Some InfluxDB gotcha's

Nov 7, 2017 *(Last modified: Mar 29, 2019)*

INFLUXDB PERFORMANCE STATS TICK · Share on:

If you've followed my vSphere performance data blog series you probably have noted that I used InfluxDB as the database for storing the performance data.

With over 4 months of performance data in the InfluxDB I've picked up some gotcha's along the way (there's probably more lying around which I've not come over yet).

In this blog post I'll outline what I've learned so far

(Save) Disk space

One of them is of course, and this is an obvious one, the amount of data and the corresponding disk space needed to store it.

Although InfluxDB is very good at compressing your data and saving a lot of space doing that it depends on you to do the right choices on how you write your data. This means that you want to use tag as they were intendent to, aka Design your Schema (although Influx is schema less)

Read the recommandations from the InfluxDB documentation (link is for version 1.3)

I won't go in to too much detail but to give a short intro:

- Data is written to a **measurement** which would be sort of like a Table in SQL
- The measurements are filled by **points** which corresponds to a row in SQL
- A point consists of three (four) things: (the measurement name), the **tag** set, the **field** set and the **timestamp**
- The **tag** set is a set of metadata tags to identify the point. For instance the name of a server, it's location etc. Stored as key/value pairs

The **timestamp**, in the wanted precision:
second/millisecond/microsecond/nanoseconds from epoch (defaults to nanosecond precision).

Particular for my use case would be that you use **tags** to add metadata, and not to include it in the measurement name or as a field. Tags should not include random data like changing identifiers and timestamps. In my case I do add the host and cluster to a VM measurement and this might change (especially the host), but this is a tradeoff I decided to make. The alternative would be to have the host as a **field**. This could also work, but here comes the fact that **tags are indexed and fields are not**. Fields cannot be used in a group by in a query which is a use case for us.

The timestamp precision is also something to think about. First it's worth noting that the default setting would work, and you might not need to ever worry about it. I've not touched that and the most important thing is that you use the same precision and that you know what precision you use! If not you could end up with using different precisions and the query you do won't pick up the correct data:

- November 1st 2017 would be 1509494400 with second precision, if you thought you used millisecond precision and use 1509494400000 as the timestamp that would be 12/23/49803

Besides this, and what really was the gotcha, is that the **more specific precision** you use the database will require **more diskspace** and this will lead to more data being transported and might give you decreased write throughput. This has to do with how the database writes the data to disk. In our use case with our 20 second vSphere metrics we could have used the second precision to save disk space

Note that you can force the precision when you write a point, see the doc's for more info

Monitor diskspace

This should be pretty self explanatory. Of course you do proper monitoring of your database?

database outgrew the initial disk space. As this project was done as a POC kind of thing I hadn't done things properly from the start with monitoring tools etc so things stopped. Luckily it was a quick fix, just give the VM more disk, extend the partition, restart Influx and things were back to normal.

Situations like this brings up the need for some insights to the database it self. InfluxDB has it's own metrics which are gathered and stored (in the InfluxDB of course) in the _internal measurement. With this being a measurement in the database I can make use of this in Grafana to build a dashboard with the internal metrics. Or even better explore some dashboards already created (community built) and available from the Grafana website. For instance InfluxDB Internals . This didn't include the database size, but it's pretty easy to add that to a panel.

Retention policy

As I mentioned in the last post of the vSphere performance series we haven't set any retention on the data, meaning that nothing gets deleted. This gives us currently a database with over 4 months of metrics at 20 second intervals for around 4000 VMs. 200 hosts and some SAN metrics.

Either way, the database is approaching 80GB. This is actually quite impressing given that we have 11 metrics for each VM. This would give us some (3 records per minute/4320 per day x 120 days x 11 metrics x 4000 VMs) 22.8 billion records only for the VMs. Add the metrics gathered for the hosts as well and you do have a huge dataset.

With that said I don't think that 4 months of 20-second interval metric data is needed, at least not in our use case. I am experimenting and getting familiar with the built in retention policy settings in Influx and we will at first probably set the retention to 2 months (this is probably also more data than we need).

I would recommend people starting with an Influx project to do some experimentation with this from day 1. The retention policy in use is actually one of the identifiers to the written point meaning that if you change to a different policy your points for the same metric could be in separate "tables". Of course this will only

Another thing to note while looking at Retention is the Shard group duration which is a part of the Retention policy. The shard is the actual data on disk (represented by a TSM file). A shard belongs to a single shard group and contains a specific set of series. All the points in a given series in a given shard group will be stores in the same shard (file) on disk. The shard group duration determines the time period each of the shard groups contains. By default this is set to 1 week and it contains all points for the given retention policy during that period.

