**LAPORAN TUGAS 10**

**PEMROGRAMAN JARINGAN**

**“Performance Test”**



**Oleh**

Nama     :  Isnaini Nurul KurniaSari

NRP       :  05111740000010

Kelas      :  Pemrograman Jaringan C

**Tahun Pelajaran 2019/2020**

**Departemen Teknik Informatika**

**Institut Teknologi Sepuluh Nopember**

**SOAL TUGAS 10**

Dengan semakin populernya aplikasi/sistem informasi berbasis web, maka potensi diakses ribuan bahkan ratusan ribu user sangat besar. Apapun jenis model processingnya (threaded / asynchronous), pasti akan mempunyai batas kemampuan. Dalam hal ini yang bisa dilakukan adalah

1. Scale up → meningkatkan kapasitas memory dan kemampuan cpu (menambah core), namun hal ini tidak murah
2. Scale horizontal → melayani request dengan memanfaatkan resource komputasi dari mesin yang berbeda beda. Biaya tidak terlalu besar, karena dapat memanfaatkan komputer2 komoditi, atau bahkan laptop untuk bisa berpartisipasi dalam melayani request.

Pattern atau pola operasi yang digunakan pada umumnya adalah

1. Load balancer → bertugas sebagai front end yang akan menerima request dari user, dan kemudian men-DISTRIBUSIKAN-nya ke BACKEND-server
2. Backend server → bertugas menerima request dari load balancer, memprosesnya dan kemudian mengembalikannya ke load balancer untuk diteruskan ke client yang memintanya

Pada folder progjar6, terdapat file async\_server.py seperti pada progjar5 yang bertindak sebagai server dengan asynchrnous processing. Ini yang akan dimanfaatkan sebagai BACKEND server. Lb.py merupakan load balancer yang bertugas mendistrubusikan request ke backend yang didefinisikan di class BackendList. Dalam lb.py, fungsi getserver pada class Backendlist akan membuat backend dipilih secara round robin atau bergantian secara fair.

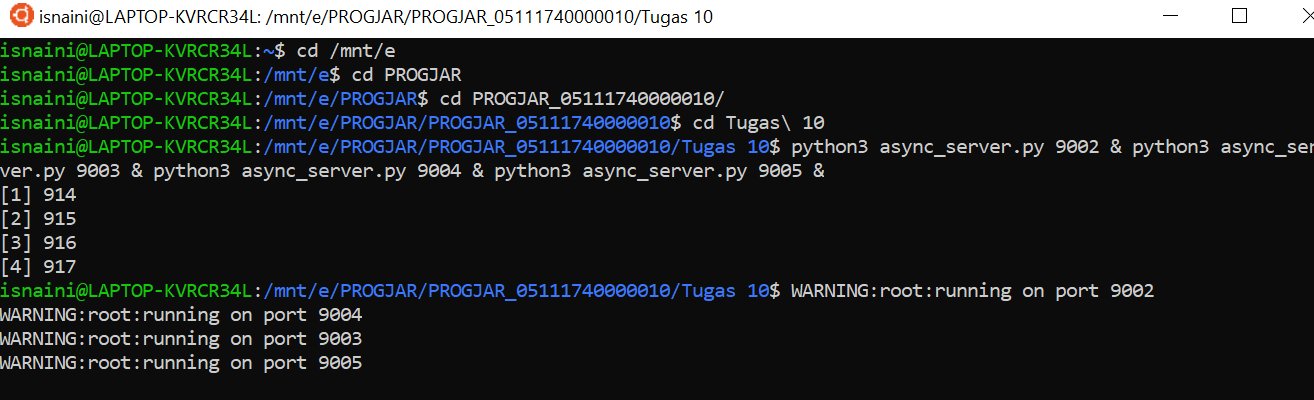
1. Pull update terbaru
2. Jalankan async\_server.py pada port 9002, 9003, 9004, 9005 (lihat pada BackendList)
3. Jalankan file lb.py, jalankan di port 44444
4. Jalankan browser, akseslah <http://localhost:44444/page.html>
5. Lihatlah di log program, bahwa setiap request akan dilayani oleh backend yang bergantian
6. Lakukan performance test seperti pada tugas 9, bandingkan penggunaan load balancer dengan async\_server dengan server\_thread\_http pada folder progjar5
7. Buatlah tabel hasilnya

**PENJELASAN**

1. Pull Update Terbaru
2. Menjalankan async\_server.py dengan port 9002, 9003, 9004, 9005 dengan menggunkan WSL

Dengan menggunakan perintah :

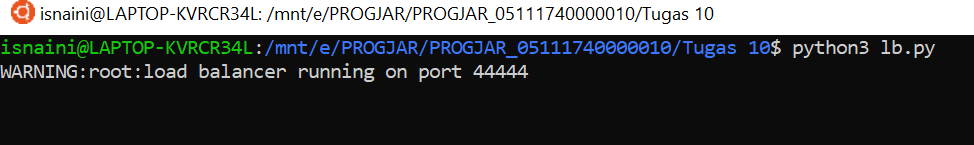
“python3 async\_server.py 9002 & python3 async\_server.py 9003 & python3 async\_server.py 9004 & python3 async\_server.py 9005 &”



