PERANCANGAN *HIGH AVAILABILITY SYSTEM* PADA SISTEM INFORMASI AKADEMIK UNIVERSITAS MUHAMMADIYAH SEMARANG BERBASIS MYSQL CLUSTER

Amran Yobioktabera
Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Semarang 50131
E-mail : amran.yobi@gmail.com

ABSTRAK

Teknologi komunikasi data semakin berkembang seiring berjalannya waktu. Saat ini, data harus dapat disajikan secara cepat dan tepat. Hal inilah yang menjadikan penyajian data menjadi suatu hal yang sangat penting. Database merupakan penyedia data yang harus dijaga stabilitas penyediaannya. Pengelolaan database yang baik dalam suatu sistem dapat meningkatkan ketersediaan data penting yang digunakan untuk menghasilkan informasi. Karena pentingnya database dalam suatu sistem, dibutuhkan perancangan server yang handal dan mumpuni untuk menjalankannya. Salah satu perancangan sistem yang baik adalah dengan menerapkan High Availability (HA). Konsep High Availability mengacu pada sistem atau komponen yang beroperasi dalam waktu yang lama hingga mencapai tingkat available sebesar 99,9 % atau sistem yang hampir tidak pernah mengalami down time (Margaret Rouse – TechTarget, 2005). Salah satu metode untuk menerapkan high availability adalah menggunakan MySQL Cluster. MySQL Cluster memungkinkan data dapat dilayani oleh sekelompok database yang bersifat live replication dan failover dalam suatu jaringan yang kompleks. Untuk mengimplementasikannya, dilakukan perancangan pada server Sistem Informasi Akademik Universitas Muhammadiyah Semarang (SiAmus) dengan menggunakan metode prototyping. Perancangan dilakukan pada prototype database Siamus dengan membuat node-node cluster yang saling melakukan live replication selama dilakukan transaksi. Setelah dilakukan pengujian, dapat disimpulkan bahwa SiAmus dengan High Availability menggunakan metode MySQL Cluster memiliki tingkat availability yang tinggi hingga mencapai 99,9% dibandingkan dengan sistem sebelumnya yang mencapai tingkat availability hanya sebesar 95,9%. Selain itu, dari sisi performance terlihat bahwa SiAmus dengan high availability memiliki throughput dan response time yang lebih baik dibanding sistem sebelumnya meskipun tidak memiliki perbedaan yang signifikan.

Kata kunci : High Availability, MySQL, MySQL Cluster

1. PENDAHULUAN

Teknologi komunikasi data semakin berkembang seiring berjalannya waktu. Saat ini, data harus dapat disajikan secara cepat dan tepat. Hal inilah yang menjadikan penyajian data menjadi suatu hal yang sangat penting dikarenakan suatu data dapat dijadikan acuan untuk pengambilan sebuah keputusan. Salah satu isu paling penting pada era teknologi informasi saat ini adalah kemampuan menyediakan layanan secara *high available* untuk semua aplikasi yang berjalan. Semua layanan dimaksudkan dapat melayani user selama 24x7 (24 jam sehari, 7 hari seminggu). Ketika dunia dapat mengakses aplikasi yang kita sediakan, maka kegagalan akses data pada aplikasi akan terekspos jauh lebih luas (Federico Calzolari, 2010).

Database merupakan penyedia data yang harus dijaga stabilitas penyediaannya. Pengelolaan *database* yang baik dalam suatu sistem dapat meningkatkan ketersediaan data penting yang digunakan untuk menghasilkan informasi dalam pengambilan keputusan. Karena pentingnya *database* dalam suatu sistem, dibutuhkan perancangan server yang handal dan mumpuni untuk menjalankannya. Perancangan server *database* yang biasanya bersifat *stand alone*, dapat mengakibatkan resiko kehilangan data (*data loss*) dikarenakan tidak adanya server *backup* yang melakukan *backup database* secara keseluruhan dan *real time*. Oleh sebab itu, diperlukan perancangan yang tepat untuk membangun sebuah server *database* yang harus bersifat *high available*. Salah satu perancangan sistem yang baik adalah dengan menerapkan *High Availability* (HA).

Konsep *High Availability* mengacu pada sistem atau komponen yang beroperasi dalam waktu yang lama hingga mencapai tingkat *available* sebesar 99,9 % atau sama artinya dengan sistem yang hampir tidak pernah mengalami *down time* (Margaret Rouse – TechTarget, 2005). Salah satu metode untuk menerapkan *high availability* adalah menggunakan MySQL Cluster. MySQL Cluster memungkinkan data dapat dilayani oleh sekelompok *database* yang bersifat *live replication* dalam suatu jaringan yang kompleks.

