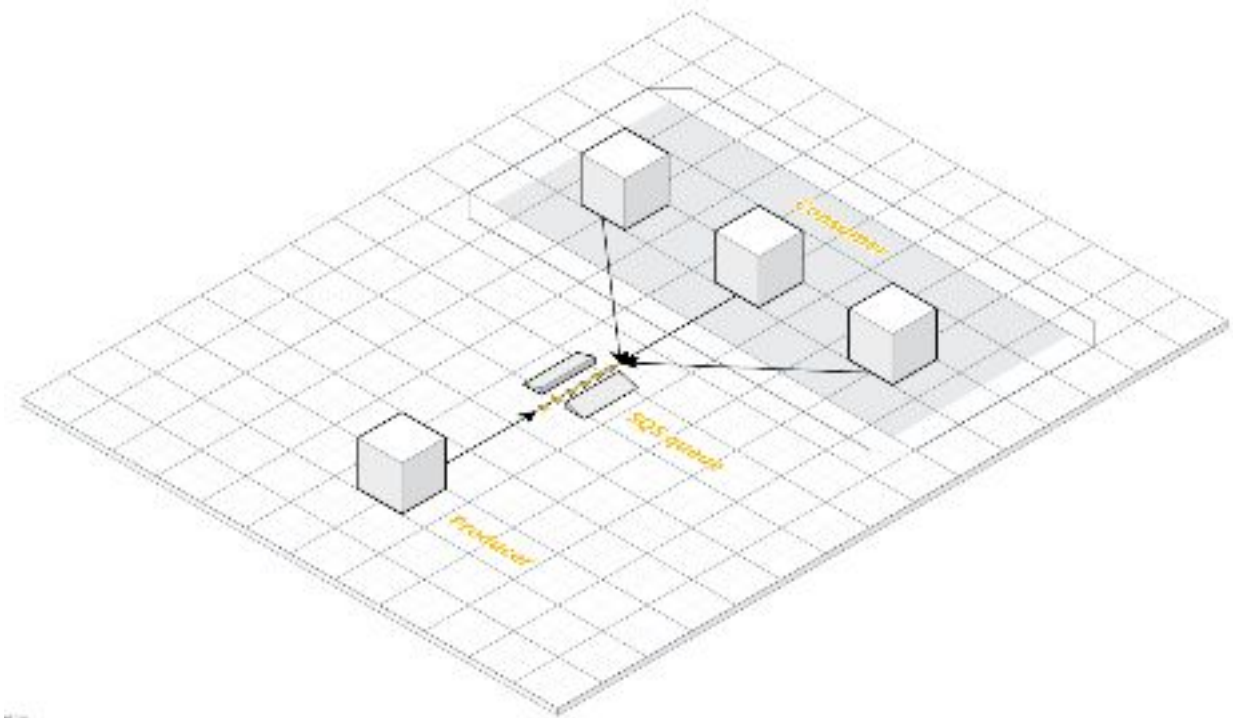
CSN-520 (Spring Semester 2018)

CLOUD COMPUTING PROJECT

Team:

14114002 ABHISHEK JAISINGH

14114009 AMBAR ZAIDI



Objective

Test and benchmark any compute intensive application on any public cloud platform like AWS, GAE or Azure.

Problem Statement

Given an integer range, L to R (inclusive of L and R) find the no. of Happy Prime Numbers in that range.

Happy Numbers (OEIS A090425)

A Happy Number is defined by the following process:

- Starting with any positive integer, replace the number by the sum of the squares of its digits in base-ten, and repeat the process until the number either equals 1 (where it will stay), or it loops endlessly in a cycle that does not include 1.

Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers (or sad numbers).

```
def square(x):
    return int(x) * int(x)

def happy(number):
    return sum(map(square, list(str(number))))

def is_happy(number):
    seen_numbers = set()
    while number > 1 and (number not in seen_numbers):
        seen_numbers.add(number)
        number = happy(number)
    return number == 1
```

Sieve of Eratosthenes

This algorithm produces all primes not greater than n. It includes a common optimization, which is to start enumerating the multiples of each prime i from i^2. The time complexity of this algorithm is **O(n log log n)** .

```
Input: an integer n > 1.

Let A be an array of Boolean values, indexed by integers 2 to n,
initially all set to true.

for i = 2, 3, 4, ..., not exceeding √n:
  if A[i] is true:
    for j = i², i²+i, i²+2i, i²+3i, ..., not exceeding n:
      A[j] := false.

Output: all i such that A[i] is true.
```

Services / Infrastructure Used

Following services provided by Amazon Web Services (AWS) have been used:

- **Amazon Simple Queue Service (SQS) - Amazon AWS**
  - Amazon Simple Queue Service (SQS) is a fully managed message queuing service that makes it easy to decouple and scale microservices, distributed systems, and serverless applications. Building applications from individual components that each perform a discrete function improves scalability and reliability, and is best practice design for modern applications.
  - https://aws.amazon.com/sqs/

- **Amazon EC2 - Amazon AWS**
  - Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction.
  - https://aws.amazon.com/ec2/

Requirements/Setup

Instances created

- ➢ 2 instances of SQS &
- ➢ Multiple instances of Amazon EC2 service

are used.

