Performance of Software-based Encrypted MPI Communication over Container Clusters

Mohsen Gavahi, Abu Naser, Mehran Sadeghi Lahijani, Cong Wu, Zhi Wang, and Xin Yuan,
Department of Computer Science, Florida State University, Tallahassee, FL 32306
Email: {gavahi,naser,sadeghil,wu,zwang,xyuan}@cs.fsu.edu

***Abstract*—We study the performance of software-based secure communication infrastructure for HPC Message Passing Interface (MPI) applications in container clusters. Specifically, extensive experiments are performed using micro- and application benchmarks to evaluate the encrypted MPI communication performance. Container built-in encrypted communication schemes including Docker Swarm and Kubernetes Antrea and Calico, as well as CryptMPI, a secure MPI library, are evaluated and compared. Our results confirm the findings in earlier studies that for some MPI applications, running in the container environment with unencrypted communication introduces only minor overheads over running on the bare metal system. However, when the communications are encrypted, all of the container built-in software-based encrypted communication mechanisms that we evaluated incur very large overheads in all of our experiments for both micro-benchmarks and application benchmarks. On the other hand, CryptMPI, which encrypts and decrypts messages in the MPI library, achieves much higher performance than the container built-in encryption schemes.**

***Keywords*-Performance, MPI, Encryption, Container, Docker, Singularity**

I. Introduction

We study the performance of encrypted communication for HPC applications developed with Message Passing Interface (MPI) on container clusters. The methods in the container environment (or cloud platforms in general) to support encrypted communication can be classified into hardware-based and software-based. The hardware-based solution is to equip compute nodes with smartNICs that have hardware-enabled encryption offload technology [1]. With the hardware solution, the encryption and decryption are performed by hardware in the smartNICs at the line rate. For systems whose compute nodes do not have smartNICs, a software-based solution, which performs encryption and decryption in the software, can be used. Major containerization technologies, including Docker Swarm [2] and Kubernetes Antrea [3] and Calico [4], have built-in software encrypted communication schemes.

Existing studies on the performance of running MPI applications on containers do not consider encrypted communication [5], [6]. These studies compare the performance of MPI applications running on container clusters to that on the bare metal system with unencrypted communication. The conclusion is that for the applications studied, the overhead introduced by container execution is not significant [5], [6]. The performance of encrypted MPI communication over container clusters, however, has not been thoroughly assessed, especially in the context of running MPI applications. This is the focus of this work.

We perform extensive experiments to evaluate the performance of software-based encrypted MPI communication over container clusters. The experiments use micro-benchmarks from OSU benchmark suite [7] as well as application benchmarks from the NAS parallel benchmarks [8]. Both Docker and Singularity are studied; major container technologies for software-based encrypted communication including Docker Swarm and Kubernetes Antrea and Calico are evaluated and compared. Additionally, we also study the performance of an alternative to the container built-in encrypted communication schemes for MPI applications, CryptMPI [9], [10], a secure MPI library that performs message encryption and decryption in the MPI library. CryptMPI supports encrypted MPI communication over unencrypted communication channels. The major findings of this study include the following:

- The performance of Docker and Singularity is similar either with unencrypted communication or with encrypted communication in our experiments. Overall, Docker Swarm has slightly better performance than Singularity with Kubernetes for both encrypted and unencrypted communications. We did not observe any performance advantage of Singularity although it was designed to run HPC applications.
- For unencrypted communication, both Docker and Singularity achieve high MPI communication performance when the message size is sufficiently large, introducing only minor overheads over the bare metal system. We observe that for BT, CG, LU, MG and SP in the NAS parallel benchmarks, the overheads introduced by container execution for both Docker and Singularity are less than 10%, which confirms the findings in earlier studies [5], [6].
- For encrypted communication, all container built-in schemes in our evaluation (Docker Swarm, Kubernates Antrea and Calico) slow down the communication dramatically, up to a factor of 15. The current container

built-in encrypted communication schemes do not provide high performance encrypted MPI communication.
- CryptMPI achieves much higher performance than all of the container built-in encryption schemes. In some situations, CryptMPI is faster by a factor of more than 10. This indicates that there is significant room for improvement in the current container built-in software-based encrypted communication schemes.