Selain itu, database dengan metode ini dibentuk dalam node-node redundan yang dapat saling mengisi satu sama lain jika terjadi kegagalan (*failure*) dan melakukan *failover* jika salah satu node terjadi *down*. Untuk mengimplementasikannya, dilakukan perancangan *high availability* menggunakan metode MySQL Cluster pada server Sistem Informasi Akademik Universitas Muhammadiyah Semarang (SiAmus).

2. TINJAUAN PUSTAKA

2.1 High Availability

High Availability (HA) merupakan konsep yang menawarkan tingkat *available* yang tinggi terhadap suatu sistem. Konsep ini biasanya berkaitan dengan kemampuan sistem mengatasi *system hang*, *crash*/*down* maupun kesalahan pada jaringan. Solusi yang ditawarkan berupa *backup data* atau *failover data* yang dilakukan secara *real time*. Saat *server* utama berhenti berjalan, maka *server slave* akan mengambil alih peran server utama dengan kualitas penanganan input dan output yang sama dengan server utama. Sistem akan selalu melakukan sinkronisasi data diantara keduanya atau mungkin lebih untuk mendapatkan *redundancy data.*

HA secara terus menerus memonitor ketersediaan dan performansi sistem. Saat terjadi error pada jaringan, HA dapat memberi peringatan pada *network administrator* untuk melakukan perbaikan *preventive* pada sistem, sementara *server slave* melakukan *take over* server utama. Hal ini akan melindungi informasi yang ada pada sistem *database* dan mengurangi kesalahan pada sistem. HA juga secara otomatis dapat memberikan pesan pada *network administrator* dan menuliskan detail kesalahannya dalam *log file.*

HA merupakan solusi yang menawarkan beberapa keuntungan, diantaranya adalah :

1. Memperkecil kesalahan pada sistem yang disebabkan oleh *system hang*, *crash/down* dan kesalahan jaringan.
2. Menyediakan *data redundancy* secara maksimal dikarenakan sinkronisasi data yang dilakukan secara *real time*

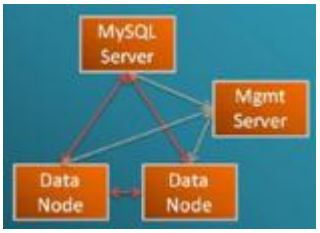
2.2 MySQL Cluster

MySQL Cluster adalah sebuah teknologi baru untuk memungkinkan clustering didalam *memory database* dalam sebuah sistem *share-nothing*. Arsitektur *share-nothing* mengijinkan sistem dapat bekerja dengan perangkat keras yang sangat murah, dan tidak membutuhkan perangkat keras dan lunak dengan spesifikasi khusus. Arsitektur tersebut juga handal karena masing-masing komponen mempunyai *memory* dan penyimpanan tersendiri.

Fungsi aplikasi MySQL Cluster adalah melakukan *backup* MySQL Server dengan sebuah mesin penyimpanan lain yang tercluster yang disebut NDB (Network Database). NDB berarti bagian dari suatu rangkaian yang dikhususkan sebagai mesin penyimpanan saja, sedangkan MySQL Cluster diartikan sebagai kombinasi atau gabungan dari aplikasi MySQL Cluster dan mesin penyimpanan yang baru tersebut.

Sebuah aplikasi MySQL Cluster terdiri dari sekumpulan komputer, masing-masing menjalankan sejumlah proses manajemen dan program-program pengakses data yang khusus. Semua program-program tersebut bekerja bersama-sama untuk membentuk MySQL Cluster. Ketika data disimpan didalam mesin penyimpan media NDB Cluster, tabel-tabel disimpan didalam node-node penyimpanan pada NDB Cluster. Tabel-tabel seperti itu dapat diakses secara langsung dari semua MySQL server yang lain didalam cluster tersebut.

Data yang disimpan didalam node penyimpanan utama pada MySQL Cluster dapat dilakukan *backup* di *node* penyimpanan *secondary* untuk mengantisipasi terjadinya kegagalan dengan teknik *failover cluster*, *cluster* tersebut dapat menangani kegagalan dari *node-node* penyimpanan *individual* dengan tidak ada dampak lain dari sejumlah transaksi dihentikan karena kegagalan proses transaksi.



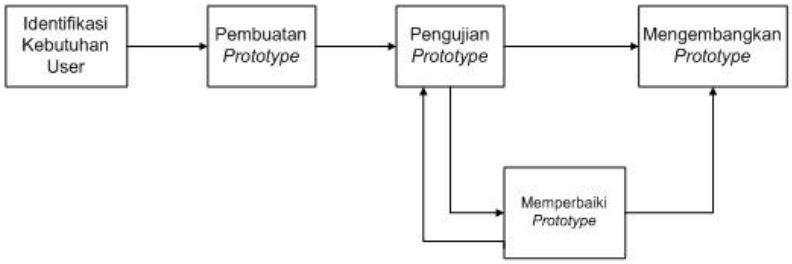
Gambar 2.3 MySQL Cluster

MySQL Cluster menggunakan tiga *node* yang berbeda, yakni:

1. *Datanode* (*ndbd process*), digunakan untuk menyimpan semua data yang menjadi milik MySQL Cluster. Semua data direplikasi di *node-node* ini
2. *Management node (ndb_mgmd process*), digunakan untuk konfigurasi dan *monitoring cluster*. *Node* ini hanya digunakan selama *node startup*.
3. *SQL node* (*mysqld process*), berfungsi sebagai pintu akses untuk masuk ke dalam *node-node* data yang ter-*cluster*.

