# Evaluasi Penggunaan *Prometheus* dan *Grafana* Untuk *Monitoring Database Mongodb*

**Ramadoni[1], Mahmud Zunus Amirudin[2], Rifki Fahmi[3], Ema Utami[4], Muhammad Syukri Mustafa[5]**

[1,2,3,4,5] Magister Teknik Informatika, Universitas Amikom Yogyakarta
[1]ramadoni.ashudi@gmail.com, [2] zunuz4@gmail.com, [3] rifkifahmi@gmail.com, [4] emma@nrar.net, [5] moh.syukri@gmail.com

**Abstrak**

Sebuah basis data dituntut untuk selalu dalam keadaan optimal, karena hal tersebut merupakan faktor penting dalam sebuah Sistem Informasi Manajemen. Untuk menjaga sebuah sistem basis data agar selalu optimal tentu saja dibutuhkan mekanisme monitoring yang dapat memberikan informasi kepada pihak yang berkepentingan sehingga setiap kejadian dapat terpantau, sehingga apabila diperlukan sebuah tindakan preventif maupun korektif dapat segera dilakukan. Pemilihan perangkat monitoring perlu mempertimbangkan beberapa faktor seperti *fleksibilitas*, kemampuan untuk diintegrasikan dengan sistem yang lain. *Prometheus* dan *Grafana* adalah kombinasi yang bagus untuk melakukan monitoring pada sistem basis data, tidak hanya terbatas pada sistem sebuah basis data *MongoDB*, kemampuan *Prometheus* untuk menyimpan basis data dalam bentuk *time series* memungkinkan *Prometheus* untuk menyimpan data dalam jumlah besar. Sedangkan tampilan visualisasi *Grafana* yang bisa dikustomisasi membuka peluang untuk kita membuat visualisasi dan mendapatkan visibilitas sesuai dengan kebutuhan. Penggunaan *Prometheus* dan *Grafana* yang bersifat *opensource* juga menjadi pilihan untuk mengurangi biaya karena tidak membutuhkan lisensi untuk penggunaannya bahkan di lingkungan komersial sekalipun.

**Kata Kunci:** ***observability, monitoring, prometheus, grafana, mongodb***

## 1. Pendahuluan

Sebuah organisasi dituntut untuk bisa memilih solusi yang tepat guna berdasarkan kebutuhan dan juga kemampuan serta fitur atas setiap solusi yang akan digunakan karena akan berdampak pada produk yang akan dihasilkan, termasuk juga memilih solusi DBMS apa yang akan digunakan dalam pengembangan aplikasinya.

DBMS sendiri diartikan sebagai perangkat lunak yang berfungsi untuk mengelola basis data sehingga pengguna dimudahkan dalam hal pengoperasian, pengolahan serta memudahkan dalam memperoleh bentuk data sesuai dengan yang diinginkan, (Malik et al., 2020).

Saat ini terdapat dua tipe basis data yang biasa digunakan yaitu basis data relasional dan basis data yang bersifat non relasional, (Palanisamy & Suvithavani, 2020). Pada era dimana hampir semua organisasi berlomba-lomba untuk mengadopsi teknologi pengolahan data besar atau yang biasa disebut sebagai *Big Data*, banyak data yang bentuknya tidak terstruktur dengan ukuran data yang sangat besar yang berasal dari berbagai macam sumber, (Patel, 2019). Oleh karena itu dapat dipastikan setiap organisasi akan mempunyai masalah dengan kapasitas, proses transmisi data, proses pengolahan data, validasi data, serta keamanan dan privasi, (Kaushik, 2020). Kebutuhan dalam penyimpanan data yang besar *(Big Data*), skalabilitas, dan kinerja adalah alasan utama sebuah organisasi atau perusahaan untuk beralih menggunakan sistem basis data non relasional (NoSQL), (Ali et al., 2019). Terdapat beberapa tipe basis data NoSQL yaitu: 1) *Key-value stores;* 2) *Column family databases;* (3) *Document stores;* (4) *Graph databases*, (Meier & Kaufmann, 2019) dimana masing-masing mempunyai karakteristik yang berbeda.

Untuk mendapatkan kinerja basis data yang selalu optimal maka diperlukan adanya sebuah mekanisme monitoring untuk memantau setiap saat kondisi dari sebuah sistem basis data agar tindakan preventif atau korektif dapat segera dilakukan jika terjadi masalah atau keadaan yang abnormal, (Candido et al., 2019). Monitoring dilakukan melalui seperangkat alat dan sistem terintegrasi yang akan menyediakan data dan informasi dengan bantuan visualisasi, (Lokawati & Widyani, 2019).