II. Encrypted Communication over Container Clusters

Containers such as Docker [2] and Singularity [11] encapsulate complex programs with their dependencies in isolated environments and offer fast, customizable, portable, and flexible deployments of applications and workloads. To deploy distributed applications that run on many containers over multiple hosts, the containers for the applications need to be managed to work in concert for the applications. This is termed *container orchestration*. The Docker containers can be orchestrated by Docker Swarm (Docker running in the Swarm mode) or Kubernetes [12] while Singularity is orchestrated by Kubernetes.

Container orchestration manages the networking among containers. Due to the importance of encrypted communication, both Docker Swarm and Kubernetes have options to use encrypted communication between nodes. For encrypted MPI communication, another choice is to use a secure MPI library that performs encryption and decryption in the MPI library. CryptMPI [9], [10] is one of such libraries. In the following, we will discuss these options to support secure communication for MPI applications over container clusters.

A. Encrypted communication with Docker Swarm

Docker Swarm supports network security in three levels. First, an isolated network for each application is provided by the overlay driver: different Docker networks are firewalled from one another. Second, the control plane of Docker networks is secured using Public Key Infrastructure (PKI). All control plane communications are encrypted with mutual transport level security (TLS). Third, the data plane is secured with a separate mechanism. The encryption mode of overlay networks can be enabled at the time of creation to encrypt all user traffic between nodes with a secured channel. The encryption method used in Docker Swarm is Internet Protocol Security (IPSec) which uses the AES algorithm in GCM mode [13]. This work evaluates the performance of this component for MPI applications.

B. Encrypted communication with Kubernetes

Networking with Kubernetes is complex. In general, networking configurations and traffic are handled through a combination of internal resources and external services such as Container Network Interface (CNI) [14] plugins. CNI is a network framework that allows the dynamic configuration of networking resources and exposes a simple set of interfaces for adding and removing a container from a network. Kubernetes uses a CNI plugin to create the virtual network interface that a container can use, and to provide added functions and features that enhance network security. Our experiments evaluate two popular plugins: Antrea [3] and Calico [4], which are known for their effective encryption methods. Antrea utilizes IPSec [13], a widely adopted encryption protocol suite, while Calico leverages WireGuard, a relatively new encryption protocol.

C. CryptMPI

CryptMPI [9], [10] is a high-performance secure MPI library that encrypts all MPI inter-node messages. With CryptMPI, encrypted MPI communication can be achieved over unsecure communication channels. CryptMPI incorporates the state-of-the-art AES-GCM encryption scheme (128-bit keys) that supports both privacy and integrity. Note that 128-bit AES-GCM is also used by IPSec. CryptMPI incorporates many techniques to improve encrypted MPI communication performance. For point-to-point communication of large messages, various techniques including overlapping communication with computation, pipelined communication, and multi-threading are employed [10]. To improve the performance of collective operations, novel algorithms were designed for many MPI collective operations [15], [16].

III. Performance of Encrypted MPI Communication over Containers

We perform extensive experiments to evaluate the performance of encrypted MPI communication over containers clusters with different technologies. Next, we will first discuss the experimental settings and then report the results.

A. Experimental setting and methodology

The hardware used in the experiments is a local cluster, called NoleLand. Each compute node on NoleLand runs CentOS-7 and is equipped with two Intel Xeon Gold 6240 CPUs at 2.60 GHz frequency, with 36 cores and 192GB DDR4-2933 RAM. Each node is equipped with a 25Gbps Broadcom BCM57414 NetXtreme-E RDMA Ethernet network for networking.

The experiments are carried out with different containers and different container orchestration systems. Both Docker [2] (version 23.0.1) and Singularity [11] (version 3.8.7-1.el7) are studied. For Docker clusters, the containers can be orchestrated with Docker Swarm or Kubernetes (kubectl version v1.17.5). For Singularity clusters, we used Singularity-CRI [17] which is Singularity-specific implementation of Kubernetes CRI. The CNI plugins for Kubernetes studied are Calico [4] (calicoctl version v3.24.5) and Antrea [3] (version v1.5.3). The same Docker image was used in all experiments, one container per node. The Docker image runs Alpine Linux (version v3.15). For Singularity, we converted

the identical Docker image into the Singularity format and performed experiments with the same settings as those on Docker clusters.

