IMPLEMENTASI *LOAD BALANCING* WEB SERVER UNTUK OPTIMALISASI KINERJA WEB SERVER DAN DATABASE SERVER

Hendrik Satria Harefa[1], Joko Triyono[2], Suwanto Raharjo[3]

[1,2,3] Informatika, Fakultas Teknologi Industri, Institut Sains & Teknologi AKPRIND Yogyakarta
Jl Kalisahak No. 28 Komplek Balapan Tromol Pos 45, Yogyakarta 55222 Telp : (0274) 563029
Email:official.hendrikharefa@gmail.com[1], zain@akprind.ac.id[2], wa2n@akprind.ac.id[3]

ABSTRACT

With a single system architecture today, it often becomes overloaded if there are many user requests simultaneously, such as when accessing academic information system services. It caused the server to be down due to the death of the web server and database server application so that many users were not served with a single server.

Multi-server architecture by utilizing web server load balancing is one solution that is considered effective and efficient to overcome these problems. The concept of web server load balancing with high availability allows the request process on the Academic Information System to be shared evenly across several servers.

The results obtained are load balancing can work well when requests come from clients have successfully distributed balancer evenly to each node cluster. So the server does not experience overload and the ability of the web server can service every client request by not experiencing an error request. In addition, the application of file synchronization works well where the files uploaded on the main server will be synchronized to node 1 and node 2 on the cluster and vice versa because synchronization of this file is two-way.

***Keywords** : Load Balancing, Cluster, Request, Server,Overload.*

INTISARI

Degan arsitektur sistem tunggal saat ini, sering ter jadi *overload* jika banyak *request user* secara bersamaan seperti pada saat mengakses layanan sistem informasi akademik. Hal itu menimbuklan kondisi *server down* karena matinya aplikasi *web server* dan *database server* sehingga banyak *user* yang tidak dilayani dengan kondisi *server tunggal*.

Arsitektur *multi server* dengan memanfaatkan *load balancing web server* merupakan salah satu solusi yang dianggap efektif dan efesien untuk mengatasi permasalah tersebut. Konsep *load balancing web server* dengan *high availbality* memungkinkan proses *request* pada Sistem Informasi Akademik dibagi secara merata kebeberapa server.

Hasil penelitian yang didapatkan yaitu *load balancing* dapat bekerja dengan baik ketika *request* datang dari *client* telah berhasil didistribusikan *balancer* secara merata kepada setiap *node cluster*. Sehingga *server* tidak mengalami *overload* dan kemampuan *web server* bisa melayani setiap *request client* dengan tidak mengalami *error request*. Selain itu penerapan *sinkronisasi file* bekerja dengan baik dimana *file* yang diupload pada *server* utama akan disinkron ke *node 1* dan *node 2* pada *cluster* begitu juga sebaliknya karena *sinkronisasi file* ini bersifat dua arah.