Tujuan penelitian ini adalah memberikan analisa penggunaan *tools* Prometheus dan Grafana diintegrasikan dengan *tools* pendukung lainnya untuk memonitor sistem basis data MongoDB, serta memberikan gambaran bagaimana membuat sistem monitoring yang mampu memberikan kemampuan *observability* yang bagus, mudah dikonfigurasikan,

mempunyai *dashboard* yang dapat dengan mudah dikustomisasi, serta proses pengadaannya tidak mempunyai dampak finansial yang tinggi atau bahkan gratis.

## 2. Landasan Teori

### 2.1 Monitoring

Salah satu manfaat penggunaan sebuah DBMS adalah untuk melakukan perlindungan dan pengamanan data, (Susanto & Meiryani, 2019). Untuk menunjang kebutuhan tersebut selain dibutuhkan desain basis data yang bagus dan berkualitas juga dibutuhkan desain infrastruktur penunjang yang juga bagus dan berkualitas. Aspek pendukung dari sebuah infrastruktur database yang bagus diantaranya terdiri dari aspek keamanan, tingkat ketersediaan dan aspek *observability*. Aspek keamanan seperti penggunaan enkripsi dan pemberian akses kontrol yang memenuhi aspek *Principle of Least Privilege*. Pemberian hak akses berlebih akan meningkatkan resiko kerusakan sistem karena faktor penyalahgunaan yang sengaja maupun tidak disengaja, sedangkan pemberian akses yang kurang juga akan menghambat seseorang melakukan tugasnya, (Sanders & Yue, 2019). Aspek tingkat ketersediaan meliputi penggunaan sistem infrastruktur yang fault tolerant dan *high available*, sedangkan aspek *observability* meliputi kemampuan mendapatkan informasi yang tepat dan cepat jika terjadi sesuatu pada sistem basis data.

Monitoring merupakan bagian penting dari aspek *observability* yang memungkinkan sebuah sistem basis data dapat dimonitor menggunakan tools yang bisa memberikan kita visibilitas terhadap kondisi basis data sehingga kita bisa melakukan aksi perbaikan dengan cepat jika terdapat masalah dalam sistem basis data tersebut. Aksi perbaikan yang cepat artinya menurunkan waktu *downtime* dan menurunkan waktu *downtime* berarti mengurangi potensi kerugian.

### 2.2 MongoDB

*MongoDB* adalah sistem manajemen basis data (DBMS) berbasis sumber terbuka yang menggunakan model basis data berorientasi dokumen yang mendukung berbagai jenis data (Jose & Abraham, 2019). MongoDB tidak menggunakan tabel dan baris seperti dalam basis data relasional, arsitektur MongoDB dibentuk dari *Collection* dan *Document*, (Chaudhary et al., 2018). MongoDB menyimpan data dalam dokumen mirip JSON yang strukturnya dapat bervariasi.

MongoDB mempunyai beberapa skema lisensi yaitu *MongoDB, Inc.'s ServerSide Public License* untuk semua versi yang rilis setelah tanggal 16 Oktober 2018 termasuk tambalan perbaikan untuk versi-versi sebelumnya. *Free Software Foundation's GNU AGPL v3.0* untuk semua versi yang rilis sebelum tanggal 16 Oktober 2018 dan tersedia juga lisensi komersial dari MongoDB, Inc.

Berdasarkan data yang ada pada situs https://db-engines.com kita bisa melihat bahwa pada bulan Oktober 2020 MongoDB menduduki peringkat pertama untuk kategori *DB-Engines of Document Stores* berdasarkan popularitasnya, sedangkan hasil survey dari *Stackoverflow Developers Survey 2020* juga menempatkan MongoDB pada peringkat pertama DBMS non relasional yang paling banyak digunakan oleh pengembang profesional.