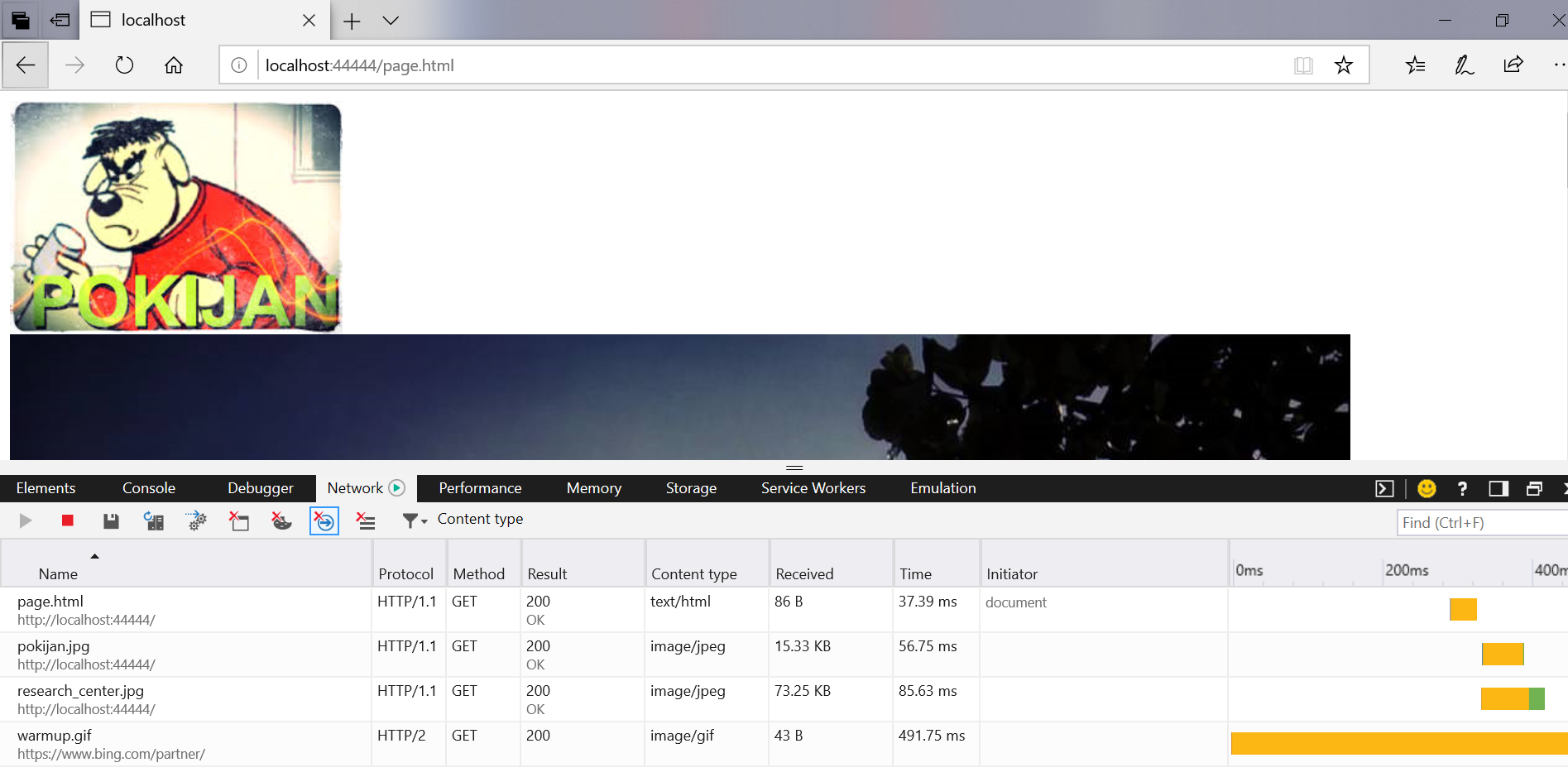
1. Menjalankan file lb.py dan menjalankan di port 44444

Dengan menggunakan perintah :

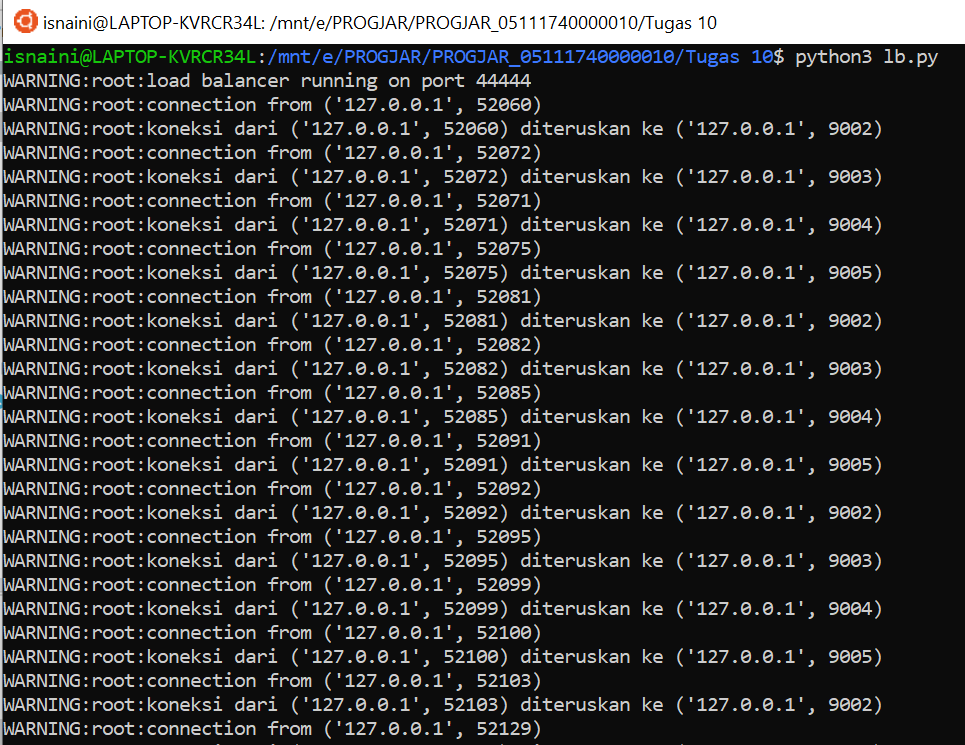
“python3 lb.py”



1. Mengakses <http://localhost:44444/page.html> pada browser



1. Mengecek dan melihat proses di log program bahwa setiap request akan dilayani oleh backend secara bergantian.



1. Melakukan performance test seperti tugas 9, dan membandingkan penggunaan antara load balancer dengan async\_server dengan server\_thread\_http pada folder progjar Tugas5

Dengan parameter sebagai berikut :

Jumlah request : 1000

Konkurensi : 1,50, 100,500,1000

1. Membuat tabel hasil performance test

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No test** | **Concurrency level** | **Time taken for test** | **Complete request** | **Failed request** | **Total transferred** | **Request per second** | **Time per request** | **Transfer rate** |
| 1 | 1 | 2.168 seconds | 1000 | 0 | 122000 bytes | 461.26 [#/sec] (mean) | 2.168 [ms] | 54.95 [Kbytes/sec] |
| 2 | 50 | 59.397 seconds | 1000 | 0 | 122000 bytes | 16.84 [#/sec] (mean) | 59.397 [ms] | 2.01 [Kbytes/sec] |
| 3 | 100 | 59.985 seconds | 1000 | 0 | 122000 bytes | 16.67 [#/sec] (mean) | 59.985 [ms] | 1.99 [Kbytes/ sec] |
| 4 | 500 | 60.406 seconds | 1000 | 0 | 122000 bytes | 16.55 [#/sec] (mean) | 60.406 [ms] | 1.97 [Kbytes/ sec] |
| 5 | 1000 | 62.970 seconds | 1000 | 0 | 122000 bytes | 15.88 [#/sec] (mean) | 62.970 [ms] | 1.89 [Kbytes/ sec] |