Kata Kunci : *Load Balancing, Cluster, Request, Server,Overload.*

PENDAHULUAN

Ketersediaan infrastrukur teknologi yang kuat (*strong*) dan handal (*reliable*) merupakan permasalahan yang dihadapi oleh perusahaan dan instansi yang mengelolah ribuan bahkan jutaan data setiap harinya. Salah satu infrastrukur yang digunakan dalam mengelolah data-data tersebut adalah server. Server merupakan sistem komputer yang menyediakan layanan-layanan tertentu seperti sistem operasi, program aplikasi maupun data-data informasi kepada komputer lain yang saling terhubung dalam sebuah jaringan komputer. Mengingat fungsi yang dimiliki server adalah memberikan layanan kepada client, maka server dituntut bisa melayani permintaan (*request*) dari semua client. UPT TIK sebagai pelaksana teknisi pengelola Sistem Informasi Akademik (SIMAK) SMA memiliki server sebagai penyedia layanan informasi yang bergantung pada satu server. Karena banyaknya permintaan (*request*) terhadap server SIMAK, maka server tersebut mengalami kelebihan kapasias (*overload*), keadaan tersebut diketahui dari hasil pengamatan pada waktu pendaftaran siswa online. Dimana *request* pada server SIMAK secara bersamaan (*Cureent Connection*) bisa mencapai 150 request secara bersamaan. Solusi yang dapat dilakukan unuk meningkatkan kemampuan server dalam melayani permintaan (*request*) yaitu dengan mengupgrade hardware server. Solusi tersebut masih terdapat kekurangan karena sebuah server memiliki batasan hardware yang bisa terpasang pada suatu server. Solusi lain yang dibutuhkan untuk meningkatkan ketersediaan sistem adalah dengan menggunakan replikasi *synchronous* adalah proses penyimpanan data pada lebih dari satu *server* disaat yang bersamaan yang pada umumnya terdiri dari *master* dan *slave*. Proses pengolahan data yang meliputi perubahan, penambahan atau pengurangan data pada *master server*, juga dilakukan secara langsung di *slave server.* Manfaat dari replikasi adalah mendistribusikan database ke beberapa mesin jadi ketika *server master* memiliki masalah ada *server slave* atau *backup* dengan data yang sama tersedia untuk menangani masalah yang terjadi di *server master* Zaenal .A(2018). *Cluster* adalah sekelompok mesin yang bertindak sebagai sebuah entitas tunggal unuk menyediakan sumber daya dan layanan ke jaringan. Pada metode *clustering* server ini terdapa dua fungsi yaitu sebagai *failover cluster dan load balancing cluster*. Fungsi failover bekerja ketika salah satu server bagian dari *cluster* (node) mengalami kerusakan maka akan diganikan oleh server yang lain, sehingga layanan tidak mengalami gangguan. Sedangkan fungsi load balancing cluster bekerja ketika server mengalami banyak permintaan, semua permintaan akan dibagi secara merata ke semua node pada cluster sehingga server tidak akan mengalami kelebihan kapasitas (*overload*).

Pada penelitian sebelumnya yang sudah dilakukan oleh Alam Rahmatulloh dkk (2017) dan Firmansyah dkk (2017) melakukan penerapan load balancing web server haproxy dan sinkronisasi file sehingga hasil yang diperoleh yaitu ketika client melakuakn request kembali dan dilayani oleh node berbeda, maka data yang pernah diupload akan tersedia kembali. Alasan menggunakan metode *load balancing* di penelitian saya ini waktu respon dengan dua buah server atau lebih yang saling berbagi beban lalu lintas web, masing-masing akan berjalan lebih cepat karena beban tidak berada pada 1 server saja, pada penilitan yang telah dilakukan Rahmatulloh dkk (2017) dan Firmansyah dkk (2017) telah menerapkan metode *load balancing haproxy* pada sistem informasi akademik secara online, yang membedakan dengan penelitian yang akan saya lakukan adalah penelitian ini menggunakan algoritma *Least Connection* dimana setiap server akan dibagi penjadwalan bebannya. Pada penelitian ini juga kita melakukan parameter yang akan diuji meliputi *troughput, delay, paket loss,jitter dan request per-detik*.

METODELOGI PENELITIAN

1. Metode Pengumpulan Data

Dalam melakukan penelitian dan laporan ini diperlukan data, adapun teknik untuk memperoleh dan mengumpulkan data yaitu sebagai berikut:

1) Studi Pustaka

Metode studi kepustakawan dipergunakan untuk mengumpulkan data mengenai penelitian terdahulu teori yang mendukung penelitian dan data pendukung lainnya, Serta mempelajari berbagai informasi tentang load balancing web server dan mysql melalui buku-buku referensi makalah, jurnal ilmiah, buku elektronik (*ebook*), dokumentasi, internet dan tugas akhir mahasiswa lain yang memiliki kesamaan topik pembahasan.

2) Observasi

Dalam penelitian ini melakukan databases yang akan di clustering dari *master server* dan *clustering server* dengan parameter tertentu, Selanjutnya akan dilakukan observasi data hasil pengujian tersebut dengan parameter status *troughput, reques* per-detik*, reques loss, jitter, delay* dan *cpu utilization.*

