8005 Asn2 - Testing and Usage

Sisyphus had a great stress test for gravity

Isaac Morneau; A00958405
Aing Ragunathan; A00765949

Building and Running

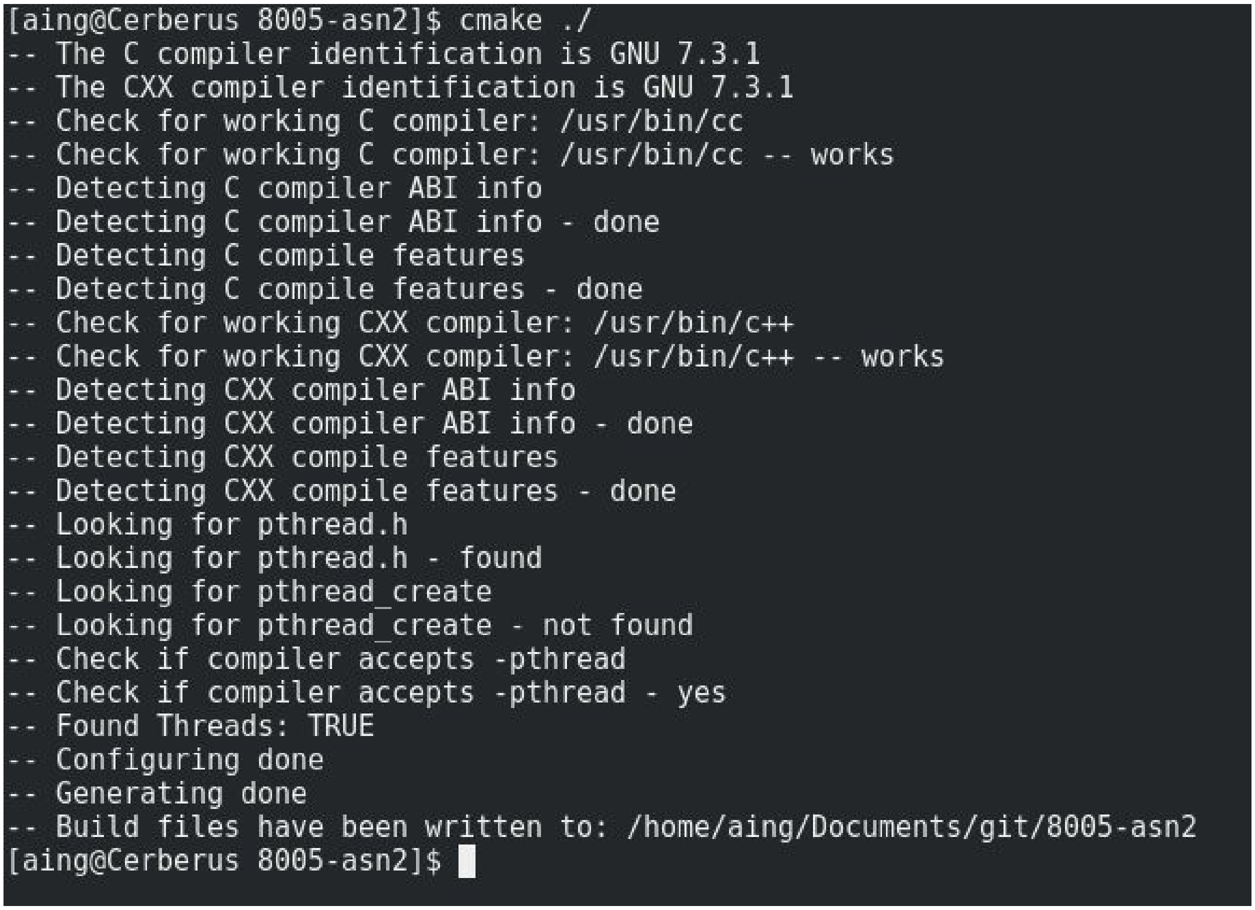
Prerequisites:

CMake is installed
Python3 is installed
C11 compliant compiler is installed

Step 1

Generate makefile by running ‘cmake ./’