* **Server Async (Port 45000)**
* **Server Thread (Port 46000)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No test** | **Concurrency level** | **Time taken for test** | **Complete request** | **Failed request** | **Total transferred** | **Request per second** | **Time per request** | **Transfer rate** |
| 1 | 1 | 483.745 seconds | 1000 | 0 | 122000 bytes | 2.07 [#/sec] (mean) | 483.745 [ms] | 0.25 [Kbytes/sec] |
| 2 | 50 | 422.244 seconds | 1000 | 0 | 122000 bytes | 2.37 [#/sec] (mean) | 422.244 [ms] | 0.28 [Kbytes/sec] |
| 3 | 100 | 492.146 seconds | 1000 | 0 | 122000 bytes | 2.03 [#/sec] (mean) | 492.146 [ms] | 0.24 [Kbytes/ sec] |
| 4 | 500 | 471.756 seconds | 1000 | 0 | 122000 bytes | 2.12 [#/sec] (mean) | 471.756 [ms] | 0.25 [Kbytes/ sec] |
| 5 | 1000 | 442.227 seconds | 1000 | 0 | 122000 bytes | 2.26 [#/sec] (mean) | 442.227 [ms] | 0.27 [Kbytes/ sec] |

* **Sever Async dengan Load Balancer (Port 44444)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No test** | **Concurrency level** | **Time taken for test** | **Complete request** | **Failed request** | **Total transferred** | **Request per second** | **Time per request** | **Transfer rate** |
| 1 | 1 | 1.579 seconds | 1000 | 0 | 0  bytes | 633.48 [#/sec] (mean) | 1.579 [ms] | 0.00 [Kbytes/sec] |
| 2 | 50 | 1.580 seconds | 1000 | 0 | 0  bytes | 632.86 [#/sec] (mean) | 1.580 [ms] | 0.00 [Kbytes/sec] |
| 3 | 100 | 1.614 seconds | 1000 | 0 | 0  bytes | 619.41 [#/sec] (mean) | 1.614 [ms] | 0.00 [Kbytes/ sec] |
| 4 | 500 | 2.285 seconds | 1000 | 0 | 0  bytes | 437.65 [#/sec] (mean) | 2.285 [ms] | 0.00 [Kbytes/ sec] |
| 5 | 1000 | 1.698 seconds | 1000 | 0 | 0  bytes | 558.81 [#/sec] (mean) | 1.698 [ms] | 0.00 [Kbytes/ sec] |

* **Kesimpulan**

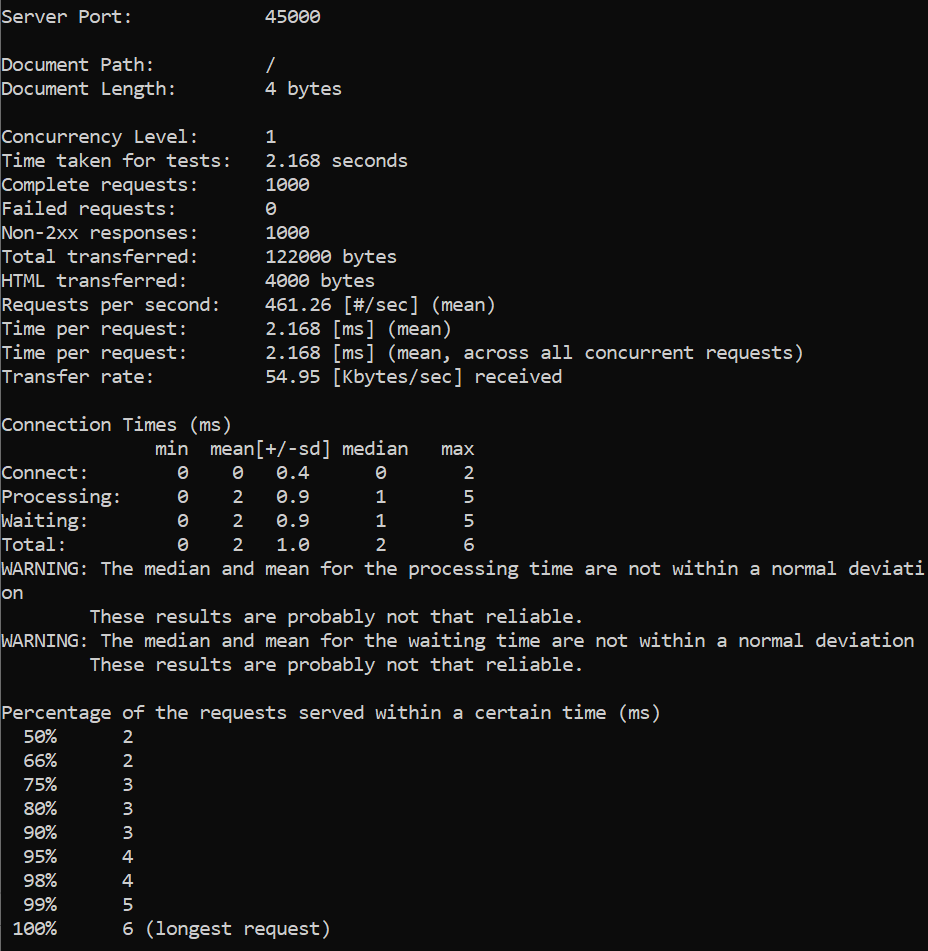
Dari ketiga table performance test diatas, dapat kita lihat bahwa asyncronus server yang menggunakan load balancer memproses lebih cepat jika dibandingkan dengan asyncronus server biasa dan multithread server yang menandakan bahwa performa asyncronus server lebih baik dibandingkan dengan asyncronus biasa dan multithread server. Load balancer bertugas untuk mendistribusikan request ke backend yag didefinisikan di class BackendList. Di dalam load balancer terdapat fungsi getserver pada class BackendList yang akan membuat backend dipilih secara round robin atau bergantian secara fair sehingga setiap request akan dilayani oleh backend yang bergantian.

