PERBANDINGAN INFLUXDB DAN PROMETHEUS UNTUK SISTEM *NETWORK MONITORING*

Rizky Saputra

Laboratorium Telematika, Sekolah Teknik Elektro dan Informatika (STEI),
Institut Teknologi Bandung, e-mail: rizkys200697@gmail.com

***Abstraksi— Paper* ini berisi tentang perbandingan *Time-Series Database* antara InfluxDB dan Prometheus jika digunakan untuk *Network Monitoring*. Pemaparan perbandingan dilakukan dalam beberapa parameter yang didapatkan dari berbagai referensi yang ada. Baik InfluxDB maupun Prometheus mempunyai kelebihan dan kekurangannya masing-masing pada situasi atau kondisi tertentu.**

Kata kunci- time-series, pull, push

1. PENDAHULUAN

Kebutuhan manajerial data semakin hari semakin meningkat. Sebagai contoh, penggunaan konsep IoT pada sistem sensor dan *monitoring data* secara berkala di Era Digital seperti sekarang ini, menuntut aliran data yang masif pada interval waktu tertentu. Bentuk aliran data yang terkoleksi dalam interval waktu tertentu inilah yang disebut dengan t*ime-series data.* Hal mendasar yang membedakan antara *time-series data* dengan *regular data* adalah bahwa *time-series data* merupakan data yang kerap diperbincangkan menggunakan waktu atau dikaitkan dengan waktu. Sebagai contoh *time-series data* adalah pegukuran temperatur pada *weather station* yang biasanya diukur per menit.

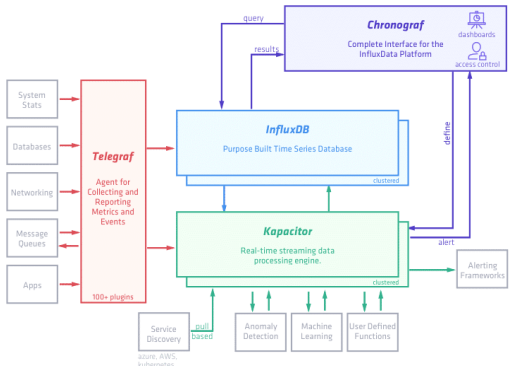
Kebutuhan akan *time-series data* ini yang mendasari peningkatan penggunaan *time-series database. Time-series database* merupakan tipe *database* yang dirancang khusus untuk menangani *time-series data* secara efisien. Dengan menggunakan, *Time-series database* memungkinkan penggunanya untuk membuat, memperbarui, menghitung, menghancurkan, dan mengatur berbagai *time-series data* dengan cara yang lebih efisien [1]. Cukup banyak *time-series database* yang digunakan di dunia yang berbasis *open source*, contohnya InfluxDB dan Prometheus. *Paper* ini akan mengulas perbandingan antara InfluxDB dan Prometheus secara umum. Parameter-parameter yang dibandingkan pun bersifat umum.

2. TIME-SERIES DATABASE

2.1 InfluxDB

InfluxDB merupakan sebuah *time-series database open source* yang dikembangkan oleh InfluxData. InfluxDB biasa digunakan untuk menyetor data-data time-stamped secara masif, seperti *DevOps monitoring, log data, application metrics, IoT sensor data,* dan *real-time analytics.* InfluxDB berbasis *Push system*, yang berarti aplikasi perlu mendorong data kepada sistem *monitoring*. InfuxDB memiliki model data yang terdapat *key-value* sebagai label, yang disebut *tags* dan *fields* sebagai *second level of labels*

data. InfluxDB *support timestamps data* sampai ke resolusi *nanosecond*, serta InfluxDB *support* tipe data *float64, int64, bool,* dan *string.*



Gambar 2.1.1 Sistem kerja InfluxDB

InfluxDB dibuat untuk menangani penulisan data dan *query* muatan yang masif atau tinggi. InfluxDB menggunakan HTTP *Representational State Transfer* (REST) API untuk proses *query* data dan menggunakan bahasa *SQL-like query language* atau biasa dikenal dengan InfluxQL. InfluxDB menggunakan sebuah *Time Structured Merge* (TSM) storage engine agar proses kompresi data lebih efektif [2].

