International Journal of Engineering Research and Advanced Technology (IJERAT)

E-ISSN : 2454-6135

Volume.4, Issue 6

June -2018

DOI:http://dx.doi.org/10.31695/IJERAT.2018.3282

Testing MPI Applications Performance in Singularity Containers

Alexey Vasyukov[1] and Katerina Beklemysheva[2]

Associate Professor [1 2]

Department of Computer Science and Numerical Mathematics

Moscow Institute of Physics and Technology

Dolgoprudny

Russia

ABSTRACT

The papers covers field experience of testing performance of MPI applications when running in Singularity containers compared with bare metal. Using computing containers is a major trend in many areas, including high performance computing. Traditionally Docker is the most commonly used container provider. However, using Docker in HPC is not that native, compared with Internet or Enterprise use cases. One of the major concerns is the fact that HPC clusters depend heavily on high speed network, that is based usually on Infiniband interconnect, and Docker was not designed with this environment in mind. Singularity container technology was created recently as an alternative to Docker, but designed from ground up for HPC use case. Among other features, Singularity uses completely different approach to organizing network communication between containers that should fit better for distributed applications that rely on MPI over high speed low latency network. This paper concentrates on testing Singularity containers performance when running High Performance Linpack test suite and using MPI over Infiniband for network communication. The paper shows that Singularity containers allow achieving performance metrics that corresponding to the values for bare metal in the default container setup, without having to perform a separate network configuration on the host system side. At the same time, it was found that libraries mismatch on the host system and on container side may cause unexpected significant performance degradation.

Key Words: *Singularity, containers, MPI, cloud computing, high performance computing.*

1. INTRODUCTION

This paper is devoted to testing the performance of distributed scientific applications when running in Singularity computing containers. Previously, the authors performed similar work for Docker containers [1]. From the previous work it should be noted that the most difficult part of tuning is the network infrastructure. Guaranteed high performance is ensured only if devices were presented to Docker container in passthrough mode, which is not a good choice from operations point of view. This fact is a consequence of the general Docker architecture that aims to provice maximum isolation of containers from each other, which causes noticeable problems when working with a high-speed network and launching MPI applications.

Similar results are given in the reports of other researchers [2], [3]. In [3], as a solution to the problem, the authors suggest the logical structure of Singularity containers, different from Docker and more adapted to the specifics of high-performance computing.

2. TESTING ENVIRONMENT

The following test environment was used:

- 2 servers, each with Intel (R) Core (TM) i7-5820K CPU @ 3.30GHz and 64 GB of RAM
- QDR Infiniband for communication between servers
- Fedora 26 x86_64 as an operating system on servers
- singularity version 2.0-10.fc25.x86_64 from Fedora repositories
- Fedora 28 and CentOS 7 in containers
- High Performance Linpack version 2.1 [4] is used as a model load for all tests.

It is known that the optimization of the results of High Performance Linpack is a separate complex subject, requiring fine-tuning of parameters for a specific environment. In this paper, we did not aim to obtain the maximum possible results with respect to theoretical limits. The aim of the work is to compare the typical results when application is running bare meta and in a computing container. In this regard, all the environments used the following High Performance Linpack configuration file:

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
5            # of problems sizes (N)
8000 16000 32000 64000 96000         Ns
1            # of NBs
128          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
4            Ps
3            Qs
16.0         threshold
1            # of panel fact
0            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

On the problems of maximum dimension, about 60% of the available RAM of the cluster was used; at these parameters, the actual performance achieved is approximately 80% of the theoretical, which is sufficient for the purposes of this work.