**Hasil Performance Test**

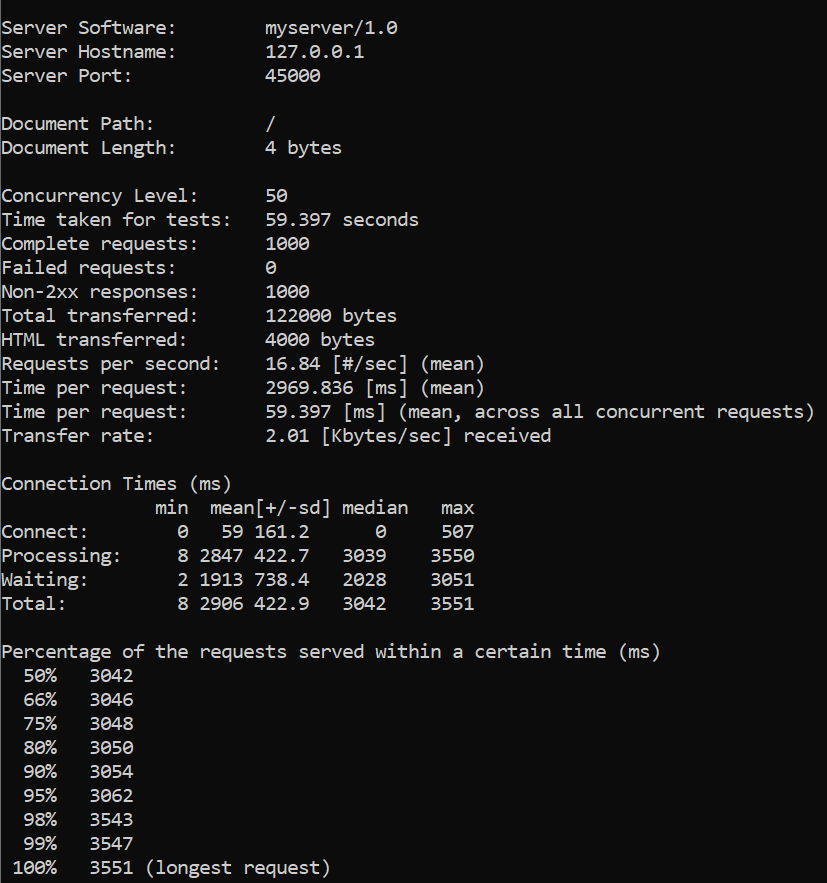
**Asyncronus Server**

**(Port 45000)**

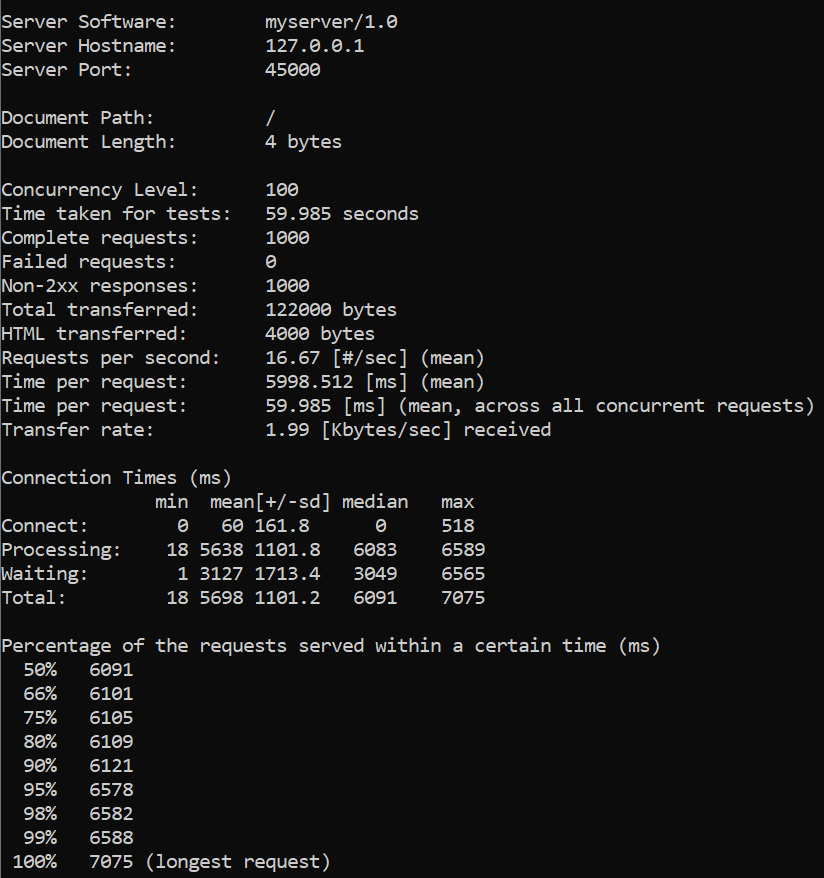
1. **Asyncronus Server Concurrency 1**



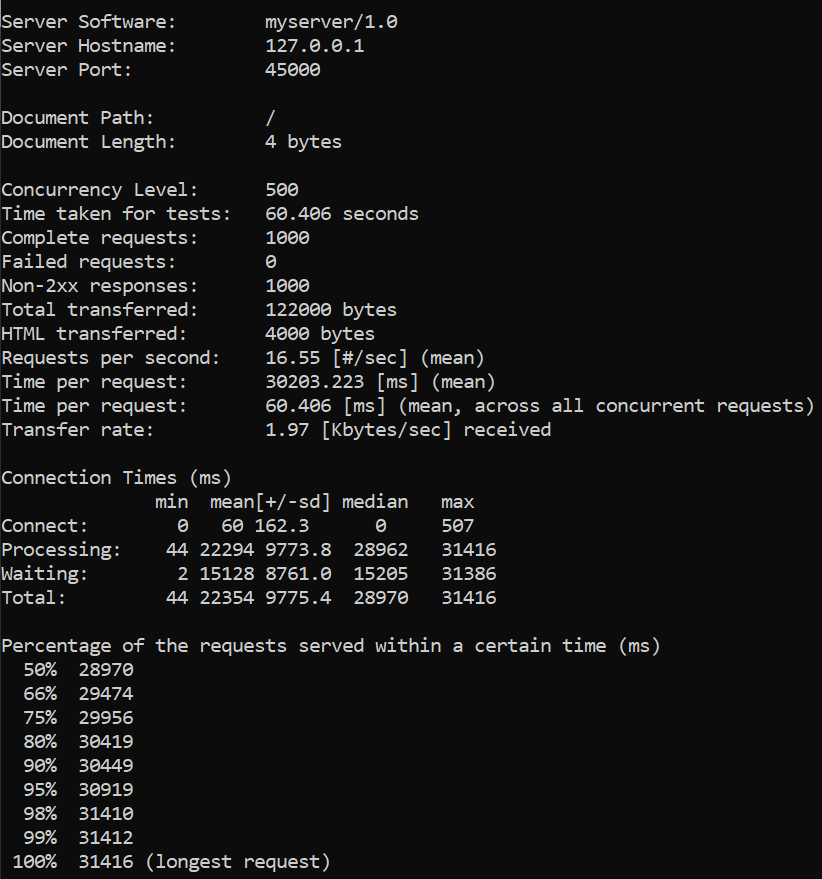
1. **Asyncronus Server Concurrency 50**



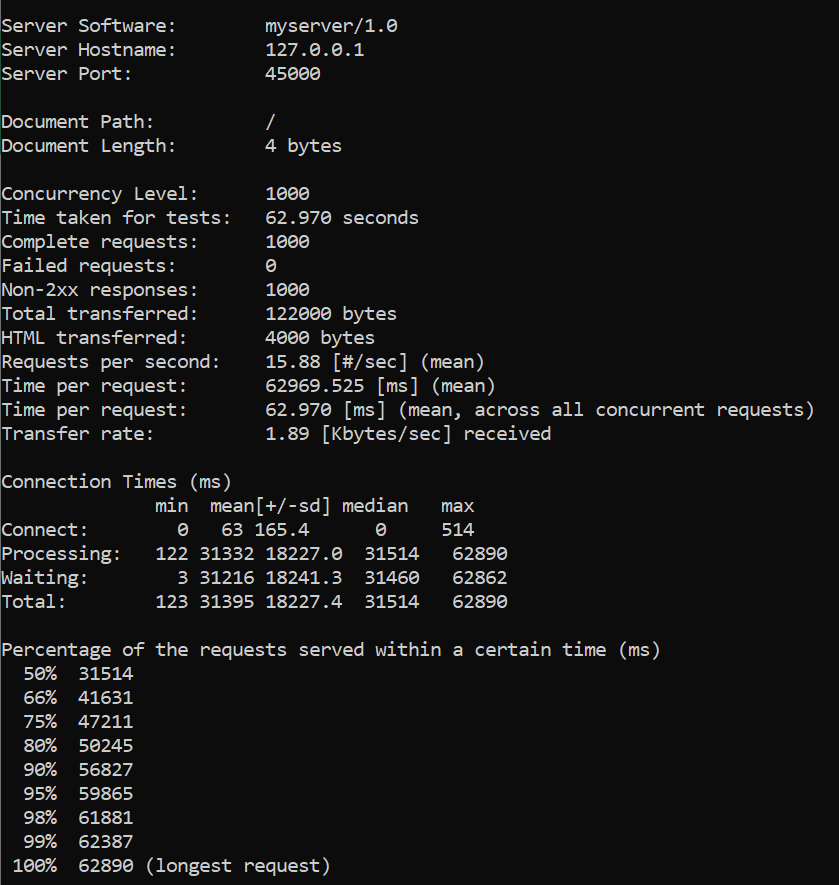
1. **Asyncronus Server Concurrency 100**



