Team 5 - Performance Evaluation

Chris Popeck, Christian Johnson, Fergus Horner, Zachary Force

Selected Benchmarks

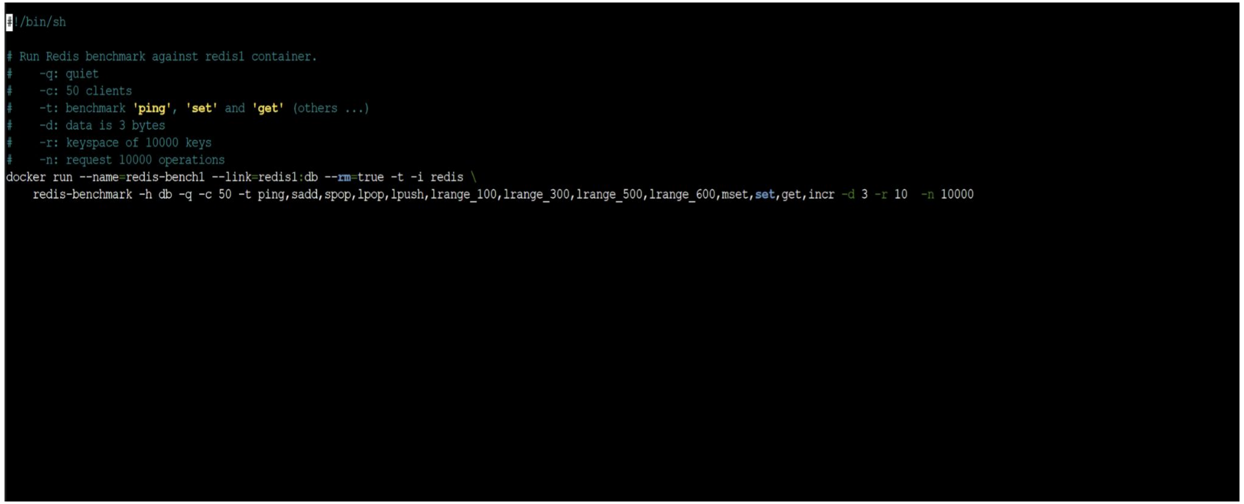
- Redis
- MySQL
- Netperf
- Linpack

Redis

- NoSql cache/database
- native vs Container
- Benchmark will run a variety of test using redis functions(ping, data entry, data retrieval. etc..) and compare their efficiency while containerized vs running natively on ubuntu

RULES

- Request 10,000 operations
- 50 parallel client
- 3 byte payload*
- Key length 10



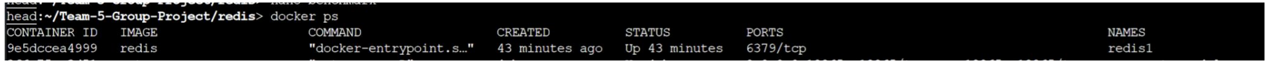
```
#!/bin/sh

# Run Redis benchmark against redis1 container.
#    -q: quiet
#    -c: 50 clients
#    -t: benchmark 'ping', 'set' and 'get' (others ...)
#    -d: data is 3 bytes
#    -r: keyspace of 10000 keys
#    -n: request 10000 operations
docker run --name=redis-bench1 --link=redis1:db --rm=true -t -i redis \
    redis-benchmark -h db -q -c 50 -t ping,sadd,spop,lpop,lpush,lrange_100,lrange_300,lrange_500,lrange_600,mset,set,get,incr -d 3 -r 10  -n 10000
```

The Process

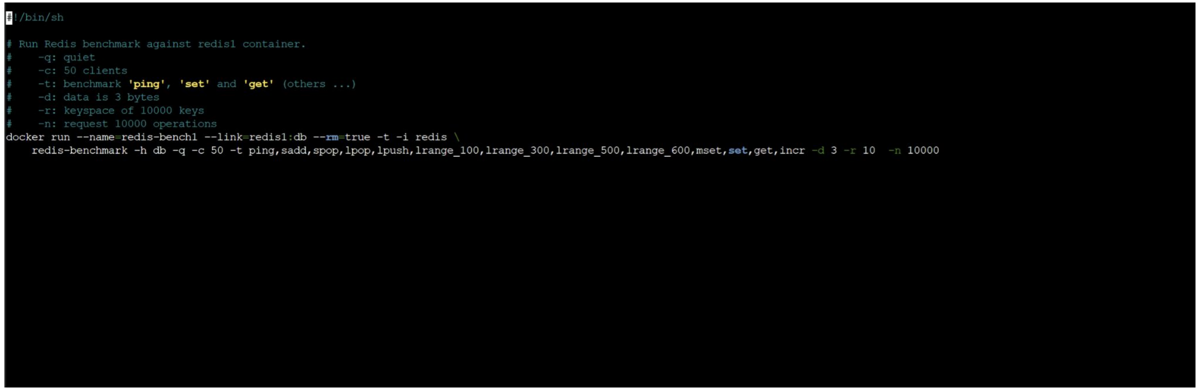
-Relatively straightforward

(Thanks Github!)

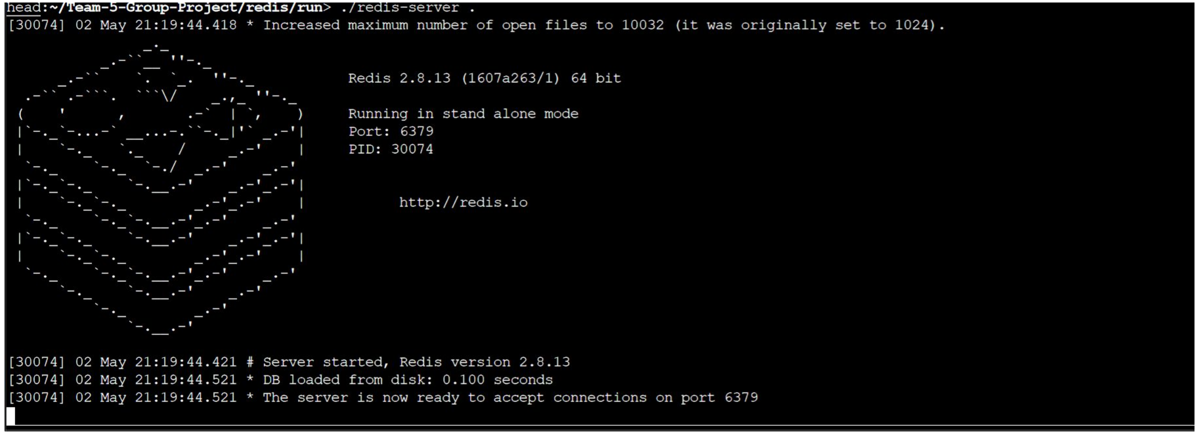
-Install, configure, and run!



```
head:~/Team-5-Group-Project/redis> docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS                                                          NAMES
9e5dccea4999   redis                "docker-entrypoint.s…"   43 minutes ago   Up 43 minutes   6379/tcp                                                       redis1
```



