

Amrul Faruq_HIGH AVAILABILITY DYNAMIC SHARDING

by Naspub Turnitin

Submission date: 20-Apr-2024 08:21AM (UTC+0700)

Submission ID: 2354573393

File name: 416-Article_Text-2536-2645-10-20220213.pdf (1.79M)

Word count: 3267

Character count: 21024



Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author: Naspub Turnitin
Assignment title: Amrul Faruq 4
Submission title: Amrul Faruq_HIGH AVAILABILITY DYNAMIC SHARDING
File name: 416-Article_Text-2536-2645-10-20220213.pdf
File size: 1.79M
Page count: 10
Word count: 3,267
Character count: 21,024
Submission date: 20-Apr-2024 08:21AM (UTC+0700)
Submission ID: 2354573393

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 5, Nomor 1, Januari 2022.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI: 10.36595/misi.v5i1
<http://e-journal.stmiklombok.ac.id/index.php/misi>



HIGH AVAILABILITY DYNAMIC SHARDING DATABASE SERVER DENGAN METODE FAIL OVER DAN CLUSTERING

Alfirda Desember Riawati¹, M Irfan², Khaeruddin³, Amrul Faruq⁴

^{1,2}Program Studi Teknik Elektro, Fakultas Teknik, Universitas Muhammadiyah Malang
^{3,4}Program Studi IIS Teknologi Elektro, Fakultas Teknik, Universitas Muhammadiyah Malang

Jl. Raya Tlogomas No. 246 Malang - Jawa Timur 65144

¹alfirda@vechmail.umma.ac.id, ²irfan@stmm.ac.id, ³khaeruddin@stmm.ac.id, ⁴faruq@stmm.ac.id

Abstract

NoSQL is a type of database that is currently developing, which has a special design in solving problems related to scalability and reliability. The document format is a document-oriented database that is supported by the NoSQL database type. MongoDB is a document-oriented DBMS. MongoDB has a GridFS feature that can store data in binary form. Sharding can distribute data to multiple machines/servers, this function is also included in the MongoDB feature. Placing data on multiple machines makes it possible to store more data and handle larger loads without the need for a high-performance machine. High Availability, one of which is the ability of a system or group of systems (cluster) to keep the application or service running. The results obtained in this study are that a NoSQL-based database sharding system using MongoDB has been successfully implemented on a server that is set with High Availability, the main function of the server is for failover and clustering.

Keywords: Sharding, Database, High Availability, Failover, Clustering

Abstrak

NoSQL merupakan salah satu jenis database yang berkembang saat ini, yang mempunyai desain khusus dalam memecahkan permasalahan yang berhubungan dengan scalability dan reliability. Format dokumen adalah salah satu jenis document-oriented database yang didukung oleh jenis database NoSQL. MongoDB merupakan salah satu jenis DBMS berbentuk document-oriented, MongoDB memiliki fitur GridFS yang dapat menyimpan data dalam bentuk biner. Sharding dapat mendistribusikan data ke dalam beberapa mesin/server, fungsi ini juga terdapat didalam fitur mongoDB. Dengan menempatkan data pada beberapa mesin memungkinkan untuk menyimpan lebih banyak data dan menangani beban lebih besar tanpa diperlukan adanya mesin dengan performansi tinggi. High Availability salah satunya merupakan kemampuan satu sistem atau kelompok sistem (cluster) menjaga aplikasi (application) atau layanan (service) berjalan. Hasil yang didapat pada penelitian ini adalah sebuah system sharding database berbasis noSQL, menggunakan mongoDB berhasil diimplementasikan pada server yang di set dengan High Availability, fungsi utama server adalah untuk failover dan clustering.

Kata kunci: Sharding, Database, High Availability, Failover, Clustering

1. PENDAHULUAN

Web menjadi media berorientasi bisnis dan antarmuka yang lebih disukai untuk sistem informasi terdistribusi. Regulator menuntut semakin besar ukuran informasi yang berbentuk teks digital. Dokumen teks dengan jumlah yang sedikit, tentunya mudah bagi manusia untuk melakukan kategorisasi secara manual, namun pada dasarnya proses tersebut menjadi

suatu kondisi yang sulit dilakukan jika jumlah dokumennya mencapai ratusan hingga ribuan. Sehingga dibutuhkan sebuah sistem yang dapat mengolah dan mengategorikan dokumen dengan jumlah besar tersebut secara otomatis. Salah satu penggunaan yaitu server clustering merupakan suatu opsi yang dapat digunakan untuk mengatasi jika terjadi permasalahan tersebut, yaitu suatu teknologi

ISSN : 2614-1701 (Cetak) - 2614-3739 (Online)

1



HIGH AVAILABILITY DYNAMIC SHARDING DATABASE SERVER DENGAN METODE FAIL OVER DAN CLUSTERING

Afirda Desember Riawati¹, M Irfan², Khaeruddin³, Amrul Faruq⁴

^{1,2}Program Studi Teknik Elektro, Fakultas Teknik, Universitas Muhammadiyah Malang

^{3,4}Program Studi D3 Teknologi Elektronika, Fakultas Teknik, Universitas Muhammadiyah Malang

Jl. Raya Tlogomas No. 246 Malang - Jawa Timur 65144

¹afirda@webmail.umm.ac.id, ²irfan@umm.ac.id, ³lamone@umm.ac.id, ⁴faruq@umm.ac.id

Abstract

NoSQL is a type of database that is currently developing, which has a special design in solving problems related to scalability and reliability. The document format is a document-oriented database that is supported by the NoSQL database type. MongoDB is a document-oriented DBMS, MongoDB has a GridFS feature that can store data in binary form. Sharding can distribute data to multiple machines/servers, this function is also included in the MongoDB feature. Placing data on multiple machines makes it possible to store more data and handle larger loads without the need for a high-performance machine. High Availability, one of which is the ability of a system or group of systems (cluster) to keep the application or service running. The results obtained in this study are that a NoSQL-based database sharding system using MongoDB has been successfully implemented on a server that is set with High Availability, the main function of the server is for failover and clustering.