1. **Asyncronus Server Concurrency 500**



1. **Asyncronus Server Concurrency 1000**

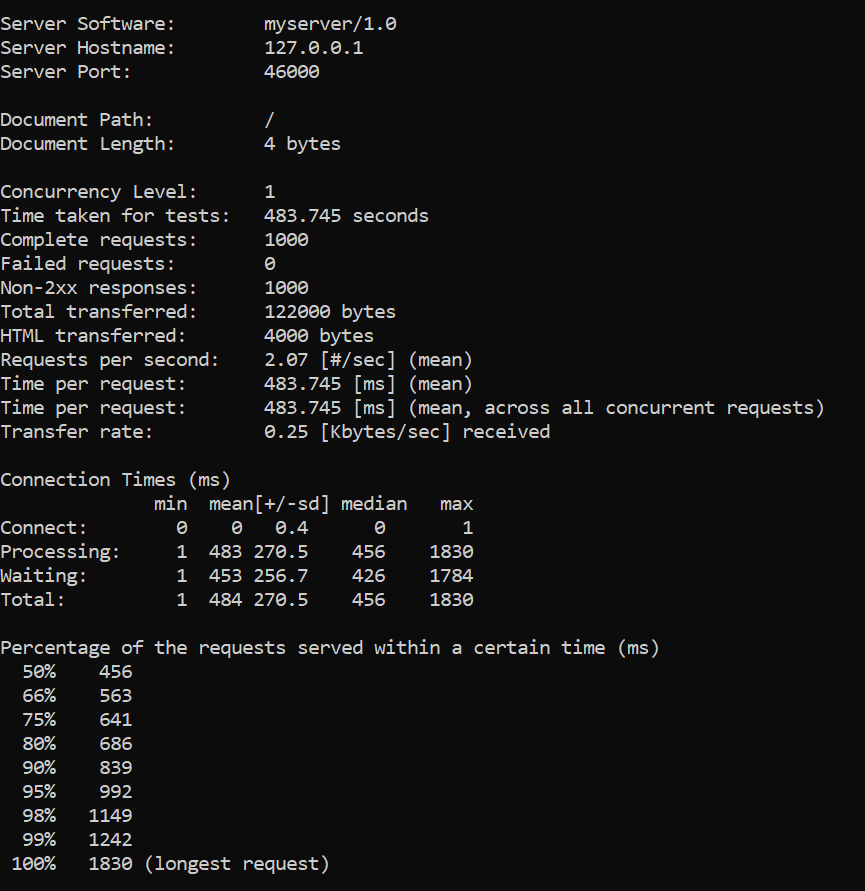


**Hasil Performance Test**

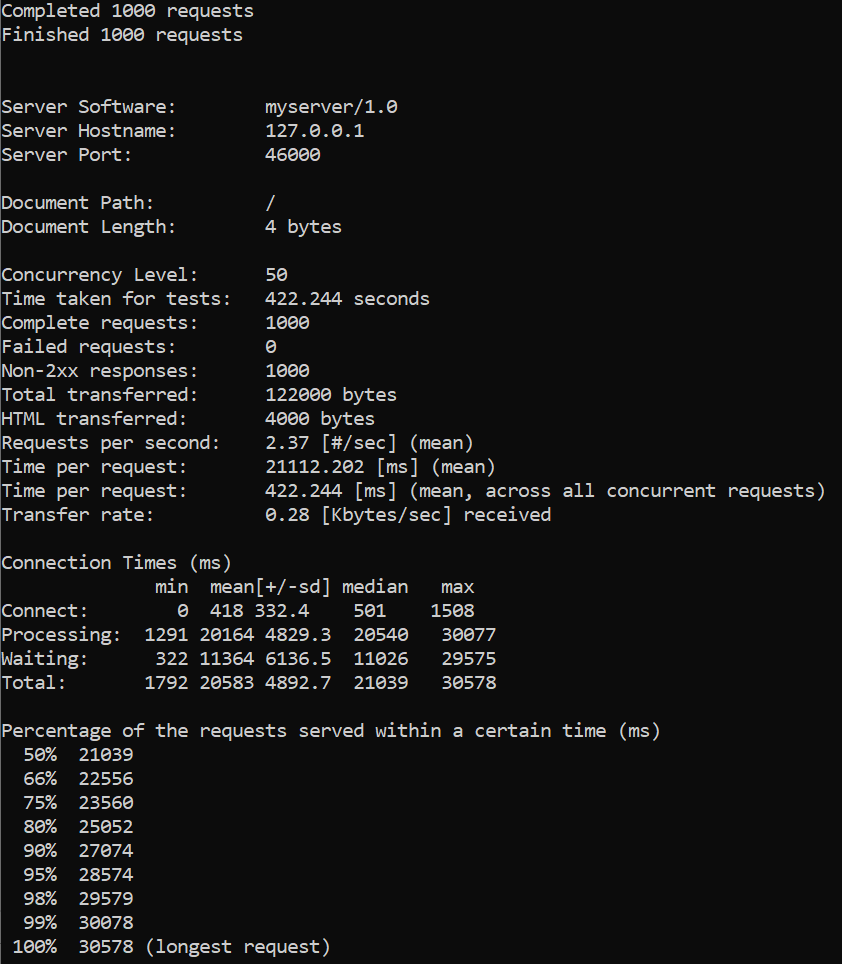
**Server Thread**

**(Port 46000)**

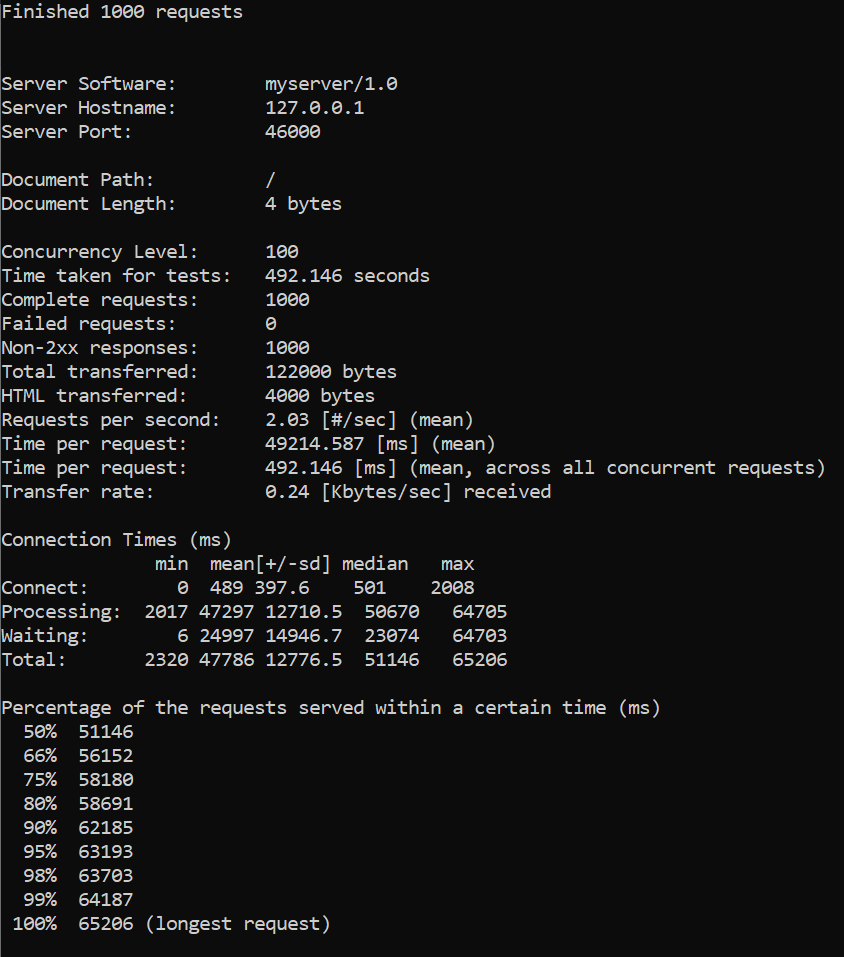
