

다중 클라이언트 연결 웹 서버 구조와 성능 테스트

컴퓨터네트워크 4주차

공과대학 5호관 633호
데이터 네트워크 연구실

조교 : 황동준

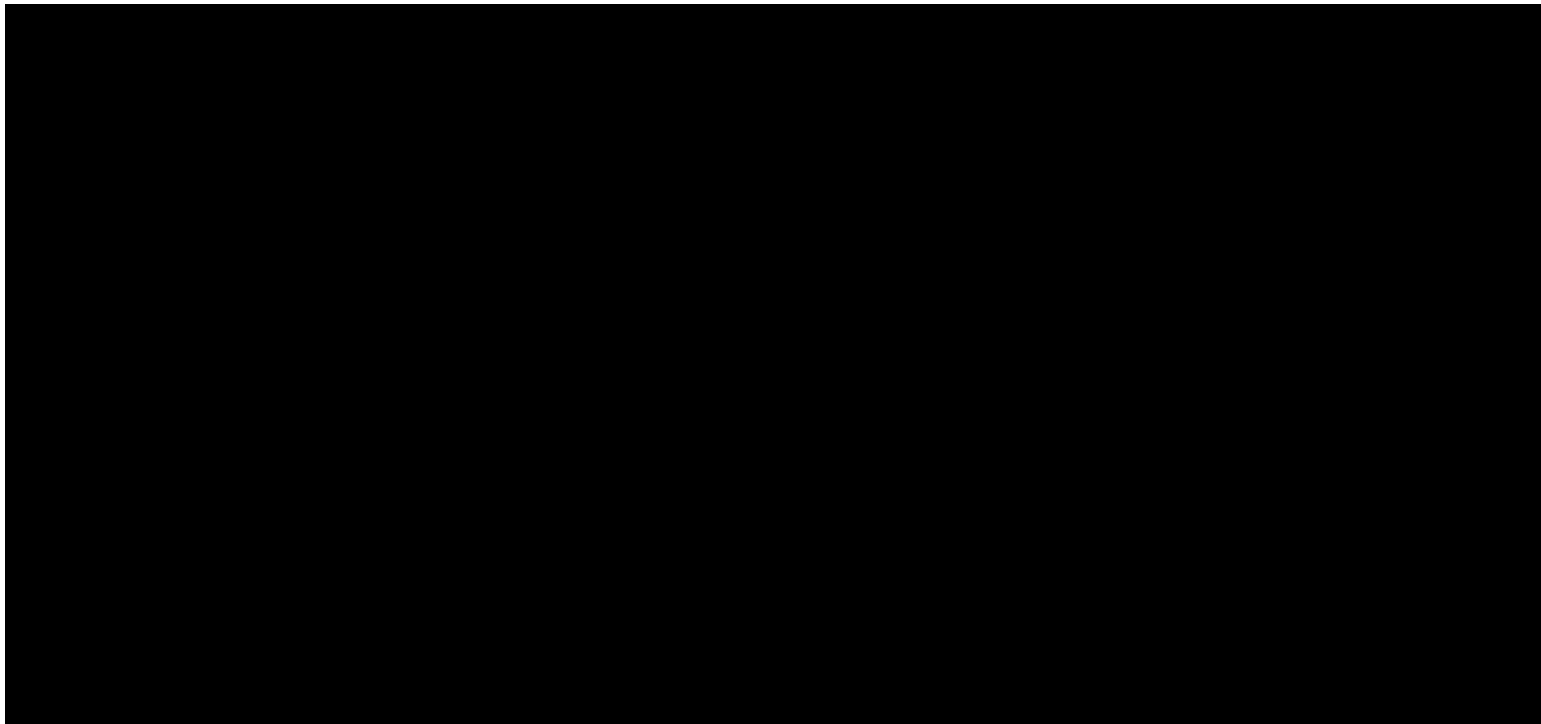
실습

1. 단일 프로세스를 사용한 클라이언트 웹 서버 구현
 - 서버는 Socket Programming
2. locust를 사용한 성능 테스트

과제

1. 다중 클라이언트 웹 서버 와 성능테스트 (10점)
 - HTTP 구현 (6점)
 - Multi Process
 - Multi Thread
 - Locust를 사용한 서버 성능 테스트 (4점)
 - Statistics와 chart이미지 캡처 및 설명