2.1.1 Model Data InfluxDB

Pada InfluxDB terdapat beberapa strukutur data berbasis JSON, contohnya adalah sebagai berikut.

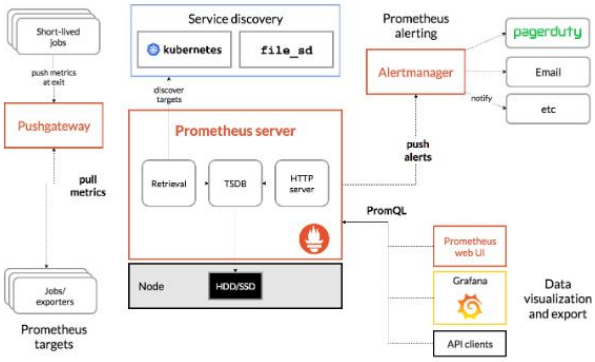
```
json_data = [
        {
                "measurement": Humidity
                "time": timestamp,
                "tags": {
                        "Station":
station_name
                },
                "fields": {
                        "value": value
yang akan dimasukkan kedalam database
                }
        }
]
```

Pada bagian *"measurement"* diisi dengan nama pengukuran yang akan dimasukkan ke dalam *database* InfluxDB. Dalam hal ini misalkan nama *measurement*-nya *Humidity*, nama *measurement* disarankan diisi dengan sebuah nama yang identik dengan hal yang diukur. Kemudian bagian berikutnya ada istilah "time", bagian ini merupakan penanda kedatangan data dengan keterangan waktu. Selanjutnya ada "*tags"* dan "*field"*. Pada "*tags"* dan "*field"* masing-masing terdapat komponen *key* dan *value*, di mana pada *key* akan berisikan nama-nama penanda baik "*tags*" maupun "*field*", sedangkan *value* merupakan nilai-nilai yang diisikan pada masing-masing *key*. Perbedaan antar "*tags"* dan "*field"* adalah, pada "*fields*" *value* akan lebih variatif nilainya dibandingkan dengan *value* pada "*tags*", misalkan jika dibandingkan, pada "*fields"* dengan *key* "temperatur" dan *value*-nya akan diisikan nilai dari temperatur ruang, sedangkan jika pada "*tags*" dengan key "*Battery Station* (BS)" dan *value*-nya akan diisikan dengan nomor ruang *Battery Station* (1, 2, dst), tentu *value* "*fields*" akan lebih variatif dibandingkan dengan "*tags*". Sehingga hal ini mengakibatkan proses *query* pada "*tags*" lebih cepat dibandingkan pada "*fields*".

```
> use PRESENTASI_AKHIR
Using database PRESENTASI_AKHIR
> precision rfc3339
> select * from Humidity
name: Humidity
time                  Station value
----                  ------- -----
2018-08-05T05:56:38Z BS      71.
2018-08-05T05:56:39Z BS      71.
2018-08-05T05:56:40Z BS      71.
2018-08-05T05:56:42Z BS      71.
2018-08-05T05:56:43Z BS      71.
2018-08-05T05:56:44Z BS      71.
2018-08-05T05:56:46Z BS      71.
2018-08-05T05:56:47Z BS      71.
2018-08-05T05:56:48Z BS      71.
2018-08-05T05:56:50Z BS      71.
2018-08-05T05:56:51Z BS      71.
2018-08-05T05:56:53Z BS      71.
```

Gambar 2.1.1.1 Contoh tampilan penyajian data pada InfluxDB

2.2 Prometheus



Gambar 2.2.1 Sistem kerja Prometheus

Prometheus adalah sebuah *time-series database open source* yang biasa digunakan untuk memonitoring sistem dan digunakan sebagai *alerting toolkit* atau alat peringatan pada sebuah sistem. Prometheus telah digunakan sejak tahun 2012 dan sudah cukup banyak perusahaan atau organisasi yang mengadopsi Prometheus untuk sistemnya. Prometheus sekarang merupakan *open source project* yang sudah berdiri sendiri. Promotheus berbasis *Pull System*, dimana *server* Promotheus yang akan "meminta" data dari aplikasi yang sedang berjalan secara berkala. Prometheus mempunyai model data berupa *key-value* sebagai label, yang disebut *tags*. Dan Prometheus *support timestamps* sampai ke resolusi *milisecond* serta hanya *support* tipe *data float64*.