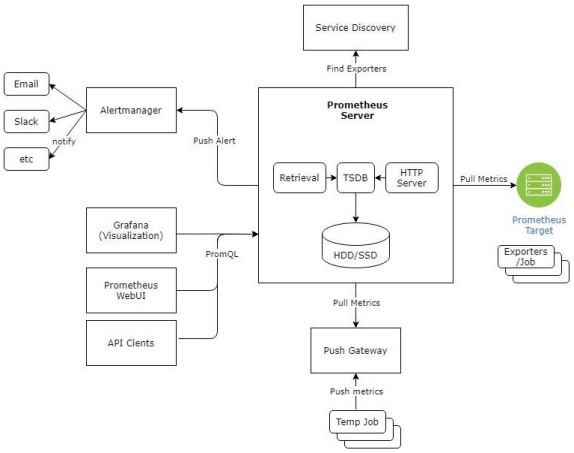
MongoDB mempunyai beberapa fitur unggulan diantaranya adalah performa yang tinggi dengan adanya *embedded data models* yang akan mengurangi aktivitas masukan dan keluaran pada sistem basis data ditambah dengan fitur *indexing* yang akan membuat kueri menjadi lebih cepat. Fitur ketersedian yang tinggi (*High Available*) yaitu kemampuan untuk melakukan replikasi serta menyediakan fitur *automatic failover* dan juga redundansi data. Fitur yang juga merupakan bagian dari fungsi intinya adalah kemampuan untuk melakukan skala secara horizontal, (Chauhan, 2019). Dengan adanya fitur-fitur tersebut banyak organisasi yang kemudian berpindah dan memilih menggunakan MongoDB untuk solusi basis data NoSQL mereka, beberapa penelitian juga telah melakukan kajian untuk membuktikan kemampuan MongoDB dibanding dengan *NoSQL* yang lain. MongoDB memiliki waktu respon kueri rata-rata terendah untuk proses baca, perbarui dan hapus dibandingkan dengan database *NoSQL* yang lain seperti *ArangoDB* dan *CouchDB*, (Gunawan et al., 2019) dan *MongoDB* menunjukkan kinerja yang relatif baik saat menangani data yang besar, (Martins et al., 2019).

### 2.3 *Prometheus*

*Prometheus* adalah perangkat lunak berbasis sumber terbuka yang berguna untuk melakukan monitoring dan alerting. Salah satu keunggulan Prometheus dibanding perangkat lunak monitoring yang lain adalah memiliki banyak metrik (pengukuran) yang dibutuhkan untuk memantau system. *Prometheus* dapat dikustomisasi dengan mudah sehingga sangat membantu saat diintegrasikan dengan perangkat lunak lainnya, (Sukhija & Bautista, 2019) seperti dikombinasikan dengan Grafana untuk kebutuhan visualisasinya.

*Prometheus* memiliki beberapa komponen utama yaitu *Prometheus* server yang berfungsi untuk mengelola dan menyimpan data. *Exporter* yang berfungsi untuk melakukan pengiriman data dari server target ke *Prometheus Server*, dan *Alertmanager* yang berfungsi untuk mengelola notifikasi. *Prometheus* server akan mengambil data target pada interval waktu yang telah ditentukan pada konfigurasi dari target tertentu dan menyimpannya dalam bentuk basis data *time series*.

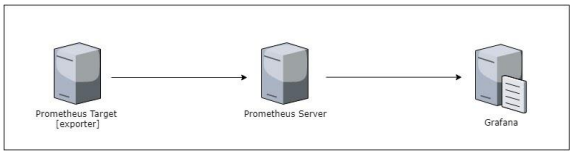
Kode sumber dari perangkat lunak *Prometheus* bisa diakses dan diunduh di halaman Githubnya.




Gambar 1. Gambar Arsitektur *Prometheus*

### 2.4 *Grafana*

*Grafana* adalah perangkat lunak berbasis sumber terbuka yang berfungsi untuk memvisualisasikan data monitoring dalam bentuk grafik dan *chart*. Grafana mempunyai dukungan untuk menggunakan berbagai macam tipe *datasource* data seperti *Graphite*, *Prometheus*, *Elasticsearch*, *OpenTSDB* and *InfluxDB*. Seperti halnya *Prometheus*, *Grafana* juga merupakan salah satu proyek berbasis sumber terbuka yang sangat populer saat ini. Kode sumber Grafana bisa diakses di halaman Github.