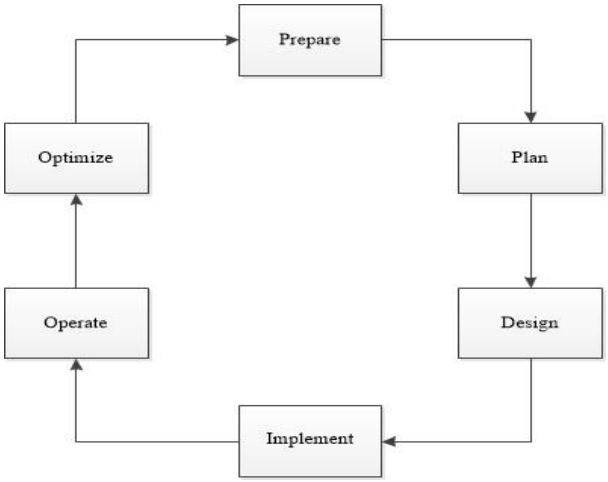
2. Metode Analisis Data

Metode analisis data yang digunakan untuk pengujian *server* adalah *benchmark* dengan parameter *transaction* dan *request client*. *Benchmark* digunakan sebagai pengujian terhadap *server* master dan *clustering* dengan parameter tertentu untuk mendapatkan hasil dan digunakan perbandingan pengujian data yang digunakan untuk pengujian clustering adalah database yang disediakan, setelah selesai dilakukan analisis hasil dengan parameter status *troughput, reques* per-detik*, reques loss, jitter, delay* dan *cpu utilization* yang dibutuhkan untuk *server master* dan *server clustering.*

3. Diagram Alir Penelitian

Untuk mengetahui jalannya implementasi load balancing web server dan database. Disini ada bererapa langkah yaitu:

1) Fase *Prepare* (Persiapan)
   Fase Prepare (persiapan) menetapkan kebutuhan organisasi, mengembangkan strategi jaringan, dan mengusulkan konsep arsitektur dengan level tingkat tinggi, untuk mendukung suatu strategi, yang didukung dengan kemampan keuangan pada organisasi atau perusahaan tersebut.
2) Fase *Plan* (Perencanaan)
   Fase Plan (perencanaan) mengidentifikasi persyaratan jaringan berdasarkan tujuan, fasilitas, dan kebutuhan pengguna. Fase ini mendeskripsikan karakteristik suatu jaringan, yang bertujuan untuk menilai jaringan tersebut. Melakukan analisis pada perancangan terbaik sebuah arsitektur, dengan melihat perilaku dari lingkungan operasional. Sebuah perencanaan proyek dikembangkan untuk mengelola tugas - tugas (tasks), pihak - pihak yang bertanggung jawab dan semua sumber daya untuk melakukan desain dan implementasi. Berdasarkan permasalahan yang dikemukakan pada fase prepare, maka perlu dilakukan penerapan clustering pada server aplikasi untuk dapat menangani permintaan dari pengguna yang terus meningkat. Untuk melakukan implementasi clustering server pada server yang telah disediakan. Server pertama difungsikan sebagai Balancer, server kedua difungsikan sebagai Node Cluster.
3) Fase *Design* (Desain)
   Fase Desain, Desain jaringan dikembangkan berdasarkan persyaratan teknis, dan bisnis yang diperoleh dari kondisi sebelumnya. Spesifikasi desain jaringan adalah desain yang bersifat komprehensif dan terperinci, yang memenuhi persyaratan teknis dan bisnis saat ini. Jaringan tersebut haruslah menyediakan ketersediaan, kehandalan, skalabilitas dan kinerja.
4) Fase *Implement* (Implementasi)
   Pada fase implement, dilakukan instalasi dan konfigurasi, sesuai spesifikasi desain. Perangkat-perangkat ini akan mengganti infrastruktur yang ada. Perubahan kebutuhan juga harus diikuti selama fase ini, jika ada perubahan seharusnya disampaikan dalam pertemuan (meeting), dengan persetujuan yang diperlukan untuk dilanjutkan.
5) Fase *Operate* (Operasional)
   Fase operasional, mempertahankan ketahahan kegiatan sehari-hari. Operasional meliputi pengelolaan dan memonitor komponen-komponan, pemeliharaan, mengelola kegiatan upgrade, mengelola kinerja, mengidentifikasi dan mengoreksi kesalahan. Tahapan ini adalah ujian akhir bagi tahapan desain. Selama operasi, load balancing harus memantau stabilitas dan kinerja web server dan database server, deteksi kesalahan, koreksi konfigurasi, dan kegiatan-kegiatan pemantauan kinerja, yang menyediakan data awal untuk fase selanjutnya, yaitu fase optimalisasi (optimize phase).
6) Fase *Optimize* (Optimalisasi)
   Fase Optimalisasi, melibatkan kesadaran proaktif seorang manajemen jaringan dengan mengindentifikasi dan menyelesaaikan masalah, sebelum persoalan tersebut mempengaruhi jaringan. Fase optimalisasi, memungkinkan untuk memodifikasi desain jaringan, jika terlalu banyak masalah jaringan yang timbul, kemudian juga untuk memperbaiki masalah kinerja, atau untuk menyelesaikan masalah-masalah pada aplikasi (software).