Expected Outcomes

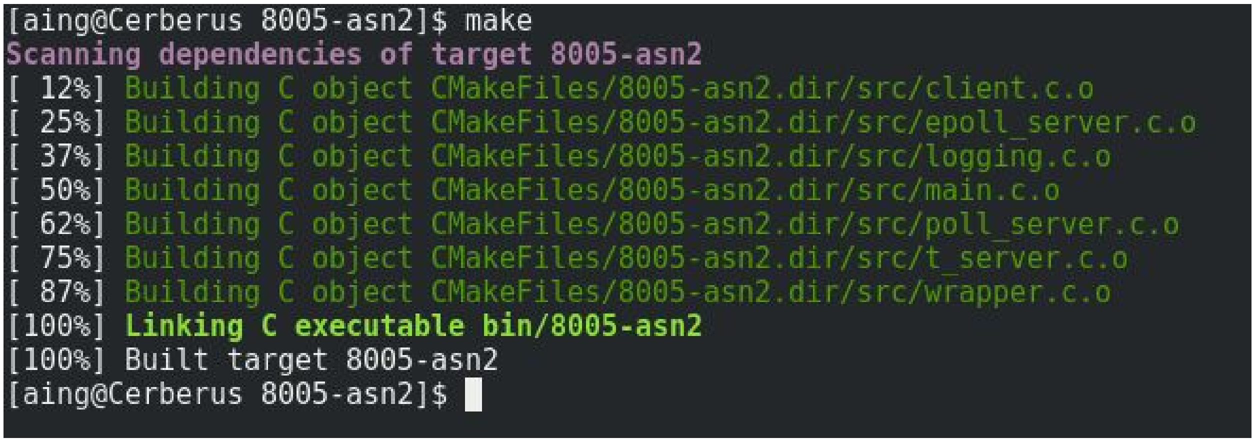


```
[aing@Cerberus 8005-asn2]$ cmake ./
-- The C compiler identification is GNU 7.3.1
-- The CXX compiler identification is GNU 7.3.1
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Check if compiler accepts -pthread
-- Check if compiler accepts -pthread - yes
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /home/aing/Documents/git/8005-asn2
[aing@Cerberus 8005-asn2]$ 
```

Step 2

Build project running ‘make’

Expected outcomes



```
[aing@Cerberus 8005-asn2]$ make
Scanning dependencies of target 8005-asn2
[ 12%] Building C object CMakeFiles/8005-asn2.dir/src/client.c.o
[ 25%] Building C object CMakeFiles/8005-asn2.dir/src/epoll_server.c.o
[ 37%] Building C object CMakeFiles/8005-asn2.dir/src/logging.c.o
[ 50%] Building C object CMakeFiles/8005-asn2.dir/src/main.c.o
[ 62%] Building C object CMakeFiles/8005-asn2.dir/src/poll_server.c.o
[ 75%] Building C object CMakeFiles/8005-asn2.dir/src/t_server.c.o
[ 87%] Building C object CMakeFiles/8005-asn2.dir/src/wrapper.c.o
[100%] Linking C executable bin/8005-asn2
[100%] Built target 8005-asn2
[aing@Cerberus 8005-asn2]$
```

Test Epoll Server

Purpose

Start an epoll echo server to establish connections with clients, echo back any received data and log data on the number of requests, total data, response time and time between reads.

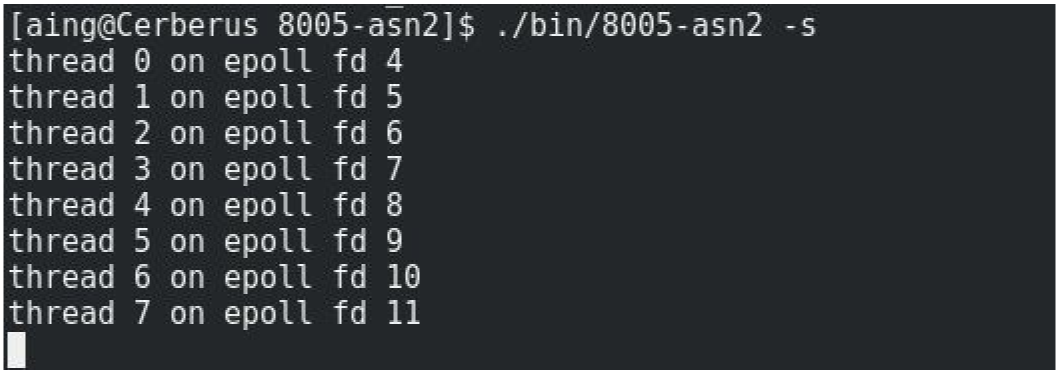
Step 1

Run server on one machine with ‘./bin/8005-asn2 -s’

