

IoTSim-Stream User Manual

Contents

1	What is IoTSim-Stream	2
2	Unique Features.....	2
3	Getting Started.....	2
3.1	System and Software Requirements.....	2
3.2	Download IoTSim-Stream	2
3.3	Directory Structure of IoTSim-Stream	2
3.4	Setup IoTSim-Stream	3
3.5	Simulation Configuration: Stream processing system simulation	10
4	Simulating	11
4.1	Example of Linear Stream Graph Application	12
4.2	Example of Branching Stream Graph Applications (App2)	13
4.3	Example of Hybrid Stream Graph Applications (App3).....	15
5	GUI	17

1 What is IoTSim-Stream

IoTSim-Stream is an IoT Simulator for Stream processing on the big data that offers an environment to model complex stream graph applications in multicloud environment, where the large-scale simulation-based studies can be conducted to evaluate and analyse these applications. It leverages the features of CloudSim and integrating real-time processing model with workflow scheduling and execution to execute the modelled stream graph application in multicloud environment.

IoTSim-Stream supports the modelling of different patterns/structures of stream workflow applications, which are linear (a multi-stage application where each stage processes input stream generated by the previous stage and produces the output stream to the following stage), branching (an application with limited precedence constraints that splits data stream to perform different parallel processing and then combining the results for further analysing) and hybrid (a mix of linear and branching patterns).

For technical detail about IoTSim-Stream, please refer to our paper entitled “IoTSim-Stream: Modelling stream graph application in cloud simulation” that has been published in Future Generation Computer Systems, Volume 99, October 2019, Pages 86-105.

2 Unique Features

- Support modelling data incentive IoT-based applications using stream processing model (aka stream graph applications).
- Support modelling multicloud environment as an execution environment for stream graph application.
- Support user-defined resource provisioning and scheduling policies.
- Provide Graphical User Interface (GUI) to simplify configuration and running the simulations

3 Getting Started

3.1 System and Software Requirements

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

3.2 Download IoTSim-Stream

The simulation toolkit (IoTSim-Stream) can be downloaded from the below link. After downloading a zip file, you need to unzip IoTSim-Stream files.

<https://github.com/mutazb999/IoTSim-Stream>

3.3 Directory Structure of IoTSim-Stream

The structure of IoTSim-Stream package is as follows:

- IoTSim-Stream/
 - dependency.repository -- Jar dependencies (i.e. jar file for Cloudsim)
 - Sample_Stream_Workflows -- Sample stream graphs with different sizes
 - src/main/java/iotsimstream/examples/ -- Some examples of stream graph applications
 - src/main/java/iotsimstream/gui/ -- Graphical user interface

- `src/main/java/iotsimstream/` -- The source code of IoTSim-Stream
- `src/main/java/iotsimstream/schedulingPolicies/` -- Scheduling policies
- `src/main/java/iotsimstream/vmOffers/` -- VM offer classes
- `src/main/java/resources/` -- The simulation properties file of IoTSim-Stream

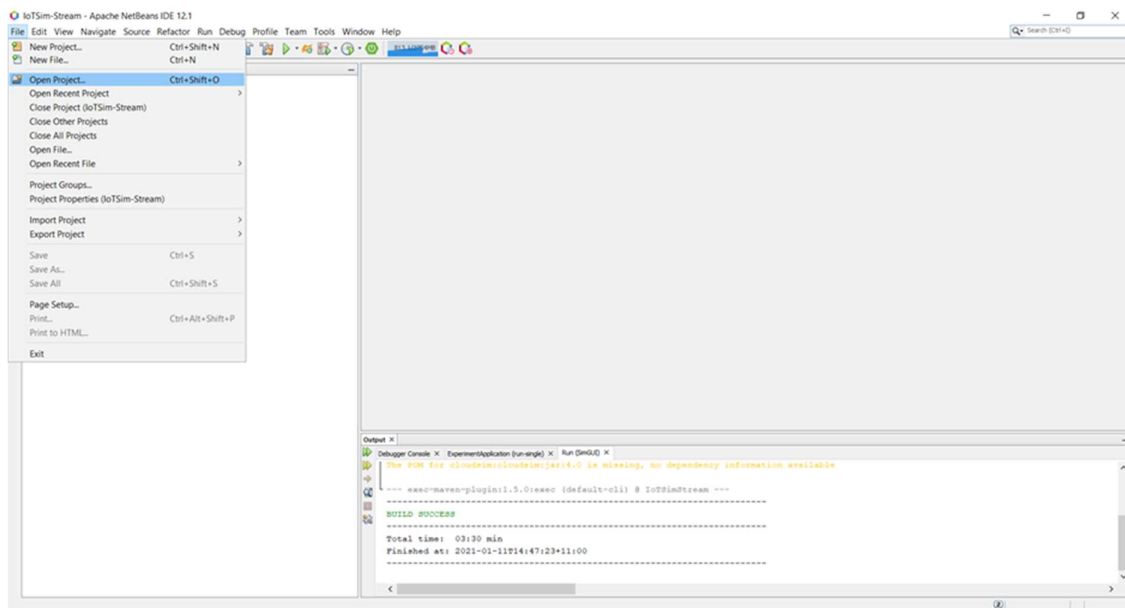
3.4 Setup IoTSim-Stream

Prior to use and work with IoTSim-Stream, you need to import and configure this project in the chosen IDE.

