BigDataSDNSim User Manual

Table of Contents

1 Explanation of BigDataSDNSim

1.1 What is BigDataSDNSim?

BigDataSDNSim is a discrete-event simulation tool designed to enable the modeling and simulating of big data management systems (YARN), MapReduce programming models, and SDN-enabled networks within cloud computing environments. It is the first tool that models and simulates the three merging technologies (MapReduce-BDMS, SDN, cloud) in a single simulated environment. The simulator is capable of capturing the key functions, characteristics, and behaviors of SDN-enabled MapReduce computing environment. It is also capable of modeling the functionalities of MapReduce applications in line with mimicking diverse SDN capabilities and interactions with BDMS systems in a seamless manner.

BigDataSDNSim provides abstract layers for enforcing new policy-based solutions. For instance, a MapReduce application might be located on different data center servers and/or racks due to insufficient resource capacity in a single server/rack. Such distribution logic requires a proper modeling of abstractions and interfaces so that the users of our tool can seamlessly deploy their scheduling policies for MapReduce server-rack placement. In another example, the same MapReduce application might require special QoS requirements (e.g. traffic prioritization, policy-based routing mechanisms) and excessive data transmission on the network layer from one server to another. BigDataSDNSim is modeled to seamlessly provide easy deployments of QoS requirements on behalf for every MapReduce application in SDN-enabled cloud environments.

For further details of BigDataSDNSim, please refer to our paper entitled "BigDataSDNSim: A Simulator for Analyzing Big Data Applications in Software-Defined Cloud Data Centers".

1.2 Unique Features of BigDataSDNSim

BigDataSDNSim differs from other simulation tools in supporting a holistic simulation framework that simulates MapReduce applications, BDMS, and SDN-related networks in cloud-based environments. In particular, BigDataSDNSim is capable of modeling and simulating:

- A generic big data approach for executing different big data programming models (e.g. MapReduce, Stream) simultaneously;
- MapReduce applications within big data cluster management (BDMS), which is one of the key prominent framework for running different big data models;
- The behaviors and features of SDN dynamic networks coupled with the coordination and interaction with MapReduce applications within cloud environments;
- Dynamic routing mechanisms based on graph theory to enable any type of network topology to be seamlessly simulated;
- Modeling several policies for SDN, MapReduce, and VM within cloud data centers for multilevel optimization**.**

2 Getting Started

2.1 Lifecycle of BigDataSDNSim

The functionalities of the BigDataSDNSim is classified into four phases: building a required infrastructure, establishing requested MapReduce application(s), carrying out task processing and data transmission, and finally reporting the results of every MapReduce application. The infrastructure is built by parsing a configuration file provided in a JSON format. Once BigDataSDNSim obtains the file, it initiates the corresponding objects of hosts, switches, and network links. At the same time, it establishes the required components such as SDN controller and resource manager along with building a required network topology.

2.2 System and Software Requirements

- Operating System: Windows, Linux or Mac OS.
- CPU: 1-GHz processor or equivalent (Minimum).
- RAM: 2GB (Minimum).
- Java Platform: JDK version 11+ (recommended)
- Any IDE for Java programming language such as Eclipse or NetBeans

2.3 Download BigDataSDNSim

BigDataSDNSim can be downloaded from https://github.com/kalwasel/BigDataSDNSim

2.4 Directory Structure of BigDataSDNSim

The structure of BigDataSDNSim framework is defined as follows:

- BigDataSDNSim/
- examples/ -- Contains examples of MapReduce applications in SDN-enabled Clouds
- sources/ -- Contains the source code of BigDataSDNSim
- inputFiles/ -- Contains the required files to be submitted to BigDataSDNSim
- outputFiles/ -- Contains all the output results of BigDataSDNSim

2.5 Main Packages of BigDataSDNSim

BigDataSDNSim is mainly developed using the following package list:

- org.cloudbus.cloudsim.bigdatasdn.bdms (Pkg_1)
- org.cloudbus.cloudsim.bigdatasdn.mapreduce (Pkg_2)
- org.cloudbus.cloudsim.bigdatasdn.bdms.polocies (Pkg_3)

Pkg_1 contains classes that models the behaviors and characteristics of big data management systems and SDN. Pkg_2 contains classes that models the behaviors and characteristics of MapReduce models. Pkg_3 contains a list of policies, such as SDN routing policy. Figure 2.1 shows the packages and their classes in detail.

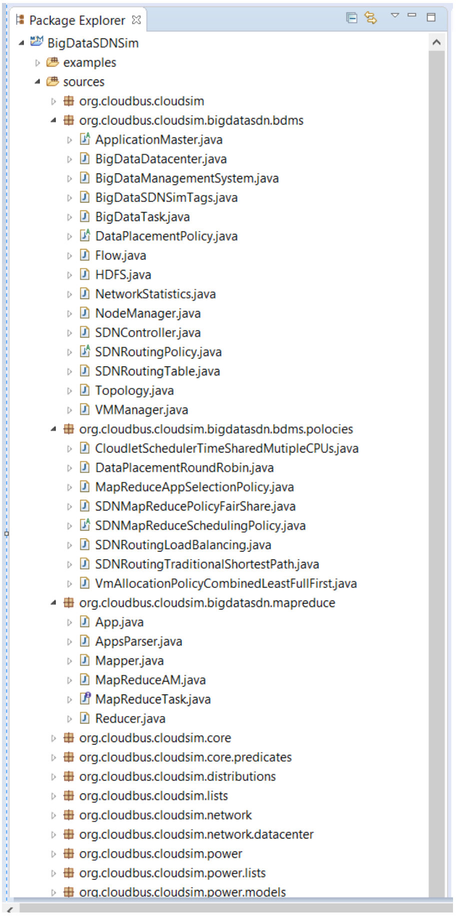

Figure 2.1

2.6 Setup BigDataSDNSim

Prior to use BigDataSDNSim, you need to import and configure the project properly. Here, we use Eclipse to illustrate how to setup the BigDataSDNSim project. The project is based on Maven. The main steps are given as follows:

