FLOWBSTER – AUTODOCK VINA TUTORIAL

Flowbster – autodock vina tutorial for MTA Cloud users v1.0

MTA Cloud team
info@mta.cloud.hu

Table of contents

Overview

About Flowbster

Flowbster is a new cloud-oriented workflow system. It was designed to create efficient data pipelines in clouds by which very large data sets can efficiently be processed. The Flowbster workflow can be deployed in the target cloud as a virtual infrastructure through which the data to be processed can flow and meanwhile it flows through the workflow it is transformed as the business logic of the workflow defines it. The Flowbster workflow can be deployed in the target cloud on-demand based on the underlying Occopus cloud deployment and orchestrator tool. Flowbster also provides an intuitive graphical user interface for end-user scientists. This interface hides the low level cloud-oriented layers and hence users can concentrate on the business logic of their data processing applications without having detailed knowledge on the underlying cloud infrastructure.

About AutoDock Vina application

In this case we have used Flowbster to set up the infrastructure for processing the Vina workflow. The setup is as follows: one VM is acting as the Generator, 5 VMs are acting as Vina processing nodes, and finally one VM is acting as the Collector node. The application used to execute the performance measurements was a workflow based on the AutoDock Vina application. The workflow consists of three nodes: a Generator, a set of Vina processing nodes, and a Collector. The input of the workflow includes the followings: a receptor molecule, a Vina configuration file, and a set of molecules to dock against the receptor molecule. The task of the generator node is to split the set of molecules to dock into a number of parts. The task of the Vina nodes is to process these parts, iterating through each molecule in the given part, by performing the docking simulation. The result of the docking includes an energy level, finally the user is interested in the docking with the lowest energy level.The task of the Collector node is to get the processing result of each molecule part from the Vina nodes, and select the best 5 energy levels. For running the experiment, we selected a molecule set of 3840 molecules. This set was split into 240 parts, so each part included 16 molecules to dock against the receptor molecule.

Features

- creating nodes through contextualisation
- using the nova resource handler
- utilising health check against a predefined port and url
- using parameters to scale up worker nodes

Prerequisites

- accessing an Occopus compatible interface
    - Occopus tool can launch virtual machines on MTA Cloud, and we recommend the use of nova interface (the url of the nova interface can be found under Compute/ Access & Security / API Access / Identity menu)
    - **Important**: **the use of Occopus tool currently works only on the SZTAKI branch of the MTA Cloud**
- target cloud contains an Ubuntu 14.04 image with cloud-init support
    - This image file can be found under Compute/Images/Public/Ubuntu 14.04 LTS image

Steps

Installation of Occopus

The deployment of Flowbster will be established with the help of Occopus tool, therefore we need to install the Occopus tool first. You can install the Occopus orchestration tool with just one command. For more information about Occopus itself, and how to install it, visit the following link. We recommend launching an Ubuntu-based virtual machine in MTA Cloud to install the Occopus tool on it.

Download

Occopus works based on descriptors. We have prepared the descriptors for the installation of Flowbster for the end-users. Based on these descriptors, Occopus will build the infrastructure in the target cloud. They can be downloaded from the following link: tutorial.examples.flowbster-autodock-vina . Install descriptors on a virtual machine which runs Occopus.

Note: This tutorial uses nova resources. However, feel free to use any Occopus-compatible cloud resource for the nodes, but we suggest to instantiate all nodes in the same cloud.

1. Fill the resource section

Open the file nodes/node_definitions.yaml and edit the resource section of the nodes labelled by node_def:.

- you must select an Occopus compatible resource plugin
    - this will be the nova resource plugin, same as in the tutorial
- you can find and specify the relevant list of attributes for the plugin
- you may follow the help on collecting the values of the attributes for the plugin
- you may find a resource template for the plugin in the resource plugin tutorials

The downloadable package for this example contains a resource template for the ec2 plugin.

It is important that end-users should personalize the node definition file to the user before launching. In this file, we add the resource identifiers we will use, such as project ID, virtual machine size, and so on. We can not provide these identifiers for the user, but they can be easily collected from the MTA Cloud web interface. For detailed assistance, visit this link or the documentation below. The downloadable package in this example contains the ec2 plugin resource template (for MTA Cloud use nova plugin).

2. Authentication

Make sure your authentication information is set correctly in your authentication file. You must set your email and password in the authentication file. Setting authentication information is described here.