How to setup IoTSim-Stream project in Netbeans

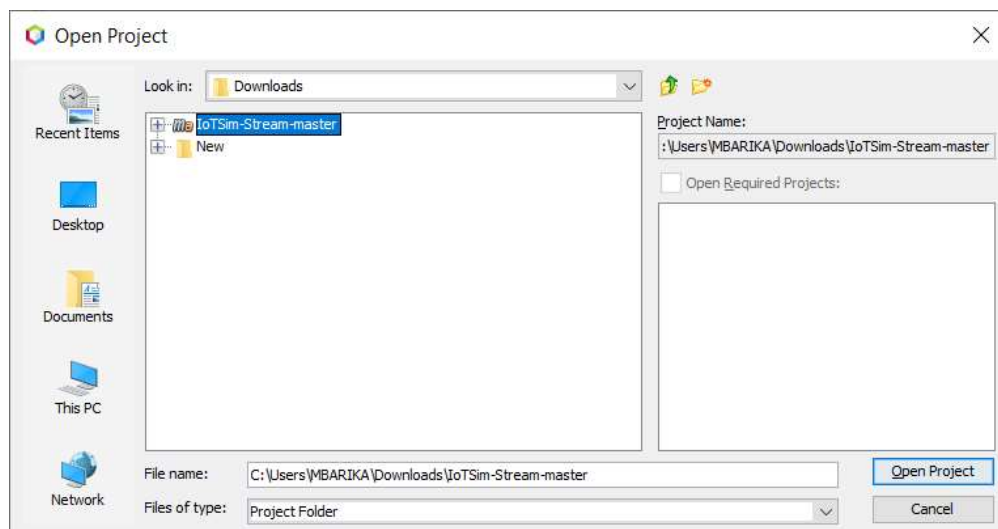
Step 1

Open NetBeans IDE -> File -> Open Project



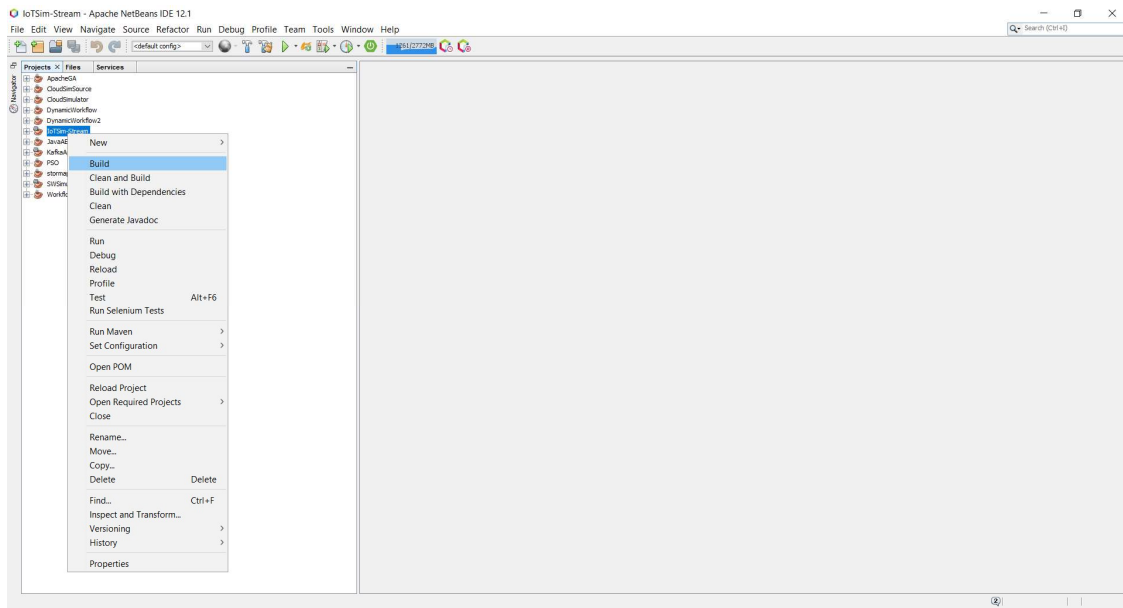
Step 2

Select the IoTSim-Stream folder, then click on Open Project

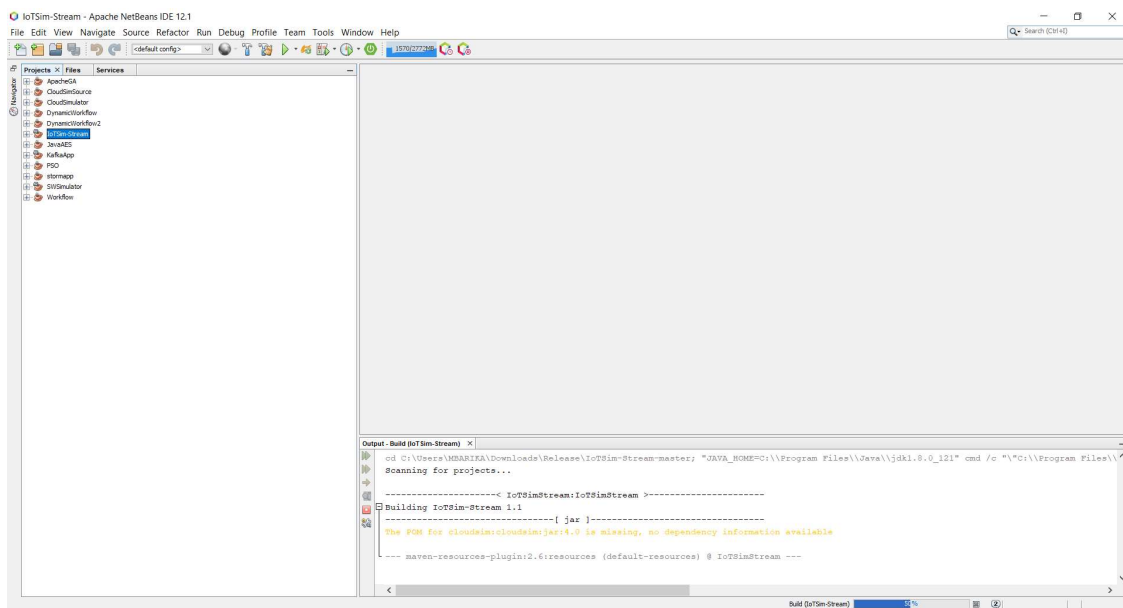


Step 3

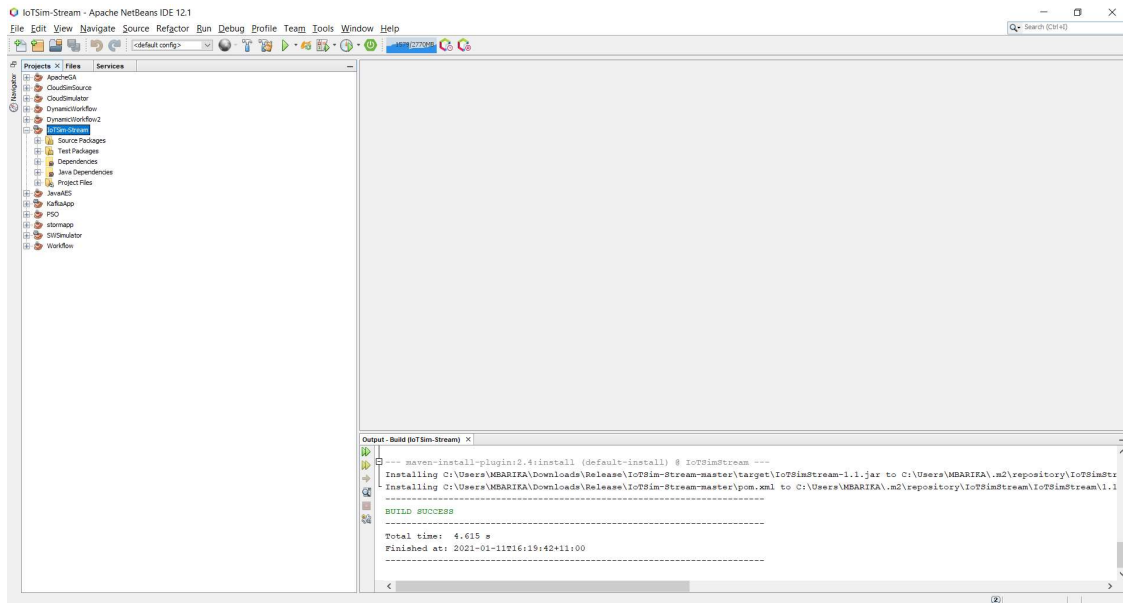
Right-click on IoTsim-Stream project and click on Build



Now, the build process is started.

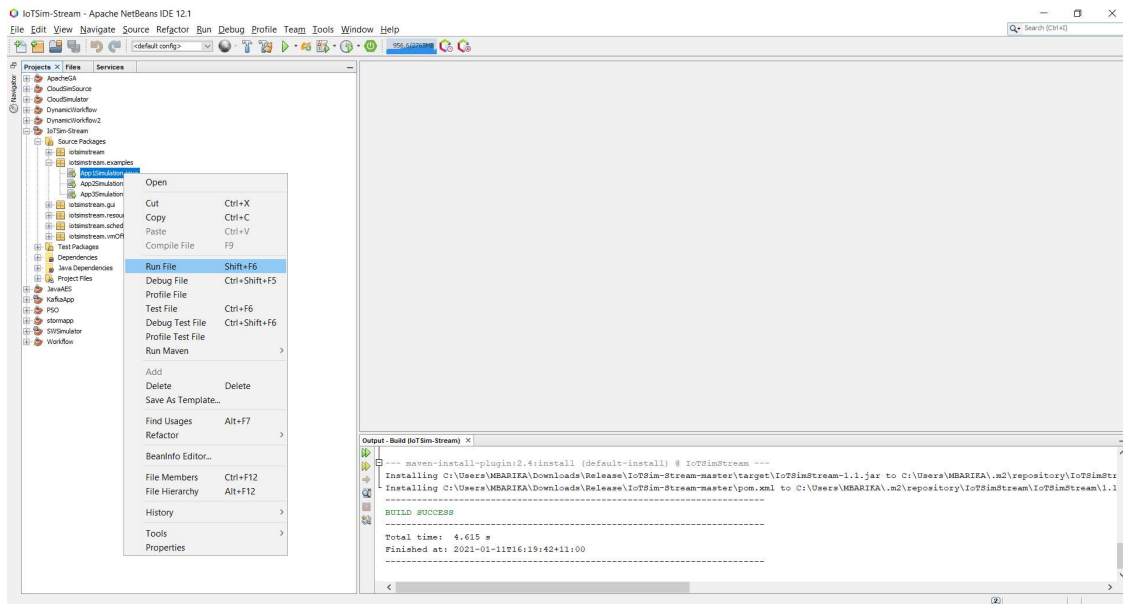


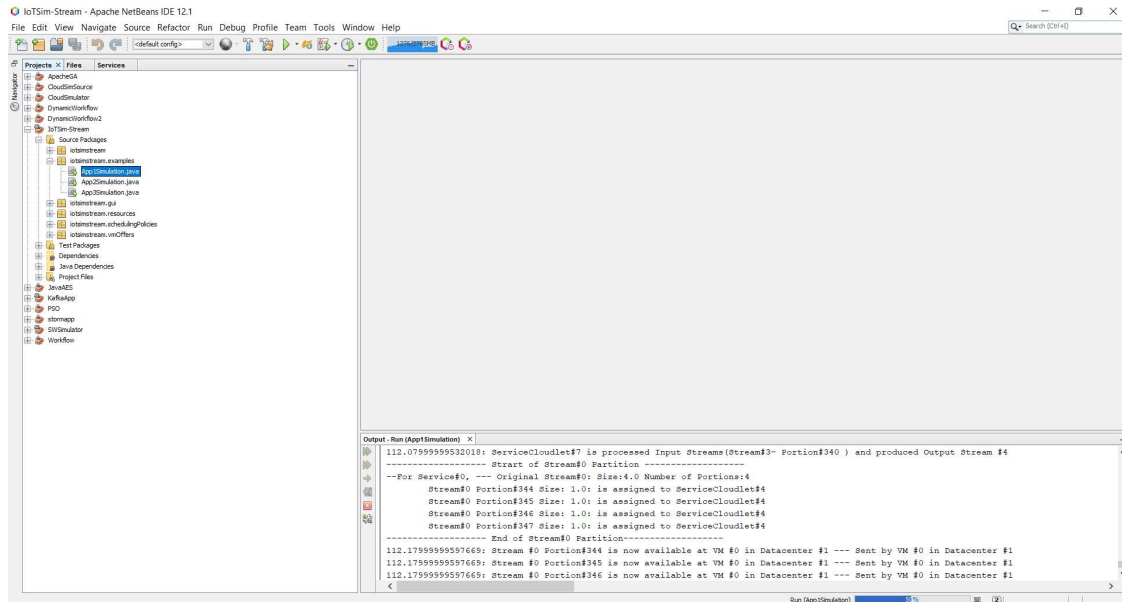
When the build process is completed, you will see “BUILD SUCCESS” as shown in the below screenshot. At this point, you successfully built and configured IoTSim-Stream.