Gambar 2. Alur kerja *Prometheus* dan *Grafana*

### 2.5 *Exporter*

*Prometheus Exporter* memungkinkan kita untuk mengekspor statistik dari sistem non-Prometheus dan mengimpornya ke Prometheus. *Prometheus Exporter* dapat diartikan sebagai pengeruk konten yang mengubah metrik dari satu format ke format lainnya. Terdapat banyak pustaka yang dapat membantu mengekspor metrik yang ada dari sistem pihak ketiga menjadi metrik Prometheus. Beberapa dari pengekspor ini dikelola sebagai bagian dari organisasi resmi Prometheus dan ada juga yang berasal dan dikelola oleh pihak eksternal. Pada tulisan ini penulis menganalisa penggunaan dua jenis pengekspor yaitu *mongodb_exporter* yang berfungsi mengekspor metrik yang berhubungan dengan sistem basis data MongoDB dan *node_exporter* yang berfungsi sebagai pengekspor metrik yang berkaitan dengan infrastruktur server yang digunakan oleh sistem basis data MongoDB.

### 2.6 *Alert Manager*

Sebuah perangkat lunak sumber terbuka yang digunakan untuk mengelola notifikasi dari Prometheus yang dapat diintegrasikan dengan Email, Slack, *PagerDutty* atau *OpsGenie*. Alertmanager melakukan pengiriman notifikasi kepada penerima yang telah ditentukan jika sebuah kondisi terpenuhi. Notifikasi berguna agar kita mendapatkan informasi secepat dan seakurat mungkin saat sesuatu terjadi pada sistem misalkan kondisi ketika replikasi pada sebuah basis data *MongoDB* terlambat lima menit dan sudah berjalan lebih dari 15 menit maka *alertmanager* akan mengirimkan notifikasi ke database administrator agar segera melakukan pengecekan dan perbaikan yang diperlukan.

## 3. Metodologi

Metode yang digunakan pada penelitian ini adalah metode studi literatur untuk mendapatkan data pendukung dan memperkuat teori-teori, metode kualitatif dengan melakukan pengujian terhadap proses pengolahan data yang dikumpulkan melalui instrumen pendukung berdasarkan variabel yang telah ditentukan, metode observasi dilakukan dengan melakukan pengujian mandiri untuk kebutuhan validasi.

### 3.1 Metode Pengumpulan Data

Metode pengumpulan data bertujuan mengumpulkan berbagai data pendukung yang dibutuhkan yang akan digunakan dalam proses pengujian, data yang dibutuhkan adalah dataset yang terkait dengan sistem basis data MongoDB, dataset tersebut akan digunakan untuk untuk proses pengujian.

### 3.2 Metode Pengolahan Data

Pada penelitian ini penulis menggunakan dataset yang tersedia secara publik pada situs *U.S. patent data* dalam bentuk berkas .txt yang kemudian diimpor ke dalam sistem basis data MongoDB. Pengolahan data juga dilakukan pada data metrik yang akan ditampilkan dalam bentuk Grafana *dashboard* menggunakan *Prometheus Query Language (PromQL)*.

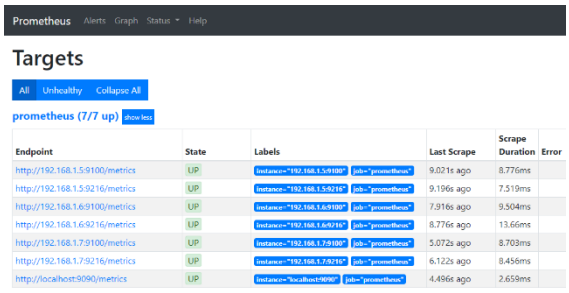
### 3.3 Metode Pengujian

Tahapan awal proses pengujian adalah dengan mempersiapkan integrasi antara MongoDB, Prometheus, Exporter dan Grafana. Proses diawali dengan instalasi dan konfigurasi *node_exporter* dan *mongodb_exporter* pada setiap server MongoDB kemudian ubah konfigurasi prometheus agar bisa mengambil metrik dari *exporter*.

```
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
    - targets: ['localhost:9090',
    '192.168.1.5:9216',
    '192.168.1.6:9216',
    '192.168.1.7:9216',
    '192.168.1.5:9100',
    '192.168.1.6:9100',
    '192.168.1.7:9100']
```

Gambar 3. Konfigurasi Prometheus

Sampai tahap ini *Prometheus* Web UI sudah bisa menampilkan metrik yang bersumber dari *node_exporter* dan *mongodb_exporter*.