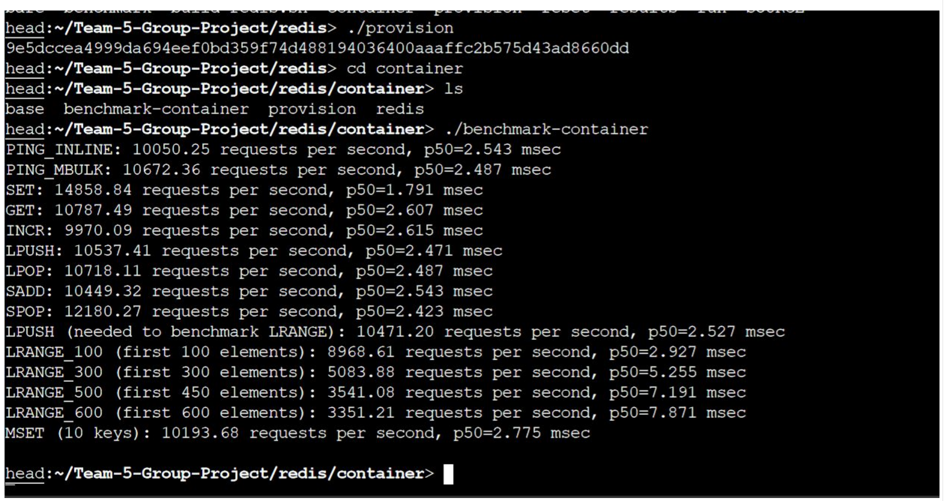
```
#!/bin/sh

# Run Redis benchmark against redis1 container.
#    -q: quiet
#    -c: 50 clients
#    -t: benchmark 'ping', 'set' and 'get' (others ...)
#    -d: data is 3 bytes
#    -r: keyspace of 10000 keys
#    -n: request 10000 operations
docker run --name=redis-bench1 --link=redis1:db --rm=true -t -i redis \
    redis-benchmark -h db -q -c 50 -t ping,sadd,spop,lpop,lpush,lrange_100,lrange_300,lrange_500,lrange_600,mset,set,get,incr -d 3 -r 10  -n 10000
```



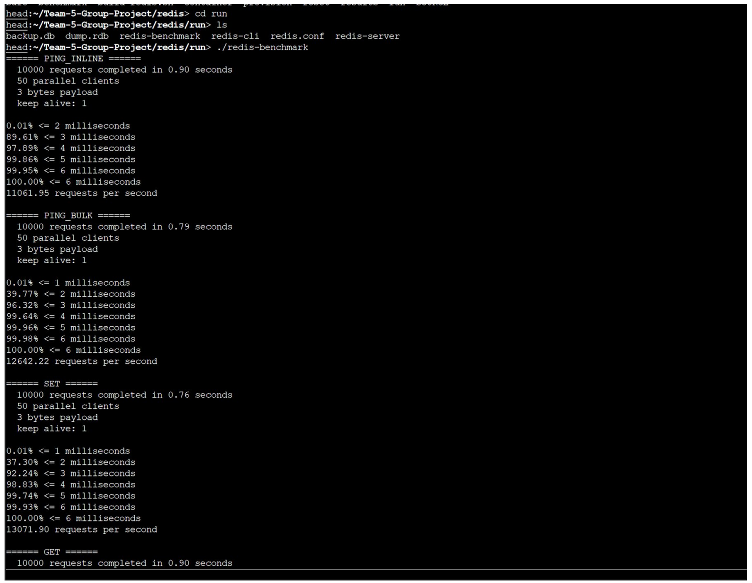
```
head:~/Team-5-Group-Project/redis/run> ./redis-server .
[30074] 02 May 21:19:44.418 * Increased maximum number of open files to 10032 (it was originally set to 1024).
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 2.8.13 (1607a263/1) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in stand alone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
 |    `-._   `._    /     _.-'    |     PID: 30074
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[30074] 02 May 21:19:44.421 # Server started, Redis version 2.8.13
[30074] 02 May 21:19:44.521 * DB loaded from disk: 0.100 seconds
[30074] 02 May 21:19:44.521 * The server is now ready to accept connections on port 6379
```

Results

-To my surprise, the containerized version of redis yielded much slower results than it’s native counterpart



```
head:~/Team-5-Group-Project/redis> ./provision
9e5dccea4999da694eef0bd359f74d488194036400aaaffc2b575d43ad8660dd
head:~/Team-5-Group-Project/redis> cd container
head:~/Team-5-Group-Project/redis/container> ls
base  benchmark-container  provision  redis
head:~/Team-5-Group-Project/redis/container> ./benchmark-container
PING_INLINE: 10050.25 requests per second, p50=2.543 msec
PING_MBULK: 10672.36 requests per second, p50=2.487 msec
SET: 14858.84 requests per second, p50=1.791 msec
GET: 10787.49 requests per second, p50=2.607 msec
INCR: 9970.09 requests per second, p50=2.615 msec
LPUSH: 10537.41 requests per second, p50=2.471 msec
LPOP: 10718.11 requests per second, p50=2.487 msec
SADD: 10449.32 requests per second, p50=2.543 msec
SPOP: 12180.27 requests per second, p50=2.423 msec
LPUSH (needed to benchmark LRANGE): 10471.20 requests per second, p50=2.527 msec
LRANGE_100 (first 100 elements): 8968.61 requests per second, p50=2.927 msec
LRANGE_300 (first 300 elements): 5083.88 requests per second, p50=5.255 msec
LRANGE_500 (first 450 elements): 3541.08 requests per second, p50=7.191 msec
LRANGE_600 (first 600 elements): 3351.21 requests per second, p50=7.871 msec
MSET (10 keys): 10193.68 requests per second, p50=2.775 msec

head:~/Team-5-Group-Project/redis/container>
```



