JasmineGraph
Distributed Graph Database Server

JasmineGraph Version : v0.1.0
Date : 25/03/2024

Setting up JasmineGraph

Installation details can find in the following link

[miyurud/jasminegraph: Distributed Graph Database Server]

Setting up configuration values

GCN model training process with federated training can be configured using the configuration file

Configuration path location : conf/jasminegraph-server.properties.

The following image shows sample configurations related to federated learning

```
#----------------------------------------------------------------------------
#Federated Learning Parameters
#----------------------------------------------------------------------------

org.jasminegraph.federated.enabled=true
org.jasminegraph.fl.location=/home/ubuntu/software/jasminegraph/src-python/
org.jasminegraph.fl.dataDir=/home/ubuntu/software/jasminegraph/src-python/data/
org.jasminegraph.fl.weights=/home/ubuntu/software/jasminegraph/src-python/weights/
org.jasminegraph.fl_clients=2
org.jasminegraph.fl.epochs=4
org.jasminegraph.fl.rounds=4
org.jasminegraph.fl.org.port=5050
org.jasminegraph.fl.num.orgs=1
org.jasminegraph.fl.weights.file=/home/ubuntu/software/jasminegraph/src-python/temp/weights.txt
org.jasminegraph.fl.flag.file=/home/ubuntu/software/jasminegraph/src-python/temp/flag.txt
org.jasminegraph.fl.aggregator=true

#----------------------------------------------------------------------------
```

Possible configurations

Org.jasminegraph.federated.enabled - enables jasmineGraph's federated mode

Org.jasminegraph.fl.aggregator - enables jasmineGraph's aggregator mode

Can define the mode of the jasmineGraph server.

1. Aggregator mode
   The server runs as a global aggregator during the training process.
2. Non-aggregator mode
   The server runs as a non-aggregator during the training process.

org.jasminegraph.fl_clients - defines local client within an organization (single cluster)

org.jasminegraph.fl.epochs - defines the number of epochs for local training within an organization (single cluster)

org.jasminegraph.fl.rounds - defines the number of rounds for local training within an organization (single cluster)

org.jasminegraph.fl.org.port - defines the organizational server port

org.jasminegraph.fl.num.orgs - defines the number of organizations (Only available for aggregator mode)

Default values

```
org.jasminegraph.fl_clients=2
org.jasminegraph.fl.epochs=4
org.jasminegraph.fl.rounds=4
org.jasminegraph.fl.org.port=5050
org.jasminegraph.fl.num.orgs=1
```

Start JamineGraph Server

JamineGraph server can run on two modes

1. Native version
2. Docker version

1. Start jasmineGraph Server in Native Version

Start jamineGraph server in Docker version

Commands descriptions

Command	Description
lst	Shows the list of graphs uploaded to the server
adgr	Upload graph into the JasmineGraph server
adgr-cust	Upload graph into JasmineGraph server(customized version)
vcnt	Shows vertex count related to the graph id
ecnt	Shows edge count related to the graph id
rmgr	Remove graph using graph id
shdn	Shutdown the jasmineGraph server
trian	Shows triangle count related to the graph id
merge	Merge central-store and local store before federated training (Preprocessing step)
train	Train the GCN models on federated learning mode or distributed mode

JasmineGraph commands detailed descriptions

1. train

Sample command (following command trains graph with id number 1 l)

```
File  Edit  View  Search  Terminal  Help
damitha@damitha-TUF-Gaming-FX505DU-FX505DU:~$ telnet localhost 7777
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
train
Available main flags:
graph_id model_id
Send --<flag1> <value1> --<flag2> <value2> ..
--graph_id 1
```

Sample Test

Number of organizations 1
Number of clients - 2 (this defines the number of local clients).
Number of epochs - 2
Number of rounds - 3
Graph ID - 1