Keywords: *Sharding, Database, High Availability, Failover, Clustering*

Abstrak

NoSQL merupakan salah satu jenis database yang berkembang saat ini, yang mempunyai desain khusus dalam memecahkan permasalahan yang berhubungan dengan *scalability* dan *reliability*. Format dokumen adalah salah satu jenis *documentorienteddatabase* yang didukung oleh jenis database *NoSQL*. *MongoDB* merupakan salah satu jenis *DBMS* berbentuk *documentoriented*, *mongoDB* memiliki fitur *GridFS* yang dapat menyimpan data dalam bentuk biner. *Sharding* dapat mendistribusikan data ke dalam beberapa mesin/server, fungsi ini juga terdapat didalam fitur *mongoDB*. Dengan menempatkan data pada beberapa mesin memungkinkan untuk menyimpan lebih banyak data dan menangani beban lebih besar tanpa diperlukan adanya mesin dengan performansi tinggi. *High Availability* salah satunya merupakan kemampuan satu sistem atau kelompok sistem (*cluster*) menjaga aplikasi (*application*) atau layanan (*service*) berjalan. Hasil yang didapat pada penelitian ini adalah sebuah system *sharding* database berbasis *noSQL* menggunakan *mongoDB* berhasil diimplementasikan pada server yang diset dengan *High Availability*, fungsi utama server adalah untuk *failover* dan *clustering*.

Kata kunci: *Sharding, Database, High Availability, Failover, Clustering*

1. PENDAHULUAN

Web menjadi media berorientasi bisnis dan antarmuka yang lebih disukai untuk sistem informasi terbaru [1]. Begitu pesatnya menuntut semakin besar ukuran informasi yang berbentuk teks digital. Dokumen teks dengan jumlah yang sedikit, tentunya mudah bagi manusia untuk melakukan kategorisasi secara manual, namun pada dasarnya proses tersebut menjadi

suatu kondisi yang sulit dilakukan jika jumlah dokumennya mencapai ratusan hingga ribuan. Sehingga dibutuhkan sebuah sistem yang dapat mengelolakan mengkategorikan dokumen dengan jumlah besar tersebut secara otomatis. Salah satu penggunaan yaitu server *Clustering* merupakan suatu opsi yang dapat digunakan untuk mengatasi jika terjadi permasalahan tersebut, yaitu suatu teknologi



yang menggabungkan beberapa server yang bekerja bersama-sama yangseolah-olah merupakan satu sistem tunggal[2].

Pada umumnya, sistem kategorisasi dokumen hanya sampai pada tahap kategorisasi saja. Pada sistem yang sudah ada dokumen hasil kategorisasi tersebut ke dalam basis data agar pengolahan data dapat dilakukan dengan mudah. Tren basis data yang berkembang saat ini yaitu NoSQL yang didesain khusus untuk memecahkan permasalahan *scalability* dan *reliability*. Salah satu jenis dari NoSQL adalah document *oriented* database yang menyimpan data dalam format dokumen. Salah satu DBMS document *oriented* adalah MongoDB. MongoDB memiliki fitur *GridFS* yang dapat menyimpan data dalam bentuk file biner[2].

Selain itu, MongoDB juga memiliki fitur *sharding* yang dapat mendistribusikan data ke dalam beberapa mesin. Dengan menempatkan data pada beberapa mesin memungkinkan untuk menyimpan lebih banyak data dan menangani beban lebih besar tanpa diperlukan adanya mesin dengan performansi tinggi. *High Availability* salah satunya merupakan kemampuan satu sistem atau kelompok sistem (*cluster*) menjaga aplikasi (*application*) atau layanan (*service*) berjalan[3][12].

2. TINJAUAN PUSTAKA DAN TEORI

2.1. Tinjauan Pustaka

High Availability merupakan server berbeda bekerja bersama-sama untuk memastikan downtime pada resource dikurangi seminimal mungkin. Tujuan dari *High Availability Cluster* adalah untuk memastikan bahwa sumber daya mencapai *Availability* semaksimal mungkin [3]. Jika server down atau jika sumber daya yang berhenti, HA cluster akan memonitor dan memastikan resource atau sumber daya dihidupkan ulang pada tempat lain dalam sistem cluster, sehingga dapat digunakan lagi setelah mengalami gangguan.

Konsep tersebut berkaitan dengan kemampuan sistem untuk mengatasi terjadinya gangguan, kerusakan hardware, crash/down, kesalahan jaringan bahkan kegagalan server yang di sebabkan software yang gagal melakukan tugas semestinya. 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 [4].

Failover merupakan sistem komunikasi dua atau lebih server ditempat yang berbeda yang dapat saling backup data dan mampu menggantikan pelayanan bila server lain down. Failover bertujuan untuk membantu menjaga akses client ke sumber daya di server, ketika server mengalami kegagalan fungsi software, atau kegagalan akses server. Failover cluster merupakan sekumpulan server yang saling bekerjasama untuk memberikan pelayanan meskipun berada ditempat yang berbeda, dan memiliki kualitas data atau sumber daya yang sama antara server yang satu dengan server lainnya. Sistem Failover akan bekerja untuk menghubungi server-server yang menjadi anggota cluster-nya untuk mengambil alih tugas server utama saat terjadi kegagalan fungsi dalam waktu tertentu [5][11].

Pada tahun 1998 pertama kalinya dikembangkan sebuah konsep penyimpanan basis data yaitu NoSQL oleh Carlo Strozzi, yang kemudian pada tahun 2009 Eric Evans memperkenalkan kembali teknologi NoSQL. Kehadiran NoSQL bukan berarti untuk menggantikan model RDBMS yang sudah ada. Awal kemunculannya dilatarbelakangi oleh beberapa masalah yang muncul dari RDBMS. NoSQL dan RDBMS memiliki kelebihan dan tempat masing-masing sehingga diharapkan dapat saling melengkapi teknologi penyimpanan basis data [6]. Pada MongoDB dokumen disimpan dalam bentuk BSON (Binary JSON) [7]. Dengan struktur yang mirip dengan JSON membuatnya cukup mudah untuk dibaca.

Database NoSQL terbukti unggul dalam proses transaksi CRUD daripada SQL namun lemah untuk menghadapi transaksi join/agregasi, database NoSQL mampu memenuhi kebutuhan aplikasi ERP Retail yang bisa mensupport perbedaan skema dari setiap tenant, database NoSQL bersifat scalable[8]. Tujuan daripada penelitian ini adalah implementasi dan membuktikan efektifitas teknologi virtualisasi Docker terintegrasi dengan openstack dalam hal penggunaan storage dan memory.