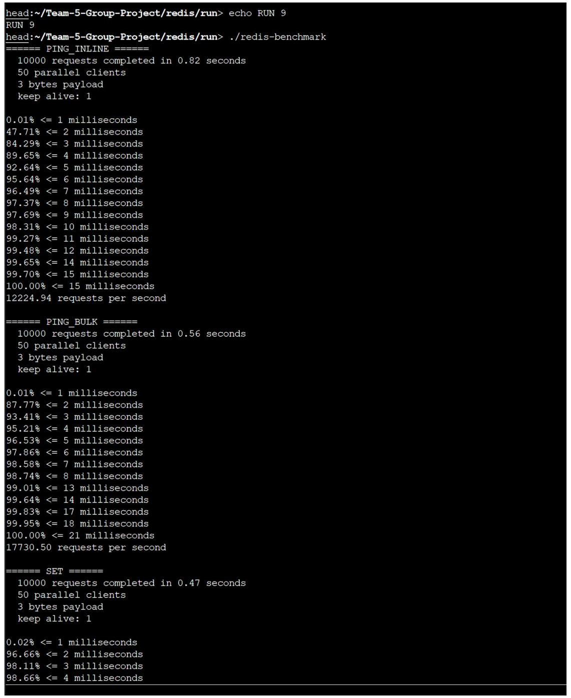
```
head:~/Team-5-Group-Project/redis> cd run
head:~/Team-5-Group-Project/redis/run> ls
backup.db  dump.rdb  redis-benchmark  redis-cli  redis.conf  redis-server
head:~/Team-5-Group-Project/redis/run> ./redis-benchmark
====== PING_INLINE ======
  10000 requests completed in 0.90 seconds
  50 parallel clients
  3 bytes payload
  keep alive: 1

0.01% <= 2 milliseconds
89.61% <= 3 milliseconds
97.89% <= 4 milliseconds
99.86% <= 5 milliseconds
99.95% <= 6 milliseconds
100.00% <= 6 milliseconds
11061.95 requests per second

====== PING_BULK ======
  10000 requests completed in 0.79 seconds
  50 parallel clients
  3 bytes payload
  keep alive: 1

0.01% <= 1 milliseconds
39.77% <= 2 milliseconds
96.32% <= 3 milliseconds
99.64% <= 4 milliseconds
99.96% <= 5 milliseconds
99.98% <= 6 milliseconds
100.00% <= 6 milliseconds
12642.22 requests per second

====== SET ======
  10000 requests completed in 0.76 seconds
  50 parallel clients
  3 bytes payload
  keep alive: 1

0.01% <= 1 milliseconds
37.30% <= 2 milliseconds
92.24% <= 3 milliseconds
98.83% <= 4 milliseconds
99.74% <= 5 milliseconds
99.93% <= 6 milliseconds
100.00% <= 6 milliseconds
13071.90 requests per second

====== GET ======
  10000 requests completed in 0.90 seconds
```

Results-Averages

Redis command	NATIVE	Container
PING	.858 seconds	2.56 seconds
SET	.686 seconds	2.5238 seconds
GET	.62 seconds	2.3414 seconds
LPOP-(Return and remove from top of queue)	.536 seconds	2.0486 seconds
MSET-(Replace existing value)	.754 seconds	1.8054 seconds
LRANGE_100-(Return within a range of 100 items)	.72 seconds	2.0736 seconds

Problems encountered

- Data leads me to believe an error occurred,but everything seems configured correctly
- Outdated github repo
  - Modifying dockerfile
  - Modifying benchmarks
  - Outputting the results to graphs
- Testing with one system
- Learning to change file permissions



```
head:~/Team-5-Group-Project/redis/run> echo RUN 9
RUN 9
head:~/Team-5-Group-Project/redis/run> ./redis-benchmark
====== PING_INLINE ======
  10000 requests completed in 0.82 seconds
  50 parallel clients
  3 bytes payload
  keep alive: 1

0.01% <= 1 milliseconds
47.71% <= 2 milliseconds
84.29% <= 3 milliseconds
89.65% <= 4 milliseconds
92.64% <= 5 milliseconds
95.64% <= 6 milliseconds
96.49% <= 7 milliseconds
97.37% <= 8 milliseconds
97.69% <= 9 milliseconds
98.31% <= 10 milliseconds
99.27% <= 11 milliseconds
99.48% <= 12 milliseconds
99.65% <= 14 milliseconds
99.70% <= 15 milliseconds
100.00% <= 15 milliseconds
12224.94 requests per second

====== PING_BULK ======
  10000 requests completed in 0.56 seconds
  50 parallel clients
  3 bytes payload
  keep alive: 1

0.01% <= 1 milliseconds
87.77% <= 2 milliseconds
93.41% <= 3 milliseconds
95.21% <= 4 milliseconds
96.53% <= 5 milliseconds
97.86% <= 6 milliseconds
98.58% <= 7 milliseconds
98.74% <= 8 milliseconds
99.01% <= 13 milliseconds
99.64% <= 14 milliseconds
99.83% <= 17 milliseconds
99.95% <= 18 milliseconds
100.00% <= 21 milliseconds
17730.50 requests per second

====== SET ======
  10000 requests completed in 0.47 seconds
  50 parallel clients
  3 bytes payload
  keep alive: 1

0.02% <= 1 milliseconds
96.66% <= 2 milliseconds
98.11% <= 3 milliseconds
98.66% <= 4 milliseconds
```

MySQL

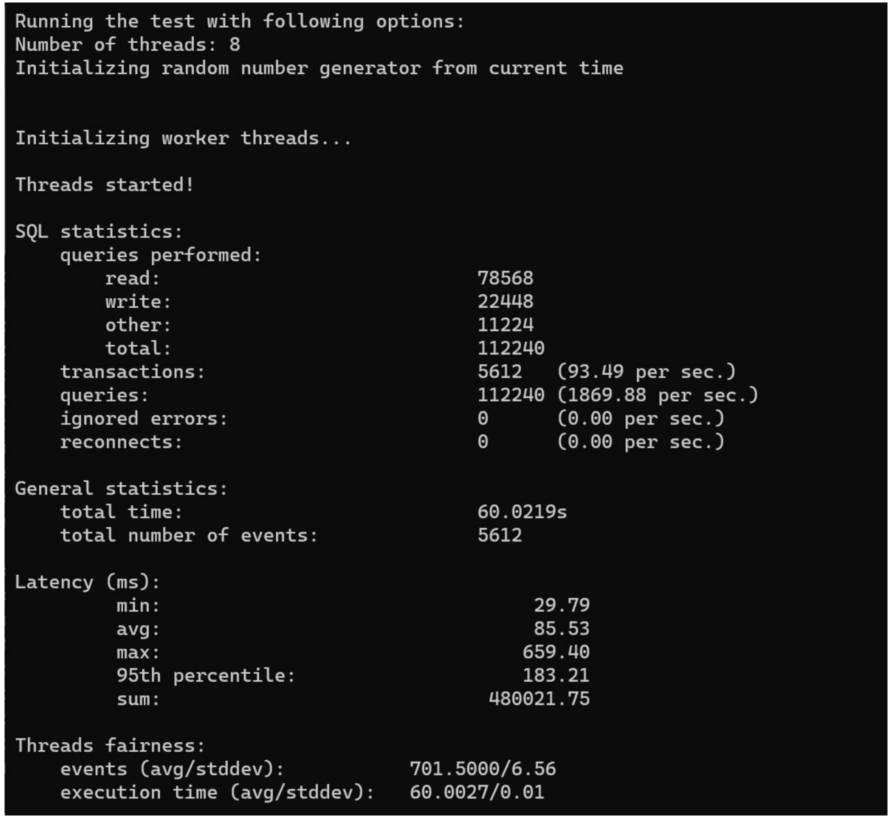
- Sysbench is an open-source benchmarking tool commonly used to test MySQL databases.
- The OTLP (OnLine Transaction Processing) test is a Sysbench test designed to test databases and is the specific benchmark used for MySQL.
- The test operates by creating a database of “transactions” to go through and sends them through the MySQL database to measure performance.

