ECE 590 HW4: Scalability

Nisarg Dabhi and Connor Grehlinger

***Abstract*— This document serves as a performance and scalability analysis of an Exchange Matching Server developed for Homework 4 of ECE590 Engineering Robust Server Software**

I. Testing Scalability of Underlying Exchange Server Infrastructure

The TCP server our Exchange Matching server runs on is a subclass of the Poco librarys Poco::Net::TCPServer. This is a multi-threaded TCP server which uses a thread pool and dispatches threads from that pool to handle connections to the server. One thread is responsible for listening for incoming connections on a queue, then accepting those connections and dispatching other threads to handle connection interaction with the client end host.

Our exchange server allows a maximum connection queue of 8192 connections. Threads waiting to accept connections which are idle for more than 10 seconds are terminated. These parameters can be adjusted to accommodate the Exchange Servers load.

The VM on which the Exchange Matching Server runs has the following specifications:

```
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  4
On-line CPU(s) list:     0-3
Thread(s) per core:      1
Core(s) per socket:      1
Socket(s):               4
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   15
Model name:              Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz
Stepping:                1
CPU MHz:                 2593.993
BogoMIPS:                5187.98
Hypervisor vendor:       VMware
Virtualization type:     full
L1d cache:               32K
L1i cache:               32K
L2 cache:                256K
L3 cache:                35840K
NUMA node0 CPU(s):       0-3
```

Fig. 1. Exchange Matching Server VM Specifications

This VM uses a portion of the resources of the Intel CPU listed. Specifications for this particular Intel chip are depicted in Figure 2. It is clear that the physical chip itself has more resources than what is allocated to the VM:

Since this is an Intel chip which supports hyper threading, based on what our virtual machine has been allocated we have 8 hardware threads:

2 threads/core * 1 core/socket * 4 sockets = 8 threads

The graphics on the next page depict the scalability of the underlying multi-threaded TCP Server our Exchange Matching Server uses. These figures represent the performance of basic TCP connections and transmissions. The Exchange Matching Server receives an example string of XML (see Figure 3) and responds to the client by sending the message Request received by exchange server.

Essentials — Export specifications

Product Collection	Intel® Xeon® Processor E5 v3 Family
Code Name	Products formerly Haswell
Vertical Segment	Server
Processor Number	E5-2697V3
Status	Launched
Launch Date	Q3'14
Lithography	22 nm
Recommended Customer Price	$2702.00 - $2706.00

Performance

# of Cores	14
# of Threads	28
Processor Base Frequency	2.60 GHz
Max Turbo Frequency	3.60 GHz
Cache	35 MB SmartCache
Bus Speed	9.6 GT/s QPI
# of QPI Links	2
TDP	145 W
VID Voltage Range	0.65V–1.30V

Fig. 2. Exchange Matching Server VM Intel CPU Specifications

```
std::string testDataStream =
  "173\r\n"
  "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\r\n"
  "<create>\r\n"
  "  <account id=\"123456\" balance=\"1000\"/>\r\n"
  "  <symbol sym=\"SPY\">\r\n"
  "     <account id=\"123456\">100000</account>\r\n"
  "  </symbol>\r\n"
  "</create>\r\n\r\n";
```

Fig. 3. Example XML string

```
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  2
On-line CPU(s) list:     0,1
Thread(s) per core:      1
Core(s) per socket:      1
Socket(s):               2
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   15
Model name:              Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz
Stepping:                1
CPU MHz:                 2593.993
BogoMIPS:                5187.98
Hypervisor vendor:       VMware
Virtualization type:     full
L1d cache:               32K
L1i cache:               32K
L2 cache:                256K
L3 cache:                30720K
NUMA node0 CPU(s):       0,1
```

Fig. 4. Separate VM Specifications

II. Connection & Transmission Stress Test

Incoming connections: Incoming connection requests and XML string transmission are made by a VM separate from the Exchange Matching server with the specifications shown in Figure 4.

A test program on this machine uses a thread pool to dispatch threads to make a new connection and send the XML string then receive a reply.

The number of threads dispatched to send requests as well as the number of requests each thread sends can be tuned with parameters `THREADS` and `REQUESTS`. The thread pool is set to a min size equal to the `THREADS` parameter and max size of 2048.

NOTE: The max thread capacity for the thread pools on both the Exchange Matching Server and the VM which is load testing are set to 4096 and 2048 respectively. This is far higher than either machine can support, but this to ensure each machine uses as much execution resources as possible for the scalability stress test)

Instructions for using our testing suite as are written below. They can also be found in the README.md file of the /testing/ directory of the repository:

III. Testing Exchange Server Scalability

To run performance tests measuring scalability in the context of execution time under certain request volumes, make changes to the following macro fields:

- **REQ in /src/ExchangeRequestHandler.cpp**

This sets the number of requests the Exchange Matching Server will handle before terminating and printing the execution time it took to process those requests in microseconds and seconds. The total running time of the Exchange Server Application is also reported.