Step 4

To run a simulation, you need to go to the examples package (iotsimstream.examples) and run any example provided. For instance, if you want to simulate App1, expand iotsimstream.examples package, right-click on “App1Simulation” and select “Run file”. The simulation process will be initialized and the simulation of App1 will begin.

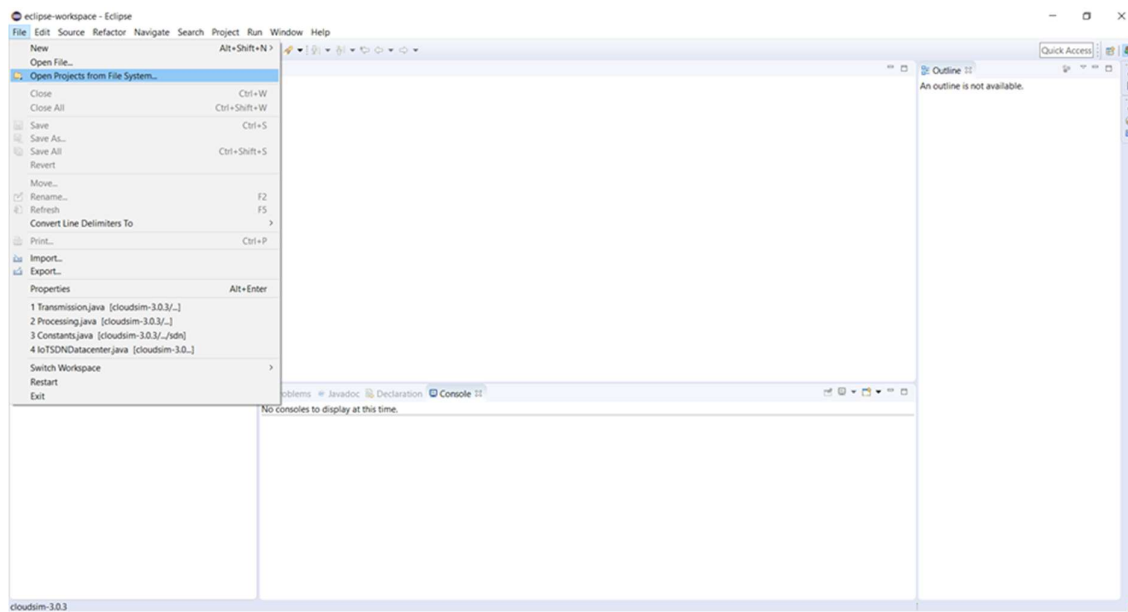




How to setup IoTSim-Stream project in Eclipse

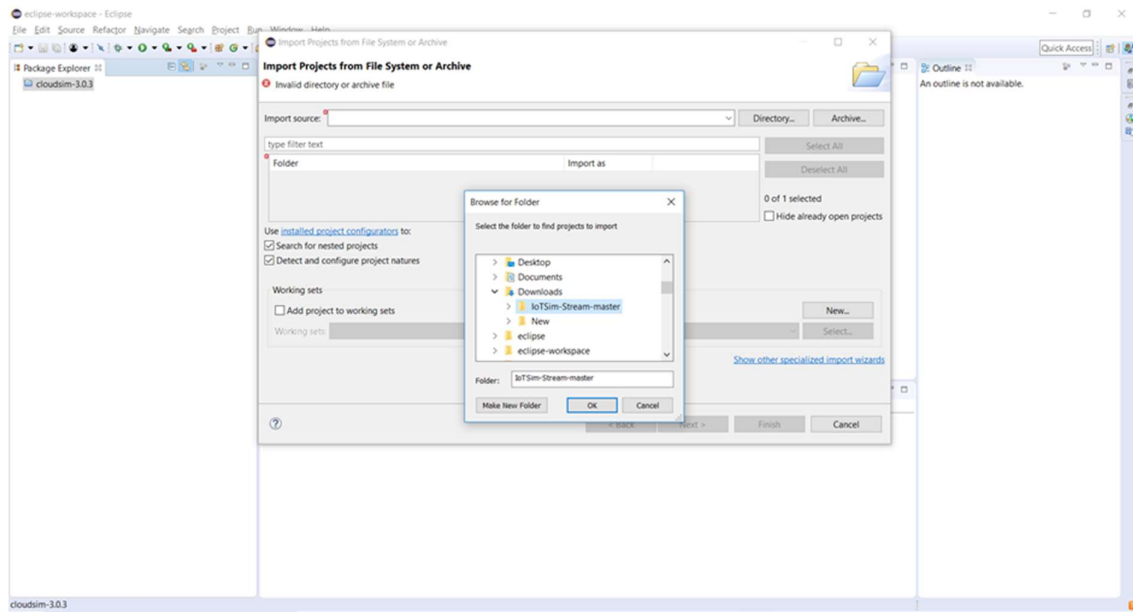
Step 1

Open Eclipse IDE -> File -> Open Projects From File System...

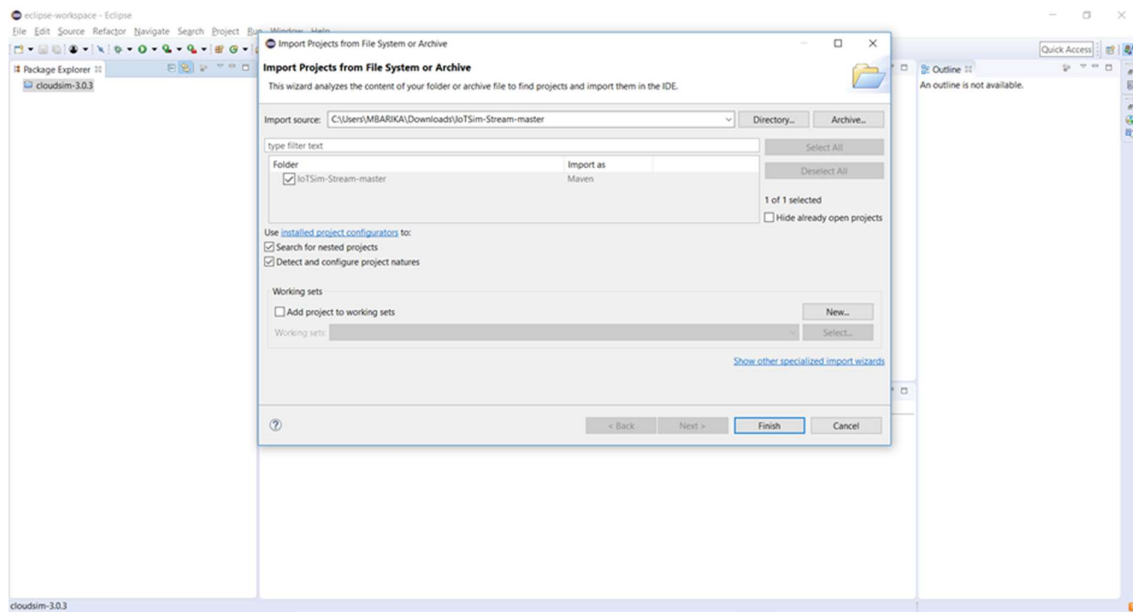


Step 2

Click on Directory and then select the folder corresponding to IoTsim-Stream. After that click on OK