Software requirements

Each EC2 instance (Ubuntu Server 16.04 LTS) is installed with the following :

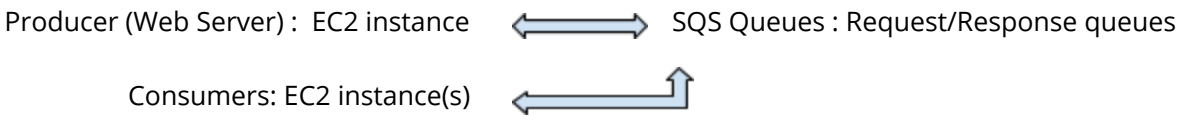
- ➢ aws-cli
- ➢ python 2.7
- ➢ AWS sdk for python - boto3

Architecture

Process

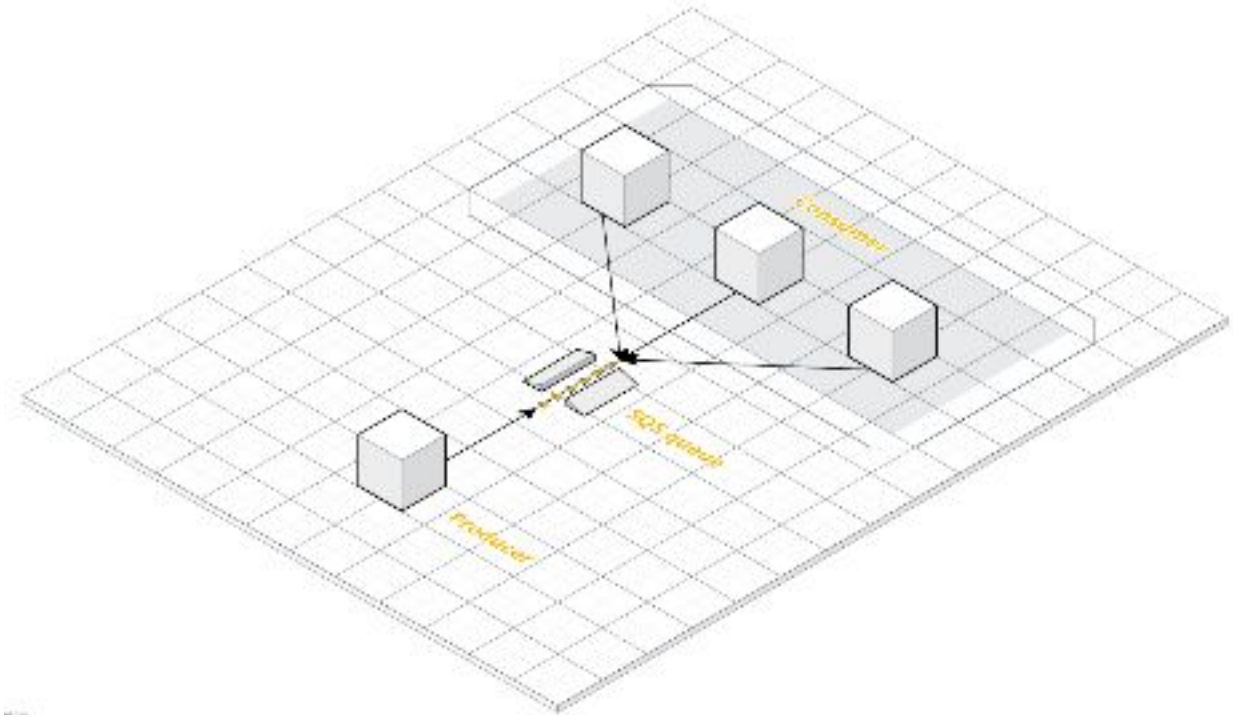
1. User enters the query range: (L,R)
2. Web Server divides the range into multiple sub-ranges, and add them to the Request queue.
3. Each Consumer node independently fetches any unprocessed entry from the request queue.
4. Results to each query are added to the Result queue.
5. Web Server combines the results in Result queue and returns it to user.