Ide penelitian ini didasarkan pada referenesi [3][4] tentang high availability dan fail over [5] yang diimplementasikan ke system database berbasis NoSQL [6][7][8] khususnya mongoDB. Penelitian ini membuktikan bahwa High



Availability Dynamic Sharding Database Server dengan metode Fail Over dan Clustering berhasil diimplementasikan.

2.2. High available (HA)

High available (HA) adalah sistem *cluster* yang terdiri dari dua buah server atau lebih mesin server yang biasa dikenal dengan node. Semua node yang terhubung dalam network saling berkomunikasi dan terhubung, pada setiap node pada cluster berkomunikasi dan menyampaikan informasi koneksi melalui *heart-beat*.

Cluster juga memiliki penyimpanan untuk menyimpan data dan data tersebut terhubung pada jaringan publik. Sebuah sistem *cluster* dapat menahan kegagalan perangkat keras dan perangkat lunak pada salah satu node dengan waktu *downtime* yang sangat kecil. Cluster dikonfigurasi untuk mendukung *high availability* dan *up time* pada server. HA cluster terdiri dari beberapa komponen hardware dan software yang rumit, dimana komponen *software*-nya adalah *operating system*, volume manager dan database[3].

2.2.1 Continuous Availability

Continuous availability merupakan sistem yang dimana didalamnya memiliki service yg dituntut agar sistem selalu pada kondisi available. Ini bertujuan untuk membuat sistem selalu siap dan tidak terjadi down. Untuk menjalankan sistem dibutuhkan 3 standar yang dapat digunakan sebagai implementasi yaitu *Programmatic*, *Shared Availability*, dan *Multiple Copy*[5].

2.2.2 Failover Availability

Proses *failover failure* membutuhkan waktu yang sedikit saat terjadi down pada sistem, dimana didalam system menggunakan 2 buah server dengan data identik pada masing-masing server. Sistem yang menggunakan konsep ini berjalan normal tanpa ada kendala, hanya saja pada master server yang melakukan suatu proses pada seluruh *user* pada suatu proses. Master server mengalami proses *failure* dan slave server menanggapi proses tersebut, slave server akan mengganti fungsi dari master server sehingga sistem server saling melakukan *back up* satu sama lain.

2.3. MongoDB

MongoDB merupakan salah satu produk open source yang banyak digunakan dimana sistem ini menggunakan struktur data JSON untuk proses penyimpanannya data. MongoDB juga

sering oleh desainer sistem untuk penggunaan aplikasi dengan sistem yang terkoneksi cloud computing, grid computing atau big data[9].

MongoDB yang merupakan suatu sistem database dengan istilah open source berbasis dokumen yang dapat digunakan oleh semua *user* tanpa perlu *license*, MongoDB awalnya dibuat menggunakan bahasa C++ dan telah dikembangkan oleh 10gen sejak oktober 2007. Dokumen yang tersimpan didalam MongoDB dapat memiliki atribut yang berbeda dengan dokumen lain walaupun berada dalam satu koleksi[6].

MongoDB yang merupakan salah satu produk NoSQL juga dibagi menurut format penyimpanan data yaitu sebagai berikut :

- Document database dimana setiap satu objek data yang ada dapat disimpan dalam dokumen-dokumen, kunci-kunci atau *keyvalue*, dan *value/nilai* dapat berupa jenis data array.
- Graph merupakan penyimpanan data dalam bentuk graph yang sering digunakan dalam suatu interaksi seperti jejaring sosial.
- Key-value atau kunci terenkripsi pada database ini salah satu contohnya adalah *hadoop cassandra*.
- Objek database merupakan suatu format dari database yang telah tersimpan dalam suatu objek[10].

3 METODOLOGI PENELITIAN

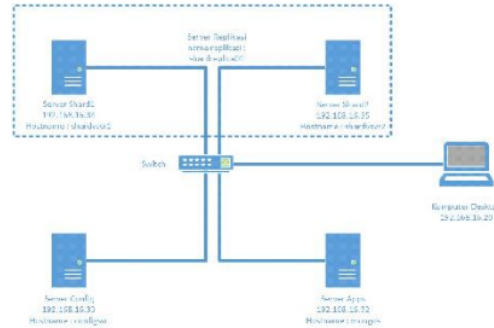
Dalam perancangan system inipembahasannya meliputi proses instalasi, konfigurasi dan integrasi perangkat lunak agar dapat berjalan pada jaringan yang dirancang. System sharding database yang dirancang adalah menggunakan empat unit server, dimana dua unit server difungsikan sebagai server *Shard* yang digunakan untuk menyimpan semua data. *Config Server* yang digunakan untuk menyimpan metadata klaster, dan server *Mongos/Query Router* yang digunakan untuk antarmuka aplikasi.

Aplikasi database yang digunakan dalam penelitian ini adalah MongoDB yang sistem basis datanya berorientasi dokumen dan dikembangkan oleh MongoDB Inc, aplikasi ini mempunyai performansi kinerja yang tinggi, ketersediaan tinggi, dan penskalaan otomatis. Banyak digunakan oleh instansi maupun perusahaan

yang mempunyai sistem basis datayang kompleks. Pada penelitian ini web server yang digunakan adalah *httpd*, webbased manajemen basis data yang digunakan adalah *Mongo Express* dengan diintegrasikan dengan PHP5.

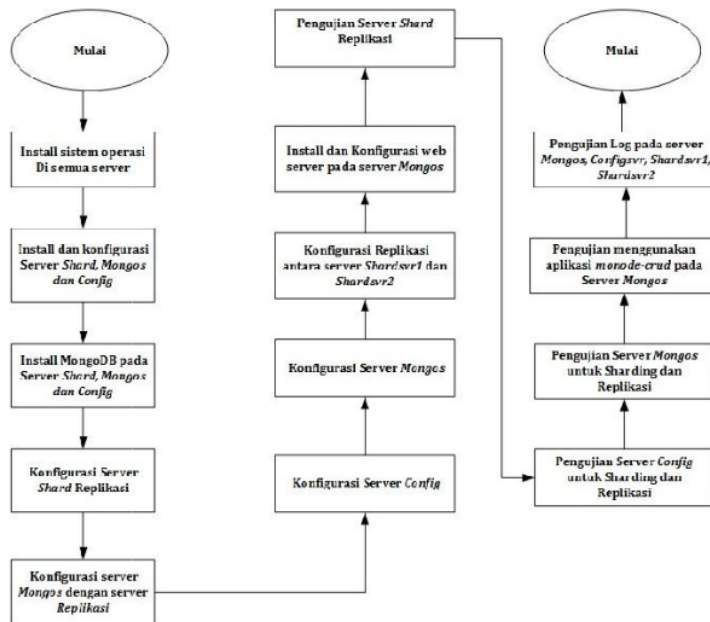
Tabel 1 IP Address Sistem

N o	IP Server	Host Server	
1.	192.168.10.30	Mongos	Mongos
2.	192.168.10.31	Configsvr	Configsvr
3.	192.168.10.32	Shardsvr1	Shardsvr1
4.	192.168.10.33	Shardsvr2	Shardsvr2