CryptMPI runs on all of the container cluster configurations. CryptMPI built over MPICH-3.3 is used in the evaluation. The library is compiled with the default compiler flags (including the -O2 flag). CryptMPI uses BoringSSL cryptographic library [18] with 128-bit keys which uses the AES-GCM encryption scheme.

Micro-benchmarks and application benchmarks are used in the evaluation. For standalone evaluation of the performance of encrypted MPI point-to-point and collective communication, the OSU benchmark suite [7] is used. In each experiment, we ran the benchmark with the default number of warm-iterations, at least 10 times, up to 100 times, until the standard deviation is within 5% of the arithmetic mean. If after 100 measurements, the standard deviation was still too large then we kept repeating the experiments until the 99% confidence interval was within 1% of the mean. The average of the experiments is reported.

For application benchmarks, we use BT, CG, LU, MG, SP, and FT from the NAS parallel benchmarks [8]. Each application benchmark is repeated 10 times and the average is reported.

B. Point-to-point communication

Table I shows the results for MPI point-to-point communication on Docker clusters with different system settings. The results are measured using the pingpong benchmark in OSU benchmark suite. As can be seen from the table, when message size is small, containers introduce large overheads (about 117% overheads for 4-byte message size). However, when the message size is large, the overhead is small. For example, for the 256KB message size, the overhead is 21%; for the 1MB message size, the overhead is negligible (less than 1%). This indicates that the unencrypted communication infrastructure for containers does not introduce significant overheads when the message size is sufficiently large.

Docker with Kubernetes Calico performs slightly worse than Antrea, and its results are omitted to save space. As can be seen in the table, the overheads for encrypted MPI communication with Docker Swarm and Kubernetes are very large regardless of the message sizes. For example for 1MB message size, the overhead of encrypted MPI over unencrypted MPI is 11.7 times for Docker Swarm and 15.0 times for Kubernetes Antrea.

For small messages, CryptMPI performs worse than Docker Swarm and Kubernetes Antrea. CryptMPI does not optimize point-to-point communication with small messages: it basically performs the encryption/decryption operations like IPSec in the user domain. Thus, it is not surprising that the performance is worse than container built-in schemes like Docker Swarm, which performs the encryption/decryption in the system. For large messages, CryptMPI employs a number of optimizations [10] as discussed in Section II, and achieves much higher performance. For 1MB messages, CryptMPI only introduces a 27.4% overhead over the unencrypted scheme and almost 9 times faster than Docker Swarm and 11 times faster than Kubernetes Antrea.

The results on Singularity are shown in Table II. The trends in the results for Singularity are very similar to those for Docker. For unencrypted communication, Singularity introduces large overheads for small messages, but not for large messages. For encrypted communication, both Calico and Antrea introduces large overheads, especially for large messages. The performance is CryptMPI is worse than Calico and Antrea for small messages, but significantly better for large messages. For 1MB messages, CryptMPI is a factor of more than 10 better than Kubernetes Calico and Antrea.

Table I: MPI point-to-point communication time (in micro seconds) on Docker clusters with different system settings

Size	Unencr. Bare Metal	Unencr. Docker Swarm	Encrypted Docker Swarm	Encrypted K8s Docker Antrea	CryptMPI Docker Swarm
4B	1.53E1	3.32E1	5.31E1	6.20E1	7.68E1
64B	1.77E1	3.47E1	5.54E1	6.46E1	7.84E1
2KB	2.70E1	5.27E1	1.11E2	1.47E2	8.18E1
16KB	4.68E1	7.30E1	2.75E2	3.70E2	9.04E1
256KB	3.00E2	3.64E2	3.57E3	5.04E3	4.75E2
1MB	9.75E2	9.77E2	1.24E4	1.56E4	1.27E3