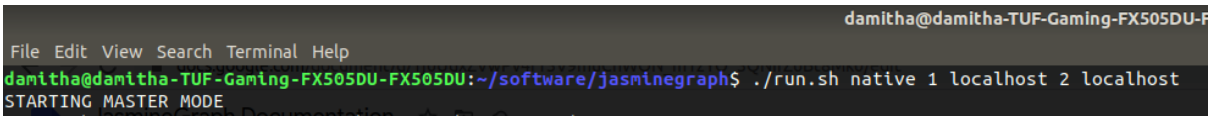
Step 1 - Edit the configuration file (conf/jasminegraph-server.properties)

```
#----------------------------------------------------------------------------
#Federated Learning Parameters
#----------------------------------------------------------------------------

org.jasminegraph.federated.enabled=true
org.jasminegraph.fl.location=/home/ubuntu/software/jasminegraph/src-python/
org.jasminegraph.fl.dataDir=/home/ubuntu/software/jasminegraph/src-python/data/
org.jasminegraph.fl.weights=/home/ubuntu/software/jasminegraph/src-python/weights/
org.jasminegraph.fl_clients=2
org.jasminegraph.fl.epochs=2
org.jasminegraph.fl.rounds=3
org.jasminegraph.fl.org.port=5050
org.jasminegraph.fl.num.orgs=1
org.jasminegraph.fl.weights.file=/home/ubuntu/software/jasminegraph/src-python/temp/weights.txt
org.jasminegraph.fl.flag.file=/home/ubuntu/software/jasminegraph/src-python/temp/flag.txt
org.jasminegraph.fl.aggregator=false
```

Step 2 - Start the JasmineGraph Server (Native or Docker version).

Please note that In this example, JasmineGraph is running in the native mode.

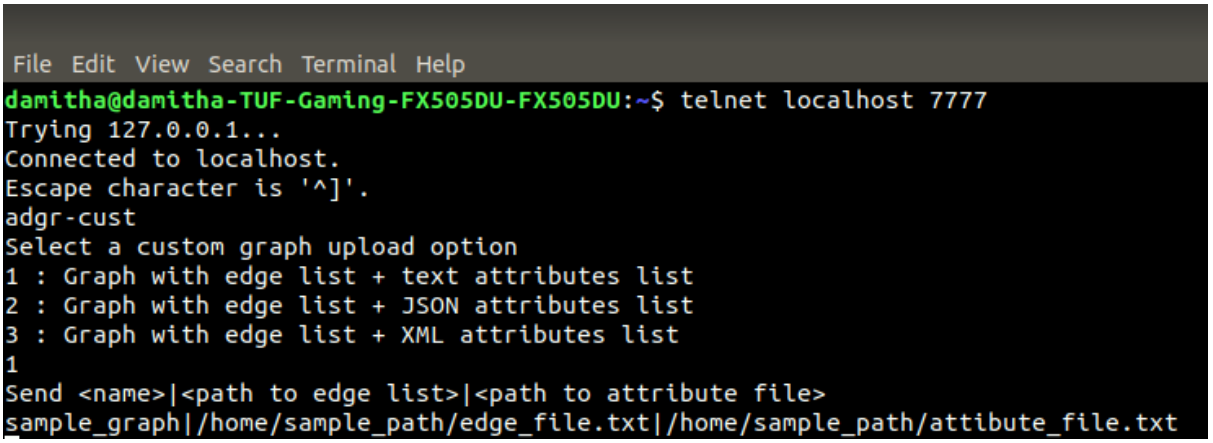


Step 3- Connect to JasmineGraph Server.

Sample command: telnet localhost 7777

Step 4 - Upload the graph into JasmineGraph server.

Please note that for this example adgr-cust command is used to upload the graph into the JasmineGraph Server



Step 5 - Run merge command

In the current jasminegraph process training data can be split between the central store and the local store due to edge cuts. To reduce this information loss, a merge command was introduced to merge the central store data and the local store data and output the final dataset.