- **THREADS in /testing/src/XMLRequestGen.cpp**

This sets the number of threads which will be dispatched from a thread pool to perform a number of iterations (defined by the REQUESTS macro) of a function (called runTest) which load tests the Exchange Matching Server. In each iteration of the load testing function:

1) A new connection is established with the Exchange Matching Server
2) A request containing an XML string is sent to the Exchange Matching Server
3) A reply sent by the Exchange Matching Server is received by the calling thread

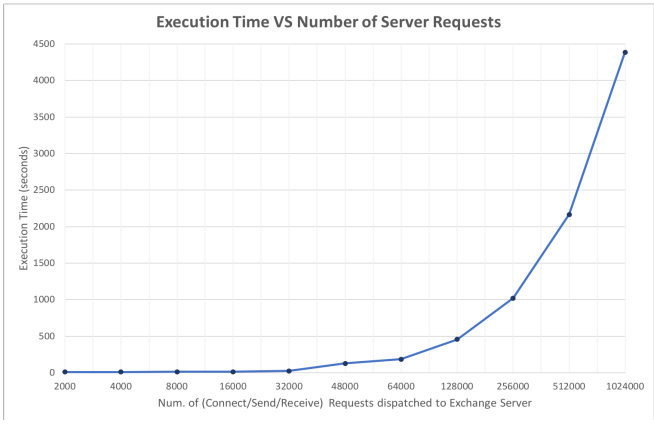
- **REQUESTS in /testing/src/XMLRequestGen.cpp**

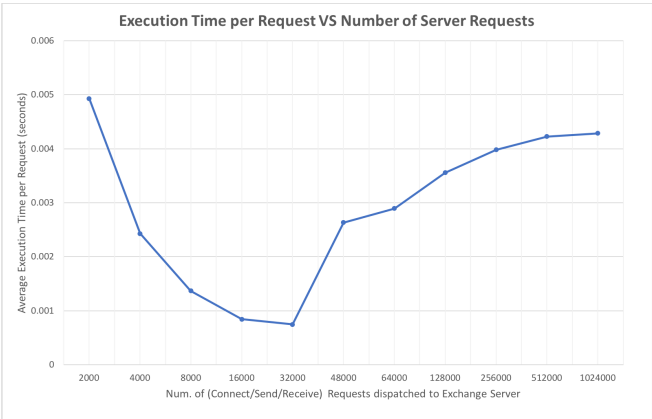
This specifies the number of iterations of the load testing function (runTest) each thread completes.This means THREADS * REQUESTS is the total number of connect + send + receive calls made to the Exchange Matching Server on each call to ./XMLRequestGen.

- **The docker-compose.yml file in /testing/**

Ensures that the bash command on line 6 iterates through and calls the ./XMLRequestGen program enough times to generate enough requests to hit the REQ macro threshold in /src/ExchangeRequestHandler.cpp:

$$(XMLRequestGencalls * THREADS * REQUESTS) \geq (REQ)$$





A. Scalability Testing Results

From both depicted graphs it is clear that the underlying TCP Server for our Exchange Matching Server is scalable within the limitations of the provided hardware.

The Execution Time vs Number of Requests dispatched to the server graph (each request established a connection, sends a message, and receives a reply) shows a near linear relationship. Each time the number of dispatched requests doubles the execution time to complete those requests also about doubles.

The Execution Time per Request vs Number of Requests graph also shows little variably in the servicing time of each request even if the load on the Exchange Matching Server is greatly increased. What little variability which can be seen is measured in milliseconds and can be attributed to variability in transmissions over TCP networks.

The table which was used to calculate these results is included below:

Scalability Test: Connect + Request + Response

Test #	Number of threads (not all concurrently executing above 32)	Requests per thread	Total requests	Execution time (seconds)	Execution time per request (seconds)	Notes & Observations:
1	2	1000	2000	9.8568	0.0049284	
3	4	1000	4000	9.7121	0.002428025	
4	8	1000	8000	10.9502	0.001368775	
5	16	1000	16000	13.5017	0.000843856	
6	32	1000	32000	23.854	0.000745438	
7	48	1000	48000	126.426	0.002633875	At around 40,000 connection requests generated from one VM stress test program, connections begin to be discarded from Exchange Server queue. Proceeding tests recall the stress test program several times using bash script (less concurrency but higher request volume)
8	64	1000	64000	185.175	0.002893359	
9	128	1000	128000	455.41	0.003557891	
10	256	1000	256000	1018.69	0.003979258	
11	512	1000	512000	2164.27	0.00422709	
12	1024	1000	1024000	4387.36	0.004284531	