

# Configuration Manual

MSc Research Project  
Cloud Computing

Shilpi Madan  
Student ID: x21145059

School of Computing  
National College of Ireland

Supervisor: Rashid Mijumbi

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Shilpi Madan
<b>Student ID:</b>	x21145059
<b>Programme:</b>	Cloud Computing
<b>Year:</b>	2022
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Rashid Mijumbi
<b>Submission Due Date:</b>	15/12/2022
<b>Project Title:</b>	Systemization and Evaluation for Data deduplication by deploying competitive chunking algorithm in polymorphic thread environment and avant-garde hashing techniques
<b>Word Count:</b>	673
<b>Page Count:</b>	5

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.

<b>Signature:</b>	Shilpi Madan
<b>Date:</b>	14th December 2022

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

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

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

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Shilpi Madan  
x21145059

## 1 Introduction

In the setup module, we will provide specifics on all of the libraries used to implement the suggested algorithm, as well as a step-by-step process for configuring the same in the AWS Cloud environment. The outcome of executing the algorithm on the cloud with a dataset on a different number of processors was described.

## 2 Software Requirements and Modules

### 2.1 Local System

For testing purpose we have deployed it on local system with below configuration:

**RAM:** 16 GB

**Operating System:** MacOS Ventura

**Hard Drive:** 512GB

**Software:** Visual Studio

### 2.2 Cloud

Implemented the overall project on AWS cloud account, under cloud the configuration used is Ubuntu 20.04 with memory of 8GB, 2VCPU.

**Processor:** Neoverse cores 64-bit Arm

**Storage:** NVMe

### 2.3 Python Modules

We have imported below modules in Python 3.7 to proceed with our algorithm functionality:

Table 1: Python Modules

Modules	Description
Time	For displaying and calculation of time
Multiprocessing	For many processor at a time
Threading	For running various function together

```
import sys
import multiprocessing
from time import time
import threading
```

Figure 1: Python

### 3 Implementation on Cloud

In this section, we will build an EC2 Akil et al. (2019) instance and deploy our application to it by following the procedures below:

1. Connect to the AWS account and create EC2 Ubuntu instance and launch the instance.

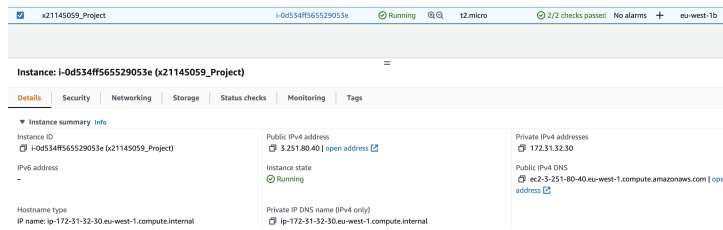


Figure 2: EC2 Instance

2. To connect our EC2 to our local terminal we used pem key and below commands with "ssh" protocol with the details of DNS connection.

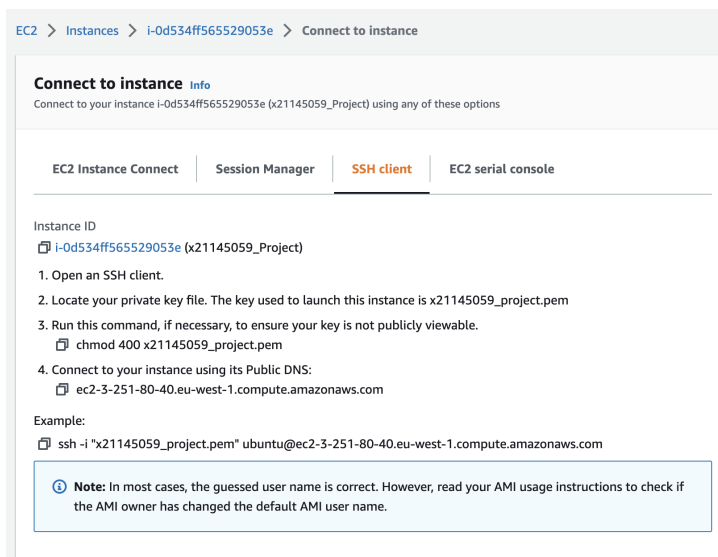
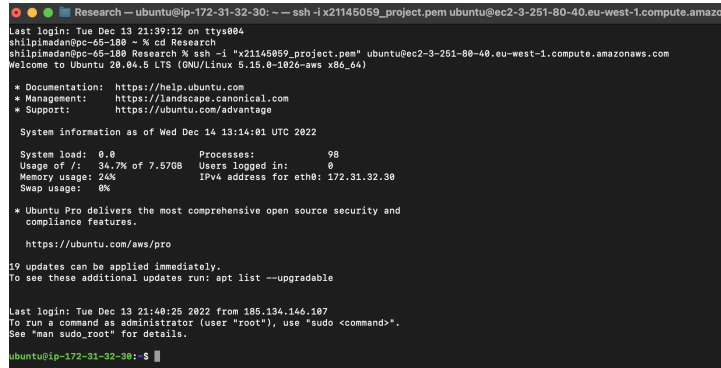