Step 1:

- Install Eclipse from https://www.eclipse.org/downloads/
- Install Maven on Eclipse, follow the steps given in https://www.eclipse.org/m2e/

Step 2: Import BigDataSDNSim as a Maven project by Opening Eclipse -> selecting File -> and selecting import (see Figure 2.2)

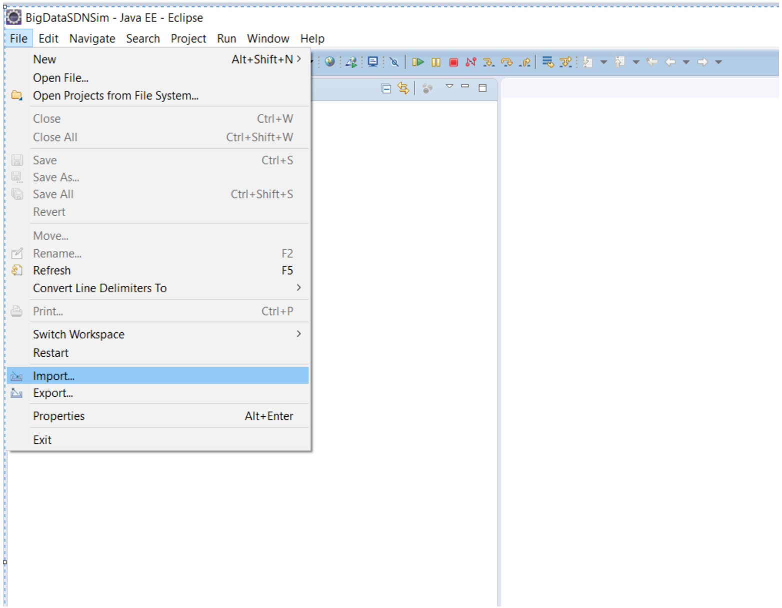

Figure 2.2

Step 3: Select Maven -> select Existing Maven Projects (see Figure 2.3)

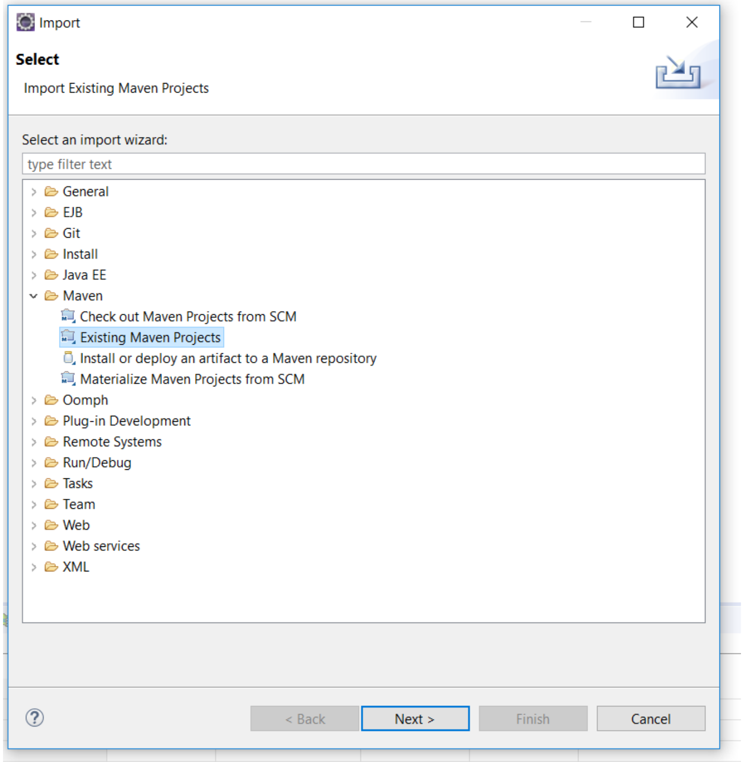

Figure 2.3

Step 4: Select the folder corresponding to BigDataSDNSim project. Next, click on Finish (see Figure 2.4)