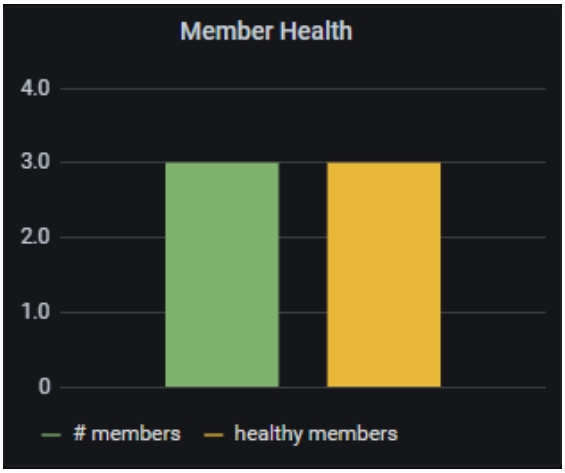

Gambar 4. Tampilan Prometheus Web UI

Langkah selanjutnya adalah mengintegrasikan *Prometheus* dengan *Grafana* agar mendapatkan tampilan visualisasi dari metrik yang ada, pada *dashboard* Grafana kita perlu membuat sebuah *datastore* yang yang diarahkan ke alamat *IP address* dari server Prometheus, kemudian buat *dashboard* sesuai dengan keinginan atau bisa juga menggunakan *dashboard* yang sudah disediakan di situs resmi Grafana. Bahasa kueri yang digunakan untuk menampilkan dashboard tergantung dari tipe data source yang digunakan, dalam penelitian ini menggunakan *Prometheus Query Language (PromQL)*. Sebagai contoh jika kita ingin menampilkan *dashboard* status member dari *replicasets* MongoDB kita bisa menggunakan kueri PromQL seperti berikut.

```
mongodb_mongod_replset_number_of
_members{instance=~"$env"}
```

Gambar 5. Kueri PromQL untuk menampilkan status member replicasets MongoDB

Hasil *query* di atas akan menampilkan tampilan *dashboard* informasi berapa banyak member replika dari sebuah *replicasets* pada sistem basis data MongodDB. Kueri di atas juga menampilkan berapa banyak member replika yang berstatus *health*. Untuk membandingkan apakah data tersebut valid atau tidak adalah dengan menjalankan perintah *rs.status()* pada *command line* di server MongoDB.




Gambar 6. Status member replika

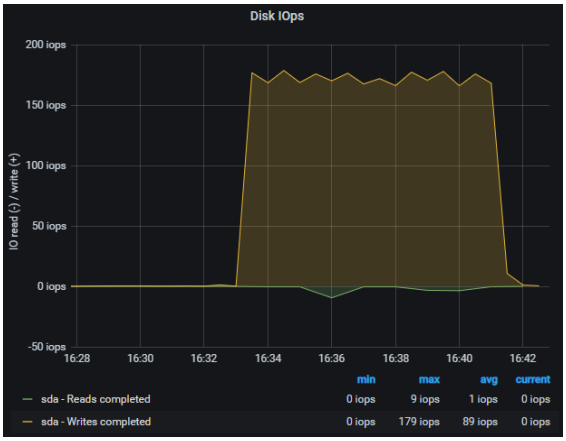
Tahapan selanjutnya adalah tahapan proses pengujian terhadap sistem basis data MongoDB. Langkah pertama adalah membuat database pada sistem basis data MongoDB dilanjutkan dengan membuat *collection*, langkah selanjutnya adalah mengimpor dataset yang sudah diunduh ke dalam database MongoDB. Proses mengimpor dataset kedalam sistem basis data MongoDB dilakukan menggunakan perintah *mongoimport*. Dalam pengujian ini penulis menggunakan tiga buah mesin virtual yang dibuat menggunakan perangkat lunak Virtualbox yang sudah terpasang mongoDB server dengan sepesifikasi yang sama untuk masing-masing mesinnya, detail konfigurasi dan spesifikasi dari masing-masing server MongoDB adalah sebagai berikut:

Tabel 1. Spesifikasi Server MongoDB

| Host Name | IP Address | CPU | RAM | Disk | Mongo DB | OS |
|---|---|---|---|---|---|---|
| mongo db01 | 192.168.1.5 | 1 CPU i5-10210U | 1 GB | 20 GB | 4.4.1 | CentOS 7.7 |
| mongo db02 | 192.168.1.6 | 1 CPU i5-10210U | 1 GB | 20 GB | 4.4.1 | CentOS 7.7 |
| mongo db03 | 192.168.1.7 | 1 CPU i5-10210U | 1 GB | 20 GB | v4.4.1 | CentOS 7.7 |