Gambar 1. Topologi Jaringan High Availability Dynamic Sharding Database Server dengan

Pada penelitian ini menggunakan sistem operasi linux dan terinstall disemua server, linux yang digunakan adalah yang berbasis server. Adapun sistem operasi linux yang digunakan adalah distro turunan redhat yaitu CentOS 7 dan merupakan versi yang sesuai dengan kebutuhan penelitian.



Gambar 2. Rancangan pembangunan sistem High Availability Dynamic Sharding Database Server dengan metode Fail Over dan Clustering



Topologi jaringan yang digunakan dalam Penelitian ini adalah seperti berikut, dimana ada dua buah server yang difungsikan sebagai server sharding dengan sistem replikasi, satu server difungsikan sebagai server config dan satu server difungsikan sebagai server apps.

4 HASIL DAN PEMBAHASAN

Untuk mengetahui instalasi dan konfigurasi yang dilakukan sudah sesuai, maka pengujian keseluruhan sistem dilakukan pertahap. Pengujian tahap pertahap adalah memastikan system replikasi pada server berfungsi dengan semestinya, didalam system replikasi ada server yang berfungsi sebagai master dan slave.

```
[root@shardsvr1 elektro]# mongo -host shardsvr1 -port 27017
shardreplica01:PRIMARY> rs.isMaster()
{
  "hosts": [
    "shardsvr1:27017",
    "shardsvr2:27017"
  ],
  "setName": "shardreplica01",
  "setVersion": 1,
  "ismaster": true,
  "secondary": false,
  "primary": "shardsvr2:27017",
  "me": "shardsvr2:27017",
  "electionId": ObjectId("7fffffff0000000000000000"),
  "lastWrite": {
    "opTime": {
      "ts": Timestamp(1523330019, 1),
      "t": NumberLong(94)
    },
    "lastWriteDate": ISODate("2018-04-10T03:26:59Z")
  },
  "maxBsonObjectSize": 16777216,
  "maxMessageSizeBytes": 48000000,
  "maxWriteBatchSize": 1000,
  "localTime": ISODate("2018-04-10T03:27:07.887Z"),
  "maxWireVersion": 5,
  "minWireVersion": 0,
  "readOnly": false,
  "ok": 1
}
```

Gambar 3. Pengujian server shardsvr1 master

```
[root@shardsvr2 elektro]# mongo -host shardsvr2 -port 27017
shardreplica01:SECONDARY> rs.isMaster()
{
  "hosts": [
    "shardsvr1:27017",
    "shardsvr2:27017"
  ],
  "setName": "shardreplica01",
  "setVersion": 1,
  "ismaster": false,
  "secondary": true,
  "primary": "shardsvr2:27017",
  "me": "shardsvr1:27017",
  "lastWrite": {
    "opTime": {
      "ts": Timestamp(1523330795, 1),
      "t": NumberLong(94)
    },
    "lastWriteDate": ISODate("2018-04-10T03:26:35Z")
  },
  "maxBsonObjectSize": 16777216,
  "maxMessageSizeBytes": 48000000,
  "maxWriteBatchSize": 1000,
  "localTime": ISODate("2018-04-10T03:26:44.303Z"),
  "maxWireVersion": 5,
  "minWireVersion": 0,
  "readOnly": false,
  "ok": 1
}
```

Gambar 4. Pengujian server shardsvr2 slave

Pada pengujian antara server shardsvr1 dan shardsvr2 yang berfungsi sebagai server master

adalah server shardsvr2 dan untuk slavenya adalah server shardsvr1. Jika computer server direstart atau dimatikan fungsi slave dan master ini kadang-kadang berubah-ubah, kadang server shardsvr1 akan berfungsi sebagai master dan shardsvr2 sebagai slave. Tetapi fungsinya sebagai server database tetap berjalan dengan normal.

```
shardreplica01:SECONDARY> db.printSlaveReplicationInfo()
source: shardsvr1:27017
  syncedTo: Tue Apr 10 2018 11:05:21 GMT+0700 (WIB)
  0 secs (0 hrs) behind the primary
```

Gambar 5. Pengujian server shardsvr2 slave perintah db.printSlaveReplicationInfo()

Informasi server slave, dimana pada pengujian dengan mengetikan perintah db.printSlaveReplicationInfo() server slave nya adalah shardsvr1 dan terjadi sinkronisasi dengan server master shardsvr2 pada jam 11:05 WIB.

```
configsvr01:PRIMARY> rs.isMaster()
{
  "hosts": [
    "192.168.16.33:27019"
  ],
  "setName": "configsvr01",
  "setVersion": 1,
  "ismaster": true,
  "secondary": false,
  "primary": "192.168.16.33:27019",
  "me": "192.168.16.33:27019",
  "electionId": ObjectId("7fffffff0000000000000000"),
  "lastWrite": {
    "opTime": {
      "ts": Timestamp(1523330931, 1),
      "t": NumberLong(15)
    },
    "lastWriteDate": ISODate("2018-04-10T03:28:51Z"),
    "majorityOpTime": {
      "ts": Timestamp(1523330931, 1),
      "t": NumberLong(15)
    },
    "majorityWriteDate": ISODate("2018-04-10T03:28:51Z")
  },
  "configsvr": 1,
  "maxBsonObjectSize": 16777216,
  "maxMessageSizeBytes": 48000000,
  "maxWriteBatchSize": 1000,
  "localTime": ISODate("2018-04-10T03:28:51.928Z"),
  "maxWireVersion": 5,
  "minWireVersion": 0,
  "readOnly": false,
  "ok": 1
}
```