This will in turn mean that even though the actual point has falling out of the Retention period it might actually still be in the database because it's inside your Shard duration

Aggregation

Following (or maybe in front of) the discussion on retention policy you will probably touch on Aggregating, "Wrapping up" or Downsampling your data. Initially this was something I had decided not to do because of the potential overhead it gives (maintaining more measurements, maintaining the downsampling logic etc). We discussed it internally in my team (which is the primary consumer of the data) and found that it wasn't worth looking into especially when you can do grouping on time directly in Grafana panels

But as data gets accessible it brings other needs and use cases. Some teams have the trend of a specific metric as their use case, and maybe from more than one VM. This would potentially bring a lot of load on the database with the amount of records at 20-second interval. What I've found is that you cannot expect all users to understand the need for doing the aggregation/grouping through Grafana, and it would require them to have a sort of deep insight into the data to know what they should group on.

I have started building my own API on top of Influx which also brings some metadata from other systems. Here I can control the grouping of data without the users needing to specify it (other than request data for a given time period). What this does is that it actually have introduced some of the downsampling logic I wanted to avoid. The difference is that I need to maintain this my self in the API backend.

the concept of Continuous queries which can downsample metrics to a new/different measurement. Check the documentation for lots of examples on how to do this.

There is a couple of things to consider while doing this which we are discussing:

- The potential need for a new measurement for each metric
- The field set might be different (i.e. avg and max as opposed to only a value field)
- Extra load on the database on regular intervals (but you have more control than getting that irregularly with large queries)
- Extra db maintenance as to ensuring that the downsampling queries are running and performing as expected
- And of course, you lose data when aggregating

Read/write metrics

This is something we discussed when starting this project. Should we sample read/write metrics as they come from vSphere (i.e. split them in read/write) or add them to a "total" metric. The answer was that we wanted the more granular view so we went along with read/write.

As with aggregation there is other use cases than just our own (and actually other use cases for us as well). We also want to see the total usage of for instance IOps or Disk Throughput for a given VM at a given period. When you store read/write as separate measurements n InfluxDB this isn't that easy to accomplish. Influx has no concept of Joins as a SQL database. This is one of the trade offs made to have a really fast and performing time-series database.

We have solved this in two ways. One in Grafana by adding the two measurements in the same panel and for some panels (for instance when we look at a specific VM) we are using the stacking feature to get a grip of the total. In our custom built dashboards outside of Grafana we have some backend code doing the join for us. This can be done and it's working, but it's kind of a messy thing as we need to traverse and align the timestamps to add the correct values together.

write and total as **fields**, or even better disk_iops with read and write as fields. Then we could have added the two fields together inside the same measurement to present the total. But it's kind of late for that now. This would also give us a different field set than for the other measurements and would require users to know that for this metric I need to use value as the field, and for this I must use the read/write or doing a summation. So we have no "one size fits all" for this.

I could also do the summation in my polling scripts and write the total to a new measurement.

Data insight

In the newer versions of InfluxDB the admin interface has been deprecated, and in the current version 1.3 it's gone. I've never tested it and the lack of a GUI can be a turn off for some. While I've managed to use the CLI and the API there has been times where I could have had use of a GUI.

In my team we have som business analysts who are very interested in all kinds of data we can feed them with. Without some kind of GUI or management tool for them to play with they've been left out of this data. They have used Grafana dashboards for some of it, but they want to explore the database more directly.

Although I'm building some APIs which can be accessed by things like PowerCLI where they can create CSV files etc this is something that you should be aware of when exploring InfluxDB.