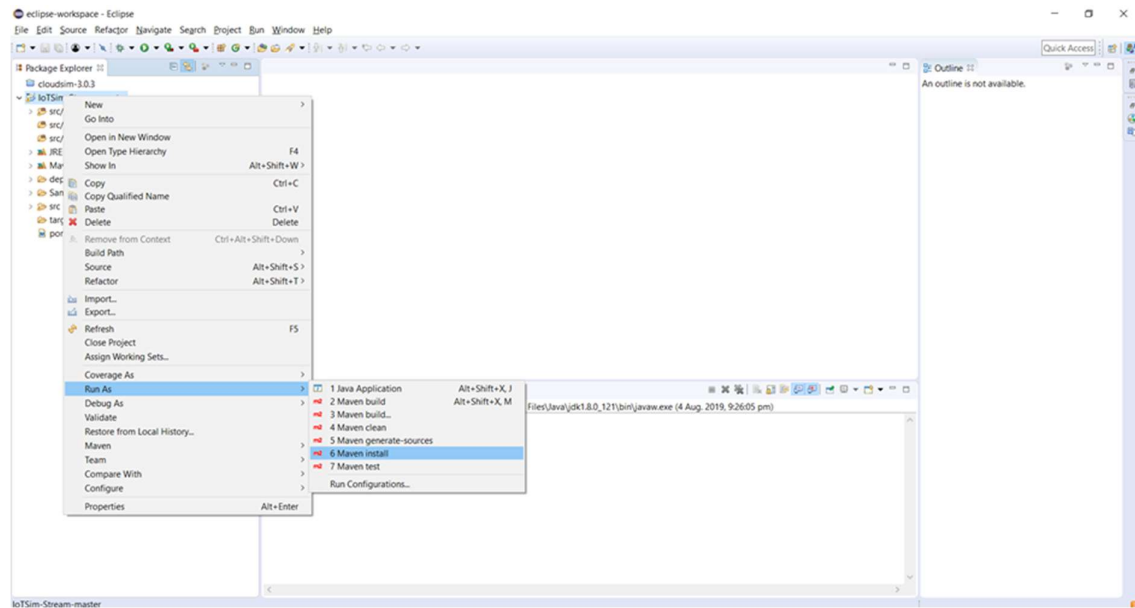


Now, click on Finish

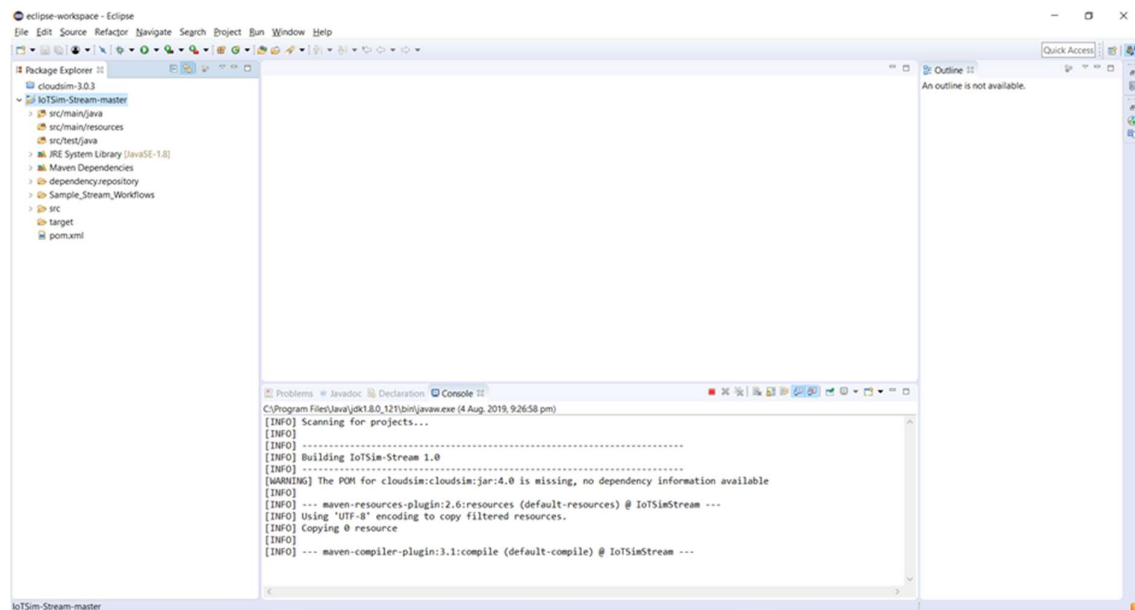


Step 3

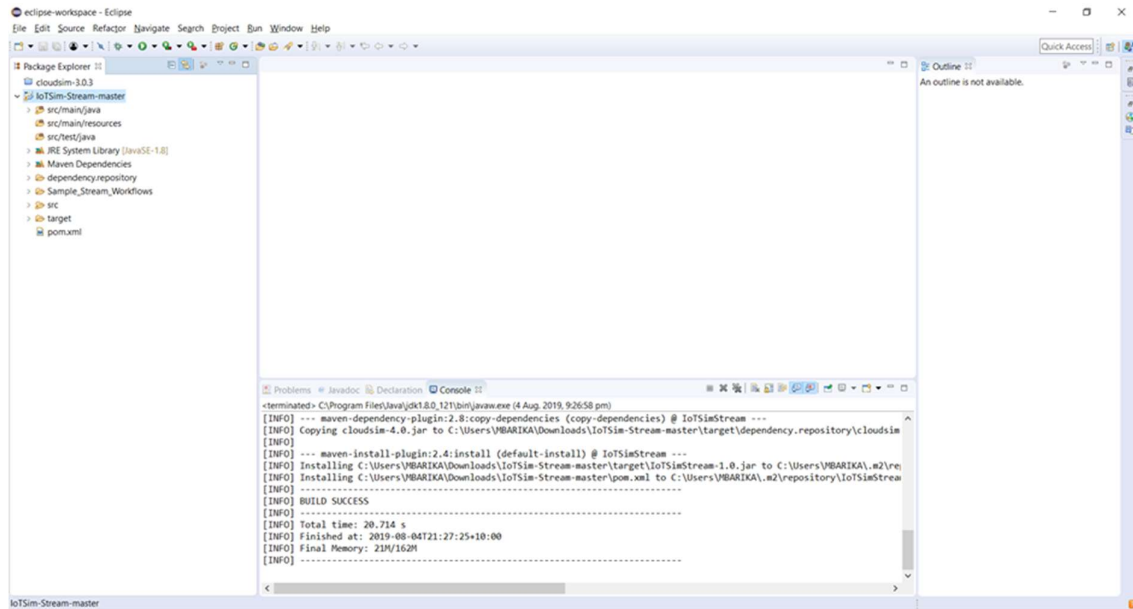
Right-click on IoTSim-Stream project and click on Maven install that found under Run As



Now, the installing project main artefact is started.