Figure 3: Connect EC2 to Terminal

```
ssh -i "x21145059_project.pem" ubuntu@ec2-3-251-80-40.eu-west1.compute.amazonaws.com
```

3. Once we define the above command that connect our EC2 instance and to terminal our local machine is running Ubuntu and all the resources linked to that cloud infrastructure.



```
Research - ubuntu@ip-172-31-32-30: ~ -- ssh -i x21145059_project.pem ubuntu@ec2-3-251-80-40.eu-west-1.compute.amazonaws.com
Last login: Tue Dec 13 21:39:12 on ttys004
shilpimadan@pc-65-180 ~ % cd Research
shilpimadan@pc-65-180 Research % ssh -i "x21145059_project.pem" ubuntu@ec2-3-251-80-40.eu-west-1.compute.amazonaws.com
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1026-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Dec 14 13:14:01 UTC 2022

System load:  0.0      Processes:    98
Usage of /:   34.7% of 7.57GB   Users logged in:  0
Memory usage: 24%      IPv4 address for eth0: 172.31.32.30
Swap usage:   6%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

https://ubuntu.com/aws/pro

19 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Last login: Tue Dec 13 21:40:25 2022 from 185.134.146.107
[O run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.]

ubuntu@ip-172-31-32-30: ~ $
```

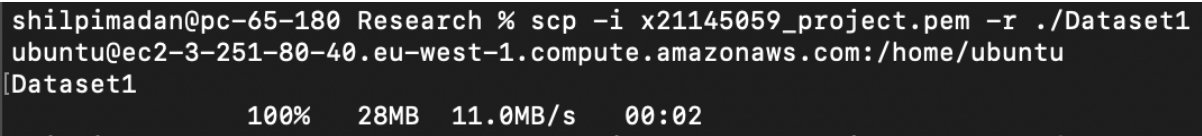
Figure 4: Connected to Ubuntu

4. To execute the chunking algorithm we need to deploy our application to AWS cloud along with the 3 datasets for evaluation.
5. Using Secure Copy Protocol (SCP)File (2022) we will transfer the data from local terminal to cloud and for that we need more parameters like: Public Pem key, file name, public IP address DNS of cloud, location of home directory.

```
$ scp -i pemfile.pem your_filename ubuntu@Public_DNS: /<path>/
```

Figure 5: Command to copy file

```
scp -i x21145059_project.pem -r ./Dataset1 ubuntu@ec2-3-251-80-40.eu-west-1.compute.amazonaws.com: /home/ubuntu
```



```
shilpimadan@pc-65-180 Research % scp -i x21145059_project.pem -r ./Dataset1
ubuntu@ec2-3-251-80-40.eu-west-1.compute.amazonaws.com: /home/ubuntu
[Dataset1
100% 28MB 11.0MB/s 00:02
```

Figure 6: Transfer file to Cloud

6. Once all the files are being copied to cloud lets verify all the files.

```
ubuntu@ip-172-31-32-30:~$ ls
Chunking-Algorithm.py  ChunkingOut  Dataset1  Dataset2  Dataset3
Parallel-Chunking.py  homes1212  result
```

Figure 7: List of files

## 4 File Execution

To obtain desired results we will be executing serial and Parallel python programs by using different number of ProcessorsSobe et al. (2015) and on all 3 datasets.