Gambar 6. Pengujian sinkronisasi server shardsvr2 slave

Jumlah database yang disinkronisasikan antara *slave* dan *master* adalah 990MB dan terjadi sinkronisasi pada jam 11:05 WIB. Pengujian tahap kedua adalah memastikan system sharding pada server berfungsi dengan semestinya. Untuk melakukan pengujian sharding dilakukan di server *mongos* dan juga di server *shardsvr1 shardsvr2*. Kenapa dilakukan juga di server *mongos*, karena pada saat penambahan server sharding dilakukan di server *mongos*. Server *mongos* juga adalah server pengontrol sistem sharding dan replikasi database *mongod*. Juga server *mongos* adalah server untuk menyimpan web aplikasi. Pada pengujian berikut diperlihatkan *Sharding Status* pada server *mongo* diperlihatkan versi sharding nya adalah *version 6*, dan untuk *shards* nya adalah *shardreplica01* dan host nya ada *shardsvr1, shardsvr2* dengan port *27017*, dimana *active mongoses* nya dengan *version 3.4.13*

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
    "id" : 1,
    "minCompatibleVersion" : 5,
    "currentVersion" : 6,
    "clusterId" : ObjectId("5a55ea9c6e8921c2646fb9ba")
  }
  shards:
    { "id" : "shardreplica01", "host" : "shardreplica01/shardsvr1:27017,shardsvr2:27017" }
  active mongoses:
    "3.4.13" : 1
  autosplit:
    Currently enabled: yes
  balancer:
    Currently enabled: yes
    Currently running: no
  Wail
  Failed balancer rounds in last 5 attempts: 5
  Last reported error: could not find host matching read preference { mode: 'primary' }
  Time of Reported error: Tue Apr 10 2018 10:13:40 GMT+0700 (WIB)
  Migration Results for the last 24 hours:
    No recent migrations
```

Gambar 7. Pengujian *Sharding Status* pada server *mongo*

Pada bagian database terdapat beberapa database yang telah dibuat, disini menggunakan database *monode-crud* dimana primary sharding nya adalah *shardreplica01*.

```
shardreplica01:SECONDARY> db.printReplicationInfo()
configured oplog size: 990MB
log length start to end: 1188988secs (330.27hrs)
oplog first event time: Tue Mar 27 2018 16:49:23 GMT+0700 (WIB)
oplog last event time: Tue Apr 10 2018 11:05:51 GMT+0700 (WIB)
now: Tue Apr 10 2018 11:05:55 GMT+0700 (WIB)
shardreplica01:SECONDARY>
```

Gambar 8. Pengujian database *Sharding Status* pada server *mongo*

Pengujian server *configsvr* tetapi tetap diset replikasinya, untuk master dan slavenya hanya *configsvr* itu sendiri, seperti terlihat primary nya adalah *192.168.16.33* begitupun slavenya.

[root@configsvr elektro]# mongo -host configsvr -port 27019

```
databases:
  { "_id" : "lemp", "primary" : "shardreplica01", "partitioned" : true }
  lemp.stack
    shard key: { "name" : 1 }
    unique: false
    balancing: true
    chunks:
      shardreplica01 1
      { "name" : { "$minKey" : 1 } } --> { "name" : { "$maxKey" :
  { "_id" : "mahasiswa", "primary" : "shardreplica01", "partitioned" : true }
  mahasiswa.grades
    shard key: { "mahasiswa_id" : 1 }
    unique: false
    balancing: true
    chunks:
      shardreplica01 1
      { "mahasiswa_id" : { "$minKey" : 1 } } --> { "mahasiswa_id"
  { "_id" : "mytrip", "primary" : "shardreplica01", "partitioned" : true }
  { "_id" : "organo", "primary" : "shardreplica01", "partitioned" : true }
  { "_id" : "monode-crud", "primary" : "shardreplica01", "partitioned" : true }
  { "_id" : "blog", "primary" : "shardreplica01", "partitioned" : true }
  { "_id" : "kampus", "primary" : "shardreplica01", "partitioned" : false }
mongos>
```

Gambar 9. Pengujian server *configsvr*

Pengujian pada server *mongos*, server ini membaca konfigurasi ke *configsvr* selain membaca konfigurasi yang berada pada server *mongos* sendiri juga membaca konfigurasi ke server *configsvr*. Untuk melihat database yang ada pada server ketikan command *show dbs*, akan muncul beberapa database yang sudah dibuat, pada pengujian ini menggunakan database *monode-crud*.


```
mongos> show dbs
admin          0.000GB
blog           0.000GB
config        0.001GB
lemp          0.000GB
mahasiswa    0.000GB
monode-crud   0.000GB
mytrip       0.000GB
organo       0.000GB
mongos>
```

Gambar 10.Database pada server

Pengujian ini berfungsi untuk melihat status sharding pada server *mongos* terlihat bahwa server *mongos* membaca konfigurasi yang ada pada server *shardreplica01*.

```
mongos> db.printShardingStatus()
--- Sharding Status ---
sharding version: {
  "_id" : 1,
  "minCompatibleVersion" : 5,
  "currentVersion" : 6,
  "clusterId" : ObjectId("5aaeea9c6e8921c8d46f98a")
}
shards:
  { "_id" : "shardreplica01", "host" : "shardreplica01/shardsvr1:27017,shardsvr2:27017" }
active mongoses:
  "3.4.13" : 1
autosplit:
  Currently enabled: yes
balancer:
  Currently enabled: yes
  Currently running: no
warn:
  Failed balancer rounds in last 5 attempts: 5
  Last reported error: could not find host matching read preference { mode: "primary" }
  Time of Reported error: Tue Apr 10 2018 10:13:40 GMT+0700 (WIB)
  Migration Results for the last 24 hours:
    No recent migrations
databases:
  { "_id" : "lemp", "primary" : "shardreplica01", "partitioned" : true }
    lemp.stack
      shard key: { "name" : 1 }
        unique: false
        balancing: true
        chunks:
          shardreplica01.1
            { "name" : { "$minkey" : 1 } } --> { "name" : { "$maxkey" : 1 } }
  { "_id" : "mahasiswa", "primary" : "shardreplica01", "partitioned" : true }
    mahasiswa.grades
      shard key: { "mahasiswa_id" : 1 }
        unique: false
        balancing: true
        chunks:
          shardreplica01.1
            { "mahasiswa_id" : { "$minkey" : 1 } } --> { "mahasiswa_id" : { "$maxkey" : 1 } }
  { "_id" : "mytrip", "primary" : "shardreplica01", "partitioned" : true }
  { "_id" : "organo", "primary" : "shardreplica01", "partitioned" : true }
  { "_id" : "monode-crud", "primary" : "shardreplica01", "partitioned" : true }
  { "_id" : "blog", "primary" : "shardreplica01", "partitioned" : true }
  { "_id" : "kampus", "primary" : "shardreplica01", "partitioned" : false }
```

Gambar 11.Database pada server