실습 데모 영상



실습1 : 단일 프로세스를 사용한 클라이언트 웹 서버 구현

Socket Programming - 단일 프로세스

- 데이터가 send / recv가 완료될 때까지 해당 프로세스가 block 상태로 전환
- 다른 프로세스에게 자원을 넘김
- 동시에 두가지 이상의 일을 처리하지 못함

```
import socket
import time

def main(port):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('', port))
    server_socket.listen()

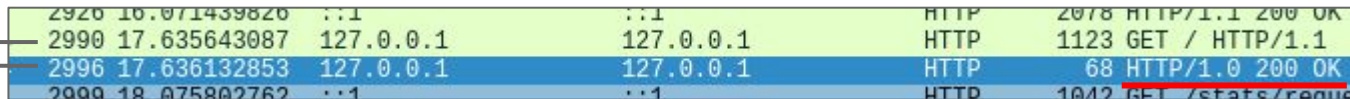
    while True:
        (csocket, address) = server_socket.accept()

        data = csocket.recv(1024)
        print(f"[Client {address} Info] {data.decode()}")
        res = "HTTP/1.1 200 OK\nContent-Type: text/html\n\n"
        csocket.send(res.encode('utf-8'))
        csocket.send(data)
        csocket.close()

if __name__ == "__main__":
    port = 8889
    main(port)
```

Socket Programming - 단일 프로세스

```
res = "HTTP/1.1 200 OK\nContent-Type: text/html\n\n"  
csocket.send(res.encode('utf-8'))  
csocket.send(data)  
csocket.close()
```



The image shows a network traffic capture with several rows. The row with sequence number 2996 is highlighted in blue and contains the text 'HTTP/1.0 200 OK' in red. An arrow points from the code block above to this specific row in the capture.

2920	10.071439820	::1	::1	HTTP	2078 HTTP/1.1 200 OK
2990	17.635643087	127.0.0.1	127.0.0.1	HTTP	1123 GET / HTTP/1.1
2996	17.636132853	127.0.0.1	127.0.0.1	HTTP	68 HTTP/1.0 200 OK
2999	18.075802762	::1	::1	HTTP	1042 GET /stats/requr

1. #2990 클라이언트는 서버에게 GET 메시지로 요청을 보냄
2. 서버는 메시지를 잘 받았다는 표시를 클라이언트에게 보내고 싶음
3. #2996 헤더에 HTTP/1.0 200 OK 를 통해 요청이 성공적으로 수행되었음을 알림