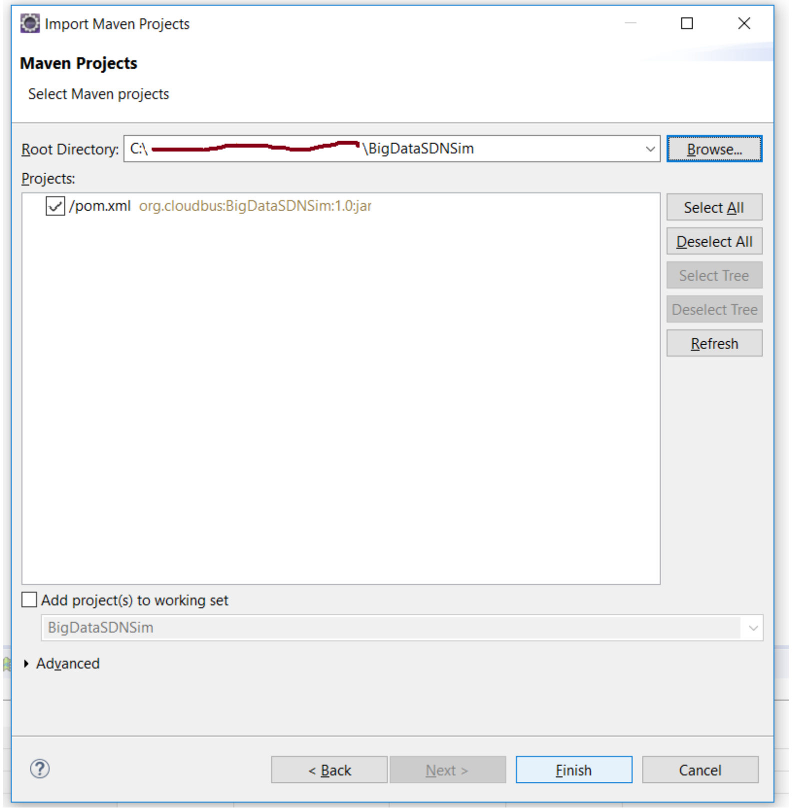

Figure 2.4

Step 5: Right click on BigDataSDNSim project and click on Update Project that found under Maven option (see Figure 2.5)

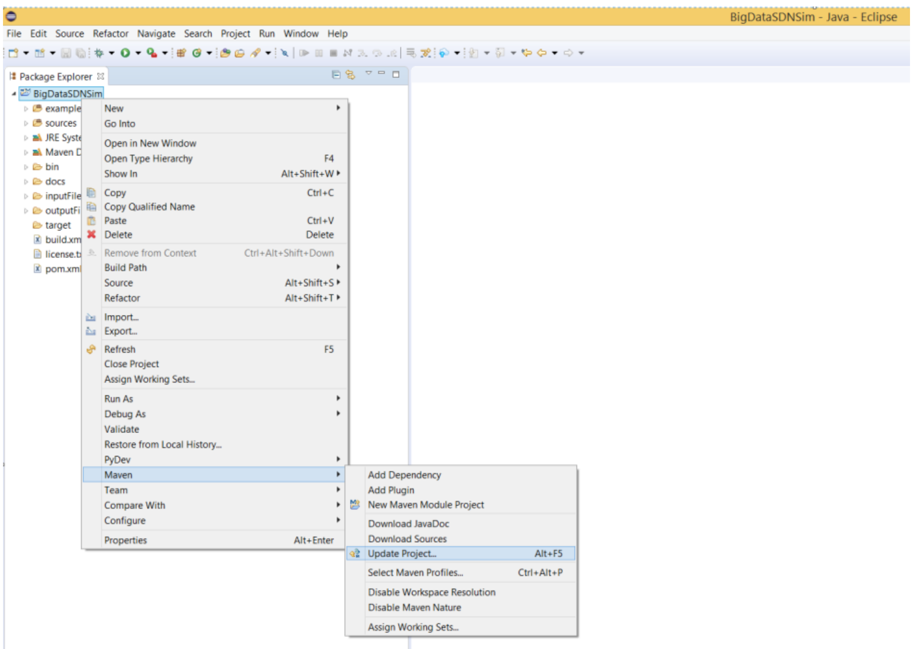

Figure 2.5

Step 6: Right click on BigDataSDNSim project and click on Maven install that found under Run As option (see Figure 2.6)

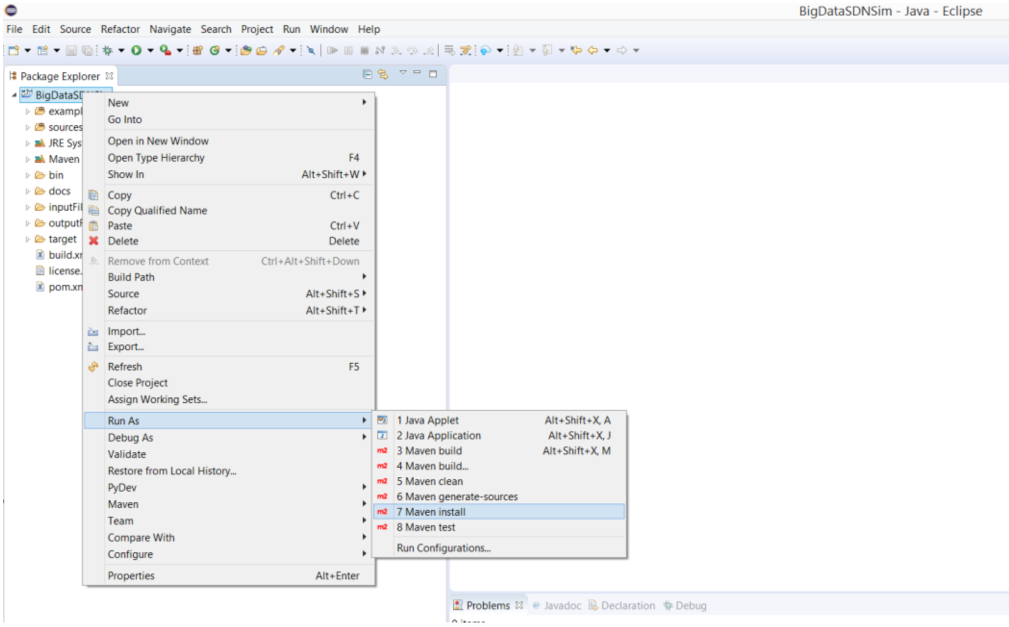

Figure 2.6

When Maven successfully builds BigDataSDNSim in your Eclipse, you will see “BUILD SUCCESS” as shown in Figure 2.7. At this point, you have successfully built and configured BigDataSDNSim.