```
mongos> use monode-crud
switched to db monode-crud
mongos> use db
mongos> db.stats()
{
  "ns" : "shardreplica01/shardsvr1:27017,shardsvr2:27017",
  "db" : "monode-crud",
  "collInfo" : {
    "totalIndexSize" : 0,
    "totalIndexSizeInMemory" : 0,
    "objects" : 0,
    "avgObjSize" : 109.160000005666667,
    "dataSize" : 855,
    "storageSize" : 3684,
    "numExtents" : 0,
    "indexes" : 4,
    "indexSize" : 3684,
    "obj" : 1,
    "avgObjSize" : {
      "totalIndex" : 3684,
      "totalIndexInMemory" : 0,
      "collection" : ObjectId("7fffffff000000")
    }
  },
  "diskSize" : 8,
  "avgObjSize" : 109,
  "dataSize" : 855,
  "storageSize" : 3684,
  "numExtents" : 0,
  "indexes" : 4,
  "indexSize" : 3684,
  "fileSize" : 0,
  "wiredTiger" : {
    "num" : 0,
    "totalSize" : 0
  },
  "ok" : 1
}
```

Gambar 12.Database pada server

Melihat stats dari database *monode-crud* menggunakan *db.stats()* terlihat bahwa database *monode-crud* disimpan di server replikasi *shardreplica01/shardsvr1:27017,shardsvr2:27017*. Pada pengujian tahap ketiga adalah untuk memastikan bahwa semua server yang terkoneksi dalam sistem sharding database bisa dimonitoring. Pada pengujian menggunakan *mongostat* terlihat terjadi koneksi antara server *mongos* dengan *localhost:27017* dan server *shardsvr1, shardsvr2* dan sistem replikasinya *shardreplica01*.

Pada pengujian *mongostat* dengan host yang lebih spesifik terlihat adanya koneksi antara server *mongos, configsvr, shardsvr1, shardsvr2* dengan ditandai adanya proses *insert, query, update, delete*. Sedangkan jika terlihat *no data received* ini menandakan tidak adanya proses *insert, query, update, delete* antara server.

Pada pengujian dengan menggunakan *mongostat* untuk tiap host bisa dilihat, untuk server *mongos* dengan ip *192.168.16.32:27017* terlihat tidak ada menggunakan replikasi, untuk server *configsvr* dengan ip *192.168.16.33:27019* terlihat menggunakan replikasi dengan set name *configsvr01*, untuk server *shardsvr1* dengan ip *192.168.16.34:27017* dan *shardsvr2* dengan ip *192.168.16.35:27017* menggunakan replikasi dengan set name *shardreplica01*.



```
[root@mongos test]# mongostat --host 192.168.16.32:27017
insert query update delete getmore command Flushed mapped vszize res faults qrw arw net_in net_out conn time
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 285b 17 0k 13 Apr 10 17:45:41.629
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 284b 17 0k 13 Apr 10 17:45:42.632
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 286b 17 1k 13 Apr 10 17:45:43.635
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 285b 17 1k 13 Apr 10 17:45:44.634
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 287b 17 2k 13 Apr 10 17:45:45.639
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 287b 17 0k 13 Apr 10 17:45:46.635
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 287b 17 2k 13 Apr 10 17:45:47.633
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 285b 17 1k 13 Apr 10 17:45:48.631
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 285b 17 1k 13 Apr 10 17:45:49.631
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 287b 17 2k 13 Apr 10 17:45:50.625
insert query update delete getmore command Flushed mapped vszize res faults qrw arw net_in net_out conn time
*0 *0 *0 *0 *0 0 21C 0 0 0 273M 16.0M 0 0 0 0 0 286b 17 1k 10 Apr 10 17:45:51.624
^C2018-04-10T17:45:51.61+09:00 signal 'interrupt' received; forcefully terminating
[root@mongos test]# mongostat --host 192.168.16.33:27017
insert query update delete getmore command dirty used flushes vszize res qrw arw net_in net_out conn set repl time
*0 *0 *0 *0 *0 0 21C 0.7% 1.3% 0 1.46G 19.3M 0 0 110 153b 47.5k 5 configsvr01 PRI Apr 10 17:45:59.747
*0 *0 *0 *0 *0 0 21C 0.7% 1.3% 0 1.46G 19.3M 0 0 110 153b 48.4k 5 configsvr01 PRI Apr 10 17:47:00.747
*0 *0 *0 *0 *0 0 21C 0.7% 1.3% 0 1.46G 19.3M 0 0 110 200b 47.5k 5 configsvr01 PRI Apr 10 17:47:01.763
*0 *0 *0 *0 *0 0 21C 0.7% 1.3% 0 1.46G 19.3M 0 0 110 655b 50.7k 5 configsvr01 PRI Apr 10 17:47:03.743
*0 *0 *0 *0 *0 0 21C 0.7% 1.3% 0 1.46G 19.3M 0 0 110 1.42k 50.6k 5 configsvr01 PRI Apr 10 17:47:04.735
*0 *0 *0 *0 *0 0 21C 0.7% 1.3% 0 1.46G 19.3M 0 0 110 158b 48.5k 5 configsvr01 PRI Apr 10 17:47:05.732
^C2018-04-10T17:47:05.73+09:00 signal 'interrupt' received; forcefully terminating
[root@mongos test]# mongostat --host 192.168.16.34:27017
insert query update delete getmore command dirty used flushes vszize res qrw arw net_in net_out conn set repl time
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.36G 15.3M 0 0 110 314b 46.7k 6 shardrealic1 SEC Apr 10 17:47:14.705
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.36G 15.3M 0 0 110 161b 46.9k 6 shardrealic1 SEC Apr 10 17:47:15.694
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.36G 15.3M 0 0 110 315b 46.8k 6 shardrealic1 SEC Apr 10 17:47:16.716
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.36G 15.3M 0 0 110 168b 46.6k 6 shardrealic1 SEC Apr 10 17:47:17.701
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.36G 15.3M 0 0 110 330b 49.0k 6 shardrealic1 SEC Apr 10 17:47:18.688
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.36G 15.3M 0 0 110 134b 46.9k 6 shardrealic1 SEC Apr 10 17:47:19.709
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.36G 15.3M 0 0 110 334b 49.3k 6 shardrealic1 SEC Apr 10 17:47:20.685
^C2018-04-10T17:47:20.68+09:00 signal 'interrupt' received; forcefully terminating
[root@mongos test]# mongostat --host 192.168.16.35:27017
insert query update delete getmore command dirty used flushes vszize res qrw arw net_in net_out conn set repl time
*0 *0 *0 *0 *0 1 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 332b 49.7k 12 shardrealic1 PRI Apr 10 17:47:27.380
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 158b 48.4k 12 shardrealic1 PRI Apr 10 17:47:28.271
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 338b 49.1k 12 shardrealic1 PRI Apr 10 17:47:30.271
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 155b 47.2k 12 shardrealic1 PRI Apr 10 17:47:31.207
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 1.52k 109.0k 12 shardrealic1 PRI Apr 10 17:47:32.273
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 155b 47.5k 12 shardrealic1 PRI Apr 10 17:47:33.281
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 325b 49.0k 12 shardrealic1 PRI Apr 10 17:47:34.293
*0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 161b 49.3k 12 shardrealic1 PRI Apr 10 17:47:35.269
^C2018-04-10T17:47:35.26+09:00 signal 'interrupt' received; forcefully terminating
[root@mongos test]#
```

Gambar 13. proses insert, query, update, delete termonitoring di server mongos