- 추가적으로 서버는 Content-Type을 통해 자원의 형식을 명시하기 위해 헤더에 정보를 넣음

<https://yunzema.tistory.com/186>

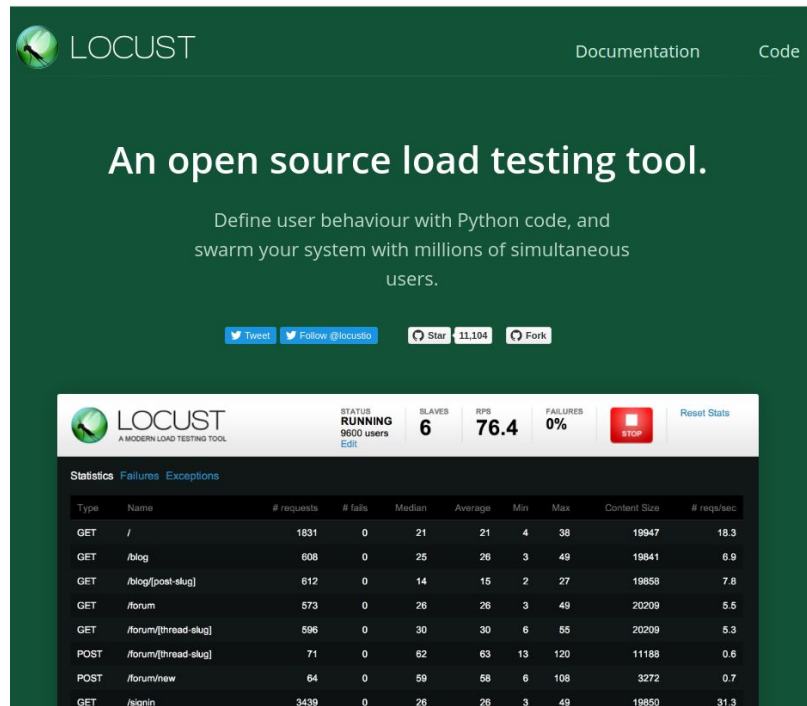
<https://hi098123.tistory.com/200>

실습2 : Locust를 사용한 서버 성능 테스트

서버 성능 테스트 도구 - Locust

- 설치 : `pip3 install locust==1.1`
 - version에 맞춰서 설치할 것
- python 코드로 작성하여 실행 가능
- 가상 유저 수(Number of users to simulate)와 초당 실행 수(users spawned/second)를 입력

⇒ Start swarming 클릭



Locust 구현 및 실행

- Locust 설치
 - pip3 install locust
- Locust 실행
 - locustfile.py 파일 생성
 - locust -f locustfile.py
- Locust 접속
 - <http://locust.ip.address:8089>
 - Number of total users to simulate = 10000와 Hatch rate = 1000 입력

The image shows a web interface titled "Start new Locust swarm" on a dark green background. It contains three input fields and a button. The first field is labeled "Number of total users to simulate" and is empty. The second field is labeled "Hatch rate (users spawned/second)" and is empty. The third field is labeled "Host" and contains the text "http://10.0.2.15:8888". Below these fields is a green button labeled "Start swarming". Three arrows point from the input fields to Korean text labels on the right: the first arrow points from the "Number of total users to simulate" field to "요청하는 전체 클라이언트 수(요청)", the second arrow points from the "Hatch rate" field to "초당 늘어나는 클라이언트 수 (요청)", and the third arrow points from the "Host" field to "요청하는 서버의 주소".

Field Label	Field Value	Annotation
Number of total users to simulate		요청하는 전체 클라이언트 수(요청)
Hatch rate (users spawned/second)		초당 늘어나는 클라이언트 수 (요청)
Host	http://10.0.2.15:8888	요청하는 서버의 주소

Start swarming

서버 성능 테스트 결과

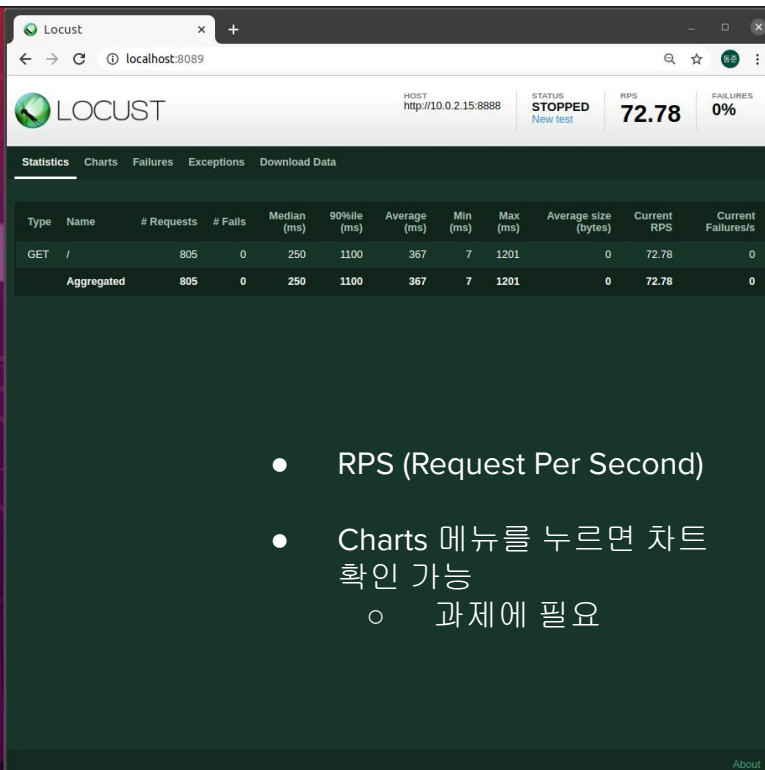
이미지에는 없지만 결과 영상과 동일하게 failure가 발생할때까지 진행해볼 것!
⇒ 단일 프로세스가 처리할 수 있는 양을 넘었기 때문

```
datanetwork@datanetwork-VirtualBox: ~/server_stress_test
p-alive\r\n\r\n'
[2020-08-28 23:35:10,014] datanetwork-VirtualBox\WARNING\urllib3.connectionpool
: Failed to parse headers (url=http://10.0.2.15:8888/): [MissingHeaderBodySepar
atorDefect()], unparsed data: 'GET / HTTP/1.1\r\nHost: 10.0.2.15:8888\r\nUser-A
gent: python-requests/2.22.0\r\nAccept-Encoding: gzip, deflate\r\nAccept: */*\r
\nConnection: keep-alive\r\n\r\n'
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/urllib3/connectionpool.py", line 441, in
_nake_request
    assert_header_parsing(httplib_response.msg)
  File "/usr/lib/python3/dist-packages/urllib3/util/response.py", line 71, in a
ssert_header_parsing
    raise HeaderParsingError(defects=defects, unparsed_data=unparsed_data)
urllib3.exceptions.HeaderParsingError: [MissingHeaderBodySeparatorDefect()], un
parsed data: 'GET / HTTP/1.1\r\nHost: 10.0.2.15:8888\r\nUser-Agent: python-requ
ests/2.22.0\r\nAccept-Encoding: gzip, deflate\r\nAccept: */*\r\nConnection: kee
p-alive\r\n\r\n'

datanetwork@datanetwork-VirtualBox: ~/server_stress_test
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive

client 3133] GET / HTTP/1.1
Host: 10.0.2.15:8888
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive

client 3133]
Traceback (most recent call last):
  File "single_process.py", line 24, in <module>
    main(port)
  File "single_process.py", line 20, in main
    send_recv(clientsocket)
  File "single_process.py", line 10, in send_recv
    client_socket.send(data)
BrokenPipeError: [Errno 32] Broken pipe
datanetwork@datanetwork-VirtualBox: ~/server_stress_test$
```