Dengan jumlah data sebesar 251MB dan jumlah baris data sebanyak 16,522,438 waktu yang dibutuhkan untuk mengimpor data tersebut adalah sekitar 7 menit. Aktivitas tersebut juga dapat dimonitor pada *dashboard* monitoring, dimana terjadi peningkatan yang signifikan pada penggunaan CPU, memory, *network*, *IOPS* dan lain-lain. Pada gambar dibawah ini terlihat adanya peningkatan *Disk IOPS* selama sekitar 7 menit saat dilakukan proses impor data. Data ini sesuai dengan

lamanya proses impor data yang dilakukan. Data penggunaan *Disk IOPS* ini juga bisa dibandingkan dengan data hasil keluaran dari *command line* dengan perintah *iostat*.




Gambar 7. Status disk I/O per detik

Dari beberapa hasil pengujian diatas terlihat bahwa Grafana dapat menampilkan grafik yang terkait dengan status basis data MongoDB dan juga status dari server MongoDB.

## 4. Hasil dan Pembahasan

Kompleksitas yang semakin meningkat dari sistem komputasi masa kini dan masa mendatang membutuhkan perangkat monitoring yang mampu memberikan informasi terpadu dan terpusat, bukan hanya mampu memonitor perangkat keras dan aspek kinerja saja tetapi juga untuk hal-hal yang lainnya seperti bagaimana sebuah perangkat monitoring dapat menyediakan mekanisme pemantauan proaktif, mempunyai fitur skalabilitas yang tinggi, kemampuan untuk memberikan perencanaan kapasitas, mampu memberikan analisa tren, pengolahan alert dan dukungan dashboard yang praktis, (Birje & Bulla, 2020).

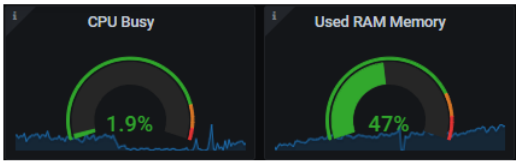
Distribusi MongoDB menyertakan sejumlah utilitas yang berfungsi untuk memonitor kinerja dan aktivitas sistem basis datanya dan berguna untuk melakukan diagnosa pada sebuah masalah serta berfungsi untuk mengukur bahwa sistem basis data MongoDB dalam keadaan normal atau tidak.. Penggunaan utilitas ini dilakukan dengan cara manual menjalankan beberapa baris perintah pada command line misalkan perintah *mongostat* yang berfungsi untuk memonitor jumlah operasi-operasi basis data saat itu berdasarkan tipenya *(insert, query, update, delete,* dll.) dan perintah *mongotop* yang berfungsi untuk melacak dan memberikan laporan berapa banyak aktifitas baca tulis pada sebuah basis data MongoDB. Untuk mempermudah monitoring pada tulisan ini penulis melakukan analisa penggunaan perangkat monitoring berbasis sumber terbuka yang mampu melakukan monitoring secara otomatis dan dapat menampilkan dashboard yang bagus dan mudah dibaca sehingga proses umpan balik dan perbaikan menjadi lebih cepat karena tim operasional dapat dengan mudah melihat ketidaknormalan dan sesegera mungkin melakukan investigasi dan perbaikan, (Candido et al., 2019).

Dalam penelitian ini penulis telah berhasil melakukan analisa tentang penggunaan Prometheus dan Grafana sebagai *tools* monitoring sistem basis data MongoDB, sistem ini terbukti mudah dalam konfigurasinya serta mudah dikustomisasi sehingga mampu menampilkan *dashboard* monitoring yang sesuai dengan kebutuhan, dan yang tak kalah penting adalah penggunaan perangkat lunak *opensource* menjadikan pengadaan sistem ini tidak membutuhkan biaya yang tinggi bahkan gratis.

Dari hasil analisa penggunaan *Prometheus* dan *Grafana* untuk monitoring basis data mongoDB ini ada setidaknya enam komponen yang menurut penulis harus dimonitor, yaitu:

1. Utilisasi Server

Komponen utilisasi server yang utama adalah komponen CPU, RAM, media penyimpanan dan jaringan, secara lebih detail lagi bisa dimonitor misalkan berapa banyak penggunaan RAM, berapa besar *network i/o*, berapa besar utilisasi penggunaan pada media penyimpanan *(Storage)*, berapa banyak operasi baca dan tulis pada media penyimpanan dan lain-lain.