Gambar 1. Diagram Alir Penelitian

HASIL DAN PEMBAHASAN

Hasil yang diperoleh dari penelitian ini berupa Optimalisasi Kinerja Web Server dan Database Server.

Hasil

Desain jaringan yang digunakan dalam optimalisasi kinerja web server dan database server dengan menggunakan Mysql ini dilakukan dalam sebuah jaringan yang langsung terhubung ke internet. Implementasi distribusi sistem pada penelitian ini menggunakan sistem operasi ubuntu 16 server. Ada beberapa komponen yang perlu diperhatikan untuk kebutuhan perangkat keras dan perangkat lunak.

Tabel 1. Tabel Kebutuhan Implementasi Sistem

	Laptop 1	PC 1	PC 2	PC 3
Prosesor	AMD A9	Intel Core 2 duo	Intel Core 2 duo	Intel Core 2 duo
Clock Speed	2.8Ghz	2.3Ghz	2.3Ghz	2.3Ghz
RAM	4GB	2GB	2GB	2GB
OS	Windows 10 64Bit	Ubuntu 16 Server	Ubuntu 16 Server	Ubuntu 16 Server
Basis Data			Mysql	Mysql
IP Address	192.168.1.101	192.168.1.30	192.168.1.11	192.168.1.12

Setelah semua persiapan yang dibutuhkan selesai dilaksanakan yang pertama yang lakukan untuk mengkonfigurasi *load balancing* yaitu melakukan pembaruan *repositori* pada *server* dengan melakukan perintah seperti Gambar 2 dibawah ini.

```
root@balancing-skripsi:/home/balancing# sudo apt-get update

Hit:1 http://ppa.launchpad.net/linrunner/tlp/ubuntu xenial InRelease
Hit:3 http://id.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
Get:4 http://id.archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
```

Gambar 2. Hasil Update Repositori

Langkah selanjutnya setelah kita melakukan proses *update* sampai selesai selanjutnya kita melakukan install paket *haproxy* dengan perintah seperti pada Gambar 3. dibawah ini.

```
root@balancing-skripsi:/home/balancing# sudo apt-get install haproxy -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
haproxy is already the newest version (1.6.3-1ubuntu0.3).
The following package was automatically installed and is no longer required:
  libllvm4.0
```

Gambar 3. Hasil Instalasi Paket Haproxy

Kemudian kita melanjutkan dengan konfigurasi *haproxy* dengan perintah seperti pada Gambar 4 dibawah ini.

```
root@balancing-skripsi:/home/balancing# systemctl restart haproxy
root@balancing-skripsi:/home/balancing#
```

Gambar 4 Setting File Haproxy

Setelah melakuan perintah seperti pada Gambar 4 diatas kita akan masuk pada file konfigurasi *haproxy* pada langkah ini kita melakukan konfigurasi load balancing yang pertama kita konfigurasi yaitu frontend dimana ini berfungsi agar user dapat langsung melihat web server hasil konfigurasi dapat dilihat pada Gambar 5 dibawah ini.

```
frontend haproxy_in
        bind 192.168.1.30:80
        mode http
        reqadd X-Forwartded-Proto:\ http
        default_backend haproxy_http
```

Gambar 5. Hasil Konfigurasi Frontend

Selanjutnya kita membuat konfigurasi banckend dimana berfungsi untuk memastikan fitur web server berjalan sesuai penjadwalan yang telah kita lakukan hasil konfigurasi dapat dilihat pada Gambar 6 dibawah ini.

```
backend haproxy_http
        mode http
        balance roundrobin
        option httpchk
        option forwardfor
        server master-skripsi     192.168.1.11:80 check
        server clustering-skripsi 192.168.1.12:80 check
```