Schema

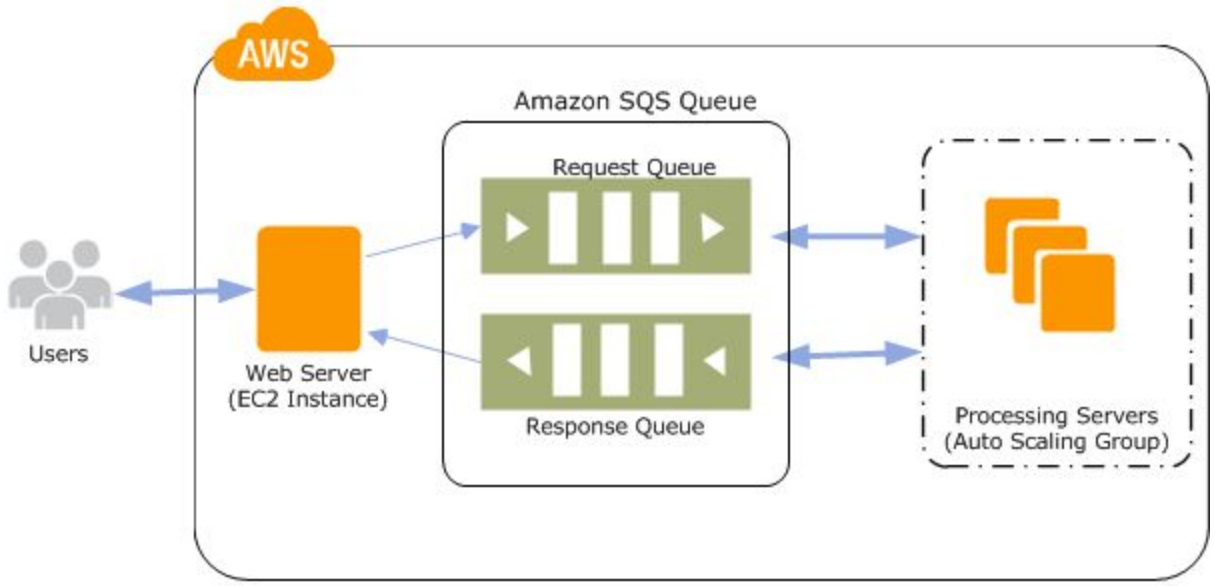


Highlights

- Independent Nodes
- Easily scalable
- Hassle-free management



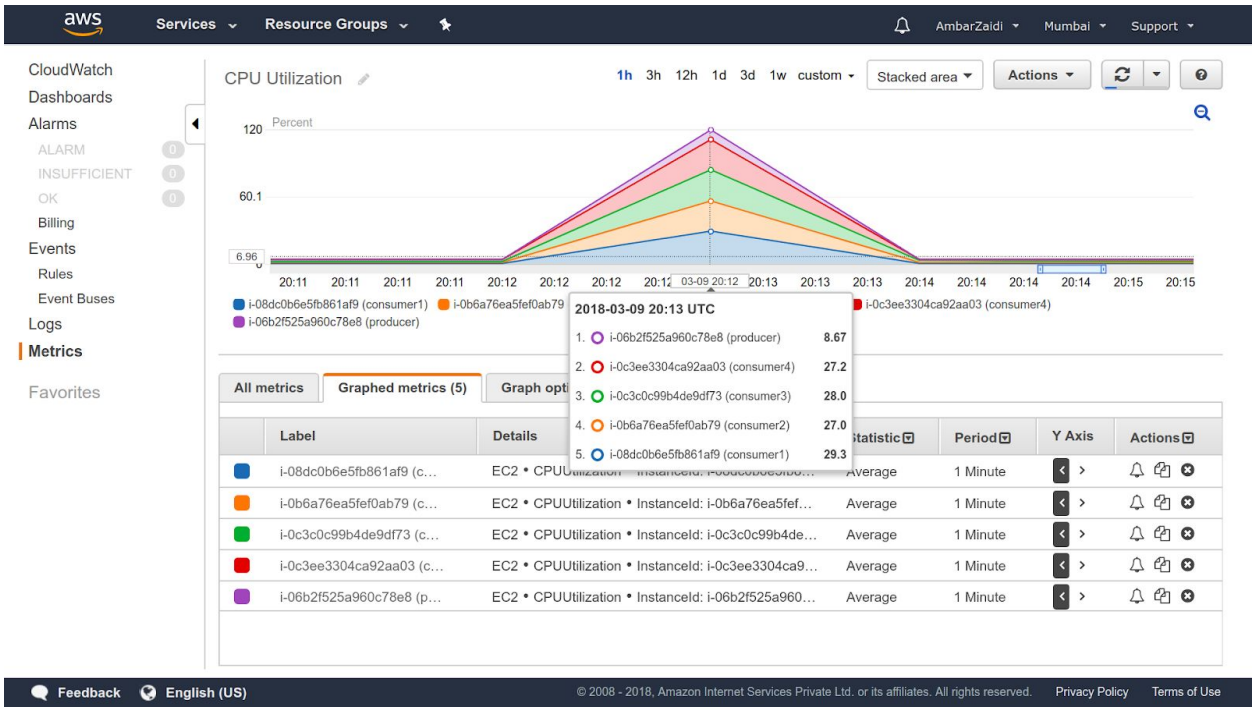
Architecture (schematic)



