National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Naseem Sultana
Student ID: 22152261

School of Computing
National College of Ireland

Supervisor: Dr.Punit Gupta

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Naseem Sultana
Student ID:	22152261
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr.Punit Gupta
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	759
Page Count:	4

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	14th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Naseem Sultana
22152261

1 Introduction

The configuration manual describes briefly how various tools were used during the implementation of algorithms in the thesis titled "Meta-Heuristic Scheduling Algorithms on Containerized Environment". The report highlights how various algorithms can be used to schedule containers on VMs and then use metaheuristic algorithms to scheduke VMs on Physical machines (PMs). This could be tested either by directly deploying applications on Cloud using services provided by the providers or by testing various workloads on simulators like CloudSim. CloudSim 4.0, has been used to implement the algorithms and run various experiments found in ContainerCloudSimExample.java to find the optimal results.

2 Setting up CloudSim

The following part of the report describes various steps to install and set up CloudSim 4.0, on the local machine.

Step1: CloudSim is written in Java, however, it is a good practice to check if any existing Java version is installed on the machines or not. If there is nothing installed then the latest Java version can be downloaded and installed from the link https://www.java.com/download/iemanual.jsp.

Step 2: Setup Java environment variables after installing the latest verison of Java.

Step 4: Downloaded an intergrated development kit to work on the cloud infrstarture. The preferred one is Eclipse oxygen IDE which can be downloaded from: https://www.eclipse.org/downl 09/r/eclipse-ide-enterprise-java-and-web-developers

Step 5: Install the eclipse IDE on the desktop.

Step 6: Downloaded CloudSim 4.0 from https://github.com/Cloudslab/cloudsim/releases/tag/clou 4.0

Step 7: Open eclipse, create a workspace.

Step 6: Unzip the downloaded CloudSim 4.0 source code file and import in Eclipse IDE using Maven for dependencies.

3 Scheduling containers on Vms

Step 8: Once successfully imported the CloudSim file structure can be seen in the project explorer tab of the IDE, as shown in 1

Step 9: The project focuses on pre-existing container algorithms, so check for the pre-existing algorithms and code them under one file. As CloudSim is a cloud simulator it

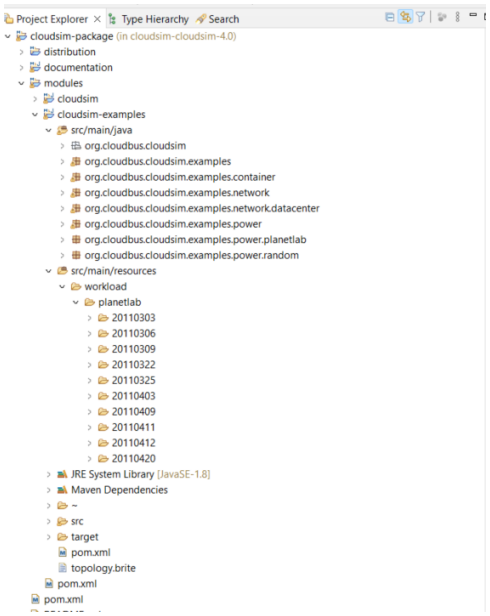


Figure 1: cloudsim-4.0 in eclipse IDE (file structure)

has an integrated open-source Planetlab workload which gives information about various dataset collected.

Step 10: Modify all the files related to the algorithms necessary and run the program as shown in the figure below: 8

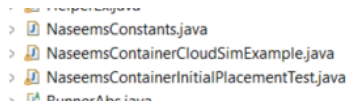


Figure 2: Files changed under containercloudsim

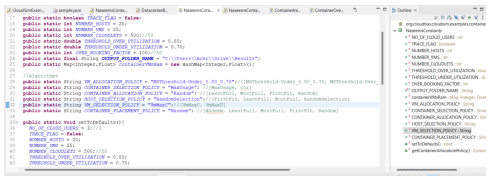


Figure 3: NaseemConstants.java

Step 11: Once the program runs successfully, the results are stored in the folder, check for the path and check the folder to get the results. 3

Step 12: Evaluate the results by checking for the maximum, minimum and average container migration time and maximum energy consumption of the containers.

4 Scheduling VMs on PMs

This section gives an overview of how the second experiment to schedule Virtual machines on physical machines using meta heuristic algorithms like ACOR, ALO and PSO is implemented.

Step 14 Using python environment on visual studios, write and run the algorithms ACOR, ALO and PSO which are improted from mealpy library.4

```
vm=10
tcount=10000
number=200
f = open('data.json')
data = json.load(f)
size=data['tcount'][0]
vmcount= data['vmcount'][0]

problem_dict1 = {
    "obj_func": Fun,
    "lb": [0 for i in range(size)] ,
    "ub": [vmcount for i in range(size)],
    "minmax": "min",
    "verbose": True,
}
print("Task Count is set to",size)
print("VM Count is set to",vmcount)
model1 = BaseACOR(problem_dict1, epoch=10, pop_size=50, pr=0.03)
model1.solve()
```

Figure 4: code for ACOR

Step 13 In Eclipse, import the file that contains meta-heuristic algorithms under CloudSim folder.

Step 14 In the project explorer of Eclipse IDE, click on CloudSimExample6.java from the list of examples, change the number of cloudlets and run it.

Step 15 The generates output is stored in a data.json file, check the data.json file and execute the ACOR algorithm in Visual Studio. The output of this algorithm is stored in a sample.json file which is uploaded to the Eclipse IDE.