Gambar 6. Hasil Konfigurasi Backend

Dan yang terakhir kita membuat konfigurasi listen stats dimana berfungsi untuk melihat statistik pada *haproxy* hasil konfigurasi dapat dilihat pada Gambar 7 dibawah ini.

```
listen stats
        bind 192.168.1.30:32700
        mode http
        stats enable
        stats hide-version
        stats realm Haproxy\ Statistics
        stats uri /  #url untuk melihat statistik haproxy
        stats auth user:password #akses ke statistik haproxy
        stats refresh 10s
        stats show-node
```

Gambar 7 Hasil Konfigurasi Listen Stats

Gambar 8 dibawah ini adalah hasil akhir dari konfigurasi yang telah kita lakukan untuk load balancing web server master dan clustering.

```
defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

frontend haproxy_in
        bind 192.168.1.30:80
        mode http
        reqadd X-Forwartded-Proto:\ http
        default_backend haproxy_http

backend haproxy_http
        mode http
        balance roundrobin
        option httpchk
        option forwardfor
        server master-skripsi     192.168.1.11:80 check
        server clustering-skripsi 192.168.1.12:80 check

listen stats
        bind 192.168.1.30:32700
        mode http
        stats enable
        stats hide-version
        stats realm Haproxy\ Statistics
        stats uri /  #url untuk melihat statistik haproxy
        stats auth user:password #akses ke statistik haproxy
        stats refresh 10s
        stats show-node
```

Gambar 8 Hasil Akhir Konfigurasi Load Balancing

Lakukan langkah berikut nya dengan restart *haproxy* dengan perintah seperti pada Gambar 9 dibawah ini dan load balancing sudah bisa digunakan.

```
root@balancing-skripsi:/home/balancing# systemctl restart haproxy
root@balancing-skripsi:/home/balancing#
```

Gambar 9 Hasil Perintah Restart Haproxy

Kedua kita melakukan *clustering database server* pada master dan clustering dengan langkah-langkah seperti berikut ini. Pertama kita install dan konfigurai pada master mengganti konfigurasi di *#mysqld.cnf* masuk dengan menggunakan perintah seperti pada Gambar 10 dibawah ini.

```
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 192.168.1.11
#
# * Fine Tuning
#
key_buffer_size         = 16M
max_allowed_packet      = 16M
thread_stack            = 192K
thread_cache_size       = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
myisam-recover-options  = BACKUP
#max_connections        = 100
#table_cache            = 64
#thread_concurrency     = 10
#
# * Query Cache Configuration
#
query_cache_limit       = 1M
query_cache_size        = 16M
```

Gambar 10. Hasil Konfigurasi Mysqld.cnf Master

Sever id adalah intuk mengidentifikasikan nama dari server log_bin yang dimana berfungsi sebagai lokasi terakhir data disimpan. Langkah selanjutnya restart mysql dengan perintah seperti pada Gambar 11 dibawah ini.

```
root@master-skripsi:/home/master# nano /etc/mysql/mysql.conf.d/mysqld.cnf
root@master-skripsi:/home/master# /etc/init.d/mysql restart
[ ok ] Restarting mysql (via systemctl): mysql.service.
root@master-skripsi:/home/master#
```

Gambar 11. Hasil Restart Mysql

Setelah *restart* selesai kemudian masuk kedalam mysql dengan perintah pada Gambar 12 dibawah ini.

```
root@master-skripsi:/home/master# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.32-0ubuntu0.16.04.1-log (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Gambar 12 Hasil Masuk Mysql

Setelah masuk pertama-tama kita membuat user *repl1* di mysql seperti Gambar 13 dibawah ini.

```
mysql> create user 'repl1'@'%' identified by 'repl1';
Query OK, 0 rows affected (0.73 sec)
```

Gambar 13. Hasil Create User Master

Selanjutnya kita memberi permision user repl1 seperti Gambar 14 dibawah ini.

```
mysql> grant replication slave on *.* to 'repl1'@'192.168.1.12' identified by 'repl1;
    '> ';
Query OK, 0 rows affected, 1 warning (0.06 sec)
```

Gambar 14 Hasil Permission User Master

Selanjutnya kita melihat status master seperti pada Gambar 15 dibawah ini.

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000154 |      449 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

Gambar 15. Hasil Show Status Master

Selanjutnya mematikan service slave seperti pada Gambar 16 dibawah ini.

