ulm university universität
uulm

Cross-organizational distributed systems and Clouds

Exercise 2: Monitoring and Stressing

Christopher B. Hauser

Institute of Information Resource Management

2018-05-03

Exercise 2: Monitoring and Stressing

In this unit, we will set up a monitoring infrastructure for our mediawiki installation. We will then establish a script-based mechanism that stresses the platform; and identify the differences between having different sized virtual machines. This session assumes that you have a working set-up of mediawiki and that you are able to easily setup additional instances. This was subject to the Hands-on I session.

Part 1: Add Monitoring System

Content: Monitoring Sink

First we need a new Instance/VM, which will host the monitoring data and provide a web-based dashboard to access it. The monitoring solution we will use is the TICK stack of InfluxData, described here [1]. While Telegraf will run on the VM with Mediawiki installed, we need another VM for InfluxDB and Chronograf.

[1] https://www.influxdata.com/products/open-source/

Question: Parts and responsibilities of the TICK stack

Please describe the components of the TICK stack and their purpose. Which of these components are needed for a minimum TICK setup?

Task: Install the monitoring data sink

We will now install the InfluxDB and Chronograf in a new VM, which will be the data sink for several Telegraf monitoring adaptors. This guide follows the official installation guide [1]

[1] https://docs.influxdata.com/influxdb/v1.2/introduction/installation/

Create a new security group on bw-cloud via “Access & Security”:

- Name: monitoring
- Create, and go to “manage rules”:
  - Add Rules for TCP ingres ports 8888, 8086, 8088

Create a new instance on bw-cloud:

- Name: monitoring
- Flavor: m1.small
- Image: Ubuntu Server 16.04
- Select both the default and the monitoring security groups

- Validate that your ssh key and the public network are selected

Log in via SSH to the new vm with username “ubuntu” and the public IP you got assigned. In the new vm, install InfluxDB and Chronograf:

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable"
    | sudo tee /etc/apt/sources.list.d/influxdb.list
sudo apt-get update && sudo apt-get install influxdb
sudo service influxdb start
wget https://dl.influxdata.com/chronograf/releases/chronograf_1.3.0_amd64.deb
sudo dpkg -i chronograf_1.3.0_amd64.deb
```

In your monitoring VM you should have two services influxdb and chronograf running now. Validate that both are running, using the `systemctl` command.

Access the chronograf dashboard via your webbrowser, pointing to your public IP of the monitoring vm and port 8888 (http://134.60.XY.YZ:8888). Tell Chronograf where InfluxDB is located:

- Connection String: http://localhost:8086
- Name: InfluxDB
- Username and Password empty!
- Telegraf database: telegraf
- Check the “Make this the default source” box
- Click “add source” button.

You will then see a empty list of hosts. Let’s install the Telegraf on the Mediawiki page, to have some data to display :-)

Question: What ports are used by influxdb?

Now that you have InfluxDB and Chronograf running, what ports are used by InfluxDB? You can get this information from your monitoring VM or from the documentation.

Hint: In the monitoring VM run `sudo netstat -tulpen` to get all ports that are open. Find the InfluxDB application.

Task: Add monitoring to mediawiki

Next, connect via SSH to the VM hosting your mediawiki. Let’s install Telegraf, which will collect monitoring statistics and send it to the monitoring VM.

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable"
    | sudo tee /etc/apt/sources.list.d/influxdb.list
sudo apt-get update && sudo apt-get install telegraf
sudo service telegraf start
```

We need to configure telegraf to point to your monitoring vm. Open as root (with sudo) e.g. with vim the file /etc/telegraf/telegraf.conf. In the `[[outputs.influxdb]]` section change the urls field from `urls = ["http://localhost:8086"]` to point to the public IP address of your monitoring vm.

Restart telegraf with `service telegraf restart` and check the Chronograf dashboard. You should see monitoring data from host "main-server" now. You will see the basic system metrics of this vm. Let's add monitoring data from apache and mariadb next. Re-open the file /etc/telegraf/telegraf.conf and uncomment the following lines:

```
...
[[inputs.apache]]
   urls = ["http://localhost/server-status?auto"]
...
[[inputs.mysql]]
   servers = ["root:PASSWORD_FOR_DATABASE@tcp(127.0.0.1:3306)/"]
...
```

Now restart again telegraf, to apply your changes to the configuration. After some seconds, check the chronograf dashboard again. You have apache2 and mysql monitoring data now as well.

Part 2: Stressing the Wiki

Task: Stressing the Wiki

Now that we have a monitoring and our mediawiki running, it is time to put some load on mediawiki.

1. Run a stressing application from your computer (client) that queries the wiki. For this purpose you can use the apache benchmarking tool. After installing it (https://httpd.apache.org/docs/2.4/programs/ab.html), run two separate load tests; one without and one with concurrency. WARNING: before running the test, make sure you are pointing it to YOUR wiki installation.
    1. `ab -n 5000 http://134.60.xy.yz/wiki/index.php/Main_Page/`
    2. `ab -c 10 -n 5000 http://134.60.xy.yz/wiki/index.php/Main_Page/`

2. Explain what consequences the permanent querying has for your application. In particular, identify the (e.g. CPU, memory and network) load caused for your server via the Chronograf dashboard.
3. How many request/per second can your server handle in both test cases (output of ab)? How many requests on the apache and the database are shown in the Chronograf dashboard?
4. What do you think is the bottleneck the current setup? The network? The apache webserver? The database? Use the Chronograf dashboard to find possible bottleneckts.

Part 3: Vertical Scaling

Question: Vertical Scaling

Vertical Scaling Seite verschieben: Vertical Scaling Seite aktualisieren: Vertical Scaling Seite anzeigen: Vertical Scaling Seite löschen: Vertical Scaling Try to scale your application vertically, by adding more virtual hardware resources. OpenStack in the Instances overview the option to “resize instance”. Before we use this option to add more CPU and Memory, we should back up our work, by creating a new snapshot, e.g. named “mediawiki-snapshot-2”. You can safely remove the first snapshot, since it has no monitoring installed. After the snapshot finished, resize your main_server instance with your mediawiki application to flavor m1.large. The resize process takes up to 1 minute and needs to be confirmed, don’t miss this step on the dashboard.

Next, validate that mediawiki and the monitoring data are still working properly.

Finally, repeat the steps for stressing the mediawiki. Compare the results of both benchmarks (without concurrency, with concurrency). What differences can you see / not see? Can you explain them?

Part 4: (Bonus) Improve performance

Question: Improve performance

Can you improve the performance of your wiki, without using another larger server?

The mediawiki guide on performance will prove helpful:

https://www.mediawiki.org/wiki/Manual:Performance_tuning

What is the best result (in terms of request/s) you can achieve? Provide the results of your best apache benchmark run!