Fun Factoring

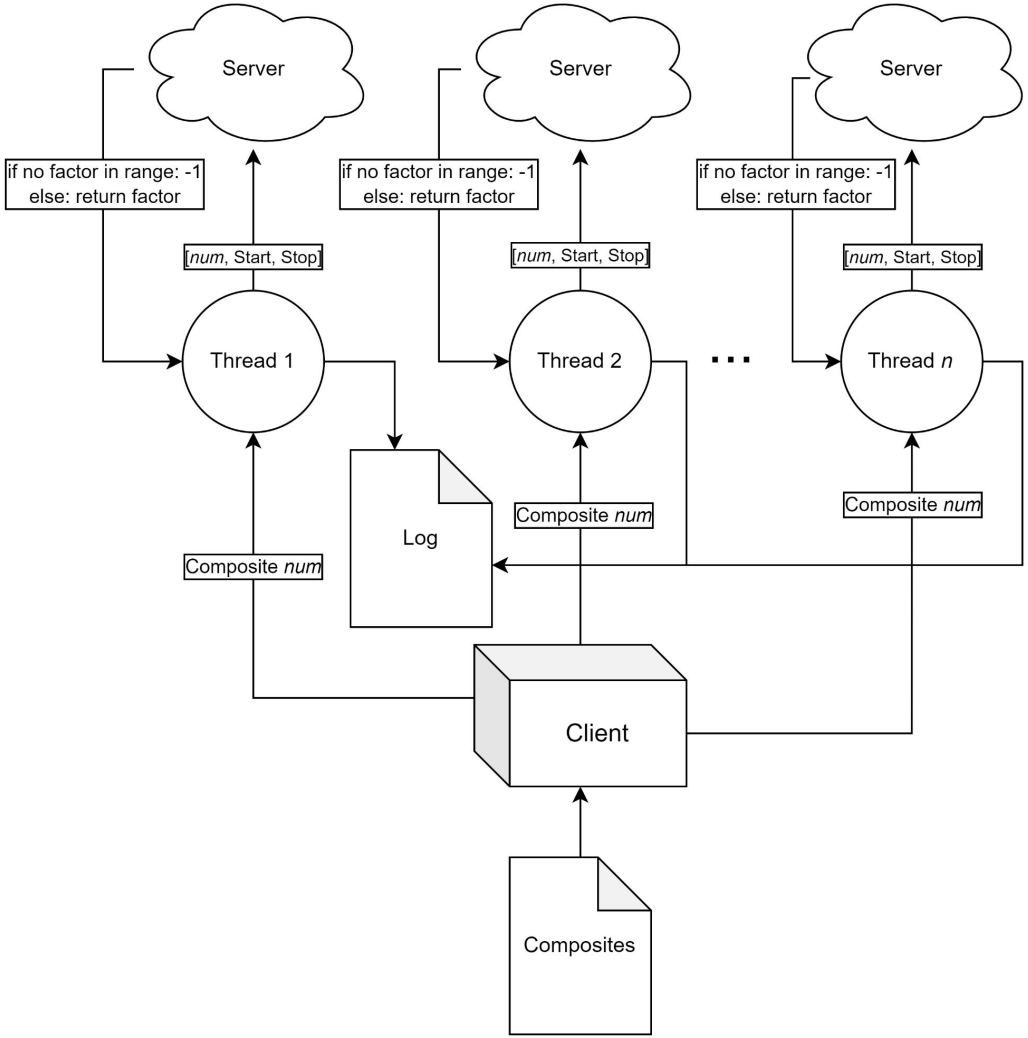
Jayden Patel
CP3

The Assignment

Using Multithreading and Socketing

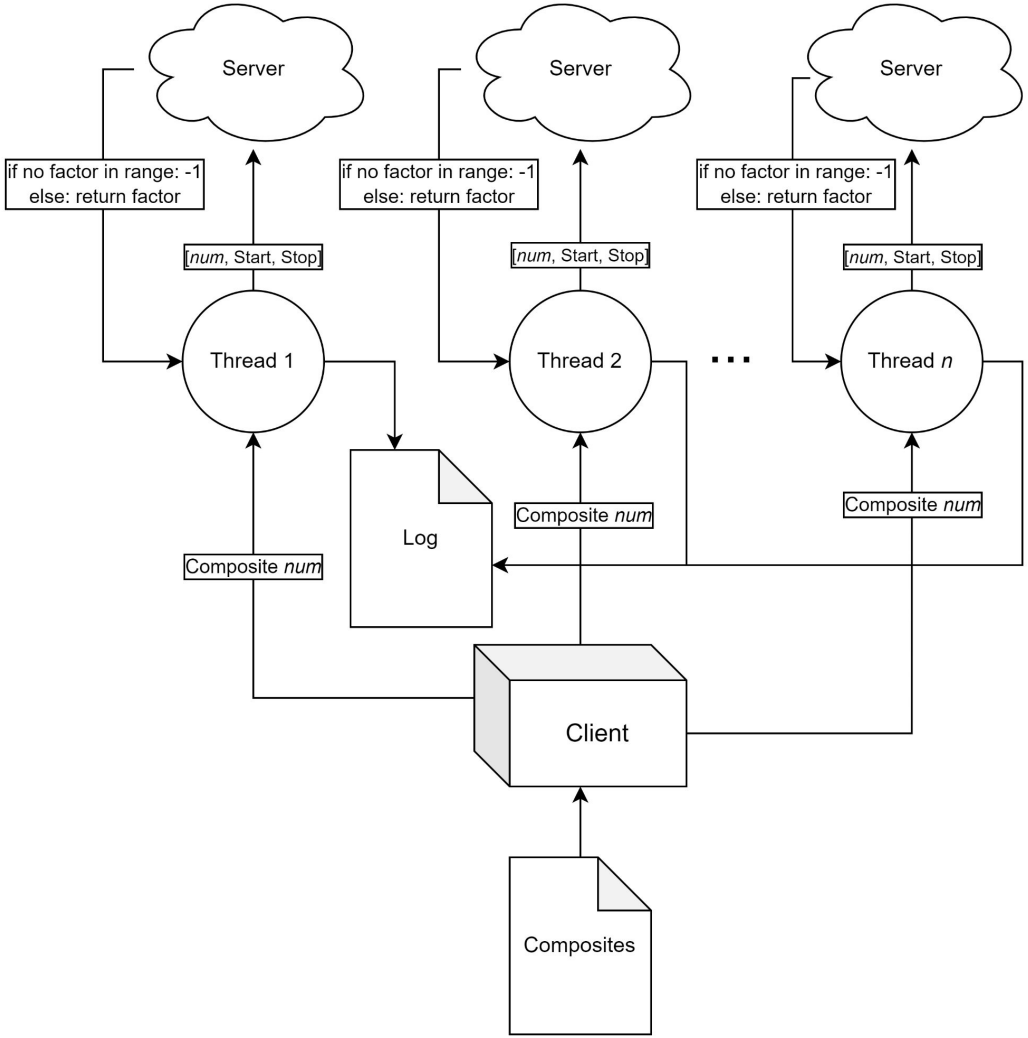
The goal of the assignment is to:

- Factor a list of composite numbers and achieve a total processing time of 500 hours
  - Offload the work to multiple servers at once
  - Requires multithreading and socketing technologies



The Solution

- The client manages up to a defined *n* number of threads
- It continuously factors composites from a file.
- Each thread handles factoring the same composite with their own server.
  - The thread sends the composite number with a search range.
  - The threads split the searching by processing their own search chunk every 50 million.
  - The server tries to find a factor of the composite within the range.



The Solution

When each thread sends or receives a packet, it logs the following data:

- [time stamp] Thread#, SENDING/RECIEVING, serverIP, composite, start, stop
- If the thread is receiving a packet, it also logs the response and the time elapsed from when it sent the initial packet.

```
private static BigInteger findFactor(BigInteger number,
      BigInteger start, BigInteger stop) {

      for (BigInteger i = start; i.compareTo(stop) < 0;
      i = i.add(BigInteger.ONE)) {
            // System.out.println(i);
            if (number.mod(i).equals(BigInteger.ZERO)) {
                  return i;
            }
      }
      return new BigInteger("-1");
}

...
String inputLine = input.readLine();
String[] parts = inputLine.split(" ");
BigInteger compositeNumber = new BigInteger(parts[0]);
BigInteger start = new BigInteger(parts[1]);
BigInteger stop = new BigInteger(parts[2]);
BigInteger factor = findFactor(compositeNumber, start,
stop);
output.println(factor.toString());
...
```