DOI : http://dx.doi.org/10.31695/IJERAT.2018.3282

3. TESTING RESULTS

The following results were obtained for bare metal run:

/usr/lib64/mpich/bin/mpirun -np 12 --hostfile bb-ib.mpi /usr/lib64/mpich/bin/xhpl_mpich

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4       8000   128     4     3               1.68              2.036e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0026161 ..... PASSED
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4      16000   128     4     3               8.65              3.157e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0021456 ..... PASSED
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4      32000   128     4     3              54.91              3.979e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0019033 ..... PASSED
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4      64000   128     4     3             383.82              4.553e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0017738 ..... PASSED
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4      96000   128     4     3            1310.05              4.502e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0016447 ..... PASSED
================================================================================
```

Containers with Fedora 28 demonstrated the following performance numbers.

/usr/lib64/mpich/bin/mpirun -np 12 --hostfile bb-ib.mpi singularity exec fedora-hpl.sapp /usr/lib64/mpich/bin/xhpl_mpich

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4       8000   128     4     3               1.61              2.114e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0026161 ..... PASSED
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L4      16000   128     4     3               8.36              3.268e+02
--------------------------------------------------------------------------------
```

```
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=       0.0021456 ..... PASSED
================================================================================
T/V            N    NB     P     Q          Time             Gflops
--------------------------------------------------------------------------------
WR00L2L4       32000  128    4    3           54.20           4.031e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=       0.0019033 ..... PASSED
================================================================================
T/V            N    NB     P     Q          Time             Gflops
--------------------------------------------------------------------------------
WR00L2L4       64000  128    4    3          381.36           4.583e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=       0.0017738 ..... PASSED
================================================================================
T/V            N    NB     P     Q          Time             Gflops
--------------------------------------------------------------------------------
WR00L2L4       96000  128    4    3         1304.40           4.522e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=       0.0016447 ..... PASSED
================================================================================
```

Thus, if we eliminate the inevitable random fluctuations, it is clear that the performance of bare metal and using containers is not different, going to 450 GFLOPS for a sufficiently large dimension of the problem. This fact fully agrees with the expectations of the Singularity architecture, in which the influence of the container on the network interaction of processes is virtually eliminated.

Separately it is worth noting that when launching in containers with CentOS 7 the first two tests demonstrated performance 4.7 GFLOPS and 10.1 GFLOPS respectively. At this point, the tests were stopped. It is seen that there is a drop in performance by one and a half orders of magnitude. A detailed study of this issue showed that the problem is caused by a bad combination of versions of libraries for working with Infiniband between the host system and the container. After the required versions of the libraries were built in the container, the problem was completely removed.

Singularity documentation mentions the fact that performance is highly dependent on the accurate configuration of OFED on the host and in the container. This restriction is completely reasonable and is caused by OFED architecture. Nevertheless, it turns out to be essential for achieving mobility of compute for distributed applications. From the above test, we can see that a simple transfer of the computational container image does not allow achieving the expected performance in the new environment, and a significant modification of the container directly on the target system is required.

4. CONCLUSION

Singularity containers allow achieving performance metrics that corresponding to the values for bare metal in the default container setup, without having to perform a separate network configuration on the host system side.

At the same time, it was found that libraries mismatch on the host system and on container side may cause unexpected significant performance degradation. The worst thing is that the issue cannot be diagnosed by obvious basic checks.

However, the ease of use of Singularity containers makes them a logical choice for HPC systems of large scale.

ACKNOWLEDGMENT

The research was supported by Russian Foundation for Basic Research grant 15-29-07096.

DOI : http://dx.doi.org/10.31695/IJERAT.2018.3282

REFERENCES

[1] A. Ermakov, A. Vasyukov, "Testing Docker Performance for HPC Applications," in *IOSR Journal of Computer Engineering*, vol. 20, pp. 36-43, 2018.

[2] Shifter: Bringing Linux containers to HPC.[Online] Available https://www.nersc.gov/research-and-development/user-defined-images/ [Accessed: June. 20, 2018]

[3] G.M. Kurtzer, "Singularity: Containers for Science" Presented at HPC Advisory Council Stanford Workshop, 2017

[4] High Performance Linpack [Online]. Available http://www.netlib.org/benchmark/hpl/ [Accessed: June. 20, 2018]