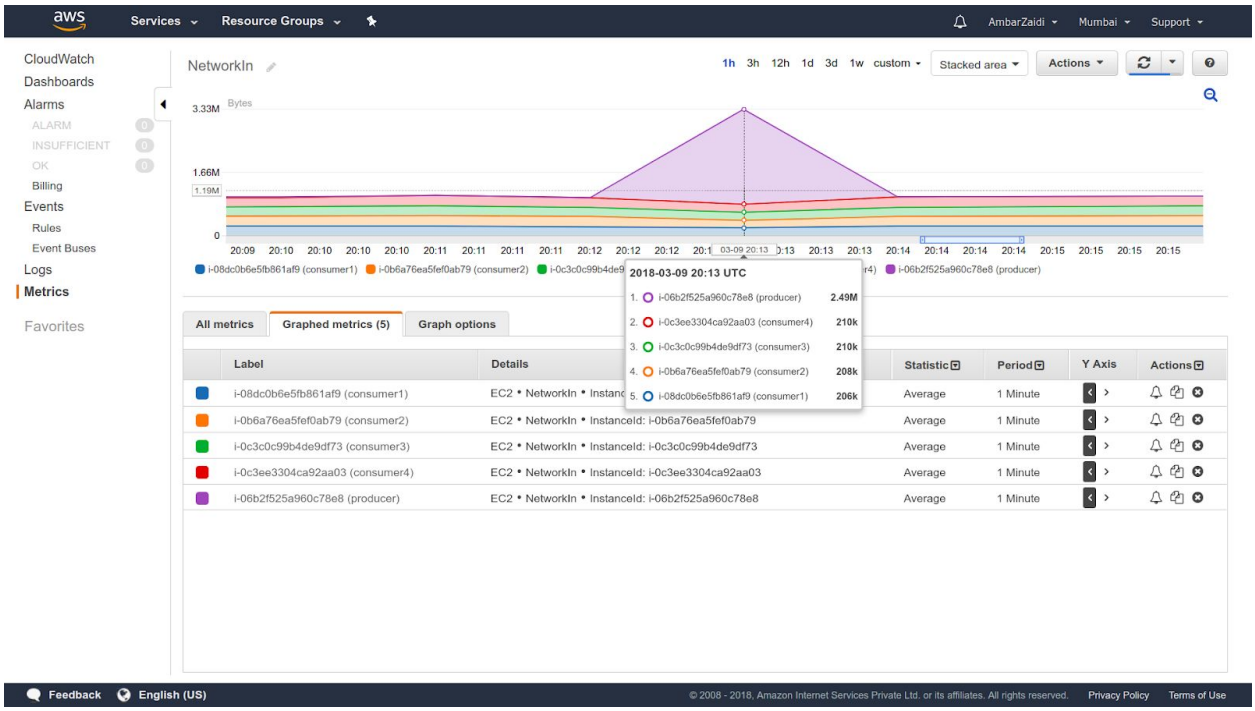
Architecture (detailed)