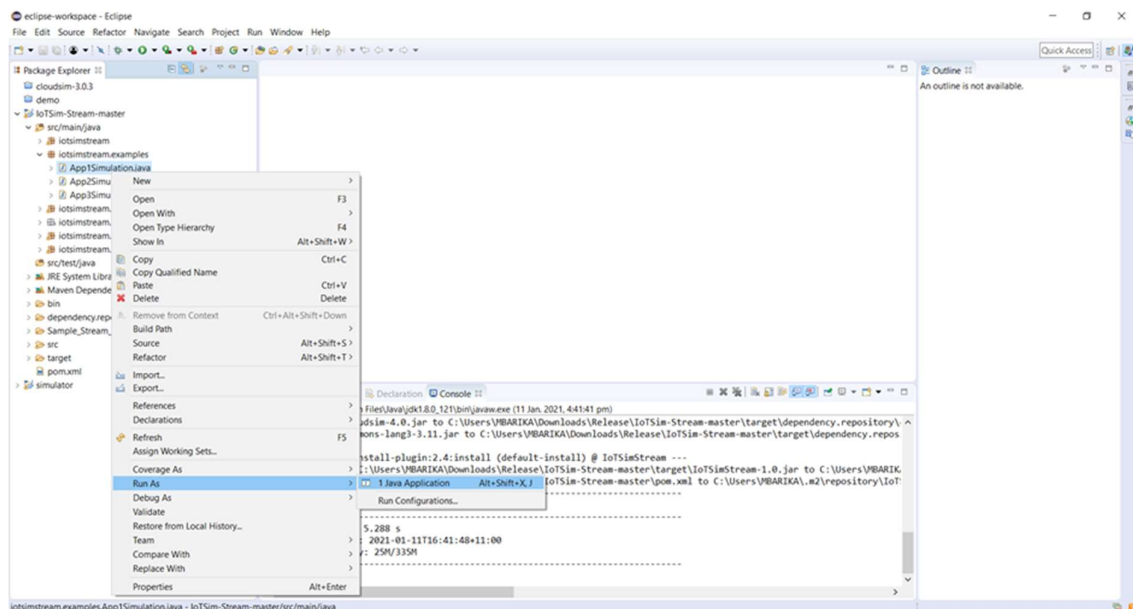


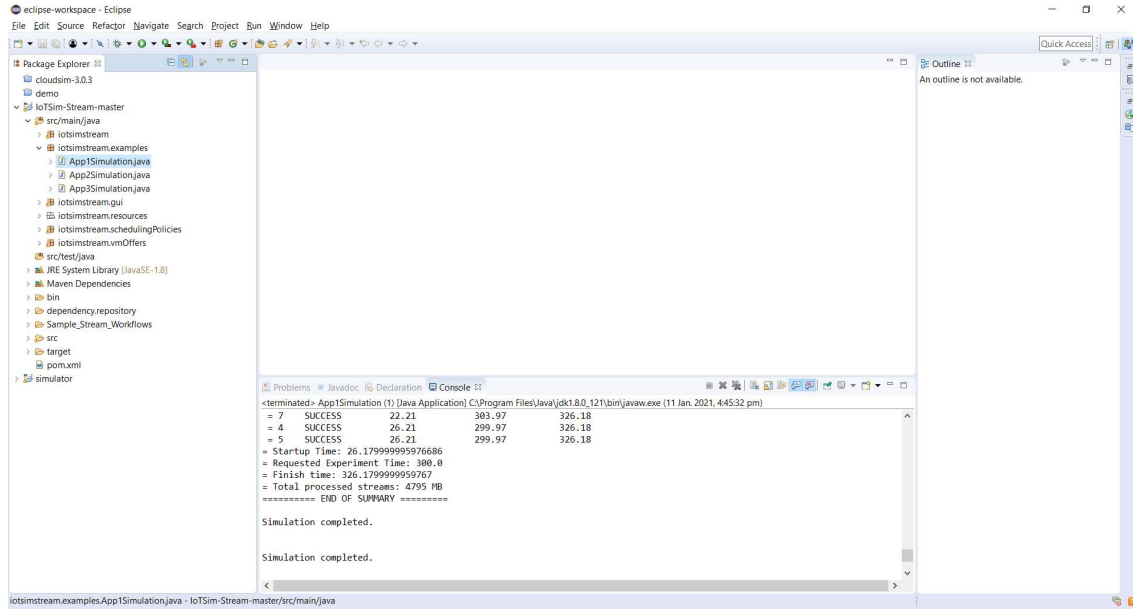
When this process is completed, you will see “BUILD SUCCESS” as shown in the below screenshot. At this point, you successfully built and configured IoTSim-Stream.



Step 4

To run a simulation, you need to go to the examples package (iotsimstream.examples) and run any example provided. For instance, if you want to simulate App1, expand the iotsimstream.examples package, right-click on "App1Simulation" and select "Run file". The simulation process will be initialized and the simulation of App1 will begin.





3.5 Simulation Configuration: Stream processing system simulation

Prior to simulating stream graph applications, you need to configure the parameters of the stream processing system and define them in the simulation properties file (named `simulation.properties`). These parameters will be read by IoTSim-Stream during initialisation to prepare the simulation environment and then simulating a given stream graph application on this environment according to the specified configurations. The below table shows the simulation parameters that are included in this file with their descriptions.

Table 1: User-defined Simulation Parameters Configuration

Parameter	Description
<code>simulation time</code>	The requested simulation time in seconds
<code>scheduling.policy</code>	Provisioning and scheduling policy
<code>dag.file</code>	Path of XML file of stream graph application
<code>cloud.datacenter</code>	Number of Clouds, where each Cloud is represented by a datacenter
<code>engine.network.bandwidth</code>	Network bandwidth of GraphAppEngine
<code>engine.network.latency</code>	Network latency of GraphAppEngine
<code>cloud.provider</code>	Index of Cloud provider in execution environment (index starting from 0)
<code>datacenter.hosts#index</code>	number of hosts in datacenter (ex. <code>datacenter.hosts#0</code>)
<code>vm.delay#index</code>	Average delay of VM boot time
<code>vm.offers#index</code>	Path of Java class for offerings of Cloud-based datacentre
<code>host.cores#index</code>	Number of cores (PEs) available for each host
<code>host.memory#index</code>	Amount of memory available for each host
<code>host.storage#index</code>	Amount of storage available for each host
<code>core.mips#index</code>	MIPS for each core or PE
<code>internal.bandwidth#index</code>	Internal network bandwidth available for each VM within Cloud-based datacentre
<code>internal.latency#index</code>	Network delay between VMs within Cloud-based datacentre
<code>external.bandwidth#index</code>	External network bandwidth available by Cloud-based datacentre for transferring data streams to other datacentres
<code>external.latency#index</code>	Network delay from Cloud-based datacentre to other datacentres

The below listing shows an example of simulation.properties file to configure the simulation environment with two datacenters (IaaS providers). This file is found in the IoTSim-Stream directory (src/main/java/resources). The VM offers provided by each datacentre can be found in the corresponding class (see VmOffersDatacenter1 and VmOffersDatacenter2).

Listing 1: An example of simulation.properties

```
simulation.time = 300
scheduling.policy = iotsimstream.SimpleSchedulingPolicy
dag.file =

cloud.datacenter = 2
engine.network.bandwidth = 250
engine.network.latency = 0.03

cloud.provider = 0
datacenter.hosts#0 = 1000
vm.delay#0 = 20
vm.offers#0 = iotsimstream.VmOffersDatacenter1
host.cores#0 = 64
host.memory#0 = 144000
host.storage#0 = 1400000
core.mips#0 = 1000
internal.bandwidth#0 = 770
internal.latency#0 = 0.00077
external.bandwidth#0 = 170
external.latency#0 = 0.028

cloud.provider = 1
datacenter.hosts#1 = 1000
vm.delay#1 = 20
vm.offers#1 = iotsimstream.VmOffersDatacenter2
host.cores#1 = 64
host.memory#1 = 176000
host.storage#1 = 1500000
core.mips#1 = 2000
internal.bandwidth#1 = 780
internal.latency#1 = 0.00075
external.bandwidth#1 = 180
external.latency#1 = 0.026
```

4 Simulating

The structure of the stream graph application involves heterogeneous services, multiple data sources, multiple input streams and multiple output streams. This structure can be expressed in DAG file by including all modelled services with their data processing requirements and performance constraints that defined by the owner of this application, and data dependencies among them. Moreover, IoTSim-Stream supports the modelling of different patterns/structures of stream workflow applications, which are linear, branching and hybrid. Linear workflow pattern (like App1) is a multi-stage application, where each stage processes the input stream generated by the previous stage and produces the output stream to the following stage. Branching workflow pattern (like App2) is an application with limited precedence constraints that splits data stream to perform different parallel processing and

then combining the results for further analysing. Hybrid workflow pattern (like App3) is a mix of linear and branching patterns. The DAG file will be parsed by the simulator to create stream graph application.

To simulate stream graph application, you need to describe stream graph application in XML structure in DAG file, and then given the path of such file to the simulator by writing it down as a value of dag.file property in simulation.properties.

4.1 Example of Linear Stream Graph Application

If your stream graph application is the one that is presented in the below figure, the XML structure of such application is depicted in the below listing.

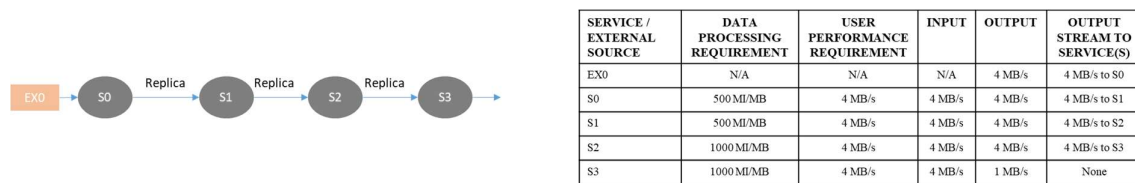


Figure 1. Example of linear stream graph application with its parameter configurations

