 *instance RAC*	1. *Login* dari mesin *testing* dengan n *login* dan dibatasi dengan jeda n detik 2. Dilakukan monitoring jumlah *login* terhadap 2 *Instance RAC* 3. *Testing* dilakukan sebanyak 30 kali, dengan jumlah *login* 10, 50, dan 200 kali *login*, dan jeda sebanyak 1 dan 2 detik	OK

Dari hasil uji coba fungsi *Load Balance* **didapatkan hasil akhir yang memuaskan dan sesuai dengan ekspektasi**, beberapa variasi jumlah *login* dengan jeda berbeda dapat terbagi dengan cukup berimbang kepada 2 *instance* yang tersedia. Pada tabel 5-4, tabel 5-6 dan tabel 5-8 terdapat selisih jumlah 1 dan 2 *login*/*session* hal ini disebabkan karena pada saat dilakukan *testing*, pada saat bersamaan juga dilakukan pengecekan berupa dijalankannya beberapa *query* pada *server* yang menyebabkan terjadinya perubahan *load* yang berakibat terjadi

perubahan informasi pada sisi *load server* statistik, sehingga *load balance advisor* menyarankan untuk melakukan koneksi pada *server* yang lebih rendah *load*nya.

5.4.2.2 Hasil Uji Coba fungsi *Failover*

Pada skenario ini akan diuji coba fungsi *Failover* dari *database* yang diuji. Fungsi *Failover* dari *database* akan memindahkan *session* yang tengah berjalan pada *node/server* yang mengalami kegagalan fungsi *instance*, *operating system* atau *hardware server* ke *instance* yang masih berjalan dengan normal. Digunakan perintah *shutdown abort* untuk mensimulasikan kegagalan fungsi pada level *database*, dan *shutdown –r now* untuk mensimulasikan kegagalan fungsi pada level *operating system* atau *hardware server*. Eksekusi dari skenario uji coba fungsi *failover* adalah sebagai berikut :

Tabel 5-12 Uji *Failover Idle Session* pada Kegagalan *Instance*

NAMA UJI : *FAILOVER IDLE SESSION IN INSTANCE FAILURE INCIDENT*				
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**		
		ORIGIN INSTANCE	***CURRENT INSTANCE***	***FAILOVER STATUS***
1	*FAILOVER*	1	2	SUKSES
2	*FAILOVER*	2	1	SUKSES

Tabel 5-13 Uji *Failover Idle Session* pada Kegagalan O/S atau *Server*

NAMA UJI : *FAILOVER IDLE SESSION IN O/S-SERVER FAILURE INCIDENT*				
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**		
		ORIGIN INSTANCE	***CURRENT INSTANCE***	***FAILOVER STATUS***
1	*FAILOVER*	1	2	SUKSES
2	*FAILOVER*	2	1	SUKSES

Tabel 5-14 Uji *Failover Running Query* pada Kegagalan *Instance*

NAMA UJI : *FAILOVER RUNNING QUERY IN INSTANCE FAILURE INCIDENT*				
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**		
		ORIGIN INSTANCE	***CURRENT INSTANCE***	***FAILOVER STATUS***
1	*FAILOVER*	1	2	SUKSES

NAMA UJI : *FAILOVER RUNNING QUERY IN INSTANCE FAILURE INCIDENT*				
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**		
		ORIGIN INSTANCE	***CURRENT INSTANCE***	***FAILOVER STATUS***
2	*FAILOVER*	2	1	SUKSES

Tabel 5-15 Uji *Failover Running Query* pada Kegagalan O/S atau *Server*

NAMA UJI : *FAILOVER RUNNING QUERY IN O/S-SERVER FAILURE INCIDENT*				
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**		
		ORIGIN INSTANCE	***CURRENT INSTANCE***	***FAILOVER STATUS***
1	*FAILOVER*	1	2	SUKSES
2	*FAILOVER*	2	1	SUKSES

Tabel 5-16 Uji *Failover Running Transaction* pada Kegagalan *Instance*

NAMA UJI : *FAILOVER RUNNING TRANSACTION IN INSTANCE FAILURE INCIDENT*				
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**		
		ORIGIN INSTANCE	***CURRENT INSTANCE***	***FAILOVER STATUS***
1	*FAILOVER*	1	2	SUKSES
2	*FAILOVER*	2	1	SUKSES