3. METODOLOGI

Karena database yang digunakan oleh aplikasi SiAmus (Sistem Informasi Akademik Unimus) sangat komplek dan terdiri dari beberapa tabel, maka digunakan metode *prototyping*. Dengan tujuan, agar proses pengembangan sub-sistem yang diambil dari sebagian tabel dalam *database* dapat dilakukan dengan lebih mudah. Pada pengembangannya, hasil *prototyping* ini dapat dilakukan pada bagian tabel yang lain.



Gambar 3.1 Metode Perancangan ‘’*Prototyping*’’

3.1 Pengumpulan Kebutuhan

Setelah mempelajari karakteristik sistem dan database SiAmus, beserta penggunaannya di lapangan, kemudian didukung pula dengan informasi yang didapatkan melalui wawancara personal dengan tim programmer Unimus, dalam tahap awal ini dapat dirumuskan sekurangnya 3 (tiga) poin *requirement*, antara lain :

a. Dibutuhkan 3 (tiga) server. Dengan rincian 1 server MySql sebagai *datanode* 1, 1 server MySql tambahan sebagai *datanode* 2, 1 server MySQL untuk *management server* (NDB MGM) dan sebagai proxy.
b. Kedua server *node* bersifat sama (*replication*) termasuk *webserver*, MySql dan *database*.
c. Ketiga server dihubungkan dalam jaringan yang *reliable*, sehingga proses *backup* dan *failover* secara *realtime* dapat berjalan dengan lancar.

3.2 Membangun Prototype

Karena prototype yang dihasilkan harus mewakili kompleksitas dari database yang digunakan dalam sebuah aplikasi, dibutuhkan 2 (dua) tahapan, yaitu :

3.2.1 *Design*

Untuk melakukan proses *prototyping high availability system* menggunakan metode MySql Cluster, dibutuhkan sebagian tabel yang mewakili kompleksitas dan banyak data dalam sebuah database. Selain tabel yang diambil sebagai prototype, diambil pula data dalam tabel tersebut untuk dilakukan pengujian keberhasilan Clustering MySql. Karena *cluster* melibatkan beberapa *server* dan memiliki variabel-variabel konfigurasi sendiri, maka konfigurasi *cluster* tersebut tidak dapat dijalankan pada engine MyISAM atau InnoDB. *MySql Cluster* hanya dapat dilakukan menggunakan *engine MySql* lain, yaitu *engine* NDBCLUSTER.

3.2.2 *Implementasi*

MySql Cluster Manager yang terdapat pada bagian management server merupakan komponen utama untuk mengimplementasikan clustering pada MySql. MySql Cluster Manager merupakan aplikasi client-server yang menjadi agen pengelolaan node-node MySql Cluster.

Datanodes dalam arsitektur mysql cluster merupakan node-node penyimpan data yang antara node satu dan node lainnya saling terhubung untuk selalu melakukan replikasi secara live (*live replication*). Banyaknya node disesuaikan dengan kebutuhan arsitektur itu sendiri.

Client sebagai pihak yang melakukan proses transaksi data dalam *server*. Data dalam *server* tersebut akan diproses oleh *management node* sehingga dapat dilakukan *replication* antara *datanodes* 1 dan *datanodes* 2 selaku *storage cluster*.