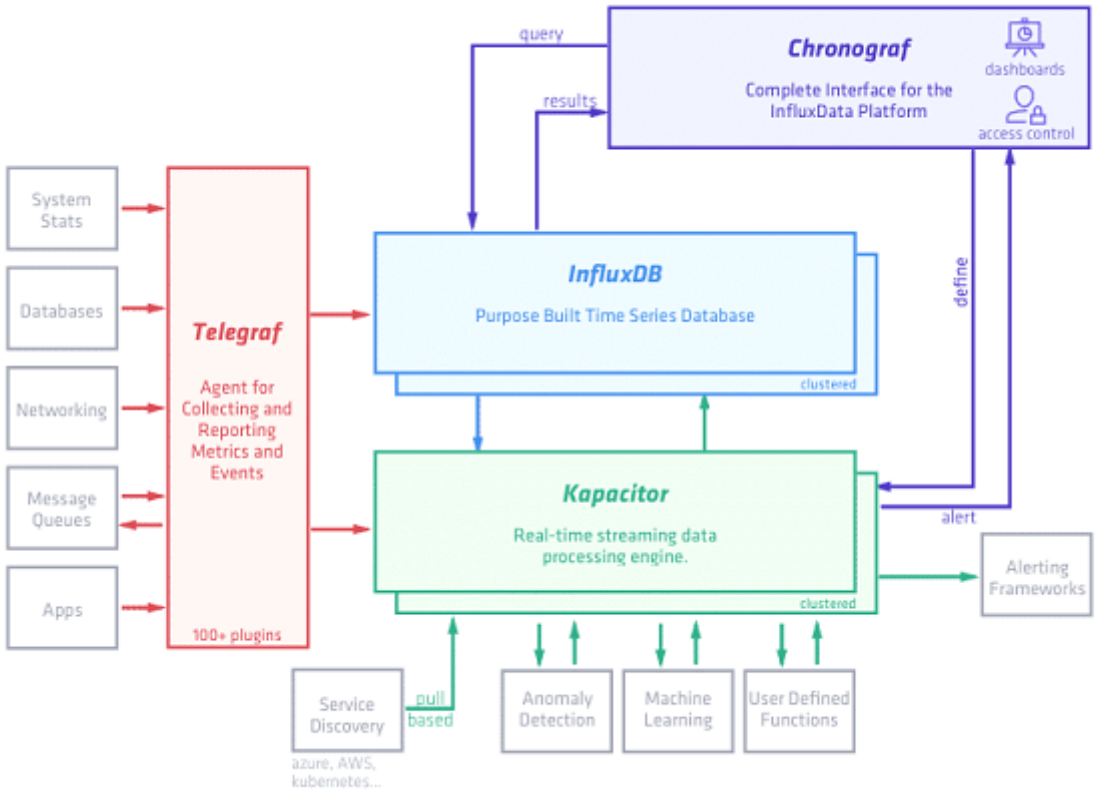
Next step

What I have found while digging into Influx and how to solve the challenges we have had so far is that InfluxData (which is the company behind InfluxDB) actually could have the answer to a lot of this already. Namely the TICK stack which InfluxDB is a part of.

The TICK stack consists of:

- InfluxDB, our time-series database
- Chronograf, a web gui to control the TICK stack
- Kapacitor, a data processing engine capable of speaking InfluxQL

Here's a flow chart from the TICK stack web page describing the individual components.



TICK Stack

I've started exploring the individual components and it looks very interesting. Stay tuned for details on if and how they will be implemented in our project

If you've read all the way here - thanks for your attention! Want to learn more about InfluxDB? Check out their recent Internals 101 blog post

This page was modified on March 29, 2019: Fixing links and markdown syntax

Related Posts

- vSphere Performance data - New vSphere plugin for Telegraf — October 1, 2018

- vSphere Performance - Visualizing data around the world with the Grafana Worldmap panel — April 16, 2019
- vSphere Performance - vCenter Server Appliance (VCSA) monitoring — December 3, 2018

Rudi Martinsen

Rudi is a VMware vExpert, a Tanzu Vanguard, and a 2xVCIX (Datacenter Virtualization & Cloud Management and Automation). Currently focusing on Cloud Automation, Tanzu and modern apps

READ MORE

Featured Posts

- CKA Study Guide - 2024 edition
- Gitops Your Scheduled Tasks
- Introducing the Tanzu Tech Zone
- VMware Retires Tanzu Community Edition

Recent Posts

- CKA Study notes - Service Accounts in Kubernetes Pods - 2024 edition
- Installing the NFS CSI Driver on a Kubernetes cluster to allow for dynamic provisioning of Persistent Volumes
- CKA Study notes - ConfigMaps and Secrets - 2024 edition
- CKA Study notes - Pod Scheduling - 2024 edition
- Gitops Your Scheduled Tasks
- Introducing the Tanzu Tech Zone
- VMware Retires Tanzu Community Edition

VMWARE 114 PERFORMANCE 35 KUBERNETES 34 VREALIZE 34

TANZU 32 HPE 16 CERTIFICATION 13 SPEAKING 13

POWERSHELL 6 BLOG 5

ALL CATEGORIES

Tags

K8S 45 KUBERNETES 38 TANZU 35 VSPHERE 34

POWERSHELL 28 CKA 27 GRAFANA 25 VCENTER 24

POWERCLI 22 VRA 20

ALL TAGS