MySQL - Difficulties

- The in-depth benchmarking tool was in-depth, but this resulted in it being difficult to set up correctly.
- Persistent permission errors were only able to be fixed by logging out and back in again.
- Setup for the testing database created multiple errors, requiring a certain degree of brute forcing errors the setup arguments.
- Encountered errors from outdated version differences during initial setup.

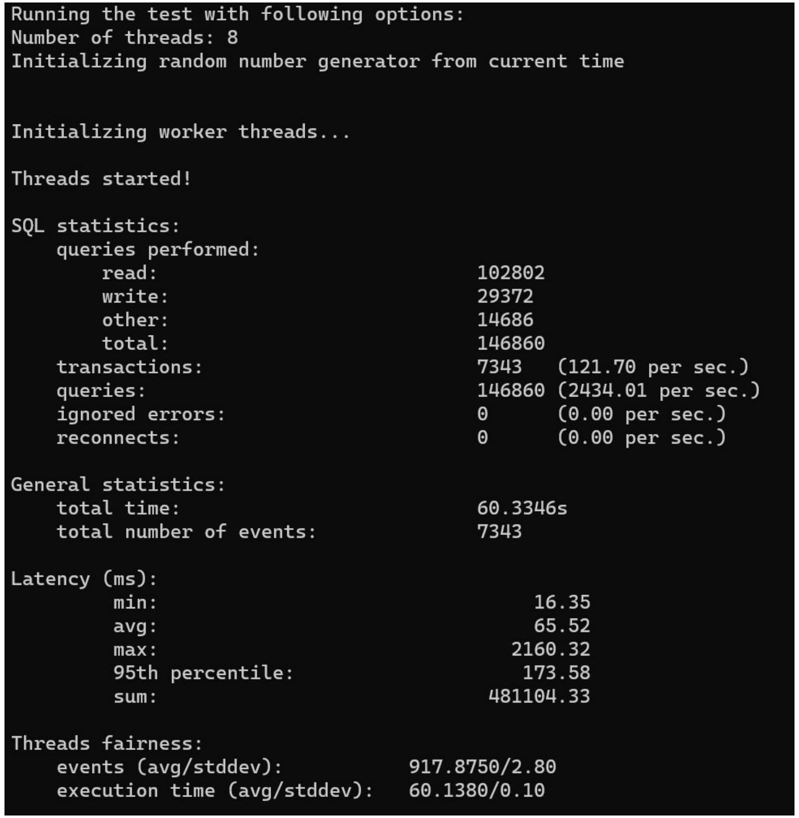
MySQL Results

[Container]



```
Running the test with following options:
Number of threads: 8
Initializing random number generator from current time


Initializing worker threads...

Threads started!

SQL statistics:
    queries performed:
        read:                           78568
        write:                          22448
        other:                          11224
        total:                          112240
    transactions:                       5612   (93.49 per sec.)
    queries:                            112240 (1869.88 per sec.)
    ignored errors:                     0      (0.00 per sec.)
    reconnects:                         0      (0.00 per sec.)

General statistics:
    total time:                         60.0219s
    total number of events:             5612

Latency (ms):
         min:                                   29.79
         avg:                                   85.53
         max:                                  659.40
         95th percentile:                      183.21
         sum:                               480021.75

Threads fairness:
    events (avg/stddev):          701.5000/6.56
    execution time (avg/stddev):  60.0027/0.01
```

[Native]

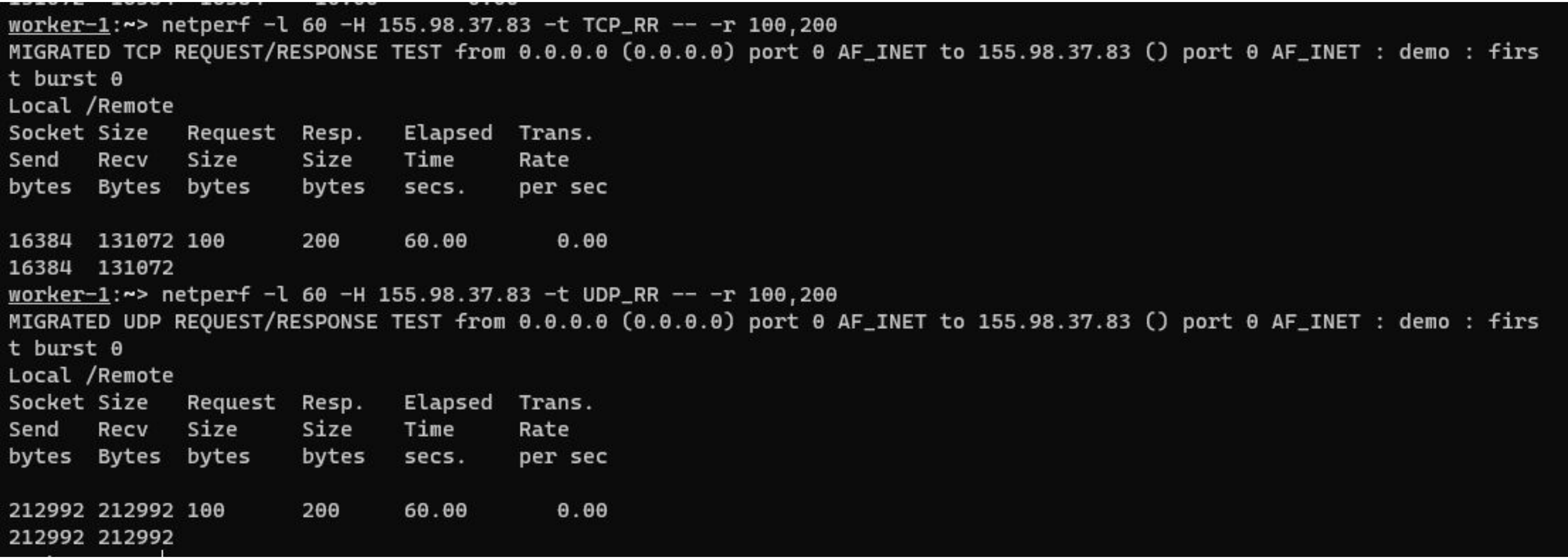


```
Running the test with following options:
Number of threads: 8
Initializing random number generator from current time


Initializing worker threads...

Threads started!

SQL statistics:
    queries performed:
        read:                           102802
        write:                          29372
        other:                          14686
        total:                          146860
    transactions:                       7343   (121.70 per sec.)
    queries:                            146860 (2434.01 per sec.)
    ignored errors:                     0      (0.00 per sec.)
    reconnects:                         0      (0.00 per sec.)

General statistics:
    total time:                         60.3346s
    total number of events:             7343

Latency (ms):
         min:                                   16.35
         avg:                                   65.52
         max:                                 2160.32
         95th percentile:                      173.58
         sum:                               481104.33

Threads fairness:
    events (avg/stddev):          917.8750/2.80
    execution time (avg/stddev):  60.1380/0.10
```