Step 2

Run client on another machine with ‘./bin/8005-asn2 -c -a <server IP>’

Expected Outcome



```
[aing@Cerberus 8005-asn2]$ ./bin/8005-asn2 -s
thread 0 on epoll fd 4
thread 1 on epoll fd 5
thread 2 on epoll fd 6
thread 3 on epoll fd 7
thread 4 on epoll fd 8
thread 5 on epoll fd 9
thread 6 on epoll fd 10
thread 7 on epoll fd 11
```

Test Poll Server

Purpose

Start a poll echo server to establish connections with clients, echo back any received data and log data on the number of requests, total data, response time and time between reads.

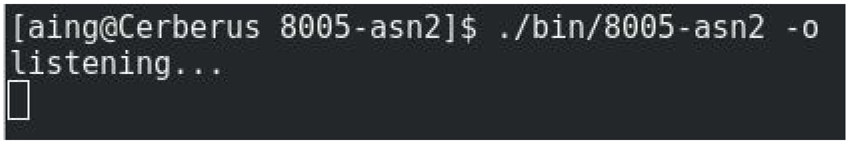
Step 1

Run server on one machine with ‘./bin/8005-asn2 -o’

Step 2

Run client on another machine with ‘./bin/8005-asn2 -c -a <server IP>’

Expected Outcome



```
[aing@Cerberus 8005-asn2]$ ./bin/8005-asn2 -o
listening...
```

Test Multithreaded Traditional Server

Purpose

Start a multithreaded traditional echo server to establish connections with clients, echo back any received data and log data on the number of requests, total data, response time and time between reads.

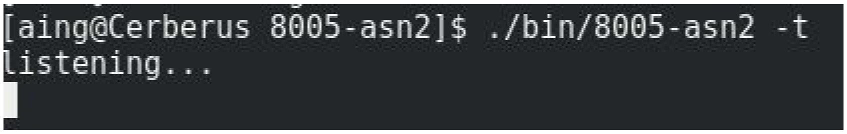
Step 1

Run server on one machine with ‘./bin/8005-asn2 -t’

Step 2

Run client on another machine with ‘./bin/8005-asn2 -c -a <server IP>’

Expected Outcome



```
[aing@Cerberus 8005-asn2]$ ./bin/8005-asn2 -t
listening...
```

Test Log Graphing Script

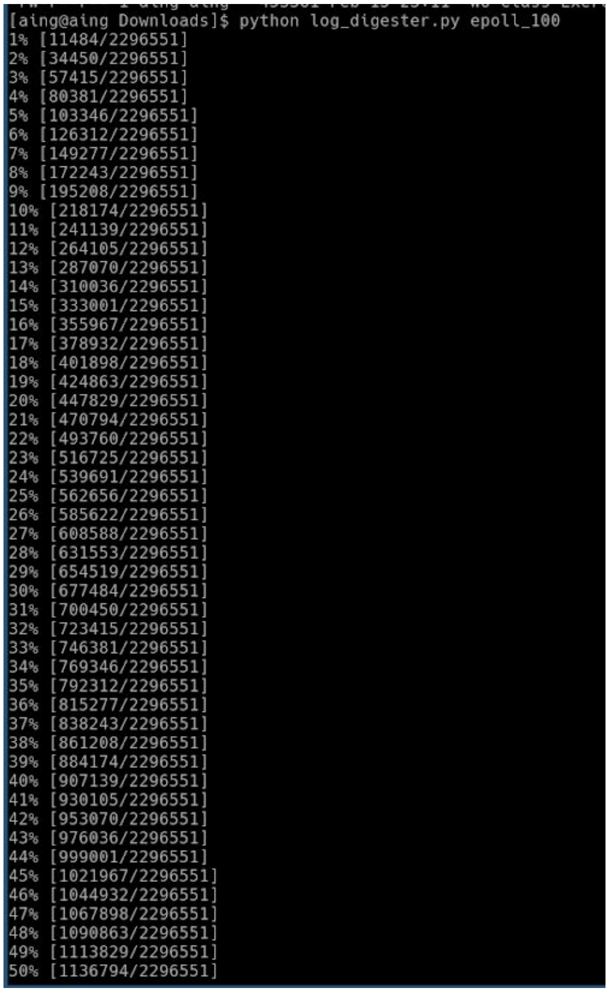
Purpose

Generate useful information from the server or client log file, such as the total established connections, total transferred data, average delay, average packets transferred, average data transferred, an average connection delay graph and an average data transferred graph .