```
[root@mongos test]# mongostat --host 192.168.16.32:27017,192.168.16.33:27017,192.168.16.34:27017,192.168.16.35:27017
host insert query update delete getmore command dirty used flushes mapped vszize res faults qrw arw net_in net_out conn set repl time
192.168.16.32:27017 *0 *0 *0 *0 *0 0 2 0 2.7% 1.3% 0 1.46G 19.3M 0 0 110 153b 47.5k 5 configsvr01 PRI Apr 10 17:36:44.745
192.168.16.33:27017 *0 *0 *0 *0 *0 0 2 0 2.7% 1.3% 0 1.46G 19.3M 0 0 110 161b 48.4k 5 configsvr01 PRI Apr 10 17:36:44.742
192.168.16.34:27017 *0 *0 *0 *0 *0 0 1 0 0 2.7% 0.3% 0 1.36G 15.3M 0 0 110 315b 46.8k 6 shardrealic1 SEC Apr 10 17:36:45.216
192.168.16.35:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 1.3% 0 0 273M 16.0M 0 0 0 0 0 288b 17.3k 13 07M Apr 10 17:36:45.266
192.168.16.32:27017 *0 *0 *0 *0 *0 0 21C 0.2% 0.3% 0 1.48G 13.3M 0 0 110 155b 47.2k 12 shardrealic1 PRI Apr 10 17:36:45.273
192.168.16.33:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.36G 15.3M 0 0 110 150b 46.2k 5 shardrealic1 SEC Apr 10 17:36:45.274
192.168.16.35:27017 *0 *0 *0 *0 *0 0 3 0 3.0 2.7% 0.3% 0 1.48G 13.3M 0 0 110 160b 49.0k 12 shardrealic1 PRI Apr 10 17:36:45.169
192.168.16.32:27017 *0 *0 *0 *0 *0 0 2 0 2.0 0 0 0 0 273M 16.0M 0 0 0 0 0 272b 16.2k 13 07M Apr 10 17:36:45.217
192.168.16.33:27017 no data received
192.168.16.34:27017 no data received
192.168.16.35:27017 no data received
192.168.16.32:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.48G 13.3M 0 0 110 339b 49.5k 12 shardrealic1 PRI Apr 10 17:36:45.255
192.168.16.33:27017 no data received
192.168.16.34:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 1.3% 0 1.46G 19.3M 0 0 110 158b 48.4k 5 configsvr01 PRI Apr 10 17:36:45.177
192.168.16.35:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.36G 15.3M 0 0 110 305b 45.3k 5 shardrealic1 SEC Apr 10 17:36:45.182
192.168.16.32:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.48G 13.3M 0 0 110 160b 49.0k 12 shardrealic1 PRI Apr 10 17:36:45.172
192.168.16.33:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.36G 15.3M 0 0 110 150b 46.2k 5 shardrealic1 SEC Apr 10 17:36:45.110
192.168.16.34:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 1.3% 0 1.46G 19.3M 0 0 110 152b 46.3k 5 shardrealic1 SEC Apr 10 17:36:45.187
192.168.16.35:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.48G 13.3M 0 0 110 160b 49.0k 12 shardrealic1 PRI Apr 10 17:36:45.183
192.168.16.32:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.0 2.7% 1.3% 0 0 273M 16.0M 0 0 0 0 0 282b 17.3k 13 07M Apr 10 17:36:45.213
192.168.16.33:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.36G 15.3M 0 0 110 155b 47.2k 5 configsvr01 PRI Apr 10 17:36:45.187
192.168.16.34:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.36G 15.3M 0 0 110 361b 51.5k 5 shardrealic1 SEC Apr 10 17:36:45.180
192.168.16.35:27017 no data received
192.168.16.32:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.0 2.7% 1.3% 0 0 273M 16.0M 0 0 0 0 0 282b 17.3k 13 07M Apr 10 17:36:45.246
192.168.16.33:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.0 2.7% 1.3% 0 1.46G 19.3M 0 0 110 160b 49.0k 5 configsvr01 PRI Apr 10 17:36:45.218
192.168.16.34:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.36G 15.3M 0 0 110 160b 49.0k 5 shardrealic1 SEC Apr 10 17:36:45.219
192.168.16.35:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.48G 13.3M 0 0 110 522b 162k 12 shardrealic1 PRI Apr 10 17:36:45.127
192.168.16.32:27017 *0 *0 *0 *0 *0 0 2 0 2.0 0 0 0 0 273M 16.0M 0 0 0 0 0 282b 16.6k 13 07M Apr 10 17:36:45.266
192.168.16.33:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.0 2.7% 1.3% 0 1.46G 19.3M 0 0 110 160b 49.0k 5 configsvr01 PRI Apr 10 17:36:45.174
192.168.16.34:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.36G 15.3M 0 0 110 160b 49.0k 5 shardrealic1 SEC Apr 10 17:36:45.217
192.168.16.35:27017 *0 *0 *0 *0 *0 0 2 0 2.0 2.7% 0.3% 0 1.48G 13.3M 0 0 110 578b 49.0k 12 shardrealic1 PRI Apr 10 17:36:45.125
^C2018-04-10T17:36:45.12+09:00 signal 'interrupt' received; forcefully terminating
[root@mongos test]#
```

Gambar 14. Status replikasi dan sharding termonitoring di server mongos



5 KESIMPULAN DAN SARAN