It is possible to check the status of the process by checking the logs.

```
[2021-03-05 14:58:45.598] [logger] [info] Operation done successfully
[2021-03-05 14:58:45.598] [logger] [info] Scheduling training order for each worker
[2021-03-05 14:58:45.599] [logger] [info] Database opened successfully
[2021-03-05 14:58:45.599] [logger] [info] Operation done successfully
[2021-03-05 14:58:45.599] [logger] [info] Merge Commands Sent
```

Step 6 - Train the model

Execute train command and input graph id as follows

```
File  Edit  View  Search  Terminal  Help
damitha@damitha-TUF-Gaming-FX505DU-FX505DU:~$ telnet localhost 7777
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
train
Available main flags:
graph_id model_id
Send --<flag1> <value1> --<flag2> <value2> ..
--graph_id 1
```

Step 7 - Link Prediction

Execute train command and input graph id as follows

Streaming graph uploading

Running PageRank

1. before execute the PageRank, execute the “idd” command to calculate the in-degree distribution. In the next line, enter the graphID value.

2. To execute the PageRank algorithm, enter the "pgrnk" command.

3. When prompted, provide the necessary parameters in the following format: `graphID|alpha|iterations`

- `graphID` is the identifier for the graph you want to run PageRank on.
- `alpha` is an optional parameter representing the damping factor. It should be a value in the range 0 <= alpha < 1.
- `iterations` is an optional parameter indicating the number of iterations for PageRank. It should be a value in the range 0 < iterations < 100.

4. You can choose to omit the `alpha` and `iterations` values if you want to use the default settings, which are as follows:
- Default alpha value: `0.85`
- Default iterations value: `10`

```
Connected to localhost.
Escape character is '^]'.
lst
|1|epinions|/home/ubuntu/software/jasminegraph/epinions.txt|nop|
|2|epinions|/home/ubuntu/software/jasminegraph/epinions.txt|nop|
|3|powergrid|/home/ubuntu/software/jasminegraph/powergrid.dl|op|
pgrnk
send
3|0.5|40
done
```

Upon the completion of PageRank score calculations, a file with the name `graphID + "_pgrnk_" + partitionID` will be stored with pageRank scores in the following path: `/var/tmp/jasminegraph-localstore`.

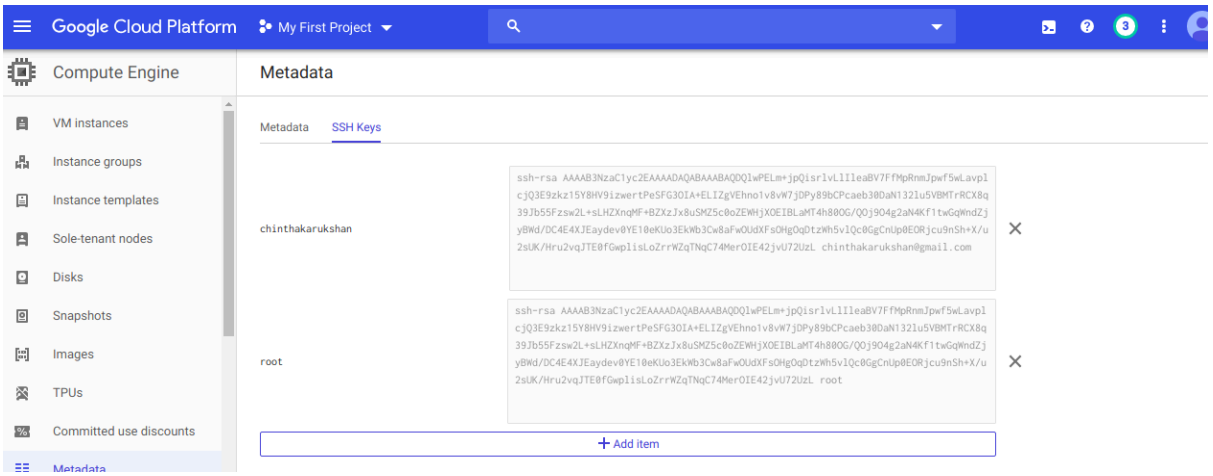
2. Jasmine Graph Docker

Docker Installation Guide

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04

Enable ssh root login in google cloud VM instances

1. Login to the GCP VM
2. switch to root (sudo su)
3. nano /etc/ssh/sshd_config
4. change “PermitRootLogin” to “yes” and save the file
5. restart SSHD (service sshd restart)
6. Configure the same ssh public key to login as root by changing username to root (Refer ssh_root_public_key.png file)