2.2.1 Model Data Prometheus

Berbeda dengan InfluxDB yang menggunakan bahasa *SQL-like query language* sehingga mempunyai tampilan dan konfigurasi yang mirip dengan *database* berbasis SQL *(Relational Database).* Pada Prometheus setiap *time-series data* teridentifikasi berbeda-beda *(uniquely identified)* dengan nama *metric*-nya dan juga dengan pasangan *key-value*-nya yang biasa disebut dengan *labels*. Data *metric* umumnya digunakan untuk menyatakan performansi *time-series data*. *Metric* juga bisa berupa diskrit atau kontinyu.

Nama *metric* pada Prometheus akan menentukan fitur umum sistem yang terukur. Nama *metric* mungkin akan berisi huruf-huruf dan digit ASCII, beserta *underscore* dan titik dua. Biasanya akan berbentuk seperti `[a-zA-Z_:][a-zA-Z0-9_:]*`

Penggunaan *Labels* memungkinkan model data Prometheus yang memiliki nama *metric* yang sama akan teridentifikasi dengan dimensi yang sama. Bahasa *query* Prometheus memungkinkan penyaringan dan agregasi data berdasarkan dimensi-dimensi ini. Mengubah nilai *label* apapun, termasuk menambahkan atau menghapus *label,* akan membuat *time-series* baru.

Nama label dapat berisi huruf ASCII, angka, serta *underscores*. Pemberian nama *Label* harus cocok dengan aturan `[a-zA-Z_][a-zA-Z0-9_]*`. Nama label yang diawali dengan "__" (*underscores)* dicadangkan untuk penggunaan internal.

Notasi penyusunan model data Prometheus adalah sebagai berikut.

```
<metric name>{<label name>=<label v
alue>, ...}
```

Sebagai contoh penulisan adalah sebagai berikut.

```
api_http_requests_total{method="POST
", handler="/messages"}
```

Nama *metric* di atas adalah `api_http_requests_total` dengan *label* `method="POST"` serta `handler="/messages".`

3. PERBANDINGAN INFLUXDB DAN PROMETHEUS

3.1 Resolusi Pengolahan Data

InfluxDB memiliki resolusi pengolahan data sampai tingkat ketelitian *nanosecond* sedangkan Prometheus hanya sampai *millisecond*. Hal ini membuat InfluxDB lebih memiliki kinerja pengolahan data yang lebih baik dibandingkan dengan Prometheus.

3.2 Kompatibel Tipe Data

InfluxDB mendukung tipe data *float64, int64, bool,* dan *string*. Sedangkan Prometheus hanya mendukung tipe data *float64.*Dengan adanya variasi tipe data yang lebih pada InfluxDB dibandingkan dengan Prometheus, variasi tipe data ini membuat InfluxDB lebih fleksibel digunakan pada saat pengoperasian atau pengolahan berbagai jenis data. Usaha untuk mengkonversikan tipe data menggunakan Bahasa pemrograman akan diminimalisir atau bahkan tidak ada sama sekali.

3.3 Delay Pemrosesan Data

Hal pertama yang dipertimbangkan adalah *dela*y pemrosesan data. InfluxDB lebih baik dalam pemrosesan data yaitu pada InfluxDB penulisan data dilakukan sehabis mendapatkan respon sukses pengiriman data kepada *client* sedangkan pada Prometheus, terdapat *buffer default* penulisan data setiap lima menit, sehingga akan ada kemungkinan *data loss* atau data hilang. InfluxDB lebih cocok digunakan untuk *data logging* karena delay lebih rendah.