3. Creating firewall rules

Components in the infrastructure connect to each other, therefore several port ranges must be opened for the VMs executing the components. Log in to the MTA Cloud OpenStack interface. Under "Compute / Access & Security" you can create a new firewall rule by clicking the "Create Security Group" button. After creation, you can edit the firewall rule by clicking the "Manage Rules / Add Rule" button. Add the following port to the security group:

- `TCP 5000 receiverport. This is used by nodes to handle incoming requests from other agents.`

4. Start the gather service

Please note that in order to receive the results, you have to run a Gather service (part of Flowbster), which will finally gather the results (the docking simulations with the lowest energy levels) from the Collector (last node in the workflow). Start the Gather service using the following command:

```
scripts/flowbster-gather.sh -s
```

By default, the Gather service is listening on port 5001.

Note: The scripts in the scripts directory need Python 2.7. Alternatively, you can activate the Occopus virtualenv!

5. Edit variables

Edit the “variables” section of the infra-autodock-vina.yaml file. Set the following attributes:

- **gather_ip** is the ip address of the host where you have started the Gather service
- **gather_port** is the port of the Gather service is listening on

```
gather_ip: &gatherip "<External IP of the host executing the Gather service>"
gather_port: &gatherport "5001"
```

6. Set the size of the cluster

Update the number of VINA nodes if necessary. For this, edit the *infra-autodock-vina.yaml* file and modify the *“min”* parameter under the *“scaling”* keyword. Currently, it is set to 5.

```
- &VINA
    name: VINA
    type: flowbster_node
    scaling:
            min: 5
```

7. Import

Load the node definitions into the database. Make sure the proper virtualenv is activated!

```
occopus-import nodes/node_definitions.yaml
```

Make sure that the proper virtualenv is activated! If you have not done this before, use the following command to activate the Occopus virtual environment:

```
source occopus/bin/activate
```

Important: Occopus takes node definitions from its database when builds up the infrastructure, so importing is necessary whenever the node definition (file) changes!

8. Deployment

Start deploying the infrastructure.

```
occopus-build infra-autodock-vina.yaml
```

9. The established infrastructure

After successful finish, the node with `ip address` and `node id` are listed at the end of the logging messages and the identifier of the newly built infrastructure is printed. You can store the identifier of the infrastructure to perform further operations on your infra or alternatively you can query the identifier using the occopus-maintain command.

```
List of nodes/ip addresses:
VINA:
  <ip-address> (2f7d3d7e-c90c-4f33-831d-91e987e8e8b2)
  <ip-address> (49bed8d2-94b0-4a7e-9672-744921dacac0)
  <ip-address> (10664026-0b31-4848-9f7a-98f880f98be7)
  <ip-address> (a0f5d091-aecc-488c-94f2-34e546f87832)
  <ip-address> (285d7efd-84a7-4ed5-a6fa-73db47bc2e87)
COLLECTOR:
  <ip-address> (4ca11ad3-a6ec-411b-89e6-d516169df9c7)
GENERATOR:
  <ip-address> (9b8dc4f1-bed4-4d1c-ba9e-45c18ee2523d)
30bc1d09-8ed5-4b7e-9e51-24ed881fc166
```

10. Send input files

Once the infrastructure is ready, the input files can be sent to the Generator node of the workflow (check the address of the node at the end of the output of the *occopus-build* command). Using the following command in the *flowbster-autodock-vina/inputs* directory:

```
../scripts/flowbster-feeder.sh -h <ip of GENERATOR node> -i input-description-for-vina.yaml -d input-ligands.zip -d input-receptor.pdbqt -d vina-config.txt
```

The -h parameter is the Generator node's address, -i is the input description file and with -d we can define data file(s).

Note: The scripts in the scripts directory need Python 2.7. Alternatively, you can activate the Occopus virtualenv!

Note: It may take a quite few minutes until the processes end. Please, be patient!

11. Check the process

With step 10, the data processing was started. The whole processing time depends on the overall performance of the VINA nodes. VINA nodes process 240 molecule packages, which are collected by the Collector node. You can check the progress of processing on the Collector node by checking the number of files under */var/flowbster/jobs/<id of workflow>/inputs* directory. When the number of files reaches 240, Collector node combines them and sends one package to Gather node which stores it under directory */tmp/flowbster/results.*

12. Stop Gather service

Once you finished processing molecules, you may stop the Gather service:

```
scripts/flowbster-gather.sh -d
```

13. Delete the infrastructure

Finally, you can destroy the infrastructure using the infrastructure id returned by *occopus-build*

```
occopus-destroy -i 30bc1d09-8ed5-4b7e-9e51-24ed881fc166
```

Report bugs or any other project related claims

Communication and support for MTA Cloud services are in the form of email. The common e-mail address is info@cloud.mta.hu. A notification form generated from this error will be generated by a designated member of the MTA Cloud team