### 4.1 Serial Chunking

To work on data de-duplication technique we have to go for chunking and hashing process. Here, in this section we will see the processing the hashing and chunking stages in single thread processor and the time it takes to execute a dataset of size 48MB. The python code for this is given in the artefacts submitted on moddle and various test are being conducted to achieve desired results.

```
for i in range(numTests):
    start = time()
    serialResult = serialCode(LBAlist)
    t = time() - start
    print("Time for 1 Processors: {}".format(t))
    avgTime.append(t)
```

Figure 8: Serial Processor

The output while executing the dataset-3 to calculate average time to chunk the entire file using 10 iterations.

```
ubuntu@ip-172-31-32-30:~$ python3 Chunking-Algorithm.py
Time for 1 Processors: 8.925203800201416
Time for 1 Processors: 8.739416599273682
Time for 1 Processors: 8.95130181312561
Time for 1 Processors: 8.890659809112549
Time for 1 Processors: 8.898515224456787
Time for 1 Processors: 8.808566808700562
Time for 1 Processors: 8.70194959640503
Time for 1 Processors: 8.883328437805176
Time for 1 Processors: 8.542064428329468
Time for 1 Processors: 8.919476985931396
Average Serial Runtime: 8.826048350334167
```

Figure 9: Serial Processor Output

### 4.2 Multi-Thread Chunking

In this section we are executing the chunking algorithm using 20 threads at a time that will execute the various section of the datastreams parallely.

```
for p in range(0,numProcs):
    q = multiprocessing.Queue()
    if p != numProcs-1:
        proc = multiprocessing.Process(target = globalCDC, args = (chunkSize*p,chunkSize*(p+1),q))
    else:
        proc = multiprocessing.Process(target = globalCDC, args = (chunkSize*p, len(LBAList),q))
```

Figure 10: Multi-Thread Processor

The output of the average time chunking process with 20 processors running at the same time.

```
ubuntu@ip-172-31-32-30:~$ python3 Parallel-Chunking.py
Testing Source File: Dataset1
Time for 20 Processors: 5.780196189880371
Time for 20 Processors: 5.573897123336792
Time for 20 Processors: 5.671887397766113
Time for 20 Processors: 5.664926767349243
Time for 20 Processors: 5.55240797996521
Time for 20 Processors: 5.567561388015747
Time for 20 Processors: 5.514532566070557
Time for 20 Processors: 5.570092439651489
Time for 20 Processors: 5.477118730545044
Time for 20 Processors: 5.622833728790283
Average Runtime for 20 Processes: 5.599545431137085
Testing Source File: Dataset1
Time for 19 Processors: 5.743909120559692
Time for 19 Processors: 5.885350942611694
Time for 19 Processors: 5.888677358627319
Time for 19 Processors: 5.815004587173462
Time for 19 Processors: 5.859370708465576
Time for 19 Processors: 5.845226526260376
Time for 19 Processors: 5.779861927032471
Time for 19 Processors: 5.762841463088989
Time for 19 Processors: 5.876147985458374
Time for 19 Processors: 5.788475751876831
```

Figure 11: Multi Processors Output

## References

- Akil, M., Mancini, L. V. and Venturi, D. (2019). Multi-covert channel attack in the cloud, *2019 Sixth International Conference on Software Defined Systems (SDS)*, IEEE, pp. 160–165.
- File, T. (2022). Local windows or mac pc to linux aws ec2, *Amazon Doc., managedservices. appguide.* .
- Sobe, P., Pazak, D. and Stiehr, M. (2015). Parallel processing for data deduplication, *PARS-Mitteilungen: Vol. 32, Nr. 1* .