Enable ssh root login in AWS instance

1. Launch new AWS instance
2. Copy your public key to the aws instance
    cat ~/.ssh/id_rsa.pub| ssh -i Chinthaka.pem ubuntu@3.88.19.172 "cat - >> ~/.ssh/authorized_keys2"
3. Login to the aws instance
4. sudo -s (to become root)
5. vi /root/.ssh/authorized_keys
6. Delete the lines at the beginning of the file until you get to the words ssh-rsa
7. vi /etc/ssh/sshd_configcat ~/.ssh/id_rsa.pub| ssh -i Chinthaka.pem root@3.88.19.172 "cat - >> ~/.ssh/authorized_keys2"
8. Set the variable PermitRootLogin to without-password
9. service sshd restart
10.Then you can login using the root user (ssh -i server.pem root@serverhost)
11. To enable pemless login
    cat ~/.ssh/id_rsa.pub| ssh -i Chinthaka.pem root@3.88.19.172 "cat - >> ~/.ssh/authorized_keys2"
12. Then you can login using ssh root@serverhost

References :

[1] https://stackoverflow.com/questions/7407333/amazon-ec2-root-login

Jasminegraph Docker Containers

1. Enable pem-less login to aws instances by copying your public key to aws instance.
    cat ~/.ssh/id_rsa.pub| ssh -i Chinthaka.pem root@3.88.19.172 "cat - >> ~/.ssh/authorized_keys2"
2. Do step 1 for all the aws instances that are going to use.
3. Copy private key, public key and known hosts to the Container

    docker cp id_rsa 2c6472ff145d:/root/.ssh
    docker cp id_rsa.pub 2c6472ff145d:/root/.ssh
    docker cp known_hosts 2c6472ff145d:/root/.ssh

Or else mount the host .ssh directory to /root/.ssh directory in docker container when starting the docker container

4. To start JasmineGraph please use below command.

docker run -v "/var/run/docker.sock:/var/run/docker.sock:rw" -v "/root/.ssh:/root/.ssh" jasmine_graph --MODE 1 --MASTERIP <MasterIP Address> --WORKERS <NO OF WORKERS> --WORKERIP <Worker IPs (Comma separated list)>

Workers will get started by the master node automatically. When starting the worker nodes the master node will pass the ports which are to be used by the worker.

If you are running on a single host makesure to use WORKERIP an IP address of your host other than any loopback IP addresses. In addition you may need to front it with your account's username. E.g., --WORKERIP miyurud@172.28.8.7

Eg:

docker run -v "/var/run/docker.sock:/var/run/docker.sock:rw" -v "/root/.ssh:/root/.ssh" -v "/tmp:/tmp" -p 7777:7777 -p 7778:7778 jasmine_graph --MODE 1 --MASTERIP 54.159.96.31 --WORKERS 1 --WORKERIP root@35.224.184.36

Command Used in Department Server:

docker run -v "/var/run/docker.sock:/var/run/docker.sock:rw" -v "/home/chinthaka/.ssh:/root/.ssh" -v "/tmp:/tmp" -v "/var/tmp/jasminegraph-localstore:/var/tmp/jasminegraph-localstore" -v "/var/tmp/jasminegraph-aggregate:/var/tmp/jasminegraph-aggregate" -v "/home/chinthaka/databases/metadb:/home/ubuntu/software/jasminegraph/metadb" -v "/home/chinthaka/databases/performancedb:/home/ubuntu/software/jasminegraph/performancedb" -p 7777:7777 -p 7778:7778 jasminegraph --MODE 1 --MASTERIP 10.8.100.245 --WORKERS 4 --WORKERIP 10.8.100.245 --ENABLE_NMON false