3.4 Bahasa *Query*

InfluxDB menggunakan Bahasa *query* seperti SQL.

```
SELECT * FROM "cpu_load_short"
WHERE "value" > 0.9
```

Sedangkan Prometheus menggunakan Bahasa *Query* yang lebih sederhana, bisa disebut juga model *direct querying*.

```
cpu_load_short > 0.9
```

Perbedaan penggunaan Bahasa *query* ini diserahkan kepada pengguna. Jika pengguna terbiasa dan lebih mudah menggunakan Bahasa *query* SQL maka InfluxDB bisa menjadi pilihan, tetapi jika pengguna ingin mengenal Bahasa *query* yang lebih sederhana namun terbilang "baru", Prometheus menyediakannya.

Untuk proses *query*, jika diinginkan proses *query* yang menawarkan *continuous queries* yang lebih baik maka InfluxDB pilihan yang lebih baik. Namun jika diinginkan *query* yang lebih *powerful* untuk masalah *graphing* dan *alerting*, maka Prometheus akan lebih baik.

3.5 *Labelling*

Pada Prometheus terdapat *key-value* sebagai label yang disebut *tags*. Sedangkan pada InfluxDB juga

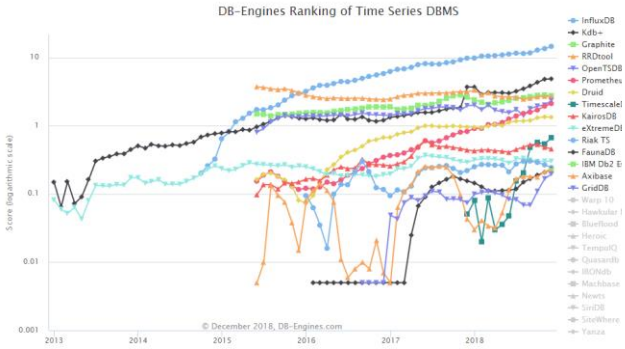
Terdapat *key-value* sebagai label, yang disebut *tags* dan juga terdapat data model lainnya yaitu *fields* sebagai *second level of labels data. Labelling* data yang lebih *expert* pada InfluxDB membuat manajerial data pada InfluxDB bisa lebih kompleks/terkategori.

3.6 Popularitas

27 systems in ranking, December 2

Rank Dec 2018	Rank Nov 2018	Rank Dec 2017	DBMS	Database Model	Score Dec 2018	Score Nov 2018
1.	1.	1.	InfluxDB	Time Series DBMS	14.70	+1.06
2.	2.	2.	Kdb+	Multi-model	4.91	+0.07
3.	3.	↑ 4.	Graphite	Time Series DBMS	2.78	-0.06
4.	4.	↓ 3.	RRDtool	Time Series DBMS	2.70	-0.02
5.	5.	5.	OpenTSDB	Time Series DBMS	2.35	+0.33
6.	6.	↑ 7.	Prometheus	Time Series DBMS	2.13	+0.18
7.	7.	↓ 6.	Druid	Multi-model	1.34	-0.02
8.	8.	↑ 15.	TimescaleDB	Time Series DBMS	0.67	+0.13
9.	9.	↓ 8.	KairosDB	Time Series DBMS	0.46	-0.03
10.	10.	↓ 9.	eXtremeDB	Multi-model	0.30	0.00

Gambar 3.6.1 Urutan 10 besar peringkat *time-series database* [5]



Gambar 3.6.2 Grafik urutan peringkat *time-series database* [5]

Per Desember 2018, InfluxDB menduduki peringkat pertama popularitas di kalangan *time-series database*, sedangkan Prometheus menduduki peringkat ke tujuh untuk popularitasnya. Hal ini secara tidak langsung membuat dokumentasi pembelajaran atau bahan belajar terkait InfluxDB lebih banyak dibandingkan dengan Prometheus.