3.3 Evaluasi Prototype

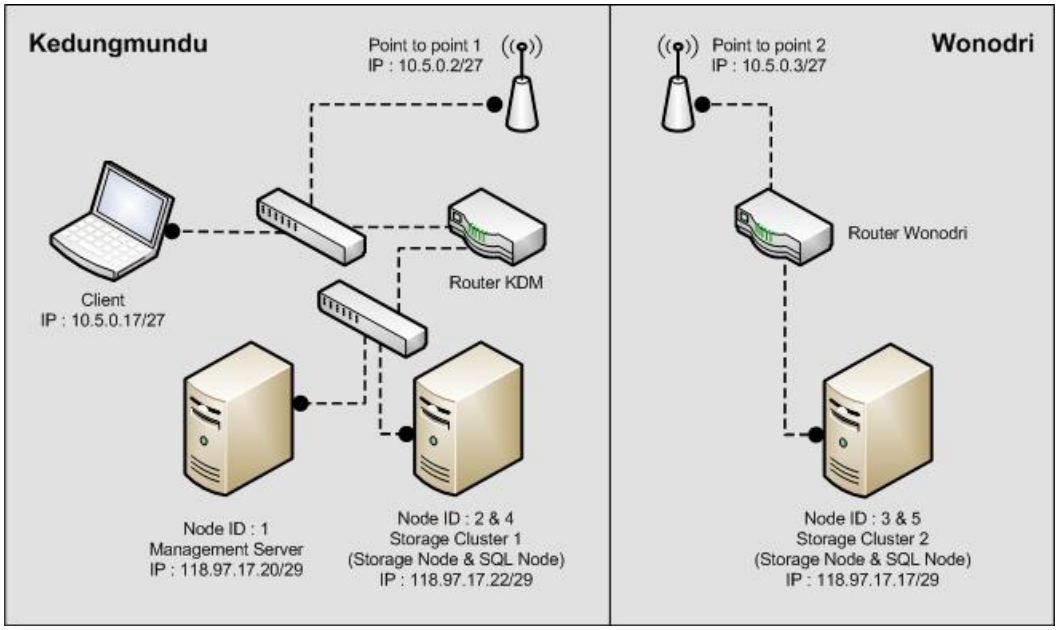
Melalui evaluasi yang dilakukan bersama dengan tim SiAmus, didapatkan kesimpulan bahwa prototype yang dihasilkan sudah memenuhi semua aspek *requirement* sistem. Yang perlu diperhatikan adalah pada sisi arsitektur. Arsitektur yang akan diterapkan yaitu antara *datanodes* 1 dan *datanodes* 2 terletak pada lokasi yang berbeda. Untuk itu perlu dijamin realibilitas koneksi antara satu *node* dengan *node* yang lainnya.

4. HASIL DAN PEMBAHASAN

4.1 Implementasi Sub Sistem

Seperti telah dijelaskan pada bab sebelumnya, untuk mengimplementasikan rancangan sub-sistem *High Availability* menggunakan metode *MySQL Server* perlu dilakukan dalam beberapa tahap. Implementasi dilakukan dengan menerapkan arsitektur pada pembahasan sebelumnya disesuaikan pada kondisi lapangan yang ada pada jaringan Unimus. Selanjutnya, dilakukan implementasi pada tiap-tiap node yang ditentukan.

4.1.1 Implementasi *MySQL Cluster* Pada Arsitektur Jaringan Unimus



Gambar 4.1 : Arsitektur MySQL Cluster Pada Jaringan Unimus

Pada gambar nampak pemetaan jaringan dibagi menjadi 2 (dua) bagian besar, yaitu pada Jaringan Kedungmundu dan Jaringan Wonodri. Pemetaan menjadi 2 (dua) bagian tersebut dilakukan pada lokasi dengan posisi geografis yang berjauhan. Hal ini dimaksudkan untuk maksimalisasi fungsi *failover* yang ada pada MySQL Cluster. Saat terjadi *failed connection* pada *datanode* 1, data secara *realtime* akan tetap ter-*update* pada *datanode* 2 serta aplikasi akan tetap berjalan tanpa terjadi *data lost.*

Pada gambar juga nampak bahwa Management Server dan Storage Cluster 1 (Datanode 1) dipusatkan pada Kampus Kedungmundu (Rektorat), dikarenakan kampus tersebut menjadi kampus pusat pengaturan jaringan Unimus. Antara Kampus Kedungmundu (Rektorat) dengan Kampus Wonodri dihubungkan dengan point-to-point melalui wireless. Lalu routing yang didapat dari Kampus Kedungmundu dihubungkan dengan switch yang selanjutnya dihubungkan langsung dengan server *Storage Cluster* 2 (*datanode* 2).

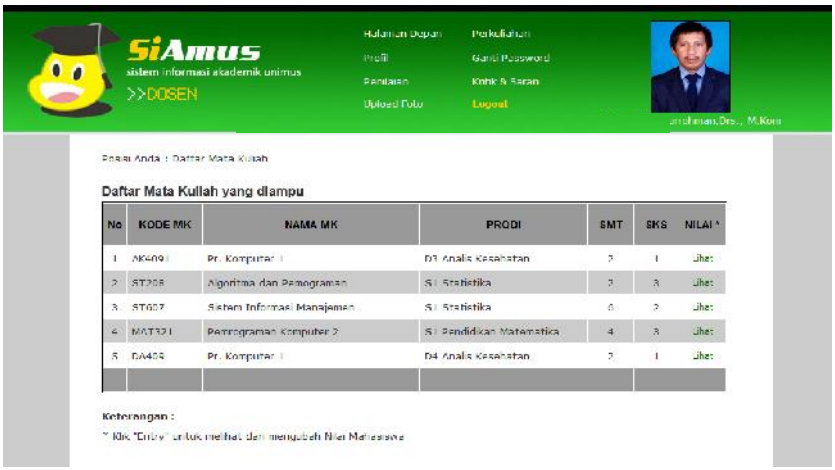
4.1.2 Implementasi Pada *Datanode 1* dan *Datanode 2*