Netperf

- Benchmark that tests unidirectional throughput and end-to-end latency.
- Benchmark mainly focuses on bulk-data transfer
- The server will run a netserver that the client connects to with netperf commands
- For this project the 2 netperf tests that were ran were the TCP_RR and UDP_RR test which test the transaction rate of the connection with a request size of 100 bytes and a response size of 200 bytes

Netperf Results - Docker



```
worker-1:~> netperf -l 60 -H 155.98.37.83 -t TCP_RR -- -r 100,200
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 155.98.37.83 () port 0 AF_INET : demo : firs
t burst 0
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

16384  131072 100      200     60.00       0.00
16384  131072
worker-1:~> netperf -l 60 -H 155.98.37.83 -t UDP_RR -- -r 100,200
MIGRATED UDP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 155.98.37.83 () port 0 AF_INET : demo : firs
t burst 0
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec

212992 212992 100      200     60.00       0.00
212992 212992
```

Netperf Results - Native

- Each test was ran 10 times, for a total of 20 tests
- Unsure why our test was so much worse than the original paper, but we did experience similar time decrease when going from TCP to UDP

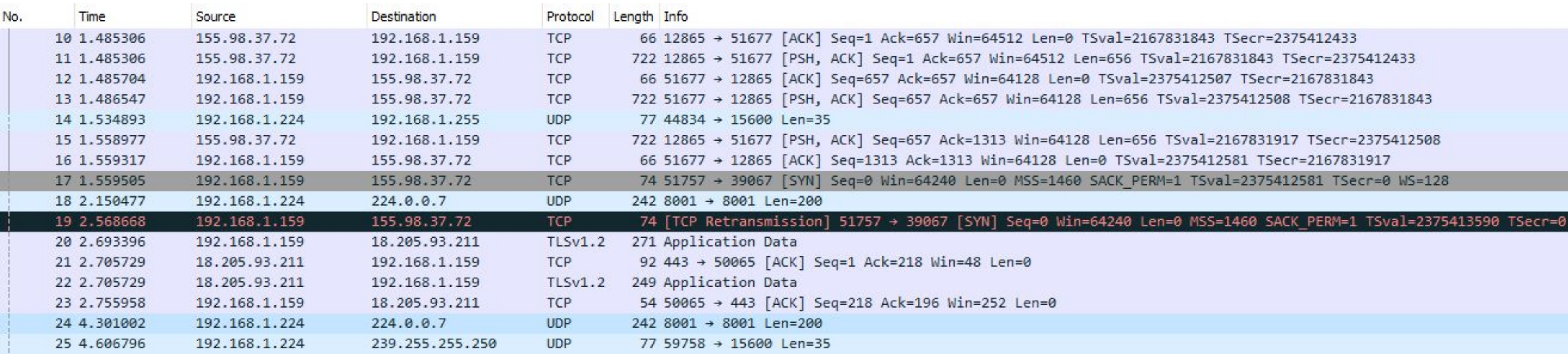
Netperf Test	Average Mean Latency	IBM Research Paper
TCP_RR	436 microseconds	~37 microseconds
UDP_RR	423 microseconds	~35 microseconds

Technical Difficulties

- Had to adjust the Dockerfile for netperf to get it to install with no errors and have it constantly running
- Confusion in figuring out how to properly set it up, whether the same node should have the netserver container and run the netperf tests
- After getting it to install there were a few firewall commands that had to be learned to allow a connection (sudo ufw allow <port number>, sudo ufw enable, sudo ufw allow ssh)
- Currently getting 0 transaction rate per second when running netperf tests with docker

Wireshark Run

- Used wireshark to try and find out why there was a 0 trans. rate



```
No.    Time       Source          Destination      Protocol  Length Info
    10 1.485306   155.98.37.72    192.168.1.159    TCP          66 12865 → 51677 [ACK] Seq=1 Ack=657 Win=64512 Len=0 TSval=2167831843 TSecr=2375412433
    11 1.485306   155.98.37.72    192.168.1.159    TCP         722 12865 → 51677 [PSH, ACK] Seq=1 Ack=657 Win=64512 Len=656 TSval=2167831843 TSecr=2375412433
    12 1.485704   192.168.1.159   155.98.37.72     TCP          66 51677 → 12865 [ACK] Seq=657 Ack=657 Win=64128 Len=0 TSval=2375412507 TSecr=2167831843
    13 1.486547   192.168.1.159   155.98.37.72     TCP         722 51677 → 12865 [PSH, ACK] Seq=657 Ack=657 Win=64128 Len=656 TSval=2375412508 TSecr=2167831843
    14 1.534893   192.168.1.224   192.168.1.255    UDP          77 44834 → 15600 Len=35
    15 1.558977   155.98.37.72    192.168.1.159    TCP         722 12865 → 51677 [PSH, ACK] Seq=657 Ack=1313 Win=64128 Len=656 TSval=2167831917 TSecr=2375412508
    16 1.559317   192.168.1.159   155.98.37.72     TCP          66 51677 → 12865 [ACK] Seq=1313 Ack=1313 Win=64128 Len=0 TSval=2375412581 TSecr=2167831917
    17 1.559505   192.168.1.159   155.98.37.72     TCP          74 51757 → 39067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2375412581 TSecr=0 WS=128
    18 2.150477   192.168.1.224   224.0.0.7        UDP         242 8001 → 8001 Len=200
    19 2.568668   192.168.1.159   155.98.37.72     TCP          74 [TCP Retransmission] 51757 → 39067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2375413590 TSecr=0
    20 2.693396   192.168.1.159   18.205.93.211    TLSv1.2     271 Application Data
    21 2.705729   18.205.93.211   192.168.1.159    TCP          92 443 → 50065 [ACK] Seq=1 Ack=218 Win=48 Len=0
    22 2.705729   18.205.93.211   192.168.1.159    TLSv1.2     249 Application Data
    23 2.755958   192.168.1.159   18.205.93.211    TCP          54 50065 → 443 [ACK] Seq=218 Ack=196 Win=252 Len=0
    24 4.301002   192.168.1.224   224.0.0.7        UDP         242 8001 → 8001 Len=200
    25 4.606796   192.168.1.224   239.255.255.250  UDP          77 59758 → 15600 Len=35
```

Linpack

- The Linpack benchmark was introduced by Jack Dongarra which measures a given system's floating-point computing power. It is used to measure how fast a computer solves a dense n by n system of linear equations Ax = b.
- The bench mark was ran twice using two dockerfiles. Dockerfile.oneSocket and Dockerfile.twoSocket. The difference between the two is that oneSocket uses 16 threads while twoSocket uses 32.
- In our local tests the twoSocket benchmark completed the trials about twice as fast as the oneSocket benchmark.
- For our docker image test I saw that it took about the same amount of time as our local tests.

Running Linpack steps.