```
mysql> stop slave
    -> ;
Query OK, 0 rows affected (0.08 sec)
```

Gambar 16. Hasil Stop Slave

Selanjutnya konfigurasi terakhir kita harus samakan *master_log_file* dan *master_log _pos* pada master, jika pada konfigurasi ini kita salah maka *clustering database* tidak akan connect, hasil konfigurasi ini bisa dilihat pada Gambar 17 dibawah ini.

```
mysql> change master to
    -> master_host='192.168.1.12',
    -> master_user='repl1',
    -> master_password='repl1',
    -> master_port=3306,
    -> master_log_file='mysql-bin.000154',
    -> master_log_pos=449,
    -> master_connect_retry=10;
Query OK, 0 rows affected, 2 warnings (0.25 sec)
```

Gambar 17. Hasil Konfigurasi Change Master

Selanjutnya nyalakan kembali service slave seperti pada Gambar 18 dibawah ini dan clustering databases sudah terhubung.

```
mysql> start slave;
Query OK, 0 rows affected (0.04 sec)
```

Gambar 18. Hasil Start Slave

Sekarang kita check apakah clustering database yang kita bat udah benar berjalan dengan sempurna atau belum hasilnya dapat dilihat pada Gambar 19 dibawah ini.

```
mysql> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State: Connecting to master
                  Master_Host: 192.168.1.12
                  Master_User: repl1
                  Master_Port: 3306
                Connect_Retry: 10
              Master_Log_File: mysql-bin.000154
          Read_Master_Log_Pos: 449
               Relay_Log_File: master-skripsi-relay-bin.000001
                Relay_Log_Pos: 4
        Relay_Master_Log_File: mysql-bin.000154
             Slave_IO_Running: Connecting
            Slave_SQL_Running: Yes
```

Gambar 19. Hasil Slave Status Terhubung

Yang berikutnya kita melakukan lagi konfigurasi yang sama seperti pada database master dengan melakukan install dan konfigurasi database clustering mengganti konfigurasi di *#mysqld.cnf* masuk dengan menggunakan perintah seperti pada Gambar 20 dibawah ini.

```
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 192.168.1.12
#
# * Fine Tuning
#
key_buffer_size         = 16M
max_allowed_packet      = 16M
thread_stack            = 192K
thread_cache_size       = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
myisam-recover-options  = BACKUP
#max_connections        = 100
#table_cache            = 64
#thread_concurrency     = 10
#
# * Query Cache Configuration
#
query_cache_limit       = 1M
query_cache_size        = 16M
#
# * Logging and Replication
#
# Both location gets rotated by the cronjob.
# Be aware that this log type is a performance killer.
# As of 5.1 you can enable the log at runtime!
#general_log_file        = /var/log/mysql/mysql.log
#general_log             = 1
#
# Error log - should be very few entries.
#
```

Gambar 20. Hasil Konfigurasi Mysqld.cnf Master

Sever id adalah intuk mengidentifikasikan nama dari server log_bin yang dimana berfungsi sebagai lokasi terakhir data disimpan. Langkah selanjutnya restart mysql dengan perintah seperti pada Gambar 21 dibawah ini.

```
root@master-skripsi:/home/master# nano /etc/mysql/mysql.conf.d/mysqld.cnf
root@master-skripsi:/home/master# /etc/init.d/mysql restart
[ ok ] Restarting mysql (via systemctl): mysql.service.
root@master-skripsi:/home/master#
```

Gambar 21. Hasil Restart Mysql

Setelah *restart* selesai kemudian masuk kedalam mysql dengan perintah pada Gambar 22 dibawah ini.

```
root@master-skripsi:/home/master# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.32-0ubuntu0.16.04.1-log (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Gambar 22. Hasil Masuk Mysql

Setelah masuk pertama-tama kita membuat user *repl1* di mysql seperti Gambar 23 dibawah ini.

```
mysql> create user 'repl2'@'%' identified by 'repl2';
Query OK, 0 rows affected (0.79 sec)
```

Gambar 23. Hasil Create User Master

Selanjutnya kita memberi permision user repl1 seperti Gambar 24 dibawah ini.

```
mysql> grant replication slave on *.* to 'repl2'@'192.168.1.11' identified by 'repl';
Query OK, 0 rows affected, 1 warning (0.04 sec)
```

Gambar 24. Hasil Permission User Cluster

Selanjutnya kita melihat status master seperti pada Gambar 25 dibawah ini.