Datanode 1 dan Datanode 2 yang konten didalamnya merupakan Storage Cluster, memiliki 2 (dua) fungsi sekaligus, yaitu sebagai Storage Node dan SQL Node. Fungsi storage node adalah untuk menyimpan data-data yang ditransaksikan melalui MySQL Cluster, data-data tersebut berupa cache query transaksi yang berjalan pada database yang dilakukan clustering. SQL Node sebagai letak SQL berada, berupa database yang di tiap-tiap node yang melakukan proses sinkronisasi.

Daemon utama dalam datanode adalah *ndbd*. *ndbd* adalah daemon yang digunakan untuk menangani semua data dalam tabel menggunakan NDB Cluster pada *storage engine*. Daemon ini memberdayakan *datanode* untuk mendistribusikan penanganan transaksi, pemulihan *datanode*, *disc checkpoint*, *online backup*, dan tugas-tugas terkait. Dalam Cluster MySQL, serangkaian daemon *ndbd* bekerja sama dalam menangani data. Daemon ini dapat dijalankan pada komputer yang sama (host) atau pada komputer yang berbeda. Korespondensi antara *datanodes* dan *Cluster host* dapat dikonfigurasi.

Sesuai dengan arsitektur yang telah dirancang pada bab sebelumnya, masing-masing datanode di set IP 118.97.17.22/29 untuk datanode 1 dan 118.97.17.17/29 untuk datanode 2. Setelah semua proses konfigurasi untuk masing-masing datanode, dilakukan import database "PDPT" yang merupakan database dari aplikasi SiAmus pada masing-masing *datanode*. Ini sama artinya dengan melakukan replikasi database yang ada pada datanode 1 dengan database yang ada pada datanode 2.

Aplikasi prototype SiAmus juga ditanam pada Datanode 1 dan Datanode 2. Aplikasi dipisahkan secara fisik dengan kedua datanode yang didalamnya terdapat database SiAmus tersebut. Hal ini dapat dimaksudkan untuk melakukan pemusatan database pada mesin cluster, sehingga saat terjadi lost connection pada pada satu datanode, datanode yang lain tetap dapat melayani transaksi database.



Gambar 4.2 Tampilan SiAmus Pada *Node 1* & *Node 2*

4.1.3 Implementasi Pada *Management Server*

Proses utama yang dilakukan oleh Management Server adalah terletak pada daemon ndb_mgmd. Daemon ndb_mgmd merupakan daemon tambahan yang tidak dimiliki oleh server-server datanode. Karena selain daemon ndb_mgmd tidak ada perbedaan antara Management Server dan Datanode Server, maka untuk proses instalasi tidak ada perbedaan dengan instalasi yang dilakukan pada Datanode Server.

Selanjutnya perlu dilakukan setting IP address pada Management Server dengan IP address 118.97.17.20/29 sesuai arsitektur yang telah dirancang pada bab sebelumnya. Konfigurasi utama dari Management Server adalah berada pada file config.ini yang terletak pada */var/lib/mysql-cluster/config.ini*. Sesuai dengan rancangan arsitektur jaringan, maka dapat dilakukan konfigurasi Management Server seperti pada gambar berikut :