The Server

- The server simply listens for incoming data, then splits the input data into three parts:
  - The Composite
  - The Start Range
  - The Stop Range
- It tries to find a factor in the range by brute force searching
- Then, it sends back the first factor it finds; if no factor is found then it sends back -1.

```
ips =  {"34.71.161.175": False,
        "34.136.161.195": False,
        ...}
```

```
def getOpenIP():
    for ip in ips:
        if not ips[ip]:
            ips[ip] = True
            return ip
```

```
class FactorizationClient(threading.Thread):
    def __init__(self, ip, port, id):
        threading.Thread.__init__(self)
        self.ip = ip
        self.port = port
        self.id = id
...
```

The Client

- A dictionary is used to store each server's IP and their availability.
- The getOpenIP function finds the first IP thats open and allocates it.
- The client is constructed with the IP from the previous function, port, and id.
    - The ID is used to label the thread's log entries.

```
...
start_range = multiple * chunk_size
multiple += 1
stop_range = multiple * chunk_size
...
```

```
...
sock.sendall(f"{composite} {start_range}
      {stop_range}\n".encode())
start = time.perf_counter()
response = sock.recv(1024).decode().strip()
end = time.perf_counter()
total_time += end - start
...
```

The Client

- Globals:
  - multiple (used to sync search ranges)
  - Total_time (total run time used to stop experiment after 500 processing hours)
- Each thread uses the multiple to create their search range, then increases it for the next thread.
- After the threads get a response from their server, they add the time elapsed to the total_time.

```
...
if composite**.5 < stop_range:
      clients -= 1
      ips[self.ip] = False
      return
```

The Client

- The thread loops this process until the composite's square root has been searched through.
  - This is the stop condition.
- Then, it:
  - subtracts from the total number of clients running
  - releases the IP it used
  - ends execution

```
with open("composites.txt", "r") as f:
    composites = f.readlines()
    while composites:
        composite = composites.pop(0).strip()
        composite = int(composite)
        multiple = 0
        clients = 0
        for i in range(max_clients):
            client = FactorizationClient(getOpenIP(),
             4321, i)
            client.start()
            clients += 1
        while clients > 0:
            pass
```

The Client

- After the program feeds the clients a composite number to factor, it waits until all threads have reached the stop_condition.
- Then, it moves onto the next composite in the list.
- The program was manually stopped once it reached 500 hours of total processing time.

[2024-05-04 18:22:37] CLIENT15, SENDING, 34.134.18.205, 395545720216423200 2933, 27650000000, 27700000000

[2024-05-04 18:22:37] CLIENT5, SENDING, 34.29.234.137, 39554572021642320029 33, 27700000000, 27750000000, -1, 21.540002300000197

[2024-05-04 18:22:37] CLIENT5, SENDING, 34.29.234.137, 39554572021642320029 33, 27700000000, 27750000000

[2024-05-04 18:22:38] CLIENT8, RECIEVING, 35.223.112.97, 39554572021642320029 33, 27700000000, 27750000000, -1, 21.726286699999037

[2024-05-04 18:22:38] CLIENT8, SENDING, 35.223.112.97, 39554572021642320029 33, 27750000000, 27800000000

[2024-05-04 18:22:42] CLIENT7, RECIEVING, 34.30.207.159, 39554572021642320029 33, 27750000000, 27800000000, -1, 10.01301550000062

[2024-05-04 18:22:42] CLIENT7, SENDING, 34.30.207.159, 39554572021642320029 33, 27800000000, 27850000000

[2024-05-04 18:22:44] CLIENT9, RECIEVING, 35.238.11.71, 39554572021642320029 33, 27800000000, 27850000000, -1, 21.726941999999326

[2024-05-04 18:22:44] CLIENT9, SENDING, 35.238.11.71, 39554572021642320029 33, 27850000000, 27900000000

[2024-05-04 18:22:45] CLIENT14, RECIEVING, 35.202.247.41, 39554572021642320029 33, 27850000000, 27900000000, -1, 21.60406989999865

[2024-05-04 18:22:46] CLIENT14, SENDING, 35.202.247.41, 39554572021642320029 33, 27900000000, 27950000000

[2024-05-04 18:22:46] CLIENT11, RECIEVING, 34.171.165.99, 39554572021642320029 33, 27900000000, 27950000000, -1, 20.72350870000082