Building and Running JasmineGraph Docker Version on a Laptop

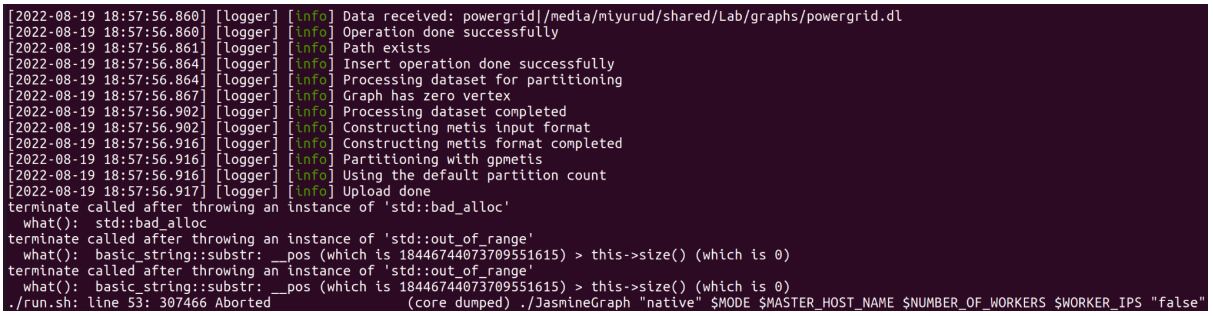
First clone the JasmineGraph repository
Next, change directory to docker and issue the following command,

docker build -t jasmine_graph .

Remote Connection Issue

Error Troubleshooting Guide

ERROR 1:
If you have not set the path to gpmetis executable you may get an error like follows while executing the adgr command,



Debugging JasmineGraph Docker

To debug JasmineGraph, GDB remote debugging can be used.
Follow the following steps to enable debugging.

1. Build JasmineGraph docker image with `**DEBUG=true**` specified as a `**build-arg**`
   docker build --build-arg DEBUG=true -t jasminegraph .

2. Start the docker container following the instructions given in the README.md. When starting use **--DEBUG <debug-port>** and expose the port from the container.
   Example:
   **docker run **
   **-v "/var/run/docker.sock:/var/run/docker.sock:rw" **
   **-v "/root/.ssh:/home/user/.ssh" **
   **-v "/tmp:/tmp" **
   -v
   **"/var/tmp/jasminegraph-localstore:/var/tmp/jasminegraph-localstore" **
   -v
   **"/var/tmp/jasminegraph-aggregate:/var/tmp/jasminegraph-aggregate" **
   -v
   **"/home/ubuntu/jasminegraph/metadb:/home/ubuntu/software/jasminegraph/metadb" **
   -v
   **"/home/ubuntu/jasminegraph/performancedb:/home/ubuntu/software/jasminegraph/performancedb" **
   **-p 7777:7777 **
   **-p 7778:7778 **
   **-p 8888:8888 **
   **jasminegraph **
   **--MODE 1 **
   **--MASTERIP 192.168.56.103 **
   **--WORKERS 2 **