Pada penelitian ini, implementasi openstack terintegrasi docker telah berhasil dijalankan. Hasil yang didapat pada penelitian ini adalah sebuah system sharding database berbasis noSQL menggunakan mongoDB berhasil diimplementasikan pada server yang diset dengan *High Availability*, fungsi utama server adalah untuk fail over dan clustering.

6 UCAPAN TERIMA KASIH

Penulis dan tim peneliti mengucapkan terima kasih kepada Fakultas Teknik Universitas Muhammadiyah Malang (FT UMM), atas dukungan pendanaan melalui skema penelitian Pusat Kajian dan Rekayasa (Puskareka) FT UMM tahun 2020-2021. Juga kepada Laboratorium Jaringan Komputer Program Studi Teknik Elektro dan Laboratorium Jaringan Komputer Program Studi D3 Teknologi Elektronika UMM atas dukungan fasilitas sehingga terlaksana kegiatan penelitian ini.

DAFTAR PUSTAKA:

- [1] Calzolari, Federico; Arezzini, Silvia; Ciampa, Alberto eds. *High Availability Using Virtualization*. IOP Science, 2010.
- [2] Lwin dan Thein. *High Availability Cluster System for Local Disaster Recovery with Markov Modeling Approach*. International Journal of Computer Science Issues, 2009.
- [3] Patil, N.V., et al. (2014). *Cost Effective Failover Clustering*. International Journal of Research in Engineering and Technology, 2014.
- [4] Michael Meadway, *storing and retrieving objects on a computer network in a distributed database*, 2014.
- [5] Rasian, R dan Mursanto, P. *Perbandingan Kinerja Pendekatan Virtualisasi*. Jurnal Sistem Informasi. Magister Teknik Informatika Univeristas Indonesia, 2009.
- [6] Jones, M. Tim. *High Availability with the Distributed Replicated Block Device*. DeveloperWorks, 2014.
- [7] Eelco Plugge, Peter Membrey dan Tim Hawkins, *The Definitive Guide to MongoDB*. Appress, 2010.
- [8] Ameya Nayak, Anil Poriya, and Dikshay Poojary . *Type of NOSQL Databases and its Comparison with Relational Databases* Dept. of Computer Engineering Thakur College of Engineering and Technology University of Mumbai, March 2013.
- [9] J. R. Lourenço, B. Cabral, P. Carreiro, M. Vieira, and J. Bernardino, "Choosing the right NoSQL database for the job: a quality attribute evaluation," *J. Big Data*, vol. 2, no. 1, p. 18, Aug. 2015.
- [10] Divya Chauhan and K. L. Bansal 'Using the Advantages of NOSQL: A Case Study on MongoDB' Himachal Pradesh University Summerhill, Shimla, India."
- [11] Iqbal A.U, Nurhadi, Lailis S., Khaeruddin; 'Implementasi System Aplikasi Docker Terintegrasi Openstack' Vol. 4 No. 1 (2021): JIRE April 2021.
- [12] Yuliadi, Nurul M. S., Herfandi; 'Rekayasa Aplikasi Center Rumah Kost Berbasis Web Di Kabupaten Sumbawa' Vol. 4 No. 2 (2021): MISI Juni 2021.
- [13] Imtihan, K., & Fahmi, H. (2020). *Analisis Dan Perancangan Sistem Informasi Daerah Rawan Kecelakaan Dengan Menggunakan Geographic Information Systems (GIS)*. Jurnal Manajemen Informatika dan Sistem Informasi, 3(1), 16-23.
- [14] Imtihan, K., & Basri, M. H. (2019). *Sistem Informasi Pembuatan Manifest Muatan Kapal Berbasis Dekstop Dan Android*. Jurnal Manajemen Informatika dan Sistem Informasi, 2(2), 69-76.
- [15] Ashari, M., Zaen, M. T. A., Putri, J. A., Imtihan, K., & Bagye, W. (2022). *Prototype Sterilisasi Virus Barang Belanjaan Online Berbasis Arduino*. Jurnal Media Informatika Budidarma, 6(1), 120-127.
- [16] Imtihan, K., Bagye, W., Asri, Z. M. T., Fadli, S., & Ashari, M. (2021, February). *Image*

Jurnal Manajemen Informatika & Sistem Informasi (MISI)

Volume 5, Nomor 1, Januari 2022.

ISSN 2614-1701 (media cetak)

ISSN 2614-3739 (media online)

DOI : [10.36595/misi.v5i1](https://doi.org/10.36595/misi.v5i1)

<http://e-journal.stmiklombok.ac.id/index.php/misi>



capture device based on Internet of Thing (IoT) technology. In IOP Conference Series: Materials Science and Engineering (Vol. 1088, No. 1, p. 012065). IOP Publishing

[17] Imtihan, K., Mutawalli, L., & Bagye, W. (2020). Pemilihan Model Scrum Dalam Pengembangan Sistem Monitoring Dengan Menggunakan Metode Agile Untuk Evaluasi Clinical Pathway. *Bianglala Informatika*, 8(1), 63-69.

Amrul Faruq_HIGH AVAILABILITY DYNAMIC SHARDING

ORIGINALITY REPORT

18%

SIMILARITY INDEX

15%

INTERNET SOURCES

5%

PUBLICATIONS

2%

STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

3%

★ **fdokumen.id**

Internet Source

Exclude quotes Off

Exclude matches < 1%

Exclude bibliography On