Metrics

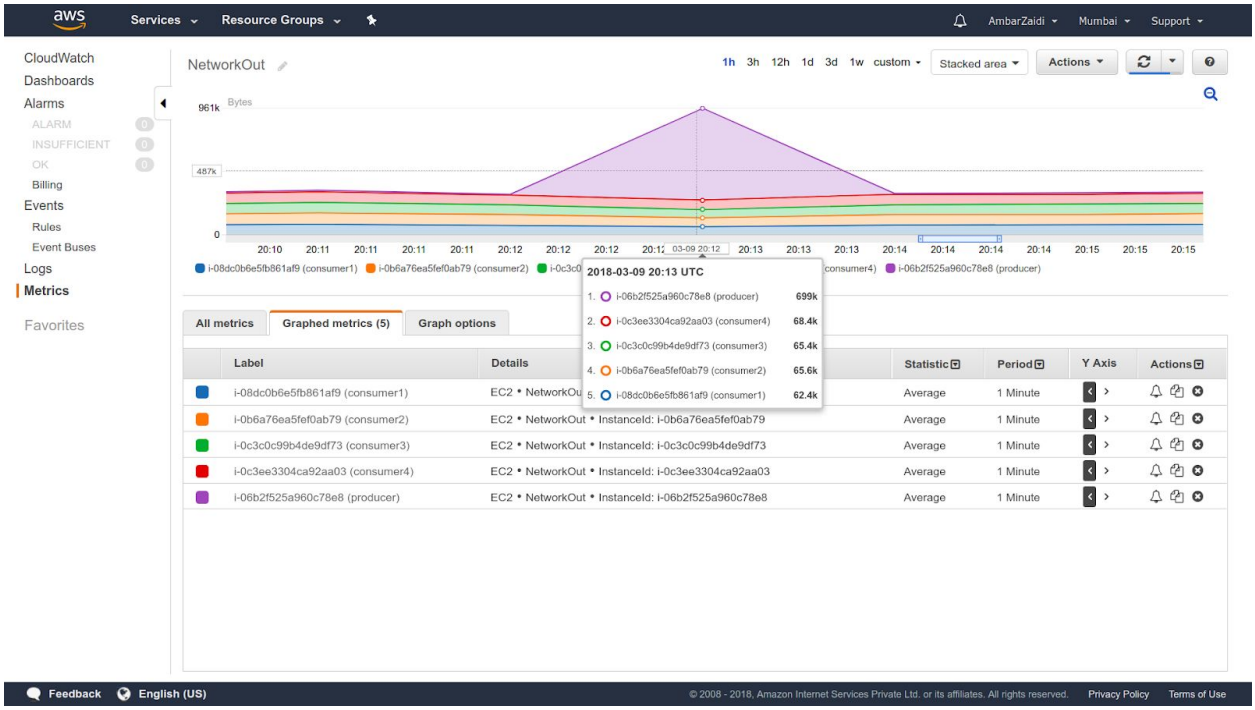
CPU Utilization



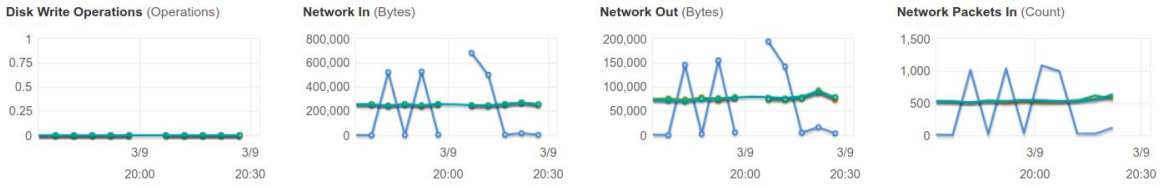
NetworkIn



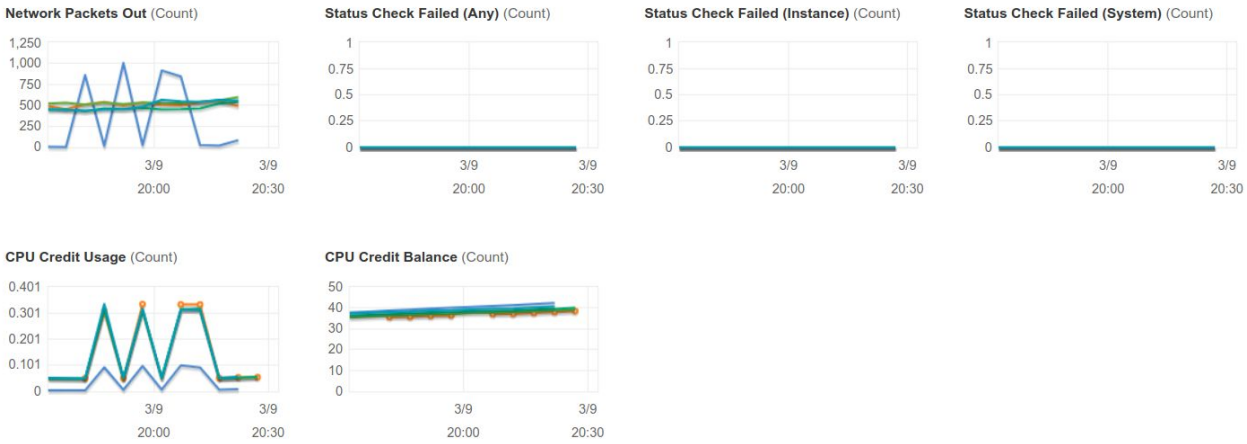
NetworkOut



Disk Usage & Some other metrics



More Metrics



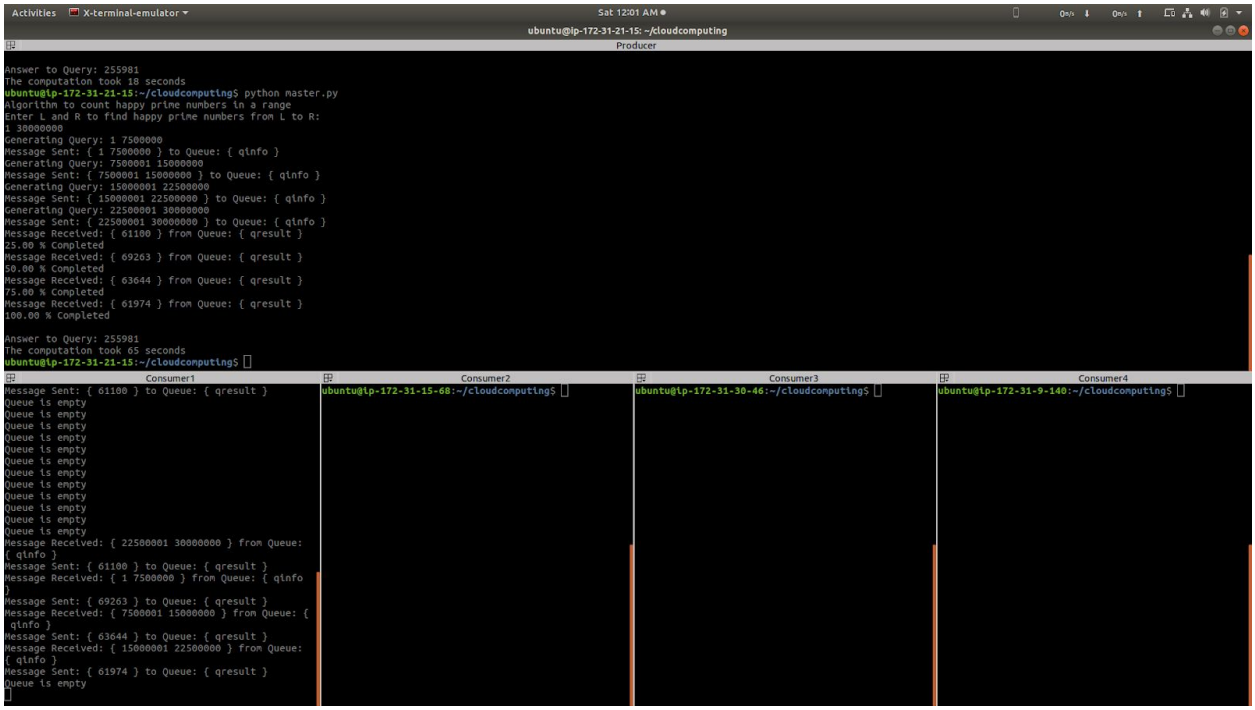
Comparison between running times of Distributed architecture vs Computation on standalone machine