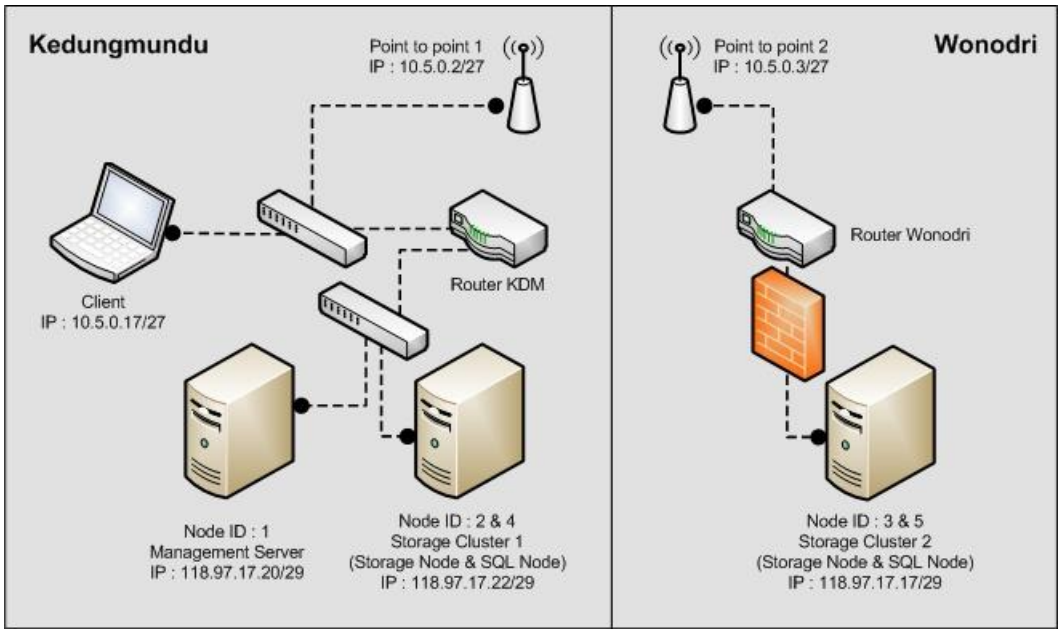
```
[ndbd default]
NoOfReplicas=2
DataMemory=80M
IndexMemory=18M
[tcp default]

# Management Servers
[ndb_mgmd]
hostname=118.97.17.20
datadir=/var/lib/mysql-cluster

# Storage Node
[ndbd]
hostname=118.97.17.22
datadir=/var/lib/mysql-cluster
[ndbd]
hostname=118.97.17.17
datadir=/var/lib/mysql-cluster

# SQL Node
[mysqld]
hostname=118.97.17.22
[mysqld]
hostname=118.97.17.17
```

Gambar 4.3 : Konfigurasi *Management Server*

Pada konfigurasi diatas terlihat *addressing* untuk masing-masing fungsi dalam MySQL Cluster. Management Server di set pada NodeId=1 dengan hostname berupa IP 118.97.17.20/29. sedangkan Storage Node dan SQL Node yang masing-masing fungsinya diletakkan pada mesin yang sama dengan node yang berbeda. Storage Node 1 + SQL Node 1 menggunakan NodeId=2 dan NodeId=4 dengan IP 118.97.17.22/29 yang berada pada 1 (satu) mesin yang sama. Sedangkan Storage Node 2 + SQL Node 2 menggunakan NodeId=3 dan NodeId=5 dengan IP 118.97.17.17/29.

4.2 Pengujian Sub-Sistem

Sebagai pengujian pada sub-sistem, dilakukan akses ke aplikasi lalu melakukan perubahan data yang ada didalamnya. Sebagai prototype, perubahan data dilakukan dengan mengubah data nilai mahasiswa yang ada pada SiAmus menggunakan login seorang dosen. Untuk menguji fungsi failover sekaligus live replication pada MySQL Cluster, maka akan diujicobakan skenario pengujian sebagai berikut :



Gambar 4.4 : Skema Skenario Uji Coba Sub-sistem

4.3 Pengujian Perbandingan

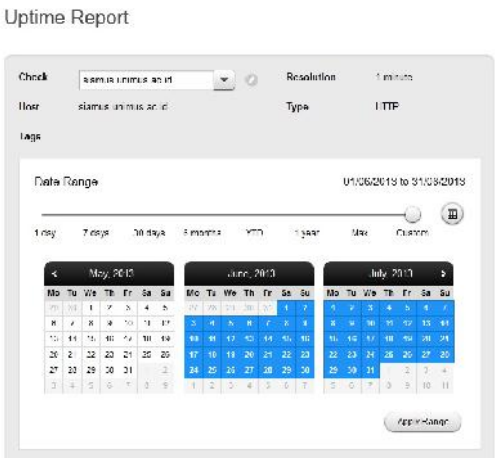
Pengujian ini dilakukan dengan membandingkan 2 (dua) kondisi pada sistem, yaitu kondisi sistem sebelum dan setelah implementasi *High Availability* menggunakan metode *MySQL Cluster*. Pengujian dibagi menjadi 2 (dua) hal, yaitu *Availability* dan *Database Benchmark*.

4.3.1 Availability

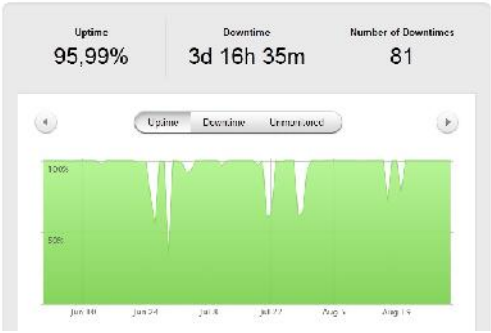
Seperti pembahasan pada bab sebelumnya, tingkat *availability* pada sistem dapat diperoleh dengan mencari data MTTR terlebih dahulu. Hal ini dikarenakan MTTR dapat katakan pula sebagai rataan *downtime* yang terjadi pada rentan waktu tertentu. Jika rataan *downtime* didapat, maka rataan *uptime* yang dijadikan sebagai data *availability* suatu sistem.