Table II: MPI point-to-point communication time in micro seconds on Singularity clusters with different system settings

Size	Unencr. Bare Metal	Unencr. K8s Sing. Calico	Encrypted K8s Sing. Calico	Encrypted K8s Sing. Antrea	CryptMPI K8s Sing Calico
4B	1.53E1	4.66E1	6.14E1	5.97E1	9.57E1
64B	1.77E1	5.08E1	6.48E1	6.31E1	9.85E1
2KB	2.70E1	7.74E1	1.61E2	1.50E2	1.05E2
16KB	4.68E1	8.56E1	4.26E2	3.88E2	1.13E2
256KB	3.00E2	5.45E2	6.22E3	5.47E3	6.81E2
1MB	9.75E2	1.30E3	1.95E4	1.71E4	1.40E3

The overheads of encrypted point-to-point communications with different system settings is plotted in Figure 1. For each system configuration, the overhead is defined as the performance of encrypted communication over unencrypted communication with the same system configuration. For CryptMPI, the overhead when it runs over Docker Swarm is reported. From the figure, it is clear that all container built-in encrypted communication mechanisms have a similar trend. The overheads are relatively small when the message size is small (less than 100%). When the message size is large, the overheads become very large (more than 1000%). In a modern computing system like our experimental system, single thread encryption and decryption throughput

is faster than 25Gbps Ethernet speed in our experiments. Thus, the overheads are more than the computation time for encryption and decryption. We believe that the encryption and decryption computation interferes with the container execution, and introduces other forms of overheads beyond the encryption and decryption calculation such as context switching overheads.

Another interesting result is that Singularity performs slightly worse than Docker overall for both unencrypted and encrypted communication. The same trend is observed in the performance of collective communication and application benchmarks in our experiments. This is a bit surprising, given that Singularity was designed for HPC workloads. We believe this is because the popularity of Docker has resulted in more improvements in the Docker platform than in the Singularity platform, and Docker is no longer at a disadvantage in terms of performance in comparison to Singularity.

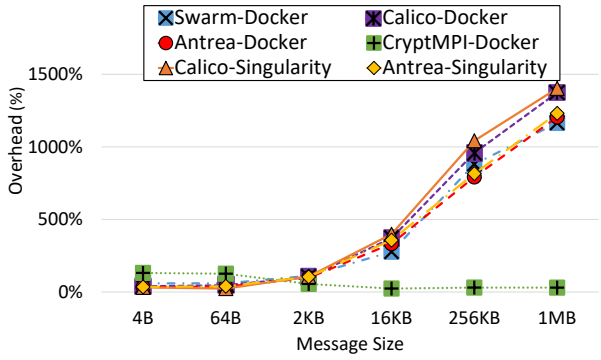

Figure 1: Overheads of encrypted point-to-point communication for different system settings.

C. Collective communication

MPI has many collective operation routines. These collective routines involve multiple parties and the communication is more intensive than point-to-point communication with many concurrent communications. Here, we will present the results for a representative collective operation MPI_Allgather. The general observation is that the container built-in encrypted communication schemes introduce large overheads for the collective operations while CryptMPI performs significantly better. This applies not only to the results shown in the paper, but also to other collective operations such as MPI_Alltoall, MPI_Allreduce, and MPI_scatter.

Table III shows the performance of MPI_Allgather over Docker clusters with different system settings. The experiments are performed on 8 physical nodes and 128 MPI processes ($N = 8$ and $p = 128$). The format of the table is the same as that in Table I. For small message sizes, the overheads for MPI_Allgather is similar to to those for the point-to-point communication. For example, for the 4 bytes message size, the overhead is 125% (4 bytes point-to-point communication has an overhead of 117%). The overheads decrease as the message size increases. For the 1MB message size, the overhead is 63%. In comparison to point-to-point communication (Table I), the overhead is slightly larger for small messages and much larger for large messages. All-gather has many concurrent point-to-point communications. For small message sizes, the system is able to handle many concurrent communications. As such the overheads are roughly the same that those for a single point-to-point communication of small messages. For large messages, the system cannot handle many concurrent communications; and the concurrent communications interfere with one another, resulting in higher overheads in comparison to communicating a single flow. So the results are not surprising.