1. **Server Thread Concurrency 1**



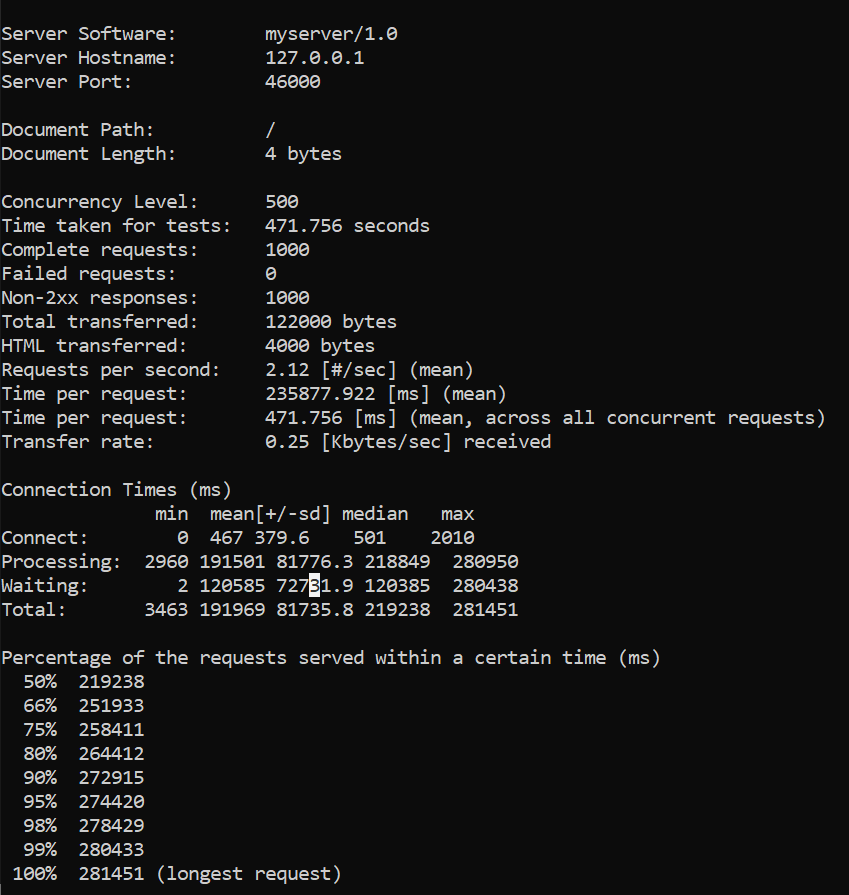
1. **Server Thread Concurrency 50**



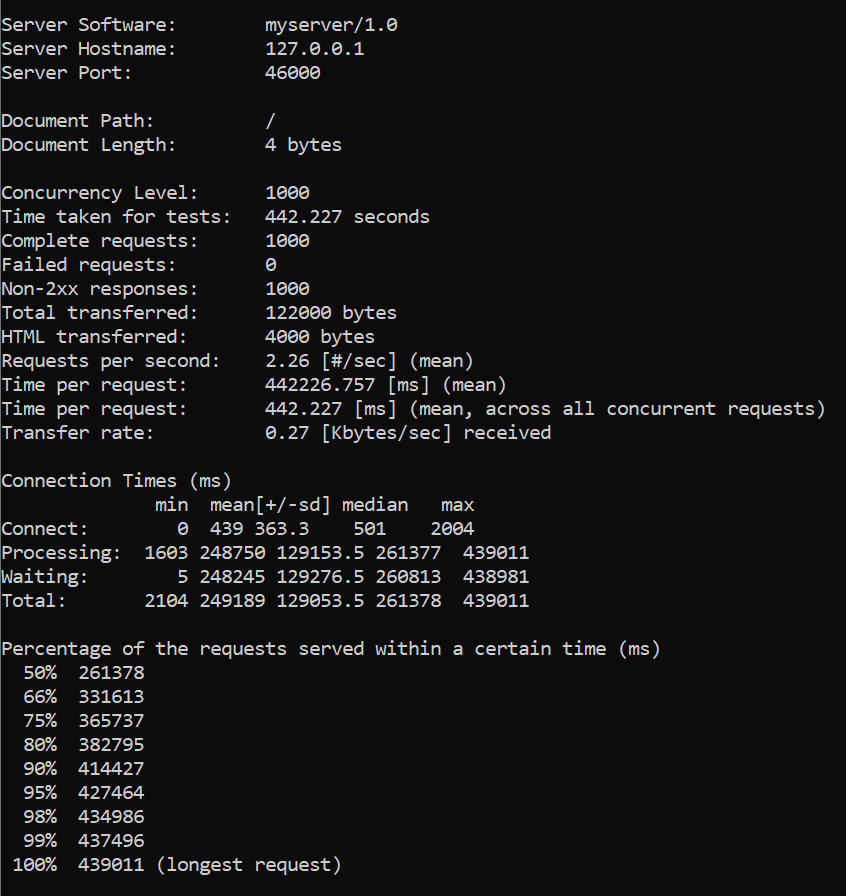
1. **Server Thread Concurrency 100**



1. **Server Thread Concurrency 500**



1. **Server Thread Concurrency 1000**