Untuk memperoleh data *downtime* pada sistem sebelum menerapkan *high availability*, sebelumnya pihak Unimus telah mendaftarkan *domain* SiAmus ke *developer* penghitung *availability* bernama Pingdom (http://pingdom.com). Saat melakukan pendaftaran, pingdom akan menempatkan file *robot* dalam domain SiAmus yang akan melakukan *report* secara *real time* tentang *availability* sistem dalam hal ini SiAmus.

Pada periode 3 (tiga) bulan terakhir dari 1 Juni 2013 - 31 Agustus 2013 diperoleh data sebagai berikut :



Gambar 4.5 Konfigurasi Uptime Report



Gambar 4.6 Uptime Report SiAmus Sebelum Penerapan HA

Dari data diatas didapakan tingkat *availability system* sebelum menerapkan *High Availability* yaitu :
- *Downtime* : 81 kali (3 hari 16 jam 35 menit)
- *Uptime* : 95,99 % (didapat dari jumlah *uptime* dibagi jumlah waktu selama 3 bulan)

Data server sebelum implementasi diatas dapat dibandingkan dengan data server setelah dilakukan implementasi. Server yang telah diimplementasikan masih berupa IP dan belum memiliki domain resmi, sehingga tidak dapat didaftarkan di *developer availability* pingdom. *Developer* lain yang dapat menghitung *availability* adalah siteuptime (http://siteuptime.com). *Developer* ini dapat menghitung *availability* meskipun belum memiliki domain.

Setelah dilakukan report availability selama periode 3 (tiga) bulan terakhir dari 1 Juni 2013 hingga 31 Agustus 2013, maka diperoleh data sebagai berikut :

Summary Statistics Report for IIA Server (116.97.17.20)
Since: June 1, 2013
Outages: 7
Total Uptime: 99.980%
Total Downtime: 0.020%

Year	Outages	Uptime	Downtime	Avg. Response Time
2013	7	99.980%	0.020%	0.046

Year	Month	Outages	Uptime	Downtime	Avg. Response Time
2013	June	2	99.984%	0.016%	0.044
2013	July	4	99.969%	0.031%	0.053
2013	August	1	99.988%	0.012%	0.042

Gambar 4.7 Uptime Report Server High Availability

Dari data diatas didapakan tingkat *availability system* setelah menerapkan *High Availability* yaitu :
- *Downtime* : 7 kali (0.02%)
- *Uptime* : 99,98 %

4.3.2 Database Benchmark

Seperti pada pembahasan bab sebelumnya, *database benchmark* dilakukan untuk mendapatkan *throughput* dan *response time* sistem SiAmus terhadap suatu *query request* tertentu dalam 2 (kondisi) yang berbeda yaitu sebelum dan sesudah menerapkan *high availability*.

Untuk mendapatkan data yang valid, dilakukan *benchmarking* dengan besar *query request* yang berbeda-beda. Data yang perlu diperhatikan adalah hasil *transactions* dan *avg* pada bagian *test execution summary*. Data *transactions* dan *avg* dapat dijadikan hasil data *throughput* dan *response time.* Hasil-hasil ini nantinya akan mewakili hasil *response* dari server dalam menerima atau mengeksekusi suatu besar *query* tertentu pada kedua kondisi yang berbeda.

Benchmark dilakukan sebanyak minimal 30 (tiga puluh) kali dengan rincian besar *query* yang berbeda. Hal ini dilakukan agar memperoleh data yang komprehensif dan aktual. Setelah dilakukan *benchmark* pada masing-masing kondisi sebelum dan sesudah menerapkan *high availability* menggunakan metode *MySQL Cluster*, maka diperoleh data hasil *benchmark* sebagai berikut :

Tabel 4.1 Hasil Pengujian Database Benchmark

No.	Query	Sebelum Penerapan		Setelah Penerapan	
		TP (per sec)	**RT (ms)**	**TP (per sec)**	**RT (per sec)**
1.	100000	3758.05	2.13	3973.12	1.85
2	200000	3784.99	2.11	3832.21	1.97
3	300000	3780.24	2.11	3956.43	1.88
4	400000	3765.12	2.12	3745.56	2.02
5	500000	3553.49	2.25	3698.34	2.03
6	600000	3616.24	2.21	3701.66	2.13
7	700000	3844.91	2.08	3601.95	2.05
8	800000	3566.08	2.24	3732.67	2.11
9	900000	3557.77	2.21	3688.45	2.03
10	1000000	3692.12	2.16	3645.67	2.12
11	1100000	3501.54	2.21	3798.56	2.05
12	1200000	3502.78	2.18	3632.39	2.13
13	1300000	3495.79	2.29	3589.45	2.16
14	1400000	3486.22	2.14	3508.23	2.17
15	1500000	3503.76	2.12	3676.87	2.14
16	1600000	3487.89	2.31	3583.96	2.18
17	1700000	3467.78	2.35	3594.96	2.24
18	1800000	3389.87	2.23	3502.67	2.15
19	1900000	3401.12	2.35	3476.18	2.25
20	2000000	3222.21	2.48	3387.78	2.19
21	2100000	3190.22	2.36	3201.65	2.24
22	2200000	3210.23	2.31	3389.54	2.26
23	2300000	3309.21	2.35	3306.34	2.28
24	2400000	3301.45	2.37	3364.89	2.25
25	2500000	3336.72	2.39	3375.24	2.31
26	2600000	3278.54	2.38	3265.75	2.33
27	2700000	3109.78	2.43	3286.34	2.31
28	2800000	3285.23	2.41	3290.56	2.34
29	2900000	3201.89	2.47	3245.35	2.37
30	3000000	3213.86	2.49	3356.34	2.41
Rata-rata		**3460.50**	**2.27**	**3546.97**	**2.16**