Table III: Communication times in micro seconds for all-gather on OSU benchmarks with different system settings of Docker (p = 128 and N = 8).

Size	Unencr. Bare Metal	Unencr. Docker Swarm	Encrypted Docker Swarm	Encrypted K8s Docker Antrea	CryptMPI Docker Swarm
4B	1.64E2	3.70E2	1.51E3	1.55E3	1.71E2
64B	1.93E2	3.59E2	2.80E3	2.85E3	2.12E2
2KB	1.45E3	1.67E3	6.98E4	7.00E4	6.04E2
16KB	3.54E3	4.58E3	1.60E4	1.61E4	1.74E3
256KB	3.84E4	6.19E4	2.85E5	3.07E5	2.76E4
1MB	1.11E5	1.82E5	1.04E6	1.06E6	8.64E4

Docker Swarm and Kubernetes Antrea exhibit similar performance. Both have poor performance. Depending on message sizes, encrypted MPI_Allgather with both Docker Swarm and Kubernetes Antrea is between 3.5 to 48.1 times slower than the unencrypted MPI_Allgather on the bare metal system and between 2.5 to 40.2 times slower than the unencrypted MPI_Allgather on Docker Swarm.

CryptMPI's performance rivals the unencrypted MPI over Bare Metal. This can be attributed to CryptMPI's highly optimized encrypted collective communication algorithms for both large and small messages. Note that CryptMPI's point-to-point performance is worse than Docker Swarm and Kubernetes Antrea, and much worse than the unencrypted MPI over bare metal, but the many communications in a collective operation like MPI_Allgather gives CryptMPI ample opportunities to improve performance, mostly through algorithm redesign. Thus, even for small messages where CryptMPI has worse encrypted point-to-point communication performance than Docker Swarm and Kubernetes Antrea, CryptMPI out-performs Docker Swarm and Kubernetes Antrea to a very large degree in the collective operation.

Figure 2 summarizes the encrypted communication overheads for MPI_Allgather with different system settings. This

is with respect to the baseline of the corresponding unencrypted communication time in the container environment. As discussed earlier, for the all-gather operation, encrypted communication introduces high overheads (from around 150% to more than 800%) over the baseline across message sizes. For this operation, among the container built-in encryption schemes, Calico-Docker and Calico-Singularity performs clearly worse than others while all other schemes have similar performance for most message sizes. All container built-in encrypted communication schemes do not perform well for this collective. CryptMPI, with its novel encrypted collective algorithms, achieves much better performance in the container environment and out-performs its baseline (the default MPI library) across all message sizes. This is becuase CryptMPI uses much better encrypted all-gather algorithms than the baseline unencrypted all-gather algorithms in the default MPI library.

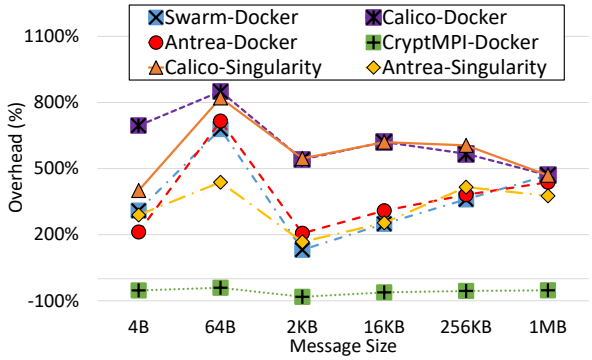

Figure 2: Overheads of encrypted all-gather.

D. Application benchmarks

We conducted experiments using the BT, CG, LU, MG, SP, and FT benchmarks in the NAS parallel benchmarks [8]. For BT, CG, LU, MG, and SP, the Class D size is used. For FT, we use the Class C size (Class D FT caused memory problem in our experiments). Among these benchmarks, BT, CG, LU, MG and SP are dominated by point-to-point communications and FT is dominated by the collective all-to-all communication. We report the results for experiments that were carried out on 8 compute nodes and 64 MPI ranks.