Step 1

Run script to generate information from a log file with ‘python log_digester.py logging’

Expected Outcomes

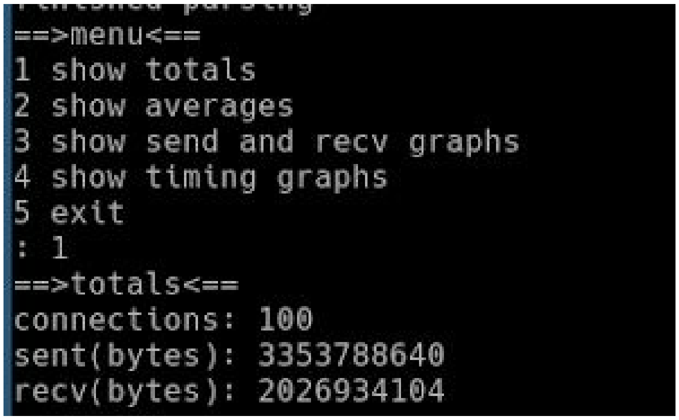


```
[aing@aing Downloads]$ python log_digester.py epoll_100
1% [11484/2296551]
2% [34450/2296551]
3% [57415/2296551]
4% [80381/2296551]
5% [103346/2296551]
6% [126312/2296551]
7% [149277/2296551]
8% [172243/2296551]
9% [195208/2296551]
10% [218174/2296551]
11% [241139/2296551]
12% [264105/2296551]
13% [287070/2296551]
14% [310036/2296551]
15% [333001/2296551]
16% [355967/2296551]
17% [378932/2296551]
18% [401898/2296551]
19% [424863/2296551]
20% [447829/2296551]
21% [470794/2296551]
22% [493760/2296551]
23% [516725/2296551]
24% [539691/2296551]
25% [562656/2296551]
26% [585622/2296551]
27% [608588/2296551]
28% [631553/2296551]
29% [654519/2296551]
30% [677484/2296551]
31% [700450/2296551]
32% [723415/2296551]
33% [746381/2296551]
34% [769346/2296551]
35% [792312/2296551]
36% [815277/2296551]
37% [838243/2296551]
38% [861208/2296551]
39% [884174/2296551]
40% [907139/2296551]
41% [930105/2296551]
42% [953070/2296551]
43% [976036/2296551]
44% [999001/2296551]
45% [1021967/2296551]
46% [1044932/2296551]
47% [1067898/2296551]
48% [1090863/2296551]
49% [1113829/2296551]
50% [1136794/2296551]
```

Step 2

View total established connections and total transferred data with the command ‘1’

Expected Outcomes

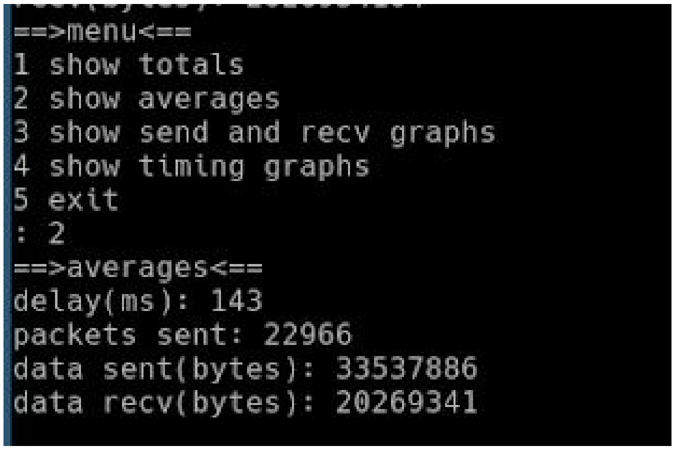


```
==>menu<==
1 show totals
2 show averages
3 show send and recv graphs
4 show timing graphs
5 exit
: 1
==>totals<==
connections: 100
sent(bytes): 3353788640
recv(bytes): 2026934104
```

Step 3

View average delay, average packets transferred and average data transferred with the command ‘2’