# nodes	Execution time (in seconds)
1	65
2	35
3	33
4	18

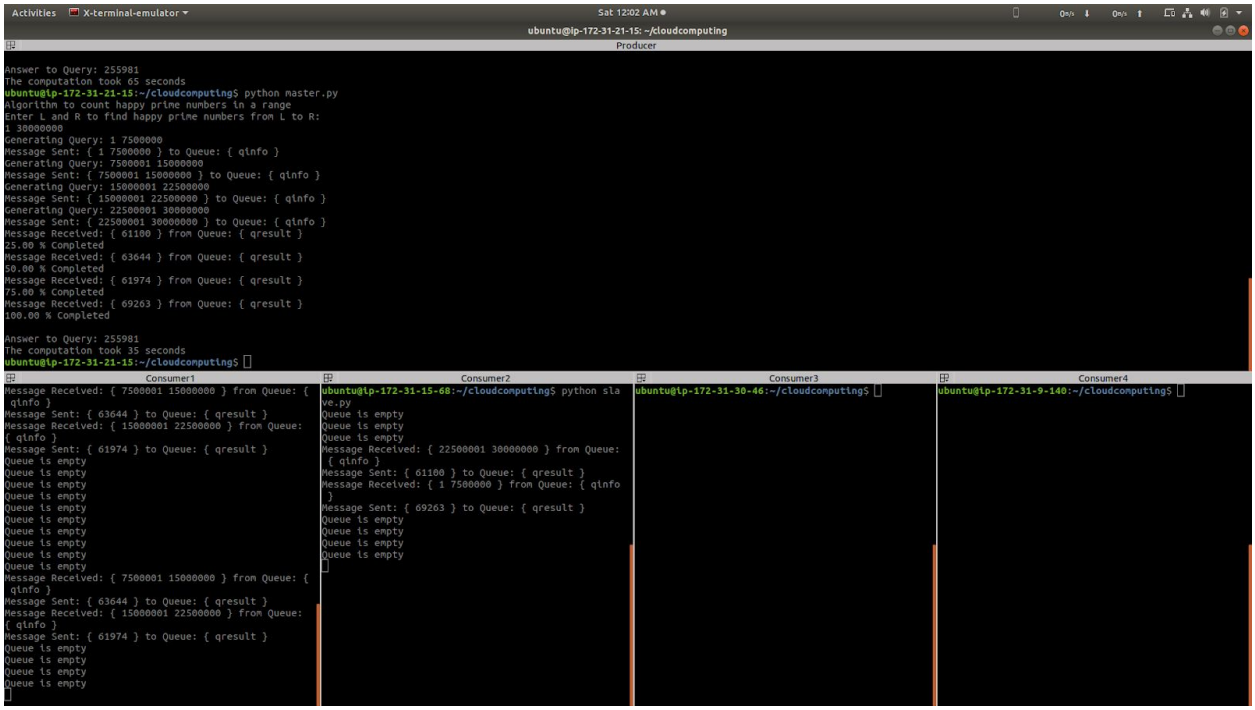
Above results clearly show that we have substantially reduced the execution times using cloud infrastructure with a simplified & scalable architecture.