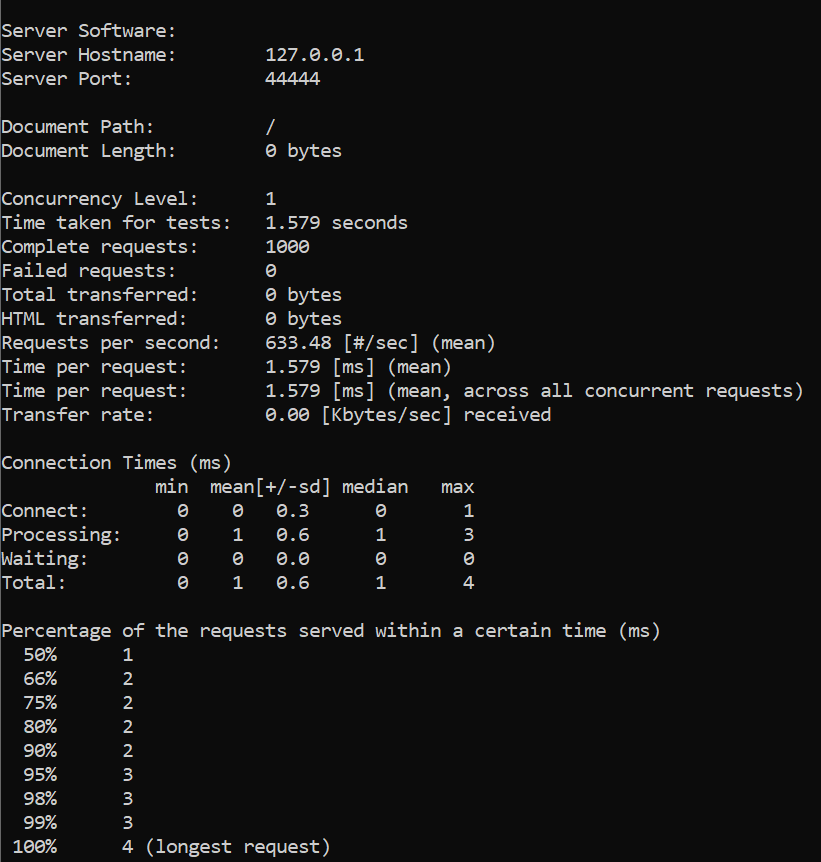


**Hasil Performance Test**

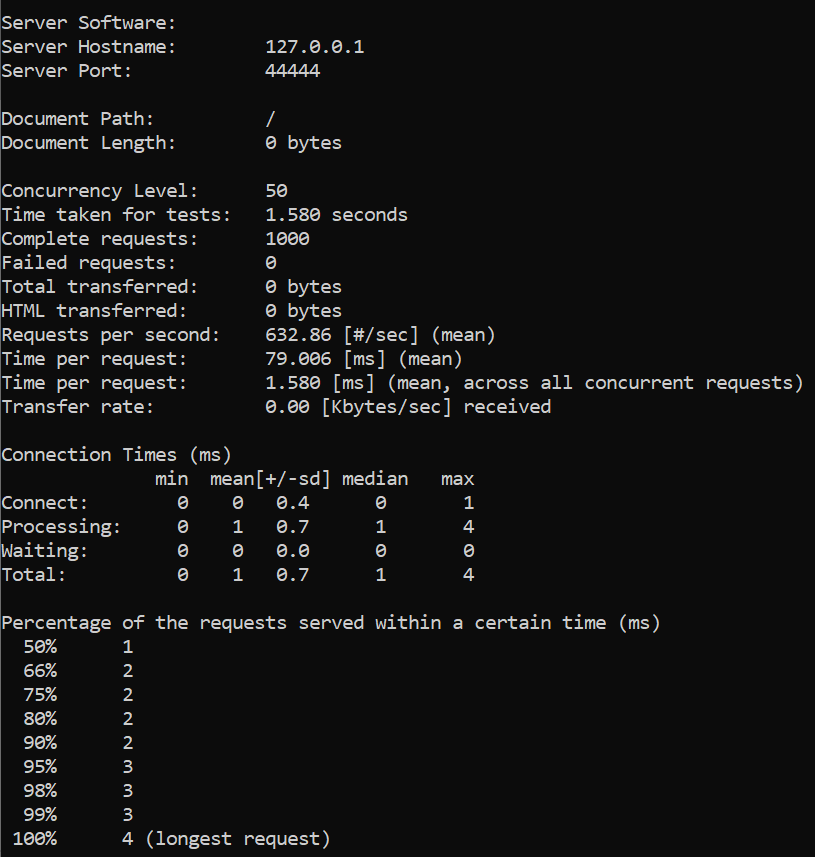
**Server Asyncronus Dengan Load Balancer**

**(Port 44444)**

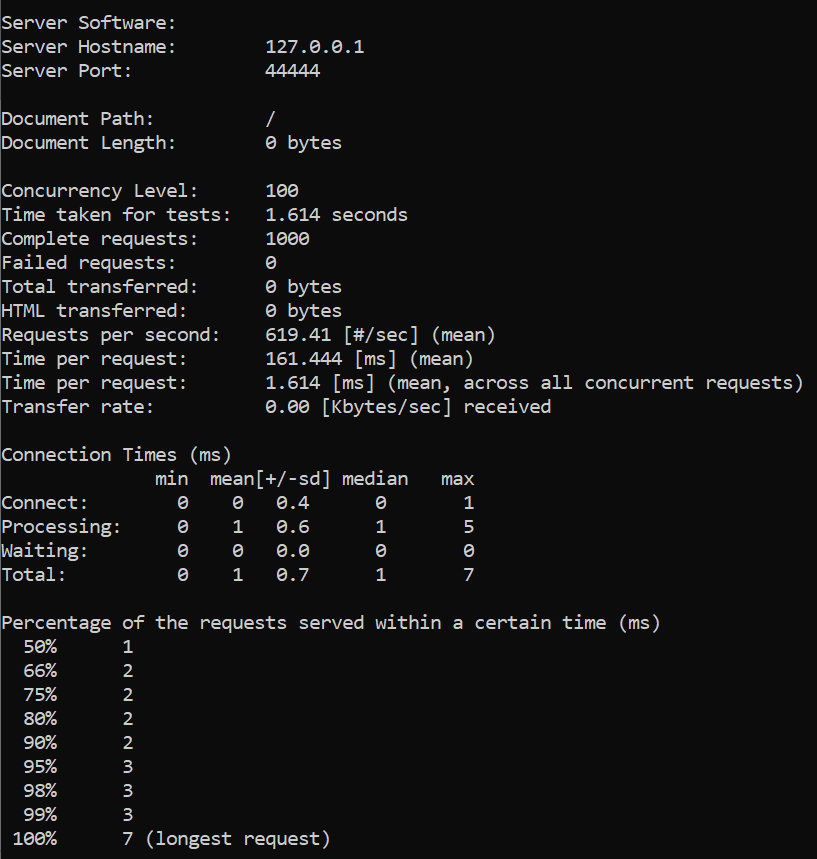
1. **Server Asyncronus Server Dengan Load Balancer 1**