Expected Outcomes

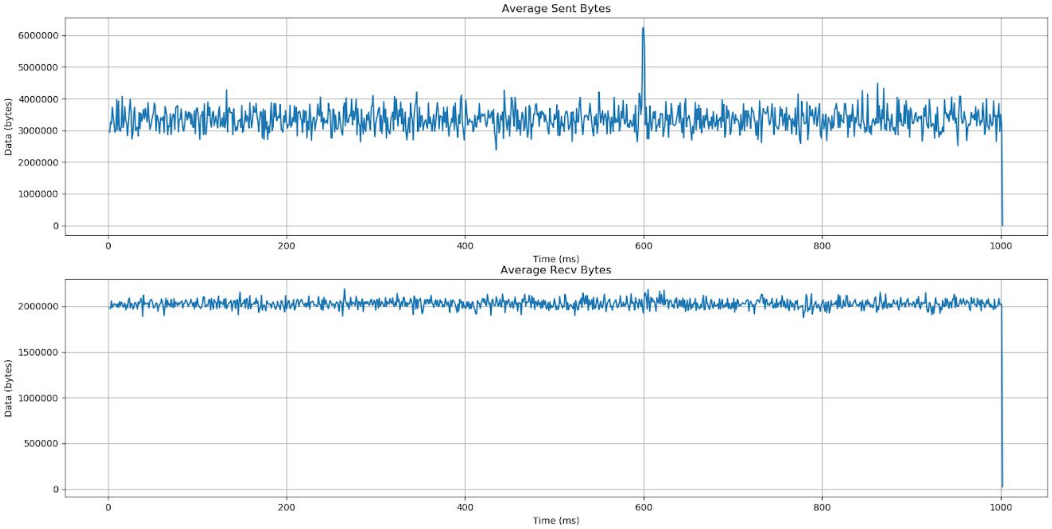


```
==>menu<==
1 show totals
2 show averages
3 show send and recv graphs
4 show timing graphs
5 exit
: 2
==>averages<==
delay(ms): 143
packets sent: 22966
data sent(bytes): 33537886
data recv(bytes): 20269341
```

Step 4

View an average delay graph with the command ‘3’

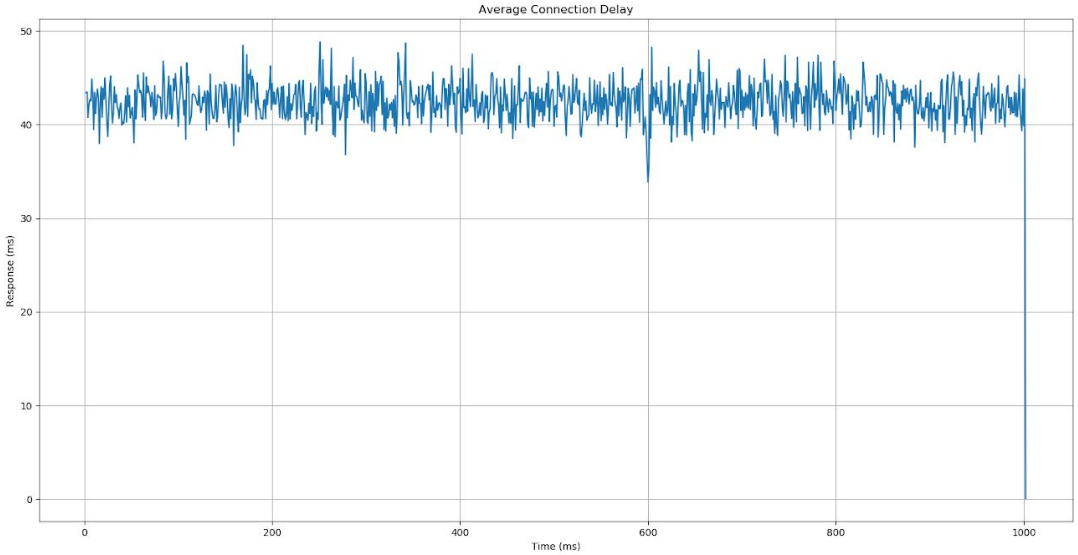
Expected Outcomes



Step 5

View an average data transferred graph with the command ‘4’

Expected Outcome


Average Connection Delay
Response (ms)
Time (ms)