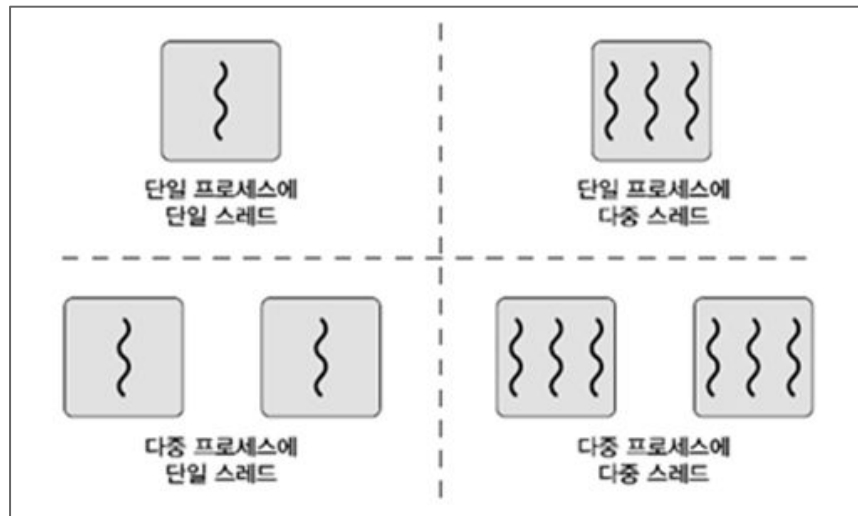


- RPS (Request Per Second)
- Charts 메뉴를 누르면 차트 확인 가능
 - 과제에 필요

과제 : 다중 클라이언트 웹 서버와 성능 테스트

다중 클라이언트란?

- 단일 프로세스가 아닌 멀티 프로세스와 멀티 쓰레드로 생성할 수 있음
- 멀티 프로세스
 - 두개 이상의 프로세스가 작업을 처리
- 멀티 쓰레드
 - 프로세스내에서 생성되는 하나의 주체



Multi Process

1. accept()
2. clients list에 client 넣기
3. 멀티 프로세스 실행
4. 1 ~ 4를 반복

-
- except (예외발생시)
 - proc.join()으로 멀티 프로세스 중지

```
from multiprocessing import Process
import socket
import time
```

```
def multiprocess_sr(csocket, addr):
```

1. Client로부터 데이터를 받음
2. HTTP 200 OK, Content-Type 전송
3. close()

```
def main(port):
```

```
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('', port))
    server_socket.listen()
    req_clients = list()
```

```
    try:
```