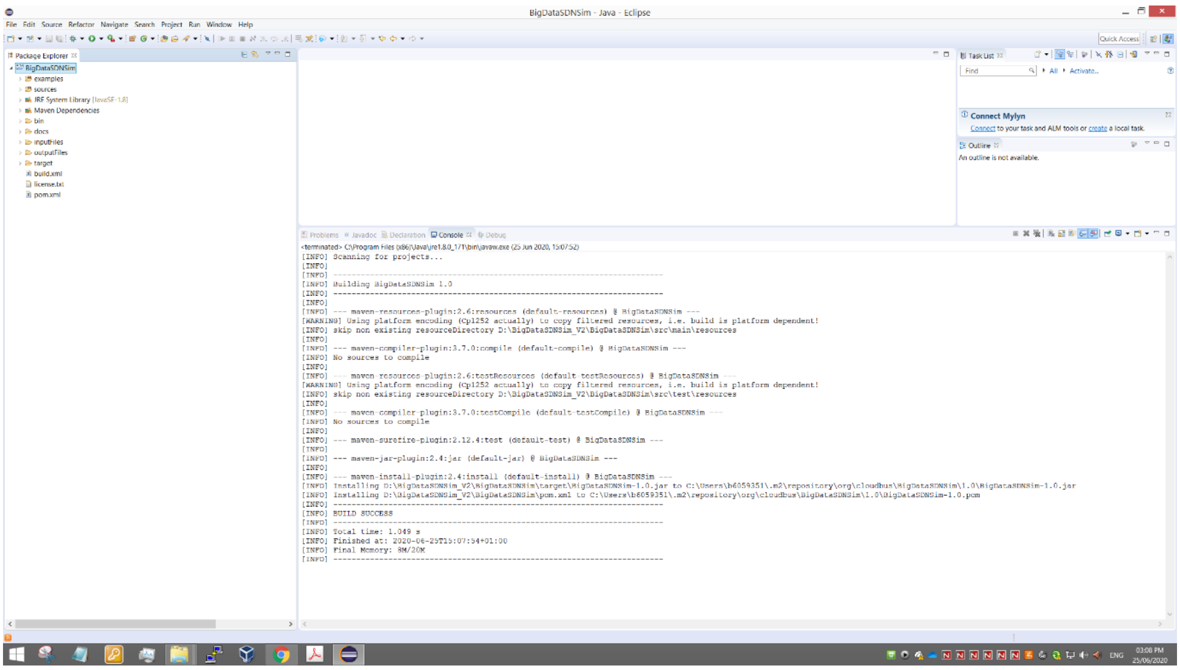

Figure 2.7

3 Simulation configuration

Before starting the actual simulation, you have to configure the infrastructure of every data center. The infrastructure of every data center can be easily configured using a configuration file named Datacenter_Configuration in the *inputFiles* folder. The parameters of the configuration file is illustrated in Table 3.1, which is defined in a JSON format. These parameters are read by the main program during initialization which configures the environment of BigDataSDNSim accordingly. A snapshot of the confirmation file is given in Figure 3.1. It shows an example of metrics used in the examples, given later.

Table 3.1 User-defined of physical configuration

Entity	Parameter	Description
nodes (hosts)	name	Node name (e.g. host1, host2)
	type	Node type (host)
	pes	Number of CPUs
	mips	Million Instructions Per Second (MIPS) rate of the host
	ram	Size of the host's memory
	storage	Size of the host's storage
nodes (switches)	iops	I/O rate of the switch
	name	Node name (e.g. core1, aggregate1)
	type	Node type (switch)
links	source	A node connected to one end of the undirected link
	destination	A node connected to the other end of the undirected link
	latency	The latency of the link
	bw	The bandwidth of the link

```
Datacenter_Configuration.json
{
  "nodes" : [
    {
      "name": "host1",
      "type" : "host",
      "pes" : 8,
      "mips" : 1250,
      "ram" : 32750,
      "storage" : 8000000000000
    },
     {
      "name": "host2",
      "type" : "host",
      "pes" : 8,
      "mips" : 1250,
      "ram" : 32750,
      "storage" : 8000000000000
    },
    {
      "name": "host3",
       "type" : "host",
      "pes" : 8,
      "mips" : 1250,
      "ram" : 32750,
      "storage" : 8000000000000
    },
```

```
    {
      "iops":1000000000,
      "name": "core1",
      "type" : "core"
    },
   {
      "iops":1000000000,
      "name": "core2",
      "type" : "core"
    },
        {
      "iops":1000000000,
      "name": "core3",
      "type" : "core"
    },
        {
      "iops":1000000000,
      "name": "core4",
      "type" : "core"
    },
    {
      "iops":1000000000,
      "name": "aggregate1",
      "type" : "aggregate"
    },
```

```
"links" : [
  { "source" : "core1" , "destination" : "aggregate1" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core1" , "destination" : "aggregate3" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core1" , "destination" : "aggregate5" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core1" , "destination" : "aggregate7" , "latency" : 1.0 , "bw" : 1000},

  { "source" : "core2" , "destination" : "aggregate1" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core2" , "destination" : "aggregate3" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core2" , "destination" : "aggregate5" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core2" , "destination" : "aggregate7" , "latency" : 1.0 , "bw" : 1000},

  { "source" : "core3" , "destination" : "aggregate2" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core3" , "destination" : "aggregate4" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core3" , "destination" : "aggregate6" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core3" , "destination" : "aggregate8" , "latency" : 1.0 , "bw" : 1000},

  { "source" : "core4" , "destination" : "aggregate2" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core4" , "destination" : "aggregate4" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core4" , "destination" : "aggregate6" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "core4" , "destination" : "aggregate8" , "latency" : 1.0 , "bw" : 1000},

  { "source" : "aggregate1" , "destination" : "edge1" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "aggregate1" , "destination" : "edge2" , "latency" : 1.0 , "bw" : 1000},

  { "source" : "aggregate2" , "destination" : "edge1" , "latency" : 1.0 , "bw" : 1000},
  { "source" : "aggregate2" , "destination" : "edge2" , "latency" : 1.0 , "bw" : 1000},
```

Figure 3.1

3.1 VM configuration

The configurations of VMs are handled by **VMManager.java** class (see Figure 3.2). We have defined different types of VMs. You can easily defined your own VM types in this class. Figure 3.2 illustrates an overview of the VMManager.java class. The required parameter for every VM is shown in Table 3.2.