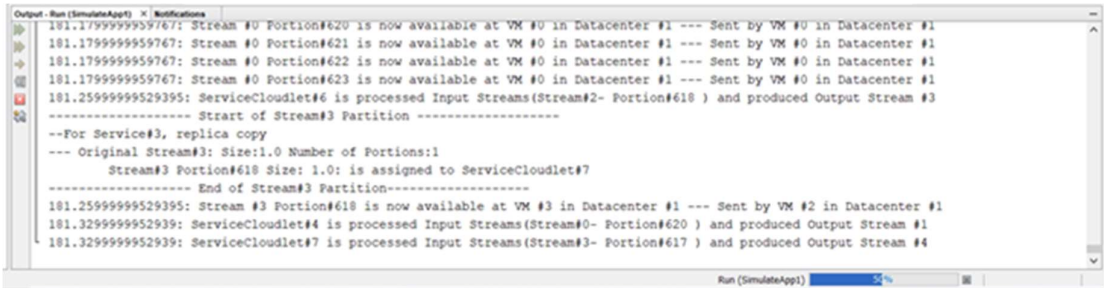
Listing 2: XML structure of linear stream graph application in DAG file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><!-- generated: 2017-06-24T14:29:13-07:00 --><!-- generated by: Mutaz[??] --><adag xmlns="http://pegasus.isi.edu/schema/DAX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" childCount="3" name="App1" serviceCount="4" version="2.1" xsi:schemaLocation="http://pegasus.isi.edu/schema/DAX http://pegasus.isi.edu/schema/dax-2.1.xsd">
<!-- part 1: list of all referenced files (may be empty) -->
<!-- part 2: definition of all jobs (at least one) -->
  <externalsources>
    <exsource datarate="4" id="PID00000" name="Producer0" type="stream"/>
  </externalsources>
  <service dataprocessingreq="500" id="ID00000" name="TmpltBank" userreq="4">
    <uses link="input" producerref="PID00000" type="stream"/>
    <uses link="output" size="4" transfer="true" type="stream"/>
  </service>
  <service dataprocessingreq="500" id="ID00001" name="TmpltBank" userreq="4">
    <uses link="input" processingtype="replica" serviceref="ID00000" transfer="true" type="stream"/>
    <uses link="output" size="4" transfer="true" type="stream"/>
  </service>
  <service dataprocessingreq="1000" id="ID00002" name="TmpBank" userreq="4">
    <uses link="input" processingtype="replica" serviceref="ID00001" type="stream"/>
    <uses link="output" size="4" type="stream"/>
  </service>
  <service dataprocessingreq="1000" id="ID00003" name="TmpBank" userreq="4">
    <uses link="input" processingtype="replica" serviceref="ID00002" type="stream"/>
    <uses link="output" size="1" type="stream"/>
  </service>
  <child ref="ID00001">
    <parent ref="ID00000"/>
  </child>
  <child ref="ID00002">
    <parent ref="ID00001"/>
  </child>
  <child ref="ID00003">
    <parent ref="ID00002"/>
  </child>
</adag>
```

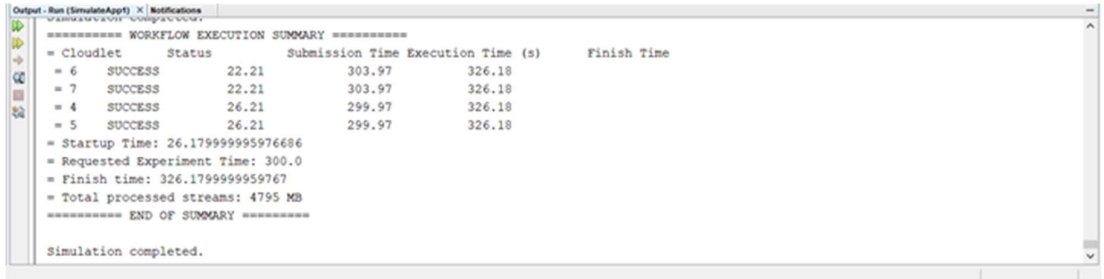
The above XML file for this stream graph application is named with “App1” and can be found in the IoTSim-Stream directory under Sample_Stream_Workflows.

Now, you can simply start the simulation by clicking on Run File “App1Simulation”.

During the simulation, you will see the detailed execution of a given stream graph application in output toolbar/pane, where IoTSim-Stream logs each event. Some of those details are scheduling plan, stream transfer from source SVM to destination SVM(s), stream replica or partition and stream scheduling on SVMs.



At the end of the simulation, you will see the summary of workflow execution like the below



4.2 Example of Branching Stream Graph Applications (App2)

If your stream graph application is the one that is presented in the below figure, the XML structure of such application is depicted in the below listing.

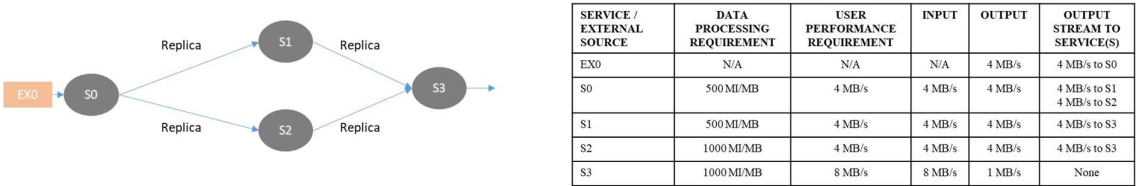


Figure 2. Example of branching stream graph application with its parameter configurations

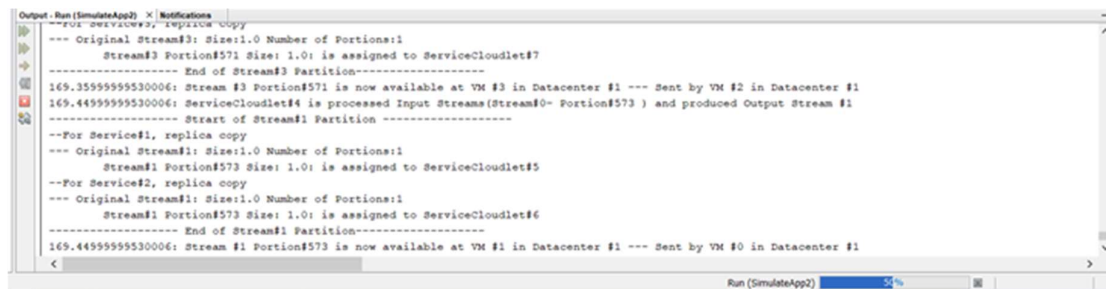
Listing 3: XML structure of branching stream graph application in DAG file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><!-- generated: 2017-06-24T14:29:13-07:00 --><!-- generated by: Mutaz[??] -
-><adag xmlns="http://pegasus.isi.edu/schema/DAX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" childCount="3"
name="App2" serviceCount="4" version="2.1" xsi:schemaLocation="http://pegasus.isi.edu/schema/DAX
http://pegasus.isi.edu/schema/dax-2.1.xsd">
<!-- part 1: list of all referenced files (may be empty) -->
<!-- part 2: definition of all jobs (at least one) -->
<externalsources>
  <exsource datarate="4" id="PID00000" name="Producer0" type="stream"/>
</externalsources>
<service dataprocessingreq="500" id="ID00000" name="TplmtBank" userreq="4">
  <uses link="input" producerref="PID00000" type="stream"/>
  <uses link="output" size="4" transfer="true" type="stream"/>
</service>
<service dataprocessingreq="500" id="ID00001" name="TplmtBank" userreq="4">
  <uses link="input" processingtype="replica" serviceref="ID00000" type="stream"/>
  <uses link="output" size="4" transfer="true" type="stream"/>
</service>
<service dataprocessingreq="1000" id="ID00002" name="TmpBank" userreq="4">
  <uses link="input" processingtype="replica" serviceref="ID00000" type="stream"/>
  <uses link="output" size="4" type="stream"/>
</service>
<service dataprocessingreq="1000" id="ID00003" name="TmpBank" userreq="8">
  <uses link="input" processingtype="replica" serviceref="ID00001" type="stream"/>
  <uses link="input" processingtype="replica" serviceref="ID00002" type="stream"/>
  <uses link="output" size="1" type="stream"/>
</service>
<child ref="ID00001">
  <parent ref="ID00000"/>
</child>
<child ref="ID00002">
  <parent ref="ID00000"/>
</child>
<child ref="ID00003">
  <parent ref="ID00001"/>
  <parent ref="ID00002"/>
</child>
</adag>
```

The above XML file for this stream graph application is named with “App2” and can be found in the IoTsim-Stream directory under Sample_Stream_Workflows.

Now, you can simply start the simulation by clicking on Run File “App2Simulation”.

During the simulation, you will see the detailed execution of a given stream graph application in output toolbar/pane, where IoTsim-Stream logs each event. Some of those details are scheduling plan, stream transfer from source SVM to destination SVM(s), stream replica or partition and stream scheduling on SVMs.