Table IV presents the results on Docker clusters (N = 8 nodes, $p = 64$ ranks). Comparing the second column (unencrypted MPI over bare metal) and the third column (unencrypted MPI over Docker cluster with Docker Swarm), we can see that for BT, CG, LU, MG, and SP the overheads for running in the Docker environment are small, ranging from 1.3% for LU to 5.1% for BT. This confirms the conclusions from other studies [5], [6] that running HPC applications in the container environment may not introduce significant overheads. For FT, whose total time is dominated the all-to-all, the overheads is large (91.7%). As showed earlier, the container execution can introduce large overheads for unencrypted collective communications.

The fourth and fifth columns show the results when encrypted communication is turned on with Docker Swarm and Kubernetes Antrea, respectively. As can be seen from the table, Docker Swarm performs slightly better than Kubernetes Antrea for all but LU. With encrypted communication, both Docker Swarm and Antrea introduce very large overheads for all of the benchmarks. This is surprising since without encryption, the communication time only accounts for a fraction of the total application time for all applications except FT and since running on the container environment does not introduce significant overheads. Even for such applications with significant computation, the overheads for using encrypted communication can be as high as more than 200% (e.g LU on Docker with Kubernetes Antrea). In the experiments, across all application benchmarks, encrypted communication significantly increases total application time for both Docker and Singularity with different container orchestration schemes. This shows how much impact these software-based encrypted communication scheme can have on MPI applications. For FT, the overheads observed is very high, but similar to those in the micro-benchmark results. This is understandable since the FT is dominated by the all-to-all operation. CryptMPI performs much better for these benchmarks than the container built-in schemes although we also observe significant overheads with CryptMPI except for FT. With CryptMPI, the overheads introduced depend on the message size for point-to-point communications: for small messages, the overhead is high; for large messages, the overhead is low. The overall application total time aggregates the time for communicating messages of different sizes in the application, showing the overall performance.

Table IV: Execution times in seconds for NAS benchmarks o n Docker clusters (p = 64 and N = 8)

	Unencr. Bare Metal	Unencr. Docker Swarm	Encrypted Docker Swarm	Encrypted K8s Docker Antrea	CryptMPI Docker Swarm
BT	3.13E2	3.29E2	5.45E2	6.12E2	3.40E2
CG	1.52E2	1.56E2	3.42E2	4.85E2	3.06E2
LU	2.22E2	2.25E2	3.60E2	3.48E2	2.67E2
MG	2.25E1	2.31E1	5.96E1	6.08E1	4.04E1
SP	2.77E2	2.88E2	6.67E2	7.74E2	3.14E2
FT	1.07E1	2.05E1	6.39E1	6.82E1	6.49E0

Figure 3 summarizes the encrypted communication overheads for the NAS parallel benchmarks. CryptMPI consistently out-performs all other schemes. Among the container built-in schemes, there is not one that performs the best in term of overheads. The applications workload affects the overheads. However, we note that overheads in this figure are with respect to the corresponding unencrypted baseline. Because the baseline for different settings performs differently, the overheads may not show the whole picture.

If we look at the absolute application total time, we can see that overall Docker Swarm has the highest performance among the container built-in schemes: it performs the best for all benchmarks except LU.

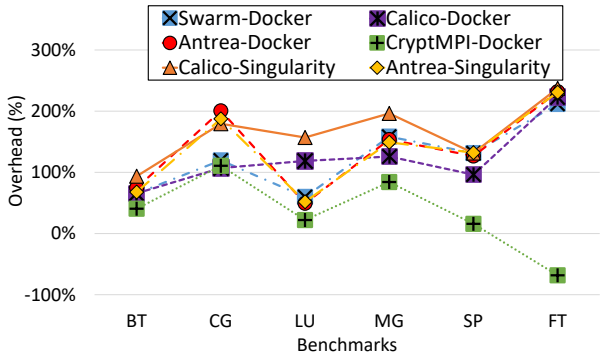

Figure 3: Overheads of encrypted communication for the NAS parallel benchmarks running on systems different settings.

