Software Architecture Documentation

Estimating Pi using the monte Carlo Method

Course	Computer Communication
Professor	Bongjun Choi Prof.
Major	Computer Science and Engineering
Student ID	20172655
Name	Kangsan Lee
Submission Date	May 21, 2021.

1. Introduction and Background

The purpose of this document is to show the advantages of the development approach using Message Queue. A simple example using 'Zero Message Queue' is included.

Message Queues has the following advantages:

- Allows you to send reliable chunks of data(message).
- Framing is done by the message queue protocol itself.
- Clients using done by the message queue do not have to loop and keep calling functions live recv() until a whole message has arrived.
- Able to mix and match entire population of clients and servers by attaching them to the same messaging framework.

2. High-level design

Message Queues typically supports the following topologies:

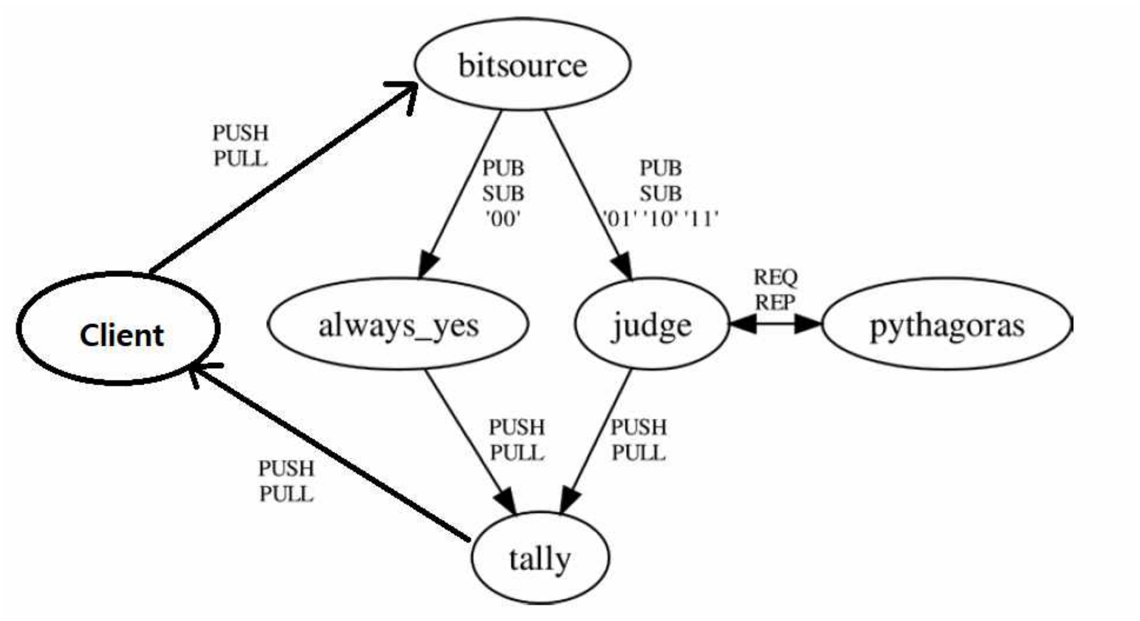
- Pipeline :

  Every queued message is delivered to exactly one customer.
- Publisher-Subscriber :

  publishers do not program the messages to be sent directly to specific subscribers. Publisher categorize messages into classes without knowledge of which subscribers are there, and subscribers specify a filter that narrows their interest to messages with a particular format.
- Request-Reply :

  Messages make a round-trip. Client need to stay connected and wait for the reply, and the queue needs some addressing scheme to correct client.

Example written with ZMQ has following network topology:



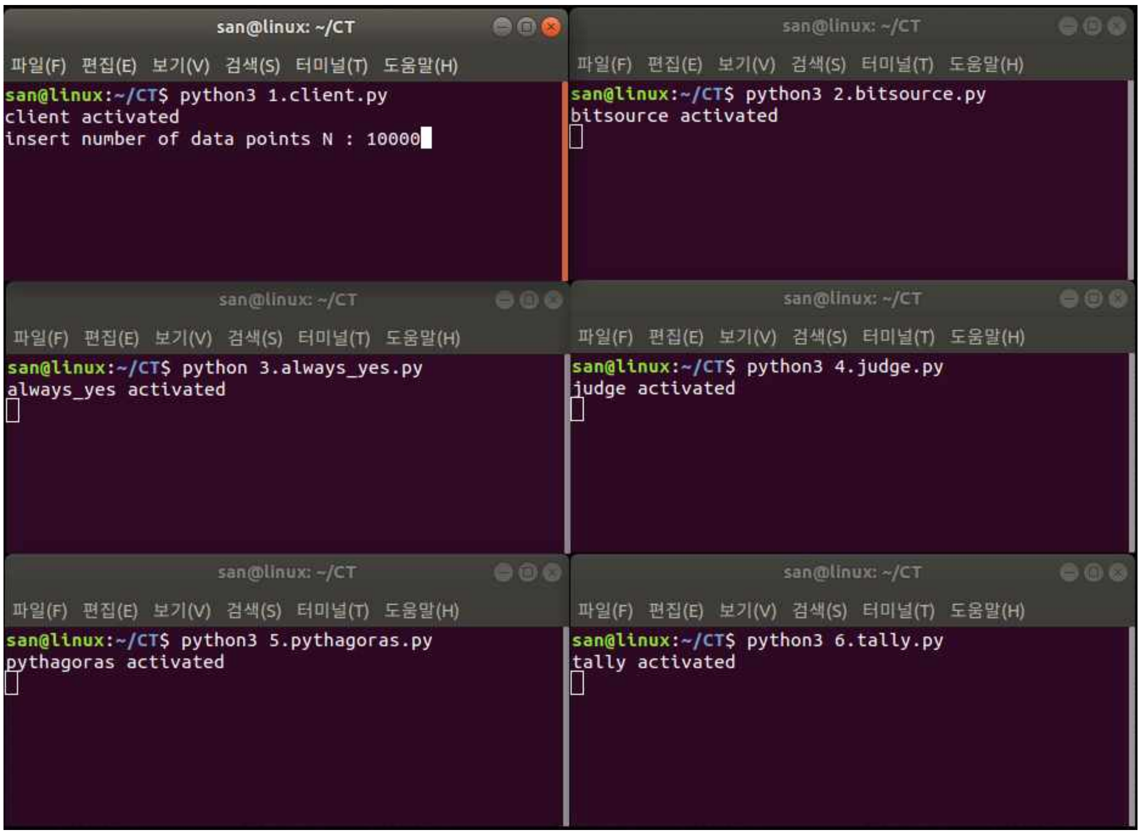
3. Logical View and Process View

- The example is an implementation of a network that estimates PI values using the Monte Carlo method using message queues.
- It has six workers : client, bitsource, always_yes, judge, pythagoras, and tally.
- Each worker has a function for its own operation, and the client prints the change of the PI value in real time.

The following is a brief explanation of the operations of workers.
client : Client asks the user the desired number of data points N, receives (number of iterations, pi value) from tally
bitsource : Produce random points in the unit square.
always_yes : Coordinates in the lower-left quadrant are inside the unit circle.
judge : Determine whether each input coordinate is inside the unit circle
pythagoras : Return the sum-of-squares of number sequences
tally : Tally how many points fall within the unit circle, print PI and send messages to client

4. Test Cases

- Inputed 10000 for N value :



```
san@linux:~/CT$ python3 1.client.py
client activated
insert number of data points N : 10000
```

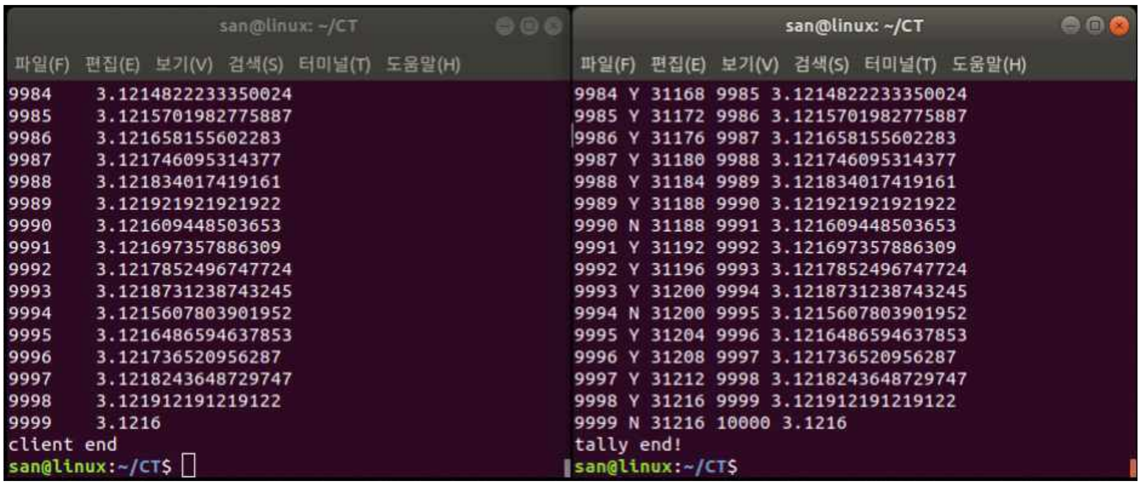
```
san@linux:~/CT$ python3 2.bitsource.py
bitsource activated
```

```
san@linux:~/CT$ python 3.always_yes.py
always_yes activated
```

```
san@linux:~/CT$ python3 4.judge.py
judge activated
```

```
san@linux:~/CT$ python3 5.pythagoras.py
pythagoras activated
```

```
san@linux:~/CT$ python3 6.tally.py
tally activated
```



```
9984     3.1214822233350024
9985     3.1215701982775887
9986     3.121658155602283
9987     3.121746095314377
9988     3.121834017419161
9989     3.121921921921922
9990     3.121609448503653
9991     3.121697357886309
9992     3.1217852496747724
9993     3.1218731238743245
9994     3.1215607803901952
9995     3.1216486594637853
9996     3.121736520956287
9997     3.1218243648729747
9998     3.121912191219122
9999     3.1216
client end
san@linux:~/CT$
```

```
9984 Y 31168 9985 3.1214822233350024
9985 Y 31172 9986 3.1215701982775887
9986 Y 31176 9987 3.121658155602283
9987 Y 31180 9988 3.121746095314377
9988 Y 31184 9989 3.121834017419161
9989 Y 31188 9990 3.121921921921922
9990 N 31188 9991 3.121609448503653
9991 Y 31192 9992 3.121697357886309
9992 Y 31196 9993 3.1217852496747724
9993 Y 31200 9994 3.1218731238743245
9994 N 31200 9995 3.1215607803901952
9995 Y 31204 9996 3.1216486594637853
9996 Y 31208 9997 3.121736520956287
9997 Y 31212 9998 3.1218243648729747
9998 Y 31216 9999 3.121912191219122
9999 N 31216 10000 3.1216
tally end!
san@linux:~/CT$
```

Each workers runs on different terminal.
The PI value was approximated using the Monte Carlo method.