Screenshots

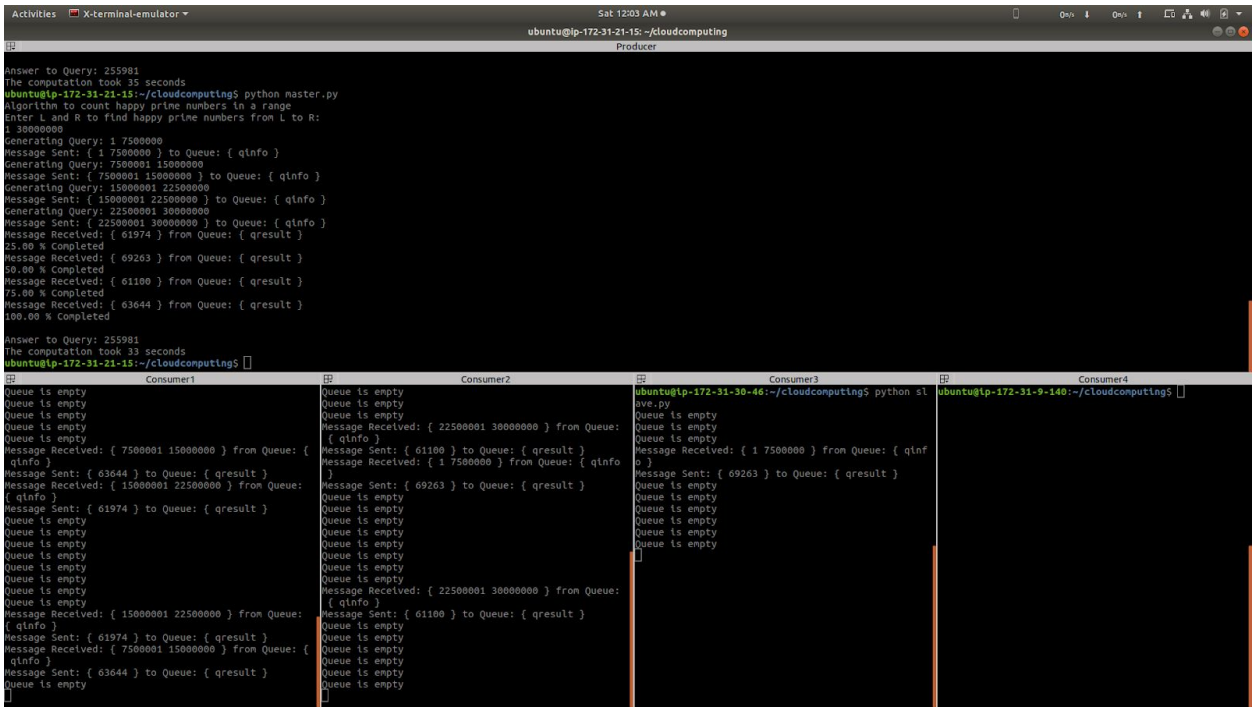
#VMs = 1



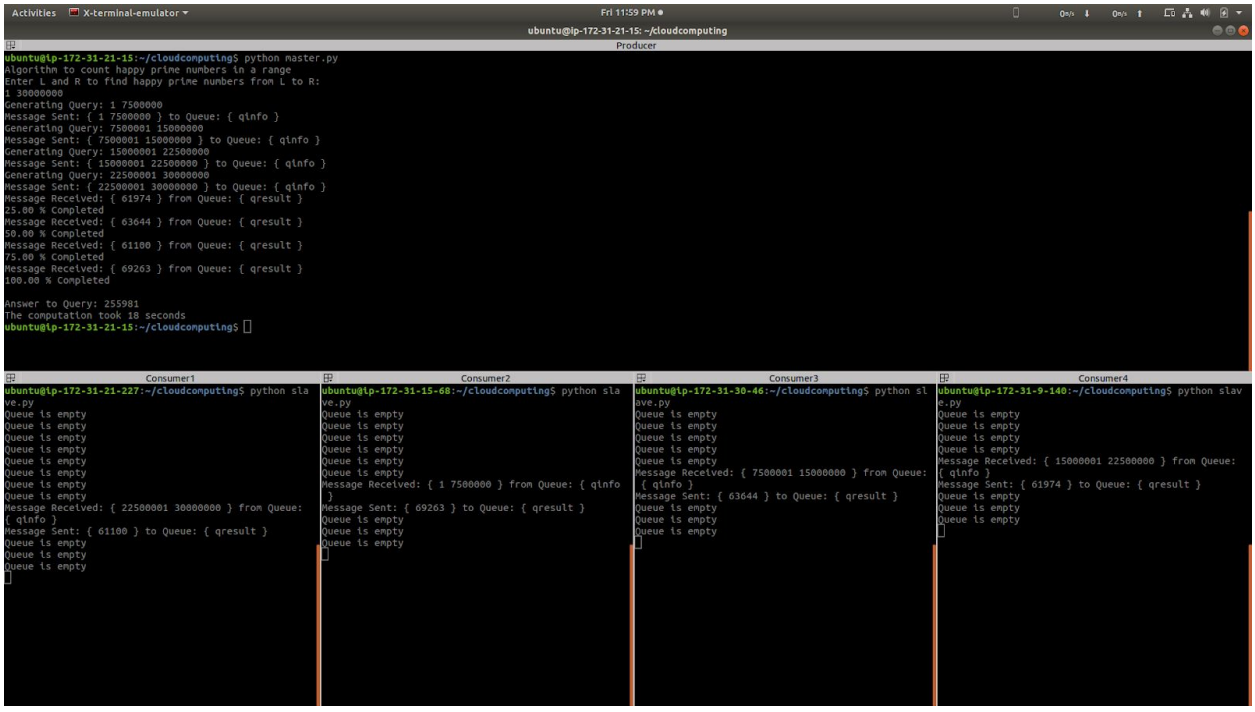
#VMs = 2



#VMs = 3


Activities
X-terminal-emulator
Sat 12:03 AM
ubuntu@ip-172-31-21-15: ~/cloudcomputing
Producer
Answer to Query: 255981
The computation took 35 seconds
ubuntu@ip-172-31-21-15:~/cloudcomputing$ python master.py
Algorithm to count happy prime numbers in a range
Enter L and R to find happy prime numbers from L to R:
1 30000000
Generating Query: 1 7500000
Message Sent: { 1 7500000 } to Queue: { qinfo }
Generating Query: 7500001 15000000
Message Sent: { 7500001 15000000 } to Queue: { qinfo }
Generating Query: 15000001 22500000
Message Sent: { 15000001 22500000 } to Queue: { qinfo }
Generating Query: 22500001 30000000
Message Sent: { 22500001 30000000 } to Queue: { qinfo }
Message Received: { 61974 } from Queue: { qresult }
25.00 % Completed
Message Received: { 69263 } from Queue: { qresult }
50.00 % Completed
Message Received: { 61100 } from Queue: { qresult }
75.00 % Completed
Message Received: { 63644 } from Queue: { qresult }
100.00 % Completed
Answer to Query: 255981
The computation took 33 seconds
ubuntu@ip-172-31-21-15:~/cloudcomputing$
Consumer1
Queue is empty
Message Received: { 7500001 15000000 } from Queue: { qinfo }
Message Sent: { 63644 } to Queue: { qresult }
Message Received: { 15000001 22500000 } from Queue: { qinfo }
Message Sent: { 61974 } to Queue: { qresult }
Message Received: { 15000001 22500000 } from Queue: { qinfo }
Message Received: { 7500001 15000000 } from Queue: { qinfo }
Message Sent: { 63644 } to Queue: { qresult }
Consumer2
Queue is empty
Message Received: { 22500001 30000000 } from Queue: { qinfo }
Message Received: { 1 7500000 } from Queue: { qinfo }
Message Sent: { 61100 } to Queue: { qresult }
Message Sent: { 69263 } to Queue: { qresult }
Message Received: { 22500001 30000000 } from Queue: { qinfo }
Message Sent: { 61100 } to Queue: { qresult }
Consumer3