Tabel 5-17 Uji *Failover Running Transaction* pada Kegagalan O/S atau *Server*

NAMA UJI : *FAILOVER RUNNING TRANSACTION IN O/S-SERVER FAILURE INCIDENT*				
ID UJI	**FUNGSI YANG DIUJI**	**HASIL UJI**		
		ORIGIN INSTANCE	***CURRENT INSTANCE***	***FAILOVER STATUS***
1	*FAILOVER*	1	2	SUKSES
2	*FAILOVER*	2	1	SUKSES

Tabel 5-18 Hasil Akhir *Uji Failover*

ID UJI	NAMA UJI	FUNGSI YANG DIUJI	HASIL YANG DIHARAPKAN	SKENARIO	HASIL PENGUJIAN
2	Uji *Failover*	Fungsi *Failover*	Session di*failover* dari mesin/*instance* yang mengalami kegagalan fungsi ke *instance* yang masih berjalan dengan normal	Dilakukan uji coba kepada : 1. *Session* yang sedang melakukan *query select* 2. *Session* yang tengah melakukan transaksi 3. *Idle session*	OK

Dari hasil uji coba fungsi *Failover* **didapatkan hasil akhir yang cukup memuaskan dan sesuai dengan ekspektasi**, skenario simulasi kegagalan fungsi pada level *instance database*, *operating system* ataupun *hardware server* tidak mengakibatkan terhentinya layanan *database*/*sistem* terhadap pengguna, dan ini seiring dengan tujuan dari diterapkannya infrastruktur *high availabilty cluster* ini.

BAB 6
PENUTUP

6.1. Kesimpulan

Merujuk kepada kebutuhan perusahaan yang tertuang pada bagian perumusan masalah yang kemudian dibandingkan dengan hasil pengujian yang telah dilakukan, maka dapat ditarik kesimpulan bahwa penerapan h*igh availability cluster* pada infrastruktur *database* menggunakan *Oracle RAC* dan proses migrasi menggunakan *Oracle Data Guard* telah memenuhi poin-poin sebagai berikut :

1. Kebutuhan perusahaan akan tersedianya infrastruktur yang mempunyai kemampuan *high availability* telah terpenuhi dengan penggunaan fitur *Transparent Application Failover* dari *Oracle RAC*.
2. Efisiensi dari sisi penggunaan resos *server* telah diakomodasi dengan penggunaan fitur *server side load balancing* dari *Oracle RAC*.
3. Proses migrasi dari sistem lama ke sistem baru yang tidak memerlukan waktu *downtime* yang lama dan berlarut-larut dengan memanfaatkan *Oracle Data Guard*.

6.2. Saran

Dari hasil penerapan *high availability cluster* ini, disarankan beberapa poin sebagai berikut :

1. Saat ini penerapan *high availability cluster* baru mencakup kemungkinan kegagalan fungsi dari sisi *operating system, hardware* dan *database*, belum mencakup akan kemungkinan datangnya kerusakan yang berasal dari faktor eksternal seperti bencana alam, maka pada pengembangan selanjutnya disarankan untuk dikembangkan penerapkan infrastruktur TI yang dapat melindungi ketersediaan layanan dari gangguan yang berasal faktor eksternal.
2. Penerapan *high availability cluster* telah mencakup kebutuhan perusahaan akan infrastruktur yang mempunyai kemampuan *high availability*, efisien dari sisi penggunaan resos *server*. Untuk

pengembangan selanjutnya, disarankan untuk dapat memaksimalkan infrastruktur yang ada untuk meningkatkan performance aplikasi.

3. Penerapan *Load Balancing* saat ini masih dalam konfigurasi *server side*, disarankan untuk dapat mengembangkan penerapan *Load Balancing* yang melibatkan peran sisi aplikasi.

DAFTAR PUSTAKA

Ault, Mike & Tumma, Madhu (2004). Oracle 10g Grid and Real Application Clusters: Oracle10g Grid Computing with RAC. Rampant TechPress.

Barker, Richard & Massiglia, Paul. (2002). Storage Area Network Essentials. New York: Wiley.