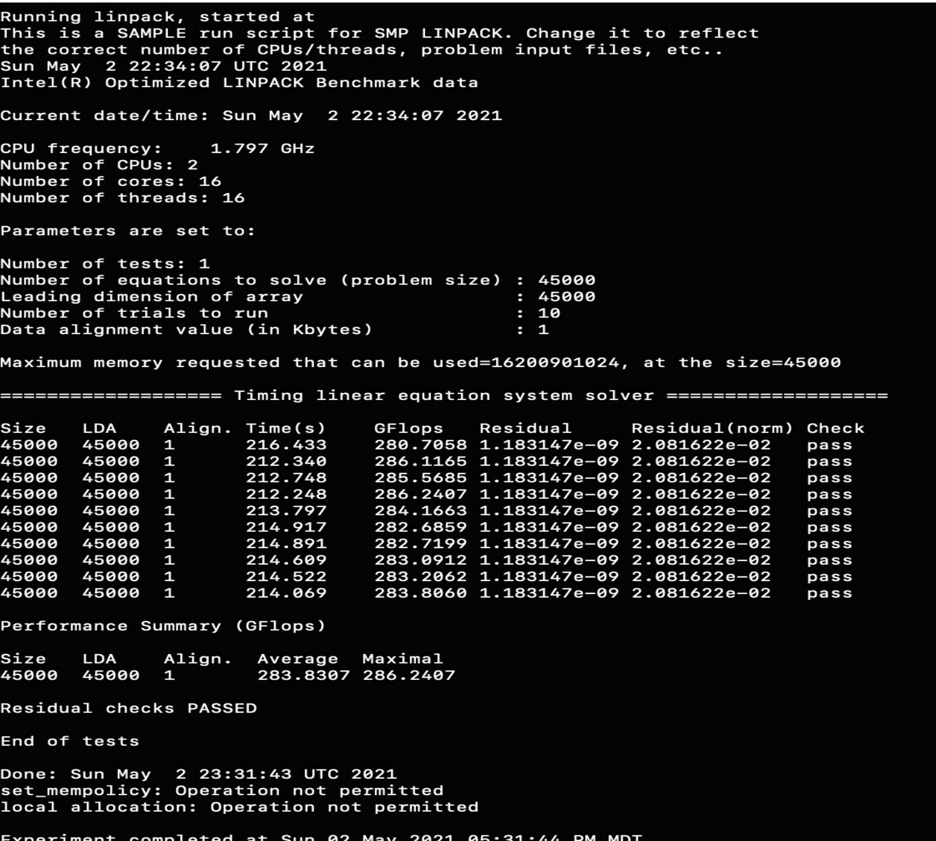
Step 1: ssh into the Raw Pc node. `ssh -p 22 cj894884@pc764.emulab.net`
Step 2: Navigate to the local/repository/linpack directory in terminal

Step 3: enable the setup script chmod +x change_files.sh (This allows access to all of the necessary files to run the benchmark.)

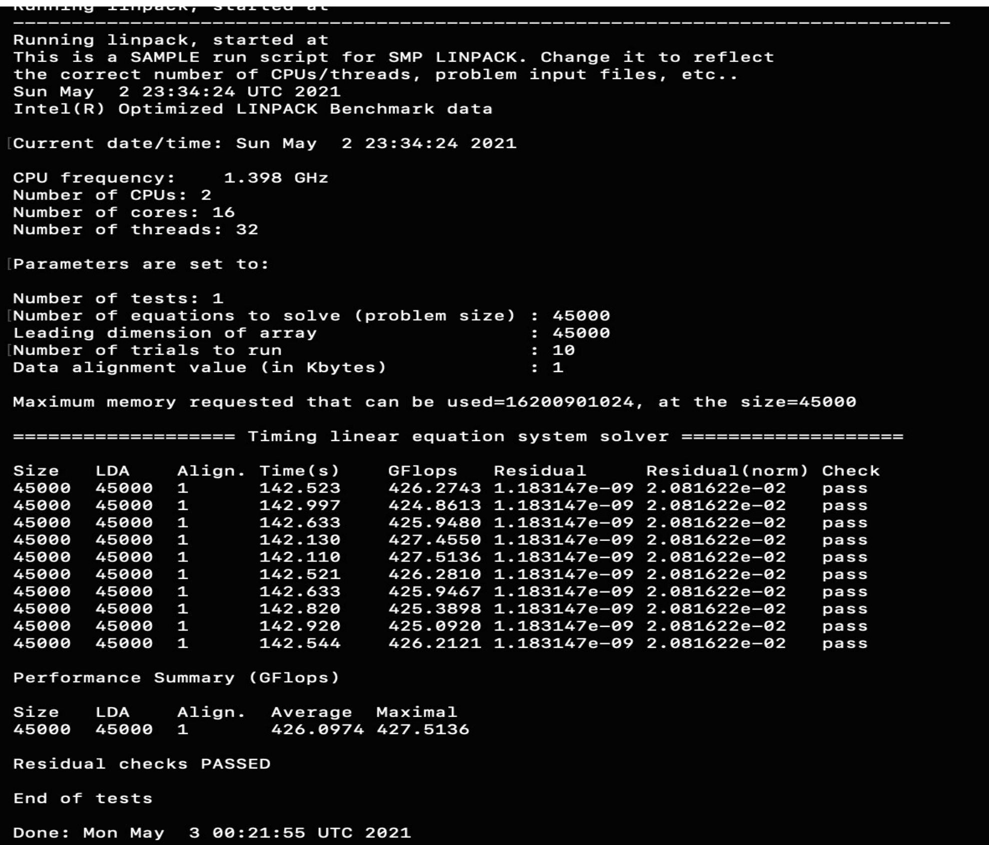
Step 4: run benchmark script ./runBench.sh (currently only works for our local test will be implementing another script to execute our docker images.)