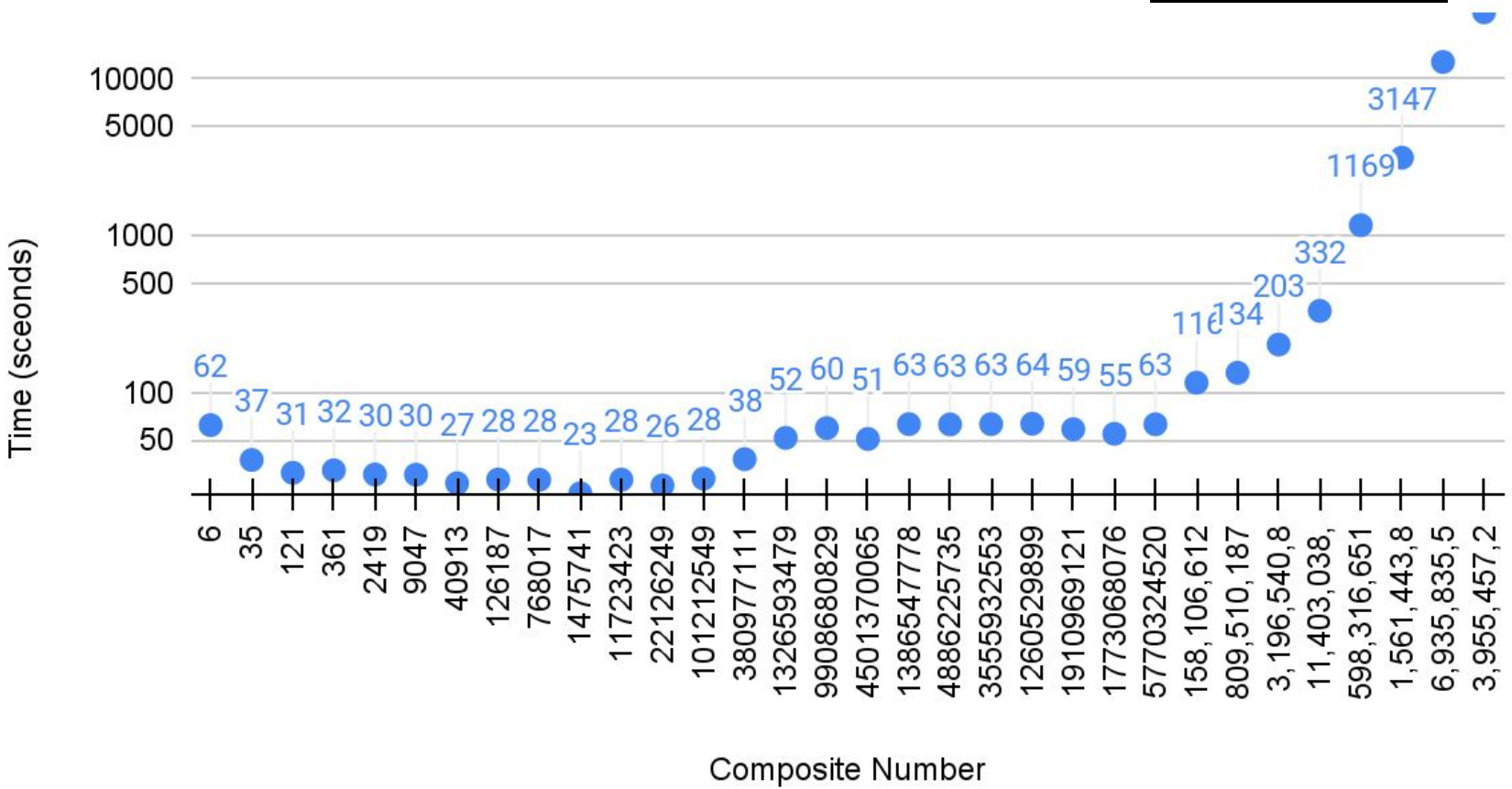
[2024-05-04 18:22:47] CLIENT15, RECIEVING, 34.134.18.205, 39554572021642320029 33, 27900000000, 27950000000, -1, 9.745997300000454

Starting at Client15 sending a packet…

It gets a response about 10 seconds later with no factors found in the given range.

Total Time for Each Composite Factored

Max: 1006664 secs



Fun Factoring

Jayden Patel
CP3