Gambar 8. Tampilan monitoring CPU dan memory server MongoDB

2. *Uptime*

Komponen *uptime* adalah salah satu komponen yang sangat perlu dimonitor karena menyangkut tentang tingkat ketersediaan infrastruktur yang digunakan untuk menjalankan sistem basis data MongoDB.

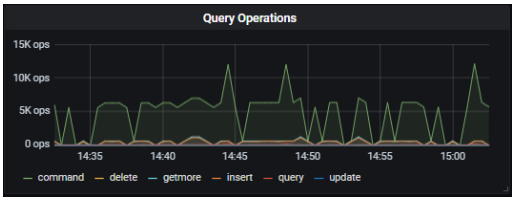
3. Koneksi

Pada komponen koneksi beberapa aspek yang perlu dimonitor adalah tentang berapa banyak koneksi pada sistem basis data MongoDB yang ada atau terjadi pada saat ini dan berapa banyak jumlah koneksi yang masih tersedia.


Gambar 9. Tampilan monitoring koneksi MongoDB.

4. *Query Operations*

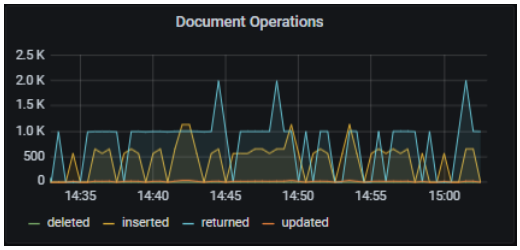
Aspek monitoring untuk operasi kueri dari sebuah sistem basis data MongoDb meliputi proses *insert, update, delete* dan lain-lain.




Gambar 10. Tampilan monitoring *Query* MongoDB

5. *Document Operations*

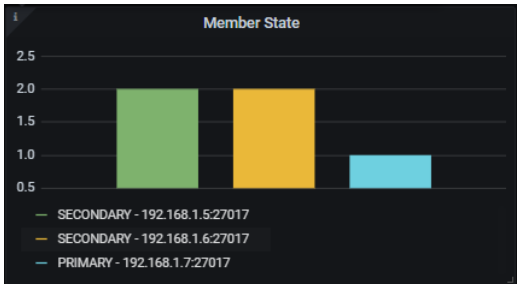
Karena MongoDB adalah sistem basis data NoSQL untuk kategori *Document Stores* sehingga penting untuk melakukan monitoring untuk aktivitas *document* yang meliputi berapa banyak document yang dihapus, diperbarui, dimasukkan dan lain-lain.




Gambar 11. Tampilan monitoring *document* MongoDB

6. Replikasi

Salah satu keunggulan sistem basis data MongoDB dibandingkan dengan sistem basis data *NoSQL* yang lain adalah kemampuan MongoDB untuk melakukan replikasi, untuk itu komponen replikasi ini menjadi sangat penting untuk dimonitor. Metrik-metrik pada komponen replikasi ini terdiri dari berapa banyak jumlah *replicasets*, berapa banyak member tiap *replicasets*, *node* mana yang menjadi *primary* dan *node* mana yang menjadi *secondary*, status dari setiap member apakah sehat atau tidak.




Gambar 12. Tampilan monitoring status anggota replika MongoDB

## 5. Kesimpulan

Berdasarkan keseluruhan proses analisis yang sudah dilakukan oleh penulis, dapat diambil kesimpulan bahwa monitoring basis data MongoDB akan sangat membantu administrator dalam memonitor kinerja basis data. Dengan monitoring sistem menggunakan Prometheus dan Grafana, maka pembacaan hasil monitoring menjadi lebih mudah dan juga akurat. Dengan adanya sistem ini, maka tidak diperlukan lagi monitoring MongoDB secara manual, sehingga diharapkan efektivitas kerja menjadi lebih baik. Penggunaan Prometheus dan Grafana adalah komposisi yang bagus untuk melakukan monitoring pada sistem basis data MongoDB, kemampuan Prometheus untuk menyimpan basis data dalam bentuk *time series* memungkinkan Prometheus untuk menyimpan data dalam jumlah besar, dan tampilan visualisasi Grafana yang bisa dikustomisasi membuka peluang untuk kita membuat visualisasi sesuai dengan kebutuhan. Sifat dari proyek Prometheus dan Grafana yang bersifat terbuka juga memungkinkan siapa saja untuk berkontribusi dalam proyek tersebut.