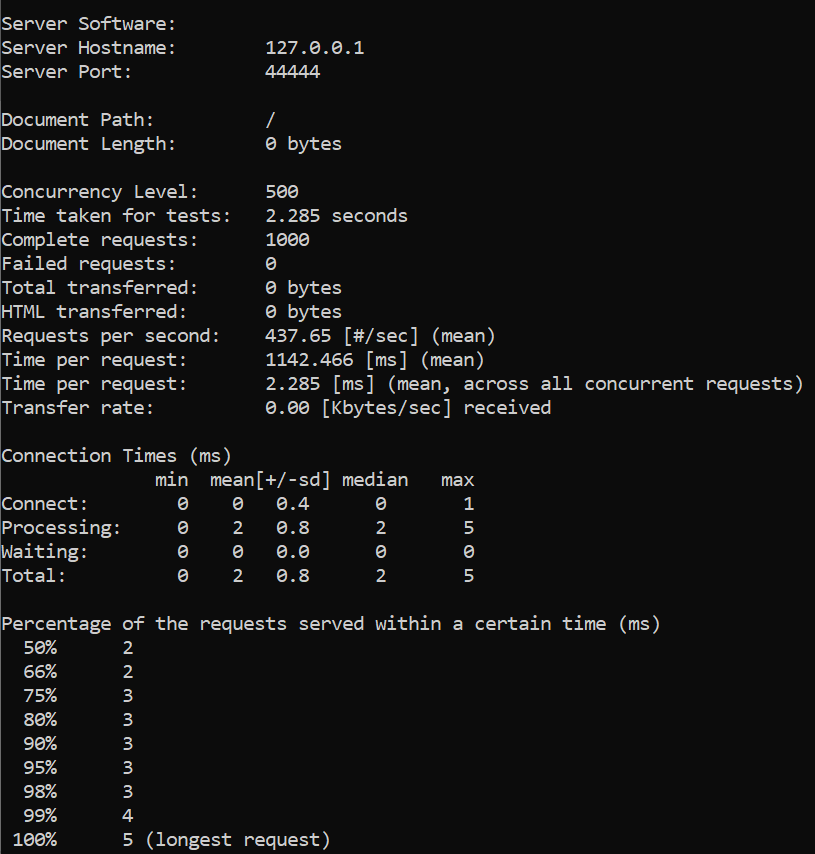
1. **Server Asyncronus Server Dengan Load Balancer 50**



1. **Server Asyncronus Server Dengan Load Balancer 100**



1. **Server Asyncronus Server Dengan Load Balancer 500**



1. **Server Asyncronus Server Dengan Load Balancer 1000**