At the end of simulation, you will see the summary of workflow execution like the below

```

Output - Run (SimulateApp2) x Notifications
Simulation completed.
===== WORKFLOW EXECUTION SUMMARY =====
= Cloudlet      Status      Submission Time Execution Time (s)      Finish Time
= 6 SUCCESS      22.21      303.97      326.18
= 7 SUCCESS      22.21      303.97      326.18
= 4 SUCCESS      26.21      299.97      326.18
= 5 SUCCESS      26.21      299.97      326.18
= Startup Time: 26.17999995976686
= Requested Experiment Time: 300.0
= Finish time: 326.179999959767
= Total processed streams: 5993 MB
===== END OF SUMMARY =====
Simulation completed.

```

4.3 Example of Hybrid Stream Graph Applications (App3)

If your stream graph application is the one that presented in the below figure, the XML structure of such an application is depicted in the below listing.

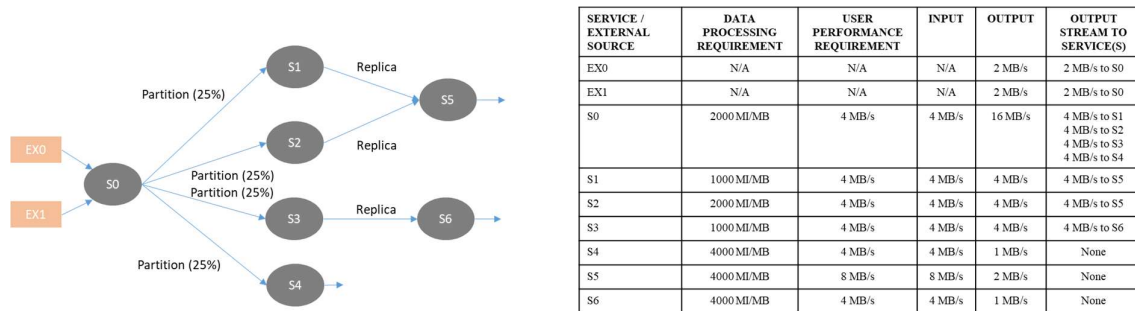


Figure 3. Example of hybrid stream graph application with its parameter configurations

Listing 4: XML structure of hybrid sample stream graph application in DAG file

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- generated: 2018-02-27:11:00 -->
<!-- generated by: Mutaz -->
<adag xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0" count="6" name="SampleStreamGraphhApplication"
serviceCount="6" childCount="5">
<!-- part 1: list of all referenced outputs of services (may be empty) -->
<!-- part 2: definition of all services (at least one) -->
  <externalsources>
    <exsource id="PID00000" name="Producer0" type="stream" datarate="10"/>
    <exsource id="PID00001" name="Producer1" type="stream" datarate="10"/>
    <exsource id="PID00002" name="Producer2" type="stream" datarate="5"/>
    <exsource id="PID00003" name="Producer3" type="stream" datarate="5"/>
    <exsource id="PID00004" name="Producer4" type="stream" datarate="5"/>
  </externalsources>
  <service id="ID00000" dataprocessingreq="400" userreq="10" namespace="Sample" name="BigService0" version="1.0">
    <uses link="input" type="stream" producerref="PID00000"/>
    <uses link="output" type="stream" size="5"/>
  </service>
  <service id="ID00001" dataprocessingreq="1000" userreq="5" namespace="Sample" name="BigService1" version="1.0">
    <uses link="input" type="stream" processingtype="replica" serviceref="ID00000"/>
    <uses link="output" type="stream" size="10"/>
  </service>
  <service id="ID00002" dataprocessingreq="500" userreq="8" namespace="Sample" name="BigService2" version="1.0">
    <uses link="input" type="stream" processingtype="replica" serviceref="ID00000"/>
    <uses link="input" type="stream" processingtype="partition" partitionpercentage="30" serviceref="ID00001"/>
    <uses link="output" type="stream" size="8"/>
  </service>

```