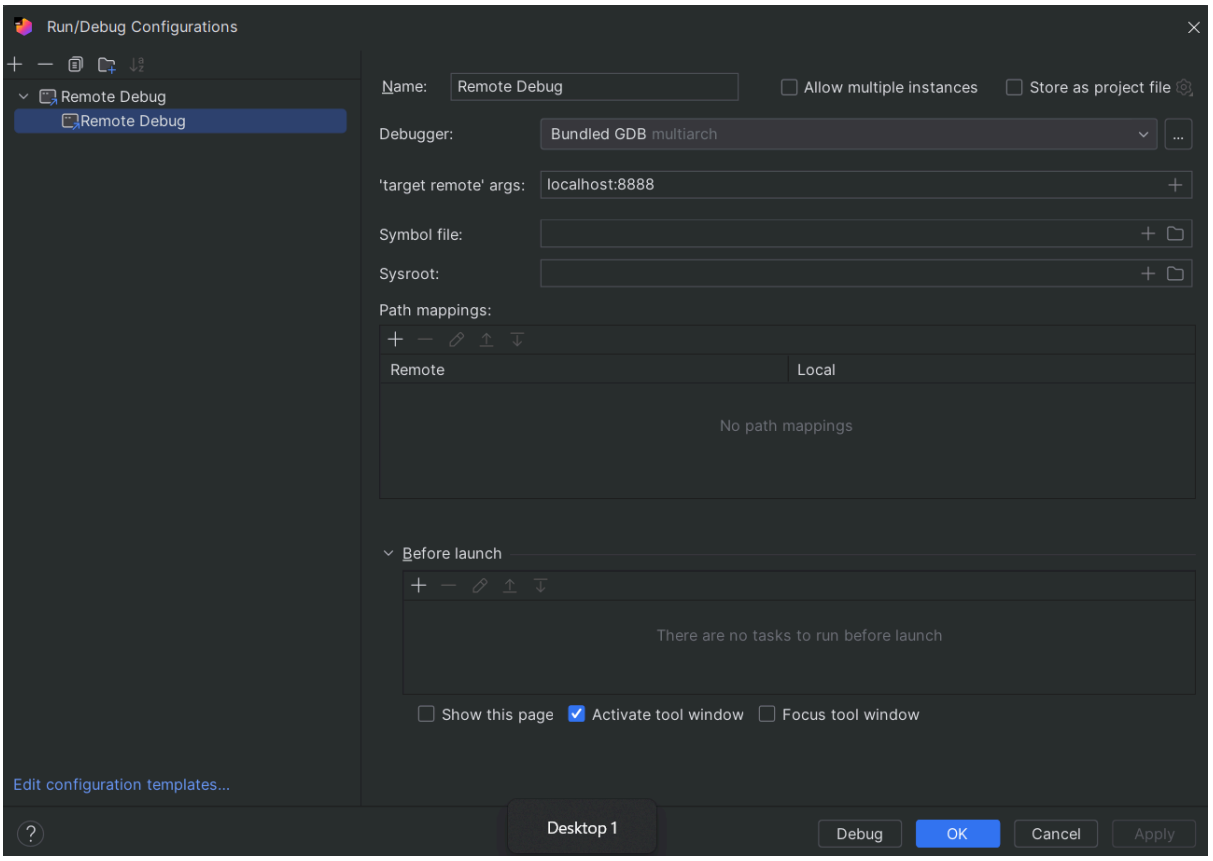
```
--WORKERIP 192.168.56.103 \
--ENABLE_NMON false \
--DEBUG 8888
```

Then it will start the docker image and wait for listening on the debug port specified as below.

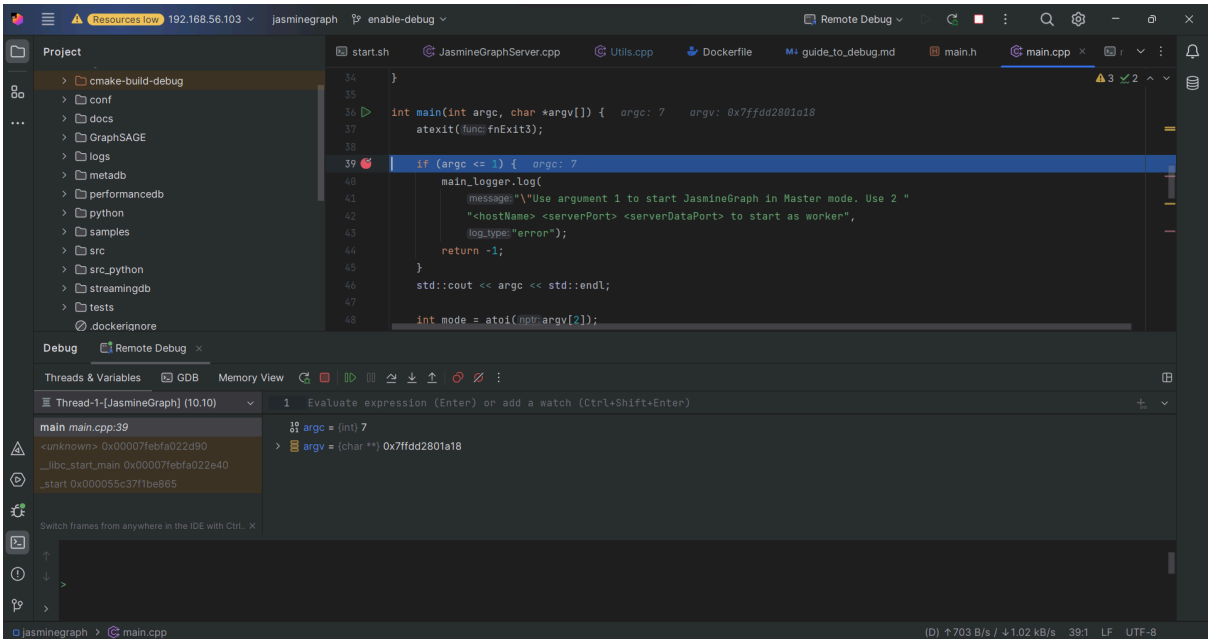
```
--MODE 1 // Optional to see the parameter:value result
--MASTERIP 192.168.56.103 // Optional to see the parameter:value result
--WORKERS 2 // Optional to see the parameter:value result
--WORKERIP 192.168.56.103 // Optional to see the parameter:value result
--ENABLE_NMON false // Optional to see the parameter:value result
--DEBUG 8888 // Optional to see the parameter:value result
gdbserver: Error disabling address space randomization: Operation not permitted
Process ./JasmineGraph created; pid = 10
Listening on port 8888
```