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000154 |      449 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

Gambar 25. Hasil Show Status Cluster

Selanjutnya mematikan service slave seperti pada Gambar 26 dibawah ini.

```
mysql> stop slave
    -> ;
Query OK, 0 rows affected (0.08 sec)
```

Gambar 26. Hasil Stop Slave

Selanjutnya konfigurasi terakhir kita harus samakan *master_log_file* dan *master_log _pos* pada master, jika pada konfigurasi ini kita salah maka *clustering database* tidak akan connect, hasil konfigurasi ini bisa dilihat pada Gambar 27 dibawah ini.

```
mysql> change master to
    -> master_host='192.168.1.11',
    -> master_user='repl2',
    -> master_password='repl2',
    -> master_port=3306,
    -> master_log_file='mysql-bin.000045',
    -> master_log_pos=1243,
    -> master_connect_retry=10;
Query OK, 0 rows affected, 2 warnings (0.15 sec)
```

Gambar 27. Hasil Konfigurasi Change Master

Selanjutnya nyalakan kembali service slave seperti pada Gambar 28 dibawah ini dan clustering databases sudah terhubung

```
mysql> start slave;
Query OK, 0 rows affected (0.04 sec)
```

Gambar 28. Hasil Start Slave

Sekarang kita check apakah clustering database yang kita bat udah benar berjalan dengan sempurna atau belum hasilnya dapat dilihat pada Gambar 29 dibawah ini.

```
mysql> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State: Connecting to master
                  Master_Host: 192.168.1.11
                  Master_User: repl2
                  Master_Port: 3306
                Connect_Retry: 10
              Master_Log_File: mysql-bin.000045
          Read_Master_Log_Pos: 1243
               Relay_Log_File: clustering-skripsi-relay-bin.000001
                Relay_Log_Pos: 4
        Relay_Master_Log_File: mysql-bin.000045
             Slave_IO_Running: Connecting
            Slave_SQL_Running: Yes
```

Gambar 29 Hasil Slave Status Terhubung

Pengujian

1. **Hasil Pengukuran *Throughput***

Data angka yang digukana pada pengujian *througput* ini diambil dari hasil statistic.

$$Throughput = \frac{Paket\ data\ yang\ diterima}{Lama\ pengamatan}$$

$$= \frac{918397001\ \text{bytes}}{1072{,}799\ \text{s}}$$

= 856069,96 byte

= 6,848 K bits/s

Hasil perhitungan *throughput* dapat dilihat pada tabel IV.2 dibawah ini sebagai berikut:

Tabel 2. Tabel Hasil Perhitungan Throughput

Parameter yang dihitung	Nilai yang didapat
Paket data yang diterima	918397001 bytes
Lama pengamatan	1072,799 s
Throughput	6,848 Kbit/s

2. **Hasil Pengukuran Paket Loss**

Data angka yang digukana pada pengujian *paket loss* ini diambil dari hasil statistic.

$$\text{Paket Loss} = \frac{(paket\ data\ dikirim - paket\ data\ diterima)x\ 100\%}{Paket\ data\ yang\ dikirim}$$

$$= \frac{441036 - 440849}{441036} \text{ X } 100\%$$

= 4,240%

3. Hasil Pengukuran Delay

Data angka yang digukana pada pengujian *delay* ini diambil dari hasil statistic.

Delay rata-rata $= \frac{Total\ Delay}{Total\ paket\ yang\ diterima}$

$= \frac{1072{,}799\ sec}{440849}$

= 0,0040 sec

= 40 ms

Total delay yang didapatkan dengan menjumlahkan keseluruhan delay yang ada antara paket satu dengan pakat lainnya.

Tabel 3. Hasil Perhitungan Rata-rata Delay

Parameter yang dihitung	Nilai yang didapat
Total paket yang diterima	440849 Paket
Total delay	1072,799 sec
Rata-rata delay	40 ms

4. Hasil Pengukuran Jitter

Data angka yang digukana pada pengujian *jitter* ini diambil dari hasil statistic.