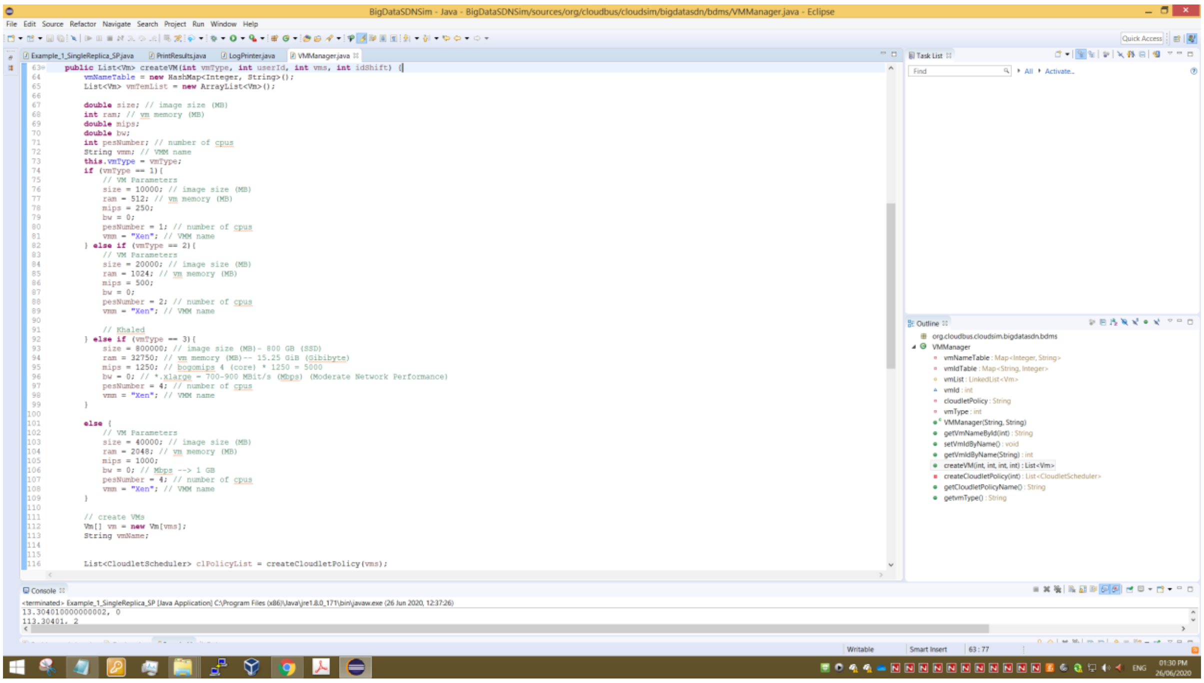

Figure 3.2

Table 3.2 The parameter description of VMs

Parameter	**Description**
size	The storage size of the VM
ram	Required memory ssize of the VM
mips	Million Instructions Per Second (MIPS) rate of the VM
bw	Bandwidth size of the VM

4 Simulation examples

Before starting the actual simulation, the processing and transmission logic of every MapReduce application must be provided. For the given examples, we have four different MapReduce logics, which is given in different CSV files. For every logic, we run two examples: one with a shortest path traditional network (SP) and the other with SDN load balancing. All the files contain similar parameters to be filled. The description of every parameter is given in Table 4.1. The file names of MapReduce logics are given as follows:

- SingleReplica_Workload (see Figure 4.1)
- ThreeReplica_Workload (see Figure 4.2)
- MultipleApps_ThreeReplica_Workload (see Figure 4.3)
- Priority_MultipleApps_ThreeReplica_Workload (see Figure 4.4)

Table 4.1 The parameter description of MapReduce applications

Parameter	**Description**
repliaction	The number of replication for every MapReduce application
appType	The application type, which is set to “MapReduce“
appID	The ID of every application
startTime	The start time of every application
CPU_Num	The required number of CPUs for every application
mapInput_GB	The data size given to every mapper in GB
Map_MI	The processing size in million instruction (MI) for every mapper
shuffleData_GB	The data input size for every reducer in GB
Reduce_MI	The processing size in million instruction (MI) for every reducer
reduceDataOutput_GB	The data output size for every reducer in GB
numberOfMappers	The number of required mappers
numberOfReducers	The number of required reducers
priority	Indicate if the MapReduce application is prioritized. The values of priority is defined as follows: 0 → no priority 1 → low priority 2 → medium priority 3 → high priority
hdfsBlockSize_MB	The data size of every MapReduce block in MB

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	repliaction	appType	appID	startTime	CPU_Num	mapInput_GB	Map_MI	shuffleData_GB	Reduce_MI	reduceDataOutput_GB	numberOfMappers	numberOfReducers	priority	hdfsBlockSize_MB
2	1	MapReduce	1	0	4	15	500000	0.75	150000	1	16	3	0	950

Figure 4.1 SingleReplica_Workload

1	repliaction	appType	appID	startTime	CPU_Num	mapInput_GB	Map_MI	shuffleData_GB	Reduce_MI	reduceDataOutput_GB	numberOfMappers	numberOfReducers	priority	hdfsBlockSize_MB
2	3	MapReduce	1	0	4	15	500000	0.75	150000	1	16	3	0	950