3.7 Push System VS Pull System

InfluxDB menerapkan *Push System* pada *database*-nya. Secara sederhana, pada *push system*, aplikasi akan terus menerus "mendorong" data kepada sistem *monitoring.* Sedangkan Prometheus menerapkan *Pull System* pada *database*-nya. Yang secara sederhana juga dapat diartikan bahwa *server* Promotheus akan "meminta" data dari aplikasi yang sedang berjalan secara berkala. Kedua sistem inilah yang cukup menjadi pembeda antara InfluxDB dan Prometheus.

Pada dasarnya, kebanyakan database menerapkan *push system.* Hal yang menjadi kelebihan *pull system* milik Prometheus adalah bahwa *pull system* memberikan standar "Bahasa" pada berbagai jenis layanan dan aplikasi untuk menarik data yang ada [3]. Sehingga hal ini membuat pengguna menjadi lebih efektif dalam proses pengolahan data. Dengan standar "Bahasa" yang ada, pengguna tidak perlu lagi repot-repot untuk "menulis" data pada *collector*. Sebenarnya pada InfluxDB memiliki sebuah sistem tambahan yang bisa berperan sebagai *data puller* yang bernama Telegraf. Telegraf merupakan sebuah *plugin-driven server agent* atau bisa disederhanakan sebagai fitur tambahan dari InfluxDB yang dapat digunakan untuk mengumpulkan dan melaporkan *metrics*. Namun pada Telegraf ini, setiap *plugin* harus men-*define* kode khusus untuk menarik data dan mengubahnya menjadi format yang sesuai. Berbeda dengan *pull system* yang utuh, dimana semua proses penarikan data sudah ada standar "Bahasa"-nya. Kelebihan lain dari *pull system* adalah sistem apapun yang berasal dari luar sistem utama kita datanya dapat terkumpul dan terkirim oleh *pull system* tanpa mekanisme pengiriman data khusus dari sistem luar tadi.

Namun, *pull system* juga memiliki kekurangan yaitu, *pull system* tidak dapat berkerja dengan baik jika menangani *time-series* yang digerakkan oleh kejadin atau biasa disebut dengan *event-driven time-series* (seperti, *individual requests* kepada sebuah API) yang mana ini menjadi kekurangannya jika dibandingkan dengan InfluxDB yang bisa menanganinya karena

menggunakan HTTP *Representational State Transfer* (REST) API untuk proses *query* data. Selain itu, ketika terdapat banyak *endpoint* yang tersebar di berbagai penjuru yang secara tidak langsung terjangkau dikarenakan *firewall* atau pengaturan jaringan yang rumit, dan di mana tidak mungkin untuk menjalankan *server* Prometheus secara langsung disetiap segmen jaringan, *pull system* pada Prometheus akan kewalahan menghadapi situasi seperti ini. Situasi seperti ini bukanlah situasi di mana Prometheus bekerja sebagaimana mestinya. Sedangkan pada InfluxDB, situasi seperti ini tidak masalah, karena InfluxDB men-*support high availability* dan skalabilitas horizontal melalui *clustering*. Dengan adanya *clustering* ini, Penguna InfluxDB dapat melakukan *query* data lintas server atau lintas *endpoint*.

4. KESIMPULAN

InfluxDB dan Prometheus memiliki kelebihan dan kekurangannya masing-masing untuk kondisi tertentu. Penulis menyarankan untuk menggunakan InfluxDB sebagai *time-series database* untuk sistem *network monitoring* ketika:

1. Dibutuhkan pemrosesan data dengan ketelitian resolusi yang detil.
   Hal ini dikarenakan InfluxDB mempunyai resolusi pengolahan data sampai ke tingkat ketelitian *nanosecond*.
2. Dibutuhkan penulisan data yang cepat dan mengurangi kemungkinan *data loss.*
   Hal ini dikarenakan pada InfluDB penulisan data dilakukan sehabis mendapatkan respon sukses pengiriman data kepada client sedangkan pada Prometheus, terdapat *buffer default* penulisan data setiap lima menit, sehingga pada Prometheus akan ada kemungkinan *data loss* atau data hilang.
3. Dibutuhkan fleksibilitas kompatibel tipe data.
   Jika pengguna mengolah data dengan berbagai tipe data, InfluxDB merupakan pilihan yang cocok untuk menjawab kebutuhan itu.
4. Pengguna lebih mengenal baik Bahasa *query* SQL.
   InfluxDB menggunakan Bahasa *query* seperti SQL dan membutuhkan *continuous queries* yang lebih baik maka InfluxDB bisa menjadi pilihan.
5. Dibutuhkan *multi-dimentional data labelling.*
   InfluxDB mempunyai *key-value* sebagai label, yang disebut tags (yang mana ini terdapat juga pada Prometheus) dan juga terdapat data model lainnya yaitu fields sebagai second level of labels data. Hal ini membuat InfluxDB mempunyai sistem manajerial data yang lebih baik.
6. Tidak membutuhkan standar “Bahasa” atau format standar pada layanan dan aplikasi untuk menarik data yang ada (kelebihan *pull system*)
   Jika pengguna merasa tidak terlalu membutuhkan efektivitas dari penggunaan standar “Bahasa” atau format standar ini pada sistem *network monitoring* (pembahasan lengkap ada pada subbab 3.7), maka InfluxDB merupakan pilihan yang tepat.
7. Terdapat banyak *endpoint* atau *node* pada sistem *network monitoring*, maka InfluxDB bisa dijadikan solusinya (pembahasan lengkap ada pada subbab 3.7).

Penulis akan menyarankan untuk menggunakan Prometheus sebagai *time-series database* untuk sistem *network monitoring* ketika:

1. Tidak terlalu membutuhkan pemrosesan data dengan ketelitian yang terlalu detil.
2. Hanya dibutuhkan penulisan data dengan interval waktu yang tidak terlalu cepat, misalnya penulisan data akan ditulis lima menit sekali.
3. Hanya dibutuhkan satu jenis tipe data yang digunakan.
4. Pengguna membutuhkan Bahasa *query* yang lebih sederhana dan lebih *powerful* untuk *alerting* dan *graphing* maka Prometheus bisa menjadi solusinya.
5. Hanya dibutuhkan satu dimensi *data labelling* saja.
6. Dibutuhkan penggunaan standar "Bahasa" atau format standar pada layanan dan aplikasi untuk menarik data yang ada (kelebihan *pull system*).

   Prometheus menawarkan efektivitas pengguna dalam melakukan penarikan data/penulisan data, sehingga pengguna tidak perlu terlalu repot untuk melakukan harus men-*define* kode khusus untuk menarik data dan mengubahnya menjadi format yang sesuai. Serta pada Prometheus, sistem apapun yang berasal dari luar sistem utama kita datanya dapat terkumpul dan terkirim oleh pull system tanpa mekanisme pengiriman data khusus dari sistem luar tadi (pembahasan lengkap ada pada subbab 3.7).
7. Tidak terdapat terlalu banyak *endpoint* atau *node* pada sistem *network monitoring*.

5. DAFTAR PUSTAKA

[1] Noor Zehra Naqvi, Syeda dan Fantidou, Sofia. 2017. *Time Series Databases and InfluxDB*. Perancis: Universite libre de Bruxelles

[2] Fixstarts. 2018. *Time Seris Database Performance Comparison Using GridDB and InfluxDB (Revision 1.8).*

[3] Dix, Paul. Monitoring with Push vs. Pull: *InfluxDB Adds Pull Support with Kapacitor*. Diperoleh 21 Desember 2018, diakses dari https://www.influxdata.com/blog/monitoring-with-push-vs-pull-influxdb-adds-pull-support-with-kapacitor/

[4] Volz, Julius. *Pull Doesn't Scale or Does It?*. Diperoleh 21 Desember 2018, diakses dari https://prometheus.io/blog/2016/07/23/pull-does-not-scale-or-does-it/

[5] DB-Engines. DB-Engines *Ranking of Time Series DBMS*. Diperoleh 21 Desember 2018, diakses dari https://db-engines.com/en/ranking/time+series+dbms