Carr, Mahil & Verner, June. (1996). Prototyping and Software Development Approaches. Retrieved May 10, 2016, from www.academia.edu: https://www.academia.edu/2563255/Prototyping_and_Software_Development_Approaches

Chandrima, Neha., et al. (2014). An Extensive Survey of Monitoring of Load Balance in Oracle Real Application Cluster. Retrived April 11, 2016 from www.ijcsit.com: http://www.ijcsit.com/docs/Volume%205/vol5issue04/ijcsit2014050452.pdf

Dell. (n.d.). PowerEdge Blades. Retrieved May 19, 2016, from www.dell.com: http://www.dell.com/en-us/work/learn/blade-server-solutions

DitJen SDPPI. (2016). Info Pelayanan. Retrieved April 26, 2016, from www.postel.go.id/: http://www.postel.go.id/artikel-izin-spektrum-frekuensi-radio-informasi-pelayanan-7-1856

Isaias, Pedro & Issa, Tomayess. (2014). High Level Models and Methodologies for Information Systems. Springer.

Jeftovic, Mark. (2014). Managing Mission Critical Domains and DNS, Mitigate risk within complex naming environments. O'Reilly Media.

Kadam, Deepali., et al. (June 2011). Oracle Real Application Clusters. Retrived April 11, 2016 from www.ijcsit.com: http://www.ijser.org/researchpaper%5Coracle_real_application_clusters.pdf

Krzyzanowski, Paul. (2007, April). Clusters Building scalable and reliable systems. Retrieved April 24, 2016, from www.cs.rutgers.edu: https://www.cs.rutgers.edu/~pxk/rutgers/notes/content/clusters.html

Mullins, Craig S. (2003). Architectures for Clustering: Shared Nothing and Shared Disk. Retrieved 20 April 2016, from mullinsconsultinginc.com: http://mullinsconsultinginc.com/db2arch-sd-sn.html

Nidhra, Srinivas & Dondeti, Jagruthi. (June 2012). *BLACK BOX AND WHITE BOX TESTING TECHNIQUES –A LITERATURE REVIEW*. Retrived May

21, 2016 from airccse.org: http://airccse.org/journal/ijesa/papers/2212ijesa04.pdf

Oracle. (2016). Introduction to Oracle Database. Retrieved May 19, 2016, from docs.oracle.com: https://docs.oracle.com/database/121/CNCPT/intro.htm#CNCPT88788

Oracle. (n.d.). Introduction to Automatic Workload Management. Retrieved May 20, 2016, from docs.oracle.com: https://docs.oracle.com/cd/E11882_01/rac.112/e41960/hafeats.htm#RACAD076

Oracle. (n.d.). Introduction to Oracle Clusterware. Retrieved May 20, 2016, from docs.oracle.com: https://docs.oracle.com/cd/E11882_01/rac.112/e41959/intro.htm#CWADD1111

Oracle. (n.d.). Introduction to Oracle Data Guard. Retrieved May 25, 2016, from docs.oracle.com: https://docs.oracle.com/cd/E11882_01/server.112/e41134/concepts.htm#SBYDB4701

Oracle. (n.d.). Virtualization. Retrieved May 19, 2016, from www.virtualbox.org: https://www.virtualbox.org/wiki/Virtualization

PAMUNGKAS, W. T. (2011). PENGEMBANGAN INFRASTRUKTUR JARINGAN CLIENT - SERVER KELURAHAN BINTARO. Retrieved April 25, 2016, from repository.uinjkt.ac.id: http://repository.uinjkt.ac.id/dspace/bitstream/123456789/5201/1/WAHYU%20TRI%20PAMUNGKAS-FST.pdf

Torvalds, Linus. (2013, October 23). LINUX's History. Retrieved May 20, 2016, from www.cs.cmu.edu: https://www.cs.cmu.edu/~awb/linux.history.html

vmware. (n.d.). SAN Conceptual and Design Basics. Retrieved May 18, 2016, from www.vmware.com: https://www.vmware.com/pdf/esx_san_cfg_technote.pdf

Vugt, Sander. v. (2014). Pro Linux High Availability Clustering. Apress.