Figure 4.2 ThreeReplica_Workload

1	repliaction	appType	appID	startTime	CPU_Num	mapInput_GB	Map_MI	shuffleData_GB	Reduce_MI	reduceDataOutput_GB	numberOfMappers	numberOfReducers	priority	hdfsBlockSize_MB
2	3	MapReduce	1	0	4	15	500000	0.75	150000	1	16	3	0	950

Figure 4.3 MultipleApps_ThreeReplica_Workload

1	repliaction	appType	appID	startTime	CPU_Num	mapInput_GB	Map_MI	shuffleData_GB	Reduce_MI	reduceDataOutput_GB	numberOfMappers	numberOfReducers	priority	hdfsBlockSize_MB
2	3	MapReduce	1	0	4	15	500000	0.75	150000	1	16	3	1	950
3	3	MapReduce	2	0	4	15	500000	0.75	150000	1	16	3	1	950
4	3	MapReduce	3	0	4	15	500000	0.75	150000	1	16	3	2	950
5	3	MapReduce	4	0	4	15	500000	0.75	150000	1	16	3	2	950
6	3	MapReduce	5	0	4	15	500000	0.75	150000	1	16	3	3	950

Figure 4.4 Priority_MultipleApps_ThreeReplica_Workload

The list of examples is shown in Figure 4.5.

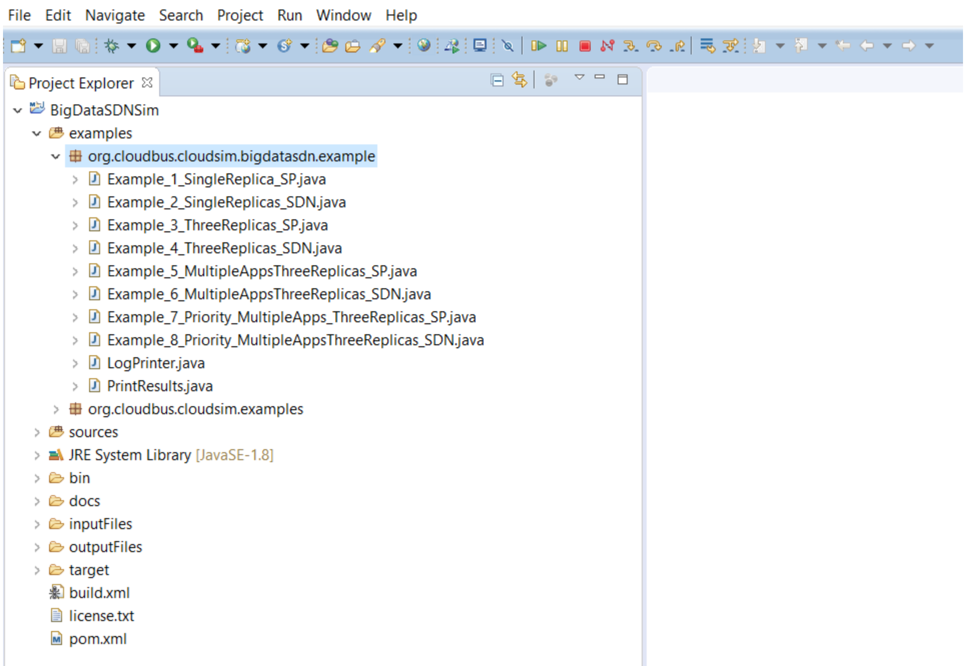

Figure 4.5

4.1 Policies

To run any example, you have to provide six policies. Figure 4.6 shows the list of policies already developed in BigDataSDNSim. The class of every example must contain the list of six policies. The following describes the purpose of every policy:

1. *VM placement:* It determines how to place VMs on a given host list.
2. *Application selection:* It is essential to be used to determine the selection criteria based on a given QoS (e.g. priority).
3. *HDFS replica placement:* It determines how to place HDFS replicas on elected VMs.
4. *VM-CPU scheduling:* It determines how to schedule MIPS of every VMs among mappers and reducers.
5. *Routing:* It is used to determine routes among VMs.
6. *Traffic:* It is used to control the sharing resources of a given network among MapReduce applications.

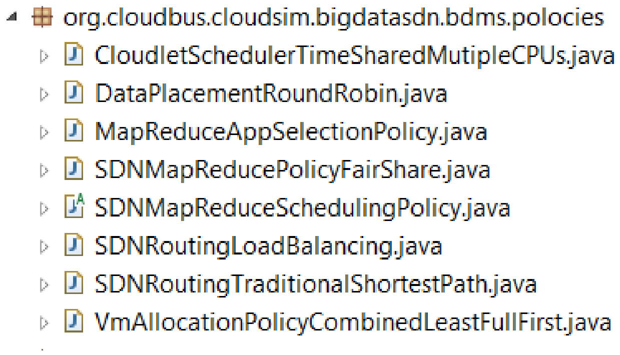

Figure 4.6

4.2 Output results of BigDataSDNSim

At the end of running every example, the output would be stored on result.txt located in the outputFiles folder. The results contains a lot of information. At the end of the result file, the results are structured as follows:

- SDN Network Results
- MapReduce Processing Outputs
- Mappers Outputs
- Reducers Outputs
- Forwarding Tables
- Host Power Consumption
- Switch Power Consumption
- Total Power Consumption

4.3 Running the first example (example 1)

Step 1: Select Example_1_SingleReplica_SP.java -> click on the small down arrow next to the play button and select Run Configurations (see Figure 4.7).