4.4 Kesimpulan Pengujian

Dari pengujian subsistem yang dilakukan, terlihat bahwa fungsi *failover* dan *replication* berjalan dengan baik saat *MySQL Clustering* diterapkan pada aplikasi SiAmus. Selain itu, fungsi-fungsi tersebut dilakukan secara *real-time.* Koneksi antar node juga *reliable* untuk menerapkan *MySQL Clustering* pada sistem meskipun antara node yang satu dengan node yang lain memiliki letak fisik yang berbeda.

Pada pengujian perbandingan bagian *availability*, dilakukan pengujian *availability* pada sistem sebelum menerapkan *high availability* dan sesudah menerapkannya selama periode 3 (tiga) bulan terakhir, yaitu dari 1 Juni 2013 hingga 31 Agustus 2013. Terlihat bahwa *availability* sistem meningkat dari yang sebelumnya mengalami *downtime* 81 kali (3 hari 16 jam 35 menit) dan *Uptime* sebesar 95,99 % (didapat dari jumlah *uptime* dibagi jumlah waktu selama 3 bulan) menjadi sebesar 99,9 % dengan hanya mengalami *downtime* sebanyak 7 kali.

Pada pengujian perbandingan bagian *database benchmark*, dapat terlihat bahwa tren *throughput* pada sistem setelah penerapan dapat mencapai hasil *throughput* yang lebih besar dengan rata-rata *throughput* 3546.97 *query per sec* dibanding sebelum penerapan dengan rata-rata *throughput* 3460.50 *query per sec*. Sedangkan *response time* sistem setelah penerapan sedikit lebih cepat dengan rata-rata 2.16 ms dibanding sebelum penerapan dengan rata-rata 2.27 ms.

5. KESIMPULAN

Berdasarkan hasil analisa dan pengujian sistem dalam hal ini Sistem Informasi Akademik Unimus (SiAmus) setelah diterapkan *High Availability* menggunakan metode *MySQL Cluster* dapat disimpulkan sebagai berikut :

a. Perancangan SiAmus dengan penerapan *high availability* menggunakan metode *MySQL Cluster* dapat memberikan solusi peningkatan *availability* SiAmus hingga 99,9%
b. Dengan metode *MySQL Cluster* yang melakukan *realtime replication*, maka metode ini dapat menjadi alternatif *backup* dalam sebuah perancangan sistem.

DAFTAR PUSTAKA

[1] Calzolari, Federico; Arezzini, Silvia; Ciampa, Alberto eds. High Availability Using Virtualization. IOP Sciense, 2010
[2] Davies, Alex, dan Harrison Fisk. MySQL Clustering. United States of America: MySQL Press, 2006
[3] Dwi, Anugrah, Periyadi dan Idham, Mohamad. Implementasi Clustering Untuk Mengatasi Kegagalan Sistem Basis Data Pada Sisi Server. Politeknik Telkom Bandung, 2011
[4] Lankford, William M. Benchmarking : Understanding The Basics. State University of West Georgia, 1996
[5] MySQL 5.0 Reference Manual :: B MySQL 5.0 Frequently Asked Questions :: B.10 MySQL 5.0 FAQ: MySQL Cluster.2011
[6] Nambiar, Raghunath; Poess, Meikel, eds . Performance Evaluation and Benchmarking. Springer. ISBN 978-3-642-10423-7. 2009
[7] Oracle Team. A MySQL White Paper : Guide to Scaling Web Databases With MySQL Cluster. Oracle. 2012
[8] Ramadhan, Fakhriy, Solikin dan Irzal Ismail, Setia Juli. Implementasi MySQL Cluster Menggunakan Pemanfaatan High Availability Pada Penginputan Data Mahasiswa Dengan Aplikasi Berbasis Java. Politeknik Telkom Bandung, 2011
[9] Rouse, Margaret. High Availability (HA). TechTarget. 2005