```

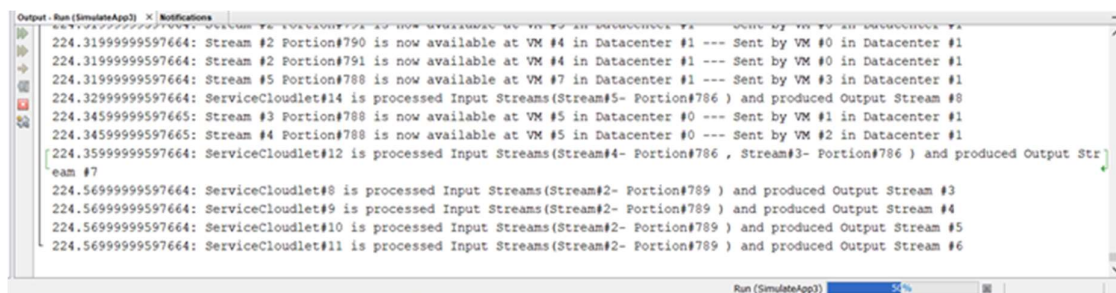
</service>
<service id="ID00003" dataprocessingreq="2000" userreq="7" namespace="Sample" name="BigService3" version="1.0">
  <uses link="input" type="stream" processingtype="partition" partitionpercentage="70" serviceref="ID00001"/>
  <uses link="output" type="stream" size="1"/>
</service>
<service id="ID00004" dataprocessingreq="3000" userreq="8" namespace="Sample" name="BigService4" version="1.0">
  <uses link="input" type="stream" processingtype="replica" serviceref="ID00002"/>
  <uses link="output" type="stream" size="2"/>
</service>
<service id="ID00005" dataprocessingreq="1500" userreq="38" namespace="Sample" name="BigService5" version="1.0">
  <uses link="input" type="stream" producerref="PID00000"/>
  <uses link="input" type="stream" producerref="PID00001"/>
  <uses link="input" type="stream" producerref="PID00002"/>
  <uses link="input" type="stream" producerref="PID00003"/>
  <uses link="input" type="stream" producerref="PID00004"/>
  <uses link="input" type="stream" processingtype="replica" serviceref="ID00003"/>
  <uses link="input" type="stream" processingtype="replica" serviceref="ID00004"/>
  <uses link="output" type="stream" size="4"/>
</service>
<!-- part 3: list of control-flow dependencies (may be empty) -->
<child ref="ID00001">
  <parent ref="ID00000"/>
</child>
<child ref="ID00002">
  <parent ref="ID00000"/>
  <parent ref="ID00001"/>
</child>
<child ref="ID00003">
  <parent ref="ID00001"/>
</child>
<child ref="ID00004">
  <parent ref="ID00002"/>
</child>
<child ref="ID00005">
  <parent ref="ID00003"/>
  <parent ref="ID00004"/>
</child>
</adag>

```

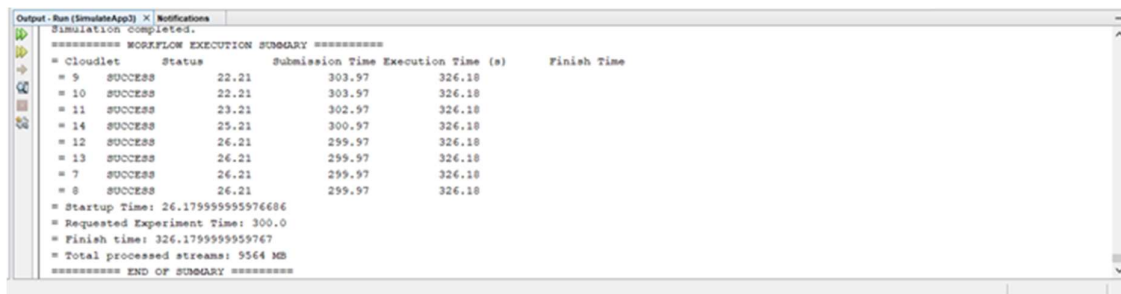
The above XML file for this stream graph application is named with “App3” and can be found in the IoTsim-Stream directory under Sample_Stream_Workflows.

Now, you can simply start the simulation by clicking on Run File “App3Simulation”.

During the simulation, you will see the detailed execution of a given stream graph application in output toolbar/pane, where IoTsim-Stream logs each event. Some of those details are scheduling plan, stream transfer from source SVM to destination SVM(s), stream replica or partition and stream scheduling on SVMs.



At the end of the simulation, you will see the summary of workflow execution like the below



The screenshot shows a terminal window titled "Output - Run (SimulateApp3) x Notifications". The text inside indicates that the simulation is completed and displays a summary of workflow execution. The summary includes a table of execution results for various cloudlets, followed by startup, requested experiment, and finish times, and the total processed streams.

===== WORKFLOW EXECUTION SUMMARY =====				
Cloudlet	Status	Submission Time	Execution Time (s)	Finish Time
= 9	SUCCESS	22.21	303.97	326.18
= 10	SUCCESS	22.21	303.97	326.18
= 11	SUCCESS	23.21	302.97	326.18
= 14	SUCCESS	25.21	300.97	326.18
= 12	SUCCESS	26.21	299.97	326.18
= 13	SUCCESS	26.21	299.97	326.18
= 7	SUCCESS	26.21	299.97	326.18
= 8	SUCCESS	26.21	299.97	326.18

===== END OF SUMMARY =====

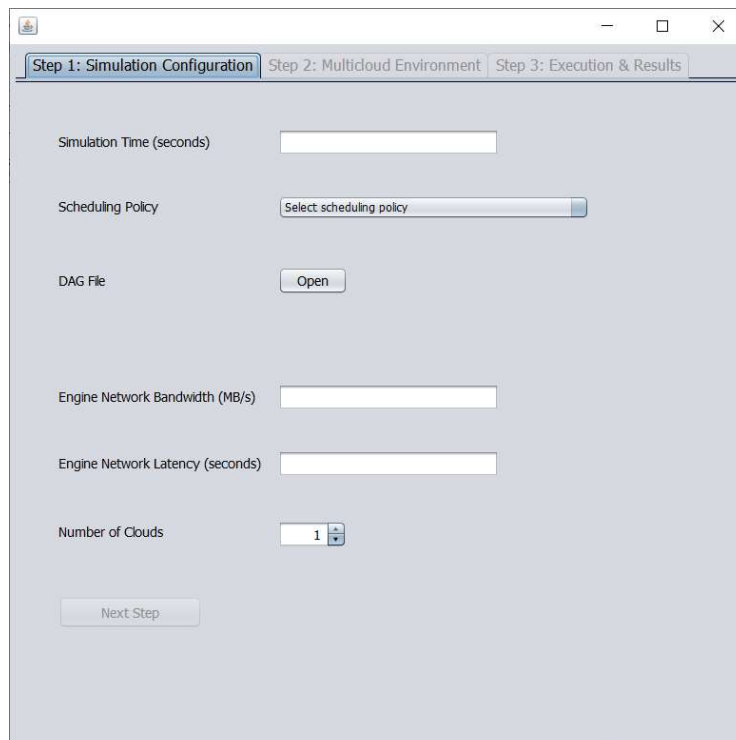
Additional summary information:
= Startup Time: 26.1795959595976686
= Requested Experiment Time: 300.0
= Finish time: 326.17959595959767
= Total processed streams: 9564 MB

Note that: to simulate your stream graph application using IoTsim-Stream, you need to provide the actual path of the application. To do this, go to an any example provided and set the value of dag.file property in runSimulation() method to the actual path of this application.

5 GUI

IoTsim-Stream now provides GUI to easily set up simulation configurations and a multicloud environment and then run a given stream graph application in the simulator. This GUI guides you using a step-by-step process starting from configuration to running the simulation without having to manually set values for simulation configuration parameters. Moreover, all inputs in the GUI are validated to ensure that the simulation configurations and the multicloud environment are successfully set.

You can simply open IoTsim-Stream GUI by clicking on Run File “SimGUI”, which is included “iotsimstream.gui” package.



The screenshot shows the "Step 1: Simulation Configuration" window of the IoTsim-Stream GUI. The window has three tabs: "Step 1: Simulation Configuration", "Step 2: Multicloud Environment", and "Step 3: Execution & Results". The "Step 1" tab is active. The configuration fields are as follows:

- Simulation Time (seconds): A text input field.
- Scheduling Policy: A dropdown menu with the text "Select scheduling policy".
- DAG File: A text input field with an "Open" button next to it.
- Engine Network Bandwidth (MB/s): A text input field.
- Engine Network Latency (seconds): A text input field.
- Number of Clouds: A text input field with the value "1" and a small up/down arrow button.

At the bottom of the window, there is a "Next Step" button.

In Step 1, you will set up simulation configurations such as simulation time and a number of cloud datacenters. If the input value passes the input validation, you will see the “Valid” word highlighted with green colour near to the entered value. Otherwise, you will see the corresponding error message highlighted with red colour instead. To move to the next step, you must type values for all simulation configurations and these values pass their input validations. If this is the case, the “Next Step” button will automatically be enabled. It is worth to note that for scheduling policy, the available police(s) is pulled from “`iotsimstream.schedulingPolicies`” package; this means if you have your own scheduling policy, you can easily put it under such package and it will automatically be added to the scheduling policy dropdown list.

The screenshot shows a software window titled "Step 1: Simulation Configuration". It features three tabs at the top: "Step 1: Simulation Configuration" (selected), "Step 2: Multicloud Environment", and "Step 3: Execution & Results". The main area contains several configuration fields, each with a label, an input field, and a "Valid" status indicator in green text. The fields are: "Simulation Time (seconds)" with a text input containing "300"; "Scheduling Policy" with a dropdown menu showing "iotsimstream.schedulingPolicies.SimpleSchedulingPol..."; "DAG File" with an "Open" button and the filename "App1.xml" displayed below; "Engine Network Bandwidth (MB/s)" with a text input containing "270"; "Engine Network Latency (seconds)" with a text input containing "0.03"; and "Number of Clouds" with a spinner input showing "2". At the bottom left, there is a "Next Step" button.

In Step 2, you will configure cloud datacenters one by one. Simply select cloud datacenter from the dropdown list and then fill in its configuration parameters. Then, add values for all configuration parameters of the selected cloud datacenter. If all values entered pass the input validation, “Add / Modify” button will be enabled. Next, click on “Add / Modify” to save these values for the selected cloud datacenter under its node in the Multicloud Environment tree. Do the same for the remaining cloud datacenters to set values for their configuration parameters. In case you want to see or modify the values of cloud datacenter configuration parameters, simply select this cloud datacenter from the cloud datacenter dropdown list. You will notice that all the values entered before are retrieved and filled in the corresponding fields/components. Now, if you want to modify the value of any configuration parameter, you can easily update the value and click on “Add / Modify” button to save this update. It is worth to note that for VM offers class, the available offer classes are pulled from “`iotsimstream.vmOffers`” package; this means if you have your own VM offer class, you can easily put it under such package and it will automatically be added to the VM offer class dropdown list.

After successfully adding the values for all configuration parameters for all cloud datacenters, “Next Step” button will be enabled. In this case, click on “Next Step” button to proceed to the last step.

Step 1: Simulation Configuration

Step 2: Multicloud Environment

Step 3: Execution & Results

Cloud Datacenter

Cloud Datacenter 1

Number of hosts

1000

Valid

VM delay

20

Valid

VM offers class

iotsimstream.vmOffers.VmOffersDatacenter1

Valid

Host cores

64

Valid

Host memeory

144000

Valid

Host storage

1400000

Valid

Core MIPS

1000

Valid

Internal bandwidth

770

Valid

Internal latency

0.00077

Valid

External bandwidth

170

Valid

External latency

0.028

Valid

Multicloud Environment

Cloud Datacenter 1

datacenter.hosts = 1000

vm.delay = 20

vm.offers = iotsimstream.vmOffers.VmOffersDatacenter1

host.cores = 64

host.memory = 144000

host.storage = 1400000

core.mips = 1000

internal.bandwidth = 770

internal.latency = 0.00077

external.bandwidth = 170

external.latency = 0.028

Add / Modify

Next Step

Step 1: Simulation Configuration

Step 2: Multicloud Environment

Step 3: Execution & Results

Cloud Datacenter

Cloud Datacenter 2

Number of hosts

1000

Valid

VM delay

20

Valid

VM offers class

iotsimstream.vmOffers.VmOffersDatacenter2

Valid

Host cores

64

Valid

Host memeory

176000

Valid

Host storage

1500000

Valid

Core MIPS

2000

Valid

Internal bandwidth

780

Valid

Internal latency

0.00075

Valid

External bandwidth

180

Valid

External latency

0.026

Valid

Cloud Datacenter 1

Cloud Datacenter 2

datacenter.hosts = 1000

vm.delay = 20

vm.offers = iotsimstream.vmOffers.VmOffersDatacenter2

host.cores = 64

host.memory = 176000

host.storage = 1500000

core.mips = 2000

internal.bandwidth = 780

internal.latency = 0.00075

external.bandwidth = 180

external.latency = 0.026

Add / Modify

Next Step

When you reach Step 3 (i.e. the last step), this means you have successfully set up the simulation configurations and multicloud environment. Now, you just need to click on “Start Execution” to run a given stream graph application in the simulator. After clicking on “Start Execution” button, IoTSim-Stream will ask you whether you want to replace the simulation configurations in the “simulation.properties” file with the values entered in the previous steps. If you select “Yes”, the simulation will begin, and you will see the summary of workflow execution. If you select “No”, a message will appear asking you to take a copy of your configuration in “simulation.properties” and try again.