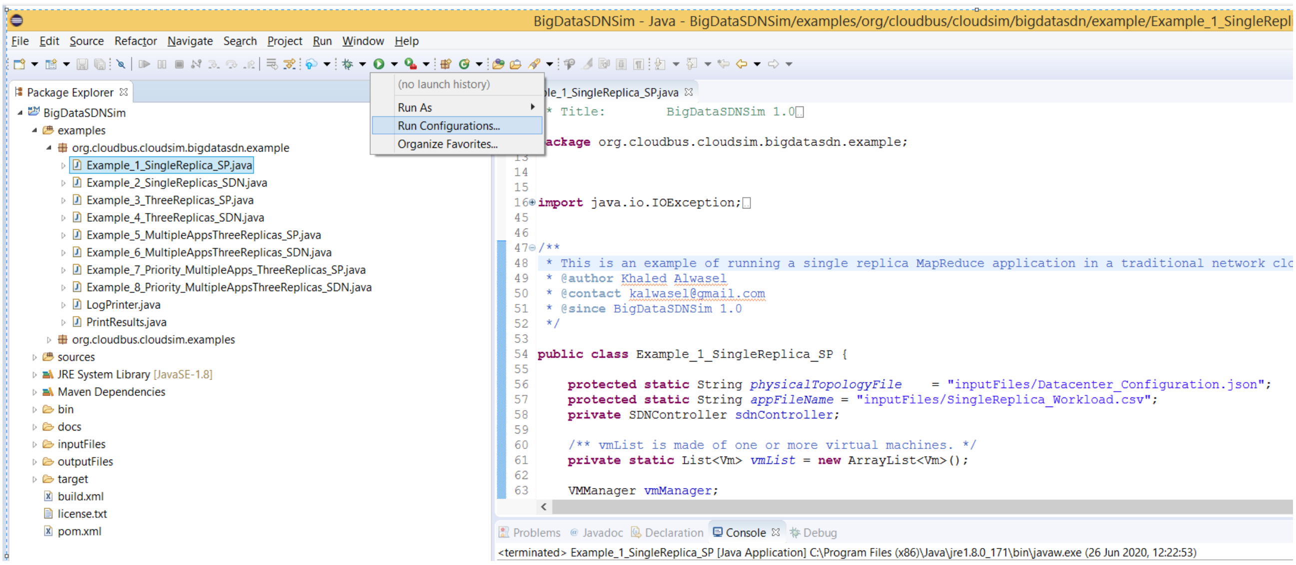

Figure 4.7

Step 2: Double click on Java Application and Eclipse will create the first example (see Figure 4.8).

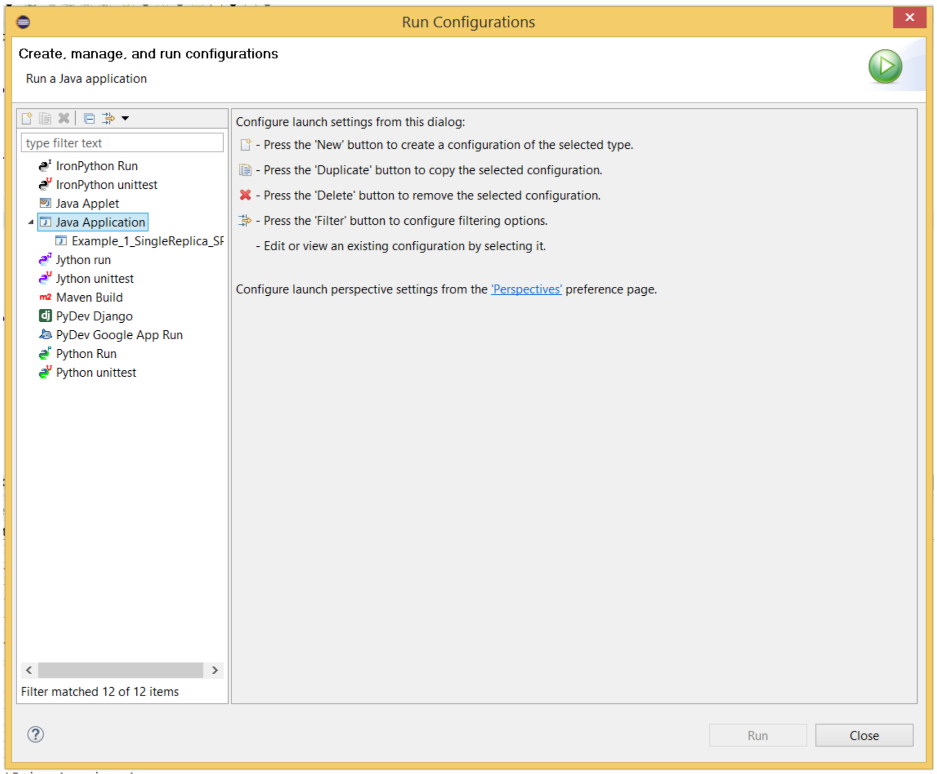

Figure 4.8

Step 2: Click on Common -> check mark Output File -> click workspace -> select BigDataSDNSim folder -> select outFiles folder -> select result.txt -> click Ok -> click Run (see Figure 4.9). This step will run the first example and store all the outputs on the result.txt file.