← 설명

```
    except:
```

```
        proc.join()
```

```
if __name__ == "__main__":
```

```
    port = 8890
    main(port)
```

Multi Thread

1. accept()
2. 멀티 쓰레드 실행
3. 1 ~ 3을 반복

-
- except (예외발생시)
 - th.join()으로 멀티 쓰레드 중지

```
from threading import Thread
import socket
import time
```

```
def multithread_sr(csocket, addr):
```

1. Client로부터 데이터를 받음
2. HTTP 200 OK, Content-Type 전송
3. close()

```
def main(port):
```

```
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('', port))
    server_socket.listen()
    req_clients = list()
```

```
    try:
```

← 설명

```
    except:
```

```
        th.join()
```

```
if __name__ == "__main__":
```

```
    port = 8891
```

```
    main(port)
```

과제 참고 사항

- Single Process // Multi Process // Multi Thread 를 모두 직접 구현하여 비교
 - Locust를 사용한 성능 테스트이며, failure가 발생할때까지 진행할 것!
 - Statistics, Charts에 대한 이미지 및 설명 필수
 - 오랫동안 failure가 발생하지 않을 경우 요청하는 총 클라이언트 수만큼 올라갔을 때, Stop을 눌러서 비교할 것
 - 설명 형식 (반드시 지킬 것)
failure일 경우 ⇒ “RPS가 XXX.X이며, XXXXX번의 요청을 했을 시 failure가 발생한다. 그리고 평균 응답시간은 XXms이다.”
failure가 아닐 경우 ⇒ “총 요청 XXXXX번의 RPS가 XXX.X이며, 그리고 평균 응답시간은 XXms이다.”
 - 수집한 데이터를 토대로 어떤 방법이 효율적인지, 왜 그렇게 생각하는지도 보고서에 작성!

과제 참고 사항

- failure이 발생하지 않아 다른 방법으로 과제를 진행했다면 failure가 발생하지 않는다는 것을 보고서에 작성할 것!
 - ex) 상당히 오랫동안 요청을 유지해도 failure이 발생하지 않는 이미지라던가 영상으로 녹화해서 유튜브 링크를 보고서에 넣어도 좋음
 - 결론적으로 failure가 아닌 다른 방법을 사용했다는 증명을 할 것
- failure를 발생시키기 위해서는 기존 Number of total users to simulate = 10000와 Hatch rate = 1000 의 값들을 늘려서 시도해볼 수 있음
 - 실험은 1번이 아닌 여러번 시도해볼 것
- Single process, Multi process, Multi thread 방법 중 어떤 것은 failure가 발생하고 어떤 것은 발생 안할 수 있음
 - 실험은 가능한 모두 진행할 것 (failure가 발생하지 않는 것들은 방법이 없지만 failure가 발생하는 것들은 실험 결과 보고서에 첨부)

과제 제출

- 과제 제출 기한
 - 2021년 10월 7일 17:59 까지 사이버 캠퍼스에 제출
- 제출 파일 (.zip)
 - python파일 (.py)
 - 보고서 (.pdf)
- 딜레이 1일당 20%감점
 - ex) 10점 만점 기준, 1일 딜레이인 경우 8점
- 과제 **카피 적발시** 보여준사람 본사람 모두 0점처리
 - 카피 적발기준 : 과제 유사도 80%이상

유의사항

- 파일명 : CN02_04_학번_이름.zip
 - python파일
 - (파일명 : 학번_multiprocess.py / 학번_multithread.py / 학번_locustfile.py)
 - 보고서 : PDF로 제출할것 (파일명 : 04_학번_이름.pdf)
 - 과제 목표 (도출해야할 결과)
 - 코드 설명과 과제 해결 방법 (실행 결과 사진도 보고서에 첨부, 코드 및 설명 필요)
 - Single Process vs. Multi Process vs. Multi Thread 에 대한 비교 값 및 설명 필수
 - 과제 느낀점 및 하고싶은 말 (선택사항)
 - 형식 지켜지지 않을시 채점대상에서 제외 (보고서도 HWP, DOC은 채점대상에서 제외)
 - python파일과 보고서 압축하여 제출
- 질문
 - 디스코드 컴퓨터네트워크 채널에서 질문을 받음