oneSocket local Results

twoSocket local Results

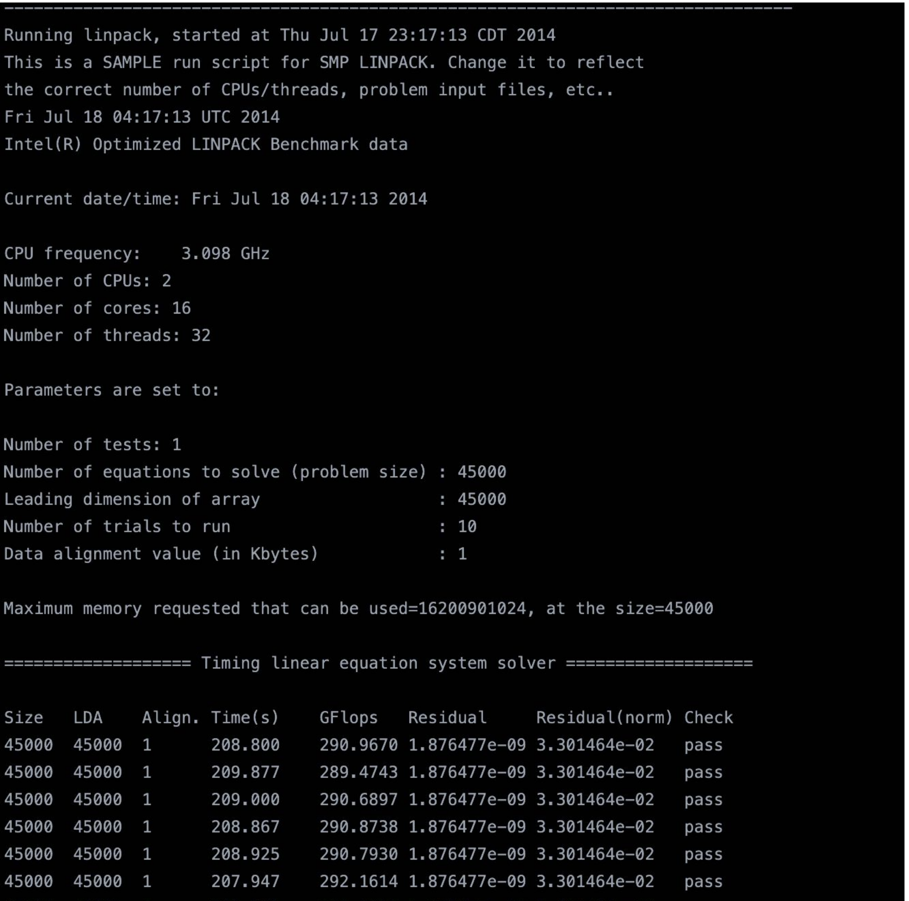


```
Running linpack, started at
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
Sun May  2 22:34:07 UTC 2021
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Sun May  2 22:34:07 2021

CPU frequency:    1.797 GHz
Number of CPUs: 2
Number of cores: 16
Number of threads: 16

Parameters are set to:

Number of tests: 1
Number of equations to solve (problem size) : 45000
Leading dimension of array                  : 45000
Number of trials to run                     : 10
Data alignment value (in Kbytes)            : 1

Maximum memory requested that can be used=16200901024, at the size=45000

=================== Timing linear equation system solver ===================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
45000  45000  1      216.433    280.7058 1.183147e-09 2.081622e-02   pass
45000  45000  1      212.340    286.1165 1.183147e-09 2.081622e-02   pass
45000  45000  1      212.748    285.5685 1.183147e-09 2.081622e-02   pass
45000  45000  1      212.248    286.2407 1.183147e-09 2.081622e-02   pass
45000  45000  1      213.797    284.1663 1.183147e-09 2.081622e-02   pass
45000  45000  1      214.917    282.6859 1.183147e-09 2.081622e-02   pass
45000  45000  1      214.891    282.7199 1.183147e-09 2.081622e-02   pass
45000  45000  1      214.609    283.0912 1.183147e-09 2.081622e-02   pass
45000  45000  1      214.522    283.2062 1.183147e-09 2.081622e-02   pass
45000  45000  1      214.069    283.8060 1.183147e-09 2.081622e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
45000  45000  1       283.8307 286.2407

Residual checks PASSED

End of tests

Done: Sun May  2 23:31:43 UTC 2021
set_mempolicy: Operation not permitted
local allocation: Operation not permitted

Experiment completed at Sun 02 May 2021 05:31:44 PM MDT
```



```
Running linpack, started at
--------------------------------------------------------------------------------
Running linpack, started at
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
Sun May  2 23:34:24 UTC 2021
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Sun May  2 23:34:24 2021

CPU frequency:    1.398 GHz
Number of CPUs: 2
Number of cores: 16
Number of threads: 32

Parameters are set to:

Number of tests: 1
Number of equations to solve (problem size) : 45000
Leading dimension of array                  : 45000
Number of trials to run                     : 10
Data alignment value (in Kbytes)            : 1

Maximum memory requested that can be used=16200901024, at the size=45000

=================== Timing linear equation system solver ===================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
45000  45000  1      142.523    426.2743 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.997    424.8613 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.633    425.9480 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.130    427.4550 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.110    427.5136 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.521    426.2810 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.633    425.9467 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.820    425.3898 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.920    425.0920 1.183147e-09 2.081622e-02   pass
45000  45000  1      142.544    426.2121 1.183147e-09 2.081622e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
45000  45000  1       426.0974 427.5136

Residual checks PASSED

End of tests

Done: Mon May  3 00:21:55 UTC 2021
```

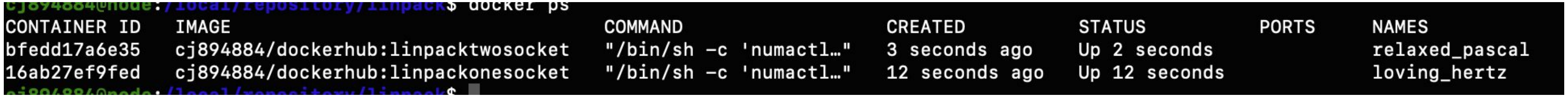
Research twoSocket local Results



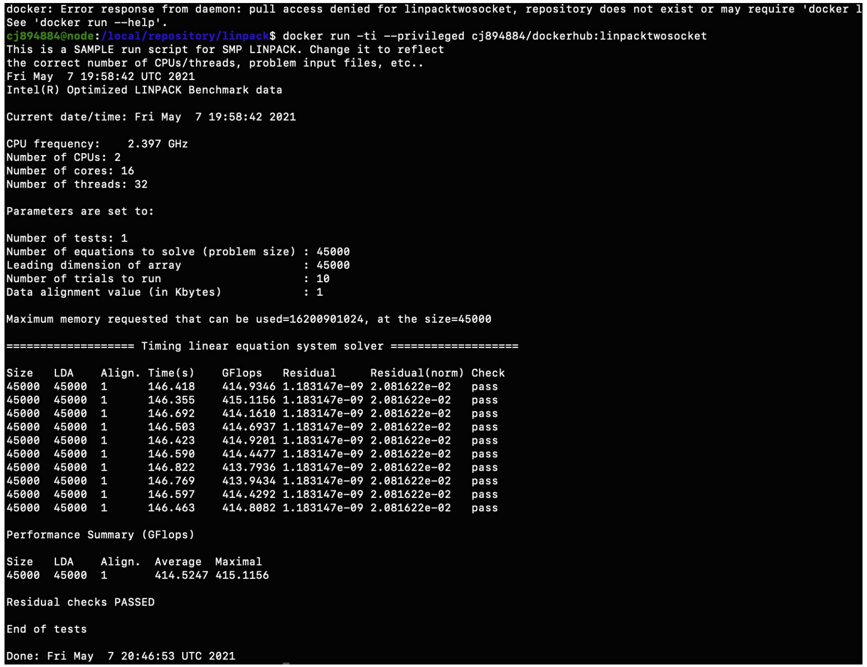
```
--------------------------------------------------------------------------------
Running linpack, started at Thu Jul 17 23:17:13 CDT 2014
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
Fri Jul 18 04:17:13 UTC 2014
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Fri Jul 18 04:17:13 2014

CPU frequency:    3.098 GHz
Number of CPUs: 2
Number of cores: 16
Number of threads: 32

Parameters are set to:

Number of tests: 1
Number of equations to solve (problem size) : 45000
Leading dimension of array                  : 45000
Number of trials to run                     : 10
Data alignment value (in Kbytes)            : 1

Maximum memory requested that can be used=16200901024, at the size=45000

=================== Timing linear equation system solver ===================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
45000  45000  1      208.800    290.9670 1.876477e-09 3.301464e-02   pass
45000  45000  1      209.877    289.4743 1.876477e-09 3.301464e-02   pass
45000  45000  1      209.000    290.6897 1.876477e-09 3.301464e-02   pass
45000  45000  1      208.867    290.8738 1.876477e-09 3.301464e-02   pass
45000  45000  1      208.925    290.7930 1.876477e-09 3.301464e-02   pass
45000  45000  1      207.947    292.1614 1.876477e-09 3.301464e-02   pass
```