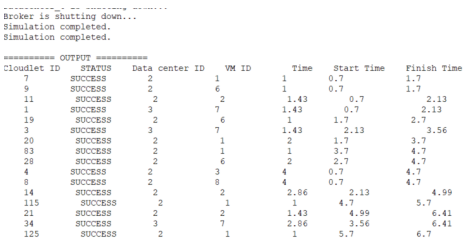
Step 16 When the CloudSimExample6.java file is executed, you notice that the output is generated halfway through and then data.json file is created. Now after executing ACOR sample.json file is generated and uploaded which acts as input to the CloudSimExample6.java. So click the console again to complete the execution of the example and check the results. 3

```
261.15942857142886: Broker: Cloudlet 357 received
264.4927619047622: Broker: Cloudlet 359 received
264.6027619047622: Broker: Cloudlet 372 received
271.15942857142886: Broker: Cloudlet 368 received
271.2694285714289: Broker: Cloudlet 382 received
274.49276190476223: Broker: Cloudlet 373 received
277.82609523809555: Broker: Cloudlet 375 received
277.93609523809556: Broker: Cloudlet 390 received
281.1594285714289: Broker: Cloudlet 377 received
284.60276190476225: Broker: Cloudlet 394 received
287.8260952380956: Broker: Cloudlet 392 received
287.9360952380956: Broker: Cloudlet 403 received
291.26942857142893: Broker: Cloudlet 407 received
294.4927619047623: Broker: Cloudlet 396 received
297.8260952380956: Broker: Cloudlet 401 received
297.9360952380956: Broker: Cloudlet 408 received
301.159428571429: Broker: Cloudlet 419 received
304.6027619047623: Broker: Cloudlet 414 received
307.82609523809566: Broker: Cloudlet 430 received
311.159428571429: Broker: Cloudlet 433 received
311.269428571429: Broker: Cloudlet 418 received
314.4927619047623: Broker: Cloudlet 437 received
317.82609523809566: Broker: Cloudlet 445 received
317.9360952380957: Broker: Cloudlet 440 received
```

Figure 5: Cloudlet received, second output

Step 17 The complete cycle of running CloudSimExample.java, storing the output in data.json, executing the meta heuristic algorithm and getting sample.json file which later

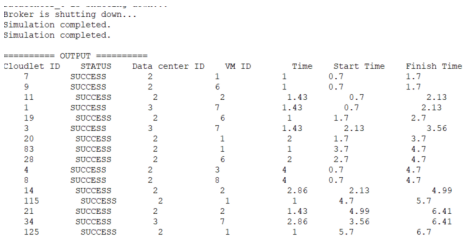
acts as input file for CloudExample6.java to complete the execution is repeated until all the algorithms are tested on different number of cloudlets and test the results. 3



```
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
     7         SUCCESS         2            1        1         0.7           1.7
     9         SUCCESS         2            6        1         0.7           1.7
    11          SUCCESS          2            2         1.43         0.7           2.13
     1         SUCCESS         3            7        1.43         0.7           2.13
    19          SUCCESS          2            6         1         1.7           2.7
     3         SUCCESS         3            7        1.43         2.13          3.56
    20          SUCCESS          2            1         2         1.7           3.7
    83          SUCCESS          2            1         1         3.7           4.7
    28          SUCCESS          2            6         2         2.7           4.7
     4         SUCCESS         2            3        4         0.7           4.7
     8         SUCCESS         2            8        4         0.7           4.7
    14          SUCCESS          2            2         2.86         2.13          4.99
   115           SUCCESS           2            1         1         4.7           5.7
    21          SUCCESS          2            2         1.43         4.99          6.41
    34          SUCCESS          3            7         2.86         3.56          6.41
   125           SUCCESS           2            1         1         5.7           6.7
```

Figure 6: Results output with wait , start and finish time



```
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
     7         SUCCESS         2            1        1         0.7           1.7
     9         SUCCESS         2            6        1         0.7           1.7
    11          SUCCESS          2            2         1.43         0.7           2.13
     1         SUCCESS         3            7        1.43         0.7           2.13
    19          SUCCESS          2            6         1         1.7           2.7
     3         SUCCESS         3            7        1.43         2.13          3.56
    20          SUCCESS          2            1         2         1.7           3.7
    83          SUCCESS          2            1         1         3.7           4.7
    28          SUCCESS          2            6         2         2.7           4.7
     4         SUCCESS         2            3        4         0.7           4.7
     8         SUCCESS         2            8        4         0.7           4.7
    14          SUCCESS          2            2         2.86         2.13          4.99
   115           SUCCESS           2            1         1         4.7           5.7
    21          SUCCESS          2            2         1.43         4.99          6.41
    34          SUCCESS          3            7         2.86         3.56          6.41
   125           SUCCESS           2            1         1         5.7           6.7
```

Figure 7: Results output with wait , start and finish time

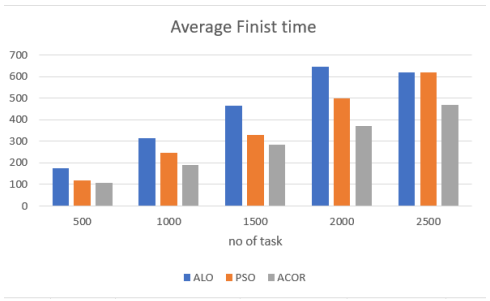


Figure 8: Results average finish time