3. Set breakpoints in IDE where the debugger needs to halt the execution.
4. Connect the IDE to the port where the debugger is listening on.
    a. Instructions in Clion.

    Create a Run/Debug Configuration as shown in below



Start debugging. It will halt the execution on breakpoint specified as below.



Running JasmineGraph in Kubernetes Environment

1. Setup Local Kubernetes Environment.

Follow 1.1 or 1.2.
Prerequisite: System should have docker installed. Follow documentation in https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

1.1. Using Minikube

Follow instructions in https://minikube.sigs.k8s.io/docs/start/

Note: Creating an alias with the following command is required.

```
alias kubectl="minikube kubectl --"
```

1.2. Using K3S

Follow instructions in https://docs.k3s.io/quick-start

Quick setup guide:

1. Install k3s server with following command.
   ```
   curl -sfL https://get.k3s.io | sh -s - --docker
   --write-kubeconfig-mode=644
   ```

Note: Following options are used in order to

- **`--docker -`** Enable the access to the local docker image repository
- **`--write-kubeconfig-mode=664 -`** Enable executing kubectl commands without the sudo access

2. To add more nodes (VMs or Physical hosts) they need to be in the same network.
   ```
   curl -sfL https://get.k3s.io | K3S_URL=https://myserver:6443
   K3S_TOKEN=mynodetoken sh -s - --docker
   --write-kubeconfig-mode=644
   ```

   `K3S_URL` – myserver need to be replaced with IP address of the k3s server node.
   `K3S_TOKEN -` Stored on `/var/lib/rancher/k3s/server/node-token`

3. Create an alias as follow.
   ```
   alias kubectl="k3s kubectl"
   ```

Starting JasmineGraph

1. Create necessary volumes by following command
   ```
   metadb_path="<path-to-metadb>" \
   performancedb_path="<path-to-performancedb>" \
   data_path="<path-to-data-files>" \
   log_path="<path-to-store-log-files>" \
   envsubst <"k8s/volumes.yaml" \
   | kubectl apply -f -
   ```

2. Start the jasminegraph master
   ```
   kubectl apply -f "k8s/master-deployment.yaml"
   ```

3. Get the jasminegraph master service URL
   ```
   kubectl get services
   ```

```
NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                         AGE
kubernetes                    ClusterIP   10.43.0.1       <none>        443/TCP                         34d
jasminegraph-master-service   NodePort    10.43.217.215   <none>        7777:30001/TCP,7778:30002/TCP   2s
jasminegraph-worker0-service  ClusterIP   10.43.16.205    <none>        7777/TCP,7778/TCP               2s
jasminegraph-worker1-service  ClusterIP   10.43.216.129   <none>        7777/TCP,7778/TCP               2s
```

4. Connect to the jasminegraph master using a telnet client.
   ```
   telnet 10.43.217.215 7777
   ```