Our experiment results show that our test's seem to have run 60 seconds faster as well as have almost twice as much GFlops compared to the KVM research paper.

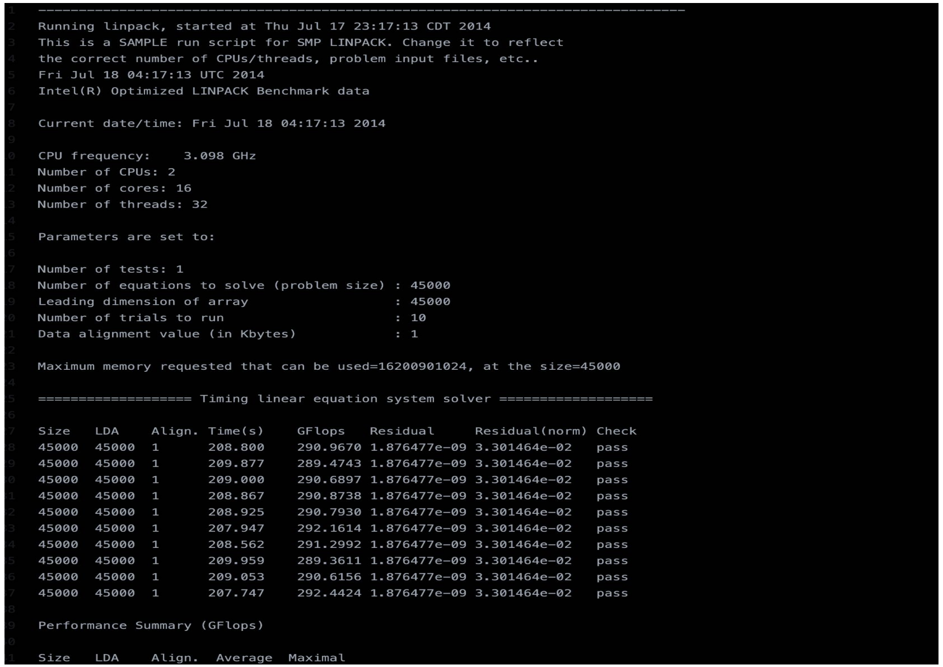
twoSocket docker Image Results vs Research Docker Results



```
cj894884@node:/local/repository/linpack$ docker ps
CONTAINER ID   IMAGE                                 COMMAND                  CREATED          STATUS          PORTS     NAMES
bfedd17a6e35   cj894884/dockerhub:linpacktwosocket   "/bin/sh -c 'numactl…"   3 seconds ago    Up 2 seconds              relaxed_pascal
16ab27ef9fed   cj894884/dockerhub:linpackonesocket   "/bin/sh -c 'numactl…"   12 seconds ago   Up 12 seconds             loving_hertz
cj894884@node:/local/repository/linpack$
```



```
docker: Error response from daemon: pull access denied for linpacktwosocket, repository does not exist or may require 'docker l
See 'docker run --help'.
cj894884@node:/local/repository/linpack$ docker run -ti --privileged cj894884/dockerhub:linpacktwosocket
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
Fri May  7 19:58:42 UTC 2021
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Fri May  7 19:58:42 2021

CPU frequency:    2.397 GHz
Number of CPUs: 2
Number of cores: 16
Number of threads: 32

Parameters are set to:

Number of tests: 1
Number of equations to solve (problem size) : 45000
Leading dimension of array                  : 45000
Number of trials to run                     : 10
Data alignment value (in Kbytes)            : 1

Maximum memory requested that can be used=16200901024, at the size=45000

=================== Timing linear equation system solver ===================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
45000  45000  1      146.418    414.9346 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.355    415.1156 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.692    414.1610 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.503    414.6937 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.423    414.9201 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.590    414.4477 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.822    413.7936 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.769    413.9434 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.597    414.4292 1.183147e-09 2.081622e-02   pass
45000  45000  1      146.463    414.8082 1.183147e-09 2.081622e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
45000  45000  1       414.5247 415.1156

Residual checks PASSED

End of tests

Done: Fri May  7 20:46:53 UTC 2021
```



```
--------------------------------------------------------------------------------
Running linpack, started at Thu Jul 17 23:17:13 CDT 2014
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
Fri Jul 18 04:17:13 UTC 2014
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Fri Jul 18 04:17:13 2014

CPU frequency:    3.098 GHz
Number of CPUs: 2
Number of cores: 16
Number of threads: 32

Parameters are set to:

Number of tests: 1
Number of equations to solve (problem size) : 45000
Leading dimension of array                  : 45000
Number of trials to run                     : 10
Data alignment value (in Kbytes)            : 1

Maximum memory requested that can be used=16200901024, at the size=45000

=================== Timing linear equation system solver ===================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
45000  45000  1      208.800    290.9670 1.876477e-09 3.301464e-02   pass
45000  45000  1      209.877    289.4743 1.876477e-09 3.301464e-02   pass
45000  45000  1      209.000    290.6897 1.876477e-09 3.301464e-02   pass
45000  45000  1      208.867    290.8738 1.876477e-09 3.301464e-02   pass
45000  45000  1      208.925    290.7930 1.876477e-09 3.301464e-02   pass
45000  45000  1      207.947    292.1614 1.876477e-09 3.301464e-02   pass
45000  45000  1      208.562    291.2992 1.876477e-09 3.301464e-02   pass
45000  45000  1      209.959    289.3611 1.876477e-09 3.301464e-02   pass
45000  45000  1      209.053    290.6156 1.876477e-09 3.301464e-02   pass
45000  45000  1      207.747    292.4424 1.876477e-09 3.301464e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
```

Technical Difficulties

When using a raw pc node I ran into a permissions error. That looked like this: docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/create: dial unix /var/run/docker.sock: connect: permission denied.
Running a simple chmod command that looked like this fixed the issue: sudo chmod 666 /var/run/docker.sock

The only other issues faced with linpack was figuring out how to get around the blocker with numactl: Running the docker image in privileged mode fixed the issue. The command look like this: docker run -ti --privileged linpack