ubuntu@ip-172-31-30-46:~/cloudcomputing$ python slave.py
Queue is empty
Message Received: { 1 7500000 } from Queue: { qinfo }
Message Sent: { 69263 } to Queue: { qresult }
Consumer4
ubuntu@ip-172-31-9-140:~/cloudcomputing$

#VMs = 4


Activities
X-terminal-emulator
Fri 11:59 PM
ubuntu@ip-172-31-21-15: ~/cloudcomputing
Producer
ubuntu@ip-172-31-21-15:~/cloudcomputing$ python master.py
Algorithm to count happy prime numbers in a range
Enter L and R to find happy prime numbers from L to R:
1 30000000
Generating Query: 1 7500000
Message Sent: { 1 7500000 } to Queue: { qinfo }
Generating Query: 7500001 15000000
Message Sent: { 7500001 15000000 } to Queue: { qinfo }
Generating Query: 15000001 22500000
Message Sent: { 15000001 22500000 } to Queue: { qinfo }
Generating Query: 22500001 30000000
Message Sent: { 22500001 30000000 } to Queue: { qinfo }
Message Received: { 61974 } from Queue: { qresult }
25.00 % Completed
Message Received: { 63644 } from Queue: { qresult }
50.00 % Completed
Message Received: { 61100 } from Queue: { qresult }
75.00 % Completed
Message Received: { 69263 } from Queue: { qresult }
100.00 % Completed
Answer to Query: 255981
The computation took 18 seconds
ubuntu@ip-172-31-21-15:~/cloudcomputing$
Consumer1
ubuntu@ip-172-31-21-227:~/cloudcomputing$ python slave.py
Queue is empty
Message Received: { 22500001 30000000 } from Queue: { qinfo }
Message Sent: { 61100 } to Queue: { qresult }
Consumer2
ubuntu@ip-172-31-15-68:~/cloudcomputing$ python slave.py
Queue is empty
Message Received: { 1 7500000 } from Queue: { qinfo }
Message Sent: { 69263 } to Queue: { qresult }
Consumer3
ubuntu@ip-172-31-30-46:~/cloudcomputing$ python slave.py
Queue is empty
Message Received: { 7500001 15000000 } from Queue: { qinfo }
Message Sent: { 63644 } to Queue: { qresult }
Consumer4
ubuntu@ip-172-31-9-140:~/cloudcomputing$ python slave.py
Queue is empty
Message Received: { 15000001 22500000 } from Queue: { qinfo }
Message Sent: { 61974 } to Queue: { qresult }