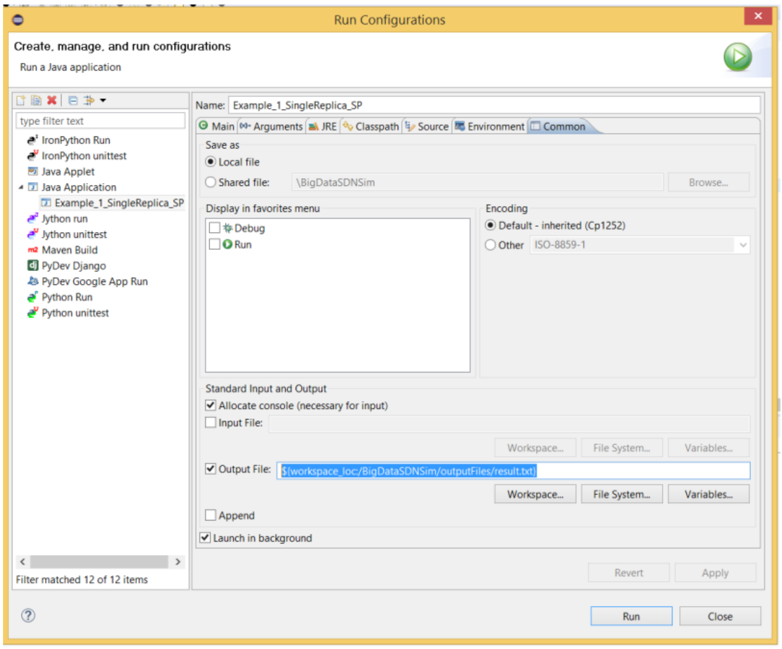

Figure 4.9

Figure 4.10 and Figure 4.11 show a sample of the result obtained by running the first example. The result is also stored in an excel file located in outputFiles folder. Similar steps given for the first example can be followed to execute the rest of the examples.

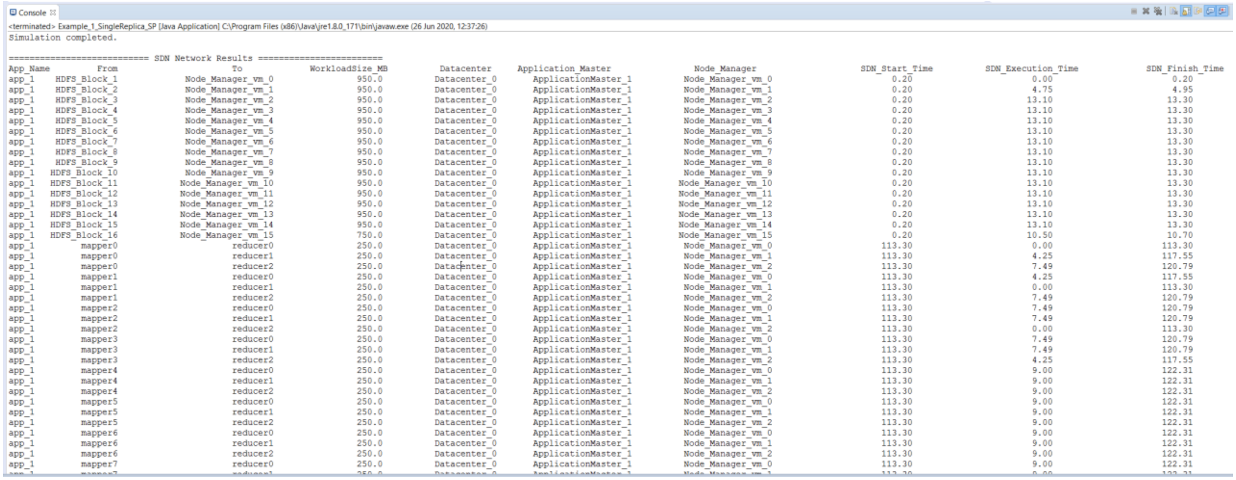

Figure 4.10

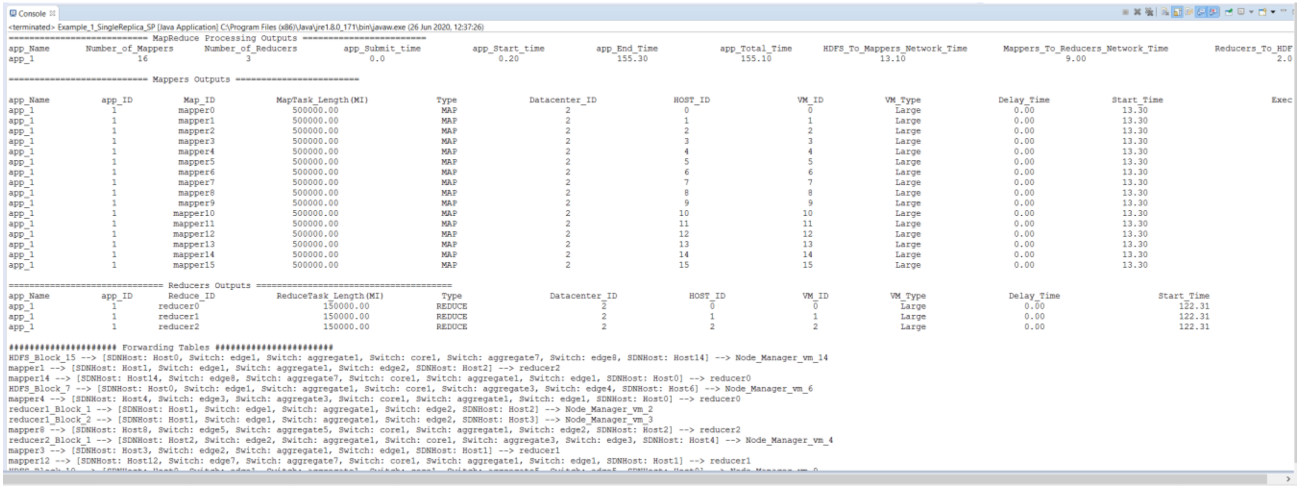

Figure 4.11

5 Contact

Please feel free to contact me if you need any further information at kalwasel@gmail.com
This manual is written by Khaled Alwasel under the supervision of Prof. Rajiv Ranjan.