Pada penelitian ini penulis tidak membahas tentang integrasi sistem monitoring Prometheus dan Grafana dengan sistem notifikasi *(alert)* menggunakan *Alertmanager*. Sebagai bahan penelitian selanjutnya penulis menyarankan untuk melakukan penelitian pada integrasi dengan sistem notifikasi *(Alert)* serta penggunaan *Prometheus* dan Grafana tidak terbatas hanya untuk monitoring sistem basis data MongoDB saja tetapi juga untuk sistem basis data yang lainnya. Masih sedikitnya *exporter* dan *dashboard* Grafana yang tersedia juga membuka peluang untuk melakukan penelitian dalam hal tersebut.

## Daftar Pustaka:

Ali, W., Shafique, M. U., Majeed, M. A., & Raza, A. (2019). Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics. *Asian Journal of Research in Computer Science*, *4*(2), 1–10. https://doi.org/10.9734/ajrcos/2019/v4i230108

Birje, M. N., & Bulla, C. (2020). *Commercial and Open Source Cloud Monitoring Tools: A Review*. 480–490. https://doi.org/10.1007/978-3-030-24322-7_59

Candido, J., Aniche, M., & Van Deursen, A. (2019). Contemporary software monitoring: A systematic literature review. *ArXiv*, *1*, 1–25.

Chaudhary, A. S., Singh, K., Kalra, S., & Kaur, P. (2018). An empirical comparison of mongoDB and hive. *2018 4th International Conference on Computing Communication and Automation, ICCCA 2018*, 1–4. https://doi.org/10.1109/CCAA.2018.8777525

Chauhan, A. (2019). A Review on Various Aspects of MongoDb Databases, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 08, Issue 05

Gunawan, R., Rahmatulloh, A., & Darmawan, I. (2019). Performance evaluation of query response time in the document stored nosql database. *2019 16th International Conference on Quality in Research, QIR 2019 - International Symposium on Electrical and Computer Engineering*, 1–6. https://doi.org/10.1109/QIR.2019.8898035

Jose, B., & Abraham, S. (2019). Performance analysis of NoSQL and relational databases with MongoDB and MySQL. *Materials Today: Proceedings*, *24*, 2036–2043. https://doi.org/10.1016/j.matpr.2020.03.634

Kaushik, S. (2020). *Big Data Issues and Challenges*. *6*(1), 1–21. https://doi.org/10.4018/978-1-7998-3049-8.ch001

Lokawati, H., & Widyani, Y. (2019). Monitoring System of Multi-Tenant Software as a Service (SaaS). *Proceedings of 2019 International Conference on Data and Software Engineering, ICoDSE 2019*. https://doi.org/10.1109/ICoDSE48700.2019.9092741

Malik, A., Burney, A., & Ahmed, F. (2020). A Comparative Study of Unstructured Data with SQL and NO-SQL Database Management Systems. *Journal of Computer and Communications*, *08*(04), 59–71. https://doi.org/10.4236/jcc.2020.84005

Martins, P., Abbasi, M., & Sá, F. (2019). A Study over NoSQL Performance. In Á. Rocha, H. Adeli, L. P. Reis, & S. Costanzo (Eds.), *New Knowledge in Information Systems and Technologies* (pp. 603–611). Springer International Publishing.

Meier, A., & Kaufmann, M. (2019). *SQL & NoSQL*.

Palanisamy, S., & Suvithavani, P. (2020). A survey on RDBMS and NoSQL Databases MySQL vs MongoDB. *2020 International Conference on Computer Communication and Informatics, ICCCI 2020*. https://doi.org/10.1109/ICCCI48352.2020.9104047

Patel, J. (2019). An Effective and Scalable Data Modeling for Enterprise Big Data Platform. *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, 2691–2697. https://doi.org/10.1109/BigData47090.2019.9005614

Sanders, M. W., & Yue, C. (2019). Mining least privilege attribute based access control policies. *ACM International Conference Proceeding Series*, *December*, 404–416. https://doi.org/10.1145/3359789.3359805

Sukhija, N., & Bautista, E. (2019). Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus. *Proceedings - 2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and Smart City Innovation, SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019*, 257–262. https://doi.org/10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00087

Susanto, A., & Meiryani. (2019). Database management system. *International Journal of Scientific and Technology Research*, *8*(6), 309–312. https://doi.org/10.5120/179-310