Jitter = $\frac{Total\ variasi\ delay}{Total\ paket\ yang\ diterima-1}$

$= \frac{40}{440849-1}$

= 0,0000907 s

=0,907 ms

Tabel 4. Tabel Hasil Perhitungan Jitter

Parameter yang dihitung	Nilai yang didapat
Total paket data yang diterima -1	440848
Total delay	40 ms
Jitter	0,907 ms

5. Hasil Pengukuran Request per-detik

Pengujian load balancing pada penelitian ini dengan menggunakan ab – Apache HTTP server benchmarking tool, dengan dilakukan mengirimkan jumlah request sebanyak 1000 (-n) secara bersamaan (-c) 150 http//:192.168.1.30

KESIMPULAN

Berdasarkan hasil pembahasan dalam bab sebelumnya dapat diambil kesimpulan yaitu:

1. Sistem *load* balancing dapat bekerja dengan baik ketika *request* datang dari *client* telah berhasil didistribusikan *balancer* secara merata kepada setiap *node cluster*. Sehingga server tidak mengalami *overload* dan kemampuan *web server* bisa melayani 10.000 *request* dengan tidak mengalami *error request*.
2. Dengan implementasi *clustering server* dan meningkatkan jumlah *current connection* yang dapat dilayani oleh server.

Saran

Terdapat beberapa hal yang dapat dikembangkan agar dapat dihasilkan kinerja sistem jaringan yang lebih baik, antara lain:

1. Untuk *balancer* bisa dibuat menjadi 2 buah salah satunya difungsikan sebagai *balancer* cadangan. Sehingga ketersediaan *server* akan lebih baik.
2. Pengembangan ke depannya dapat membandingkan performa 2 jenis load balancing yang ada pada penelitian ini dengan jenis yang lain agar dapat mengetahui konfigurasi performa server yang lebih baik

DAFTAR PUSTAKA

Rahmatulloh, A. & Firmansyah (2017). "*Implementasi Laoad Balancing Web Server menggunakan Haproxy dan Singkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi*". Teknosi Jurnal Teknologi dan Sistem Informasi Vol. 03, No.02, hal 241-248. Retrieved from

https://teknosi.fti.unand.ac.id/index.php/teknosi/article/download/348/117. Diakses pada tanggal 10 Maret 2020 jam 13.20 WIB.

Taufiq, A.G., Aulia, A & Melinda. (2015). "*Analisis Kinerja MySQL Cluster Menggunakan Metode Loaad Balancing"*. Neliti Jurnal Rekayasa Eletrika Vol.11, No.4, agustus 2015, hal 129-134. Retrieved from https://www.neliti.com/publications/129157/analisis-kinerja-mysql-cluster-menggunakan-metode-load-balancing.ISSN. 1412-4785; e-ISSN. 2252-620x Diakses tanggal 10 Maret 2020 jam 15.00 WIB.

Arifin (2018)."*Membangun Server Dan Analisis Backup Database Postgresql Menggunakan Teknik Replication Master/Slave*".Skripsi. Teknologi Industri, Teknik Informatika, Institut Sains & Teknoligi AKPRIND Yogyakarta.

Hindro. (2018, November 16). *Pengertian Database.* Retrieved from Termas Media:

https://www.termasmedia.com/lainnya/software/69-pengertiandatabase.html

Ubuntu, I. (2018, November 17). *Pengertian Ubuntu* Retrieved from Komunitas Ubuntu Indonesia: https://id.wikipedia.org/wiki/Ubuntu

Yuliano, T. (2007, Januari 1). *Ilmu Komputer* Retrieved from Pengenalan PHP: https://ilmukomputer.org/wpcontent/uploads/2009/03/triswanpengenalanphp.pdf

Bobanto., dkk (2014). "Analisis Kualitas Layanan Jaringan Internet" (Studi Kasus PT. Kawanua Internetindo Manado), UNSRAT:Manado.

Zaenal .A (2018). "*Perancangan dan Implementasi Replikasi Server Menggunakan Metode FAILOVER Cluster*".Skripsi. Teknologi Industri, Teknik Informatika, Institut Sains & Teknoligi AKPRIND Yogyakarta.