IV. Conclusion

We evaluate the performance of encrypted communication for MPI applications using micro-benchmarks and application benchmarks. Our results indicate that the current container built-in encrypted communication schemes including Docker Swarm, Kubernetes Antrea and Calico, all incur very high overheads on MPI applications. On the other hand, an encrypted MPI library, CryptMPI, achieves much better performance. Given the large performance gap between the system built-in schemes and the library based approach, we believe that it would be beneficial to examine whether it is possible to incorporate some of the techniques used in the library based approach into the systems.

Acknowledgment

We thank Prof. Viet Tung Hoang at Florida State University for extensive discussions on this research. This material is based upon work supported by the National Science Foundation under Grants CICI-1738912, CRI-1822737, and SHF-2007827. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. This work used the XSEDE Bridges resource at the Pittsburgh Supercomputing Center (PSC) through allocations ECS190004 and CCR200042. This work also used the Bridges-2 resource at Pittsburgh Supercomputing Center (PSC) through allocation CIS230062 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259,#2138286, #2138307, #2137603, and #2138296.

References

[1] Mellanox innova ipsec adapter card documentation. https://network.nvidia.com/related-docs/prod_adapter_cards/PB_Innova_IPsec4_Lx_EN_Card.pdf, accessed Apr 1, 2022.

[2] Dirk Merkel et al. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 239(2):2, 2014.

[3] Antrea. https://antrea.io/, accessed Aug 9, 2022.

[4] Calico. https://projectcalico.docs.tigera.io/, accessed Aug 9, 2022.

[5] VÃŇctor Sande Veiga, Manuel Simon, Abdulrahman Azab, Carlos Fernandez, Giuseppa Muscianisi, Giuseppe Fiameni, and Simone Marocchi. Evaluation and benchmarking of singularity mpi containers on eu research e-infrastructure. In *2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*, pages 1–10, 2019.

[6] Tomasz Kononowicz and Pawel Czarnul. Performance assessment of using docker for selected mpi applications in a parallel environment based on commodity hardware. *Applied Sciences*, 12(16), 2022.

[7] DK Panda. OSU micro-benchmark suite, 2011.

[8] David Bailey, Tim Harris, William Saphir, Rob Van Der Wijngaart, Alex Woo, and Maurice Yarrow. The nas parallel benchmarks 2.0. Technical report, Technical Report NAS-95-020, NASA Ames Research Center, 1995.

[9] Abu Naser, Mohsen Gavahi, Cong Wu, Viet Tung Hoang, Zhi Wang, and Xin Yuan. An empirical study of cryptographic libraries for mpi communications. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–11, 2019.

[10] Abu Naser, Cong Wu, Mehran Sadeghi Lahijani, Mohsen Gavahi, Viet Tung Hoang, Zhi Wang, and Xin Yuan. Cryptmpi: A fast encrypted mpi library. *arXiv preprint arXiv:2010.06471*, 2020.

[11] Singularity Container Platform. https://sylabs.io/guides/3.5/userguide/introduction.html, accessed Mar 31, 2022.

[12] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE cloud computing*, 1(3):81–84, 2014.

[13] Naganand Doraswamy and Dan Harkins. *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall Professional, 2003.

[14] Ubaid Abbasi, El Houssine Bourhim, Mouhamad Dieye, and Halima Elbiaze. A performance comparison of container networking alternatives. *IEEE Network*, 33(4):178–185, 2019.

[15] Mehran Sadeghi Lahijani, Abu Naser, Cong Wu, Mohsen Gavahi, Viet Tung Hoang, Zhi Wang, and Xin Yuan. Efficient algorithms for encrypted all-gather operation. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 372–381, 2021.

[16] Mohsen Gavahi, Abu Naser, Cong Wu, Mehran Sadeghi Lahijani, Zhi Wang, and Xin Yuan. Encrypted all-reduce on multi-core clusters. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–7, 2021.

[17] SingularityCRI. https://docs.sylabs.io/guides/cri/1.0/user-guide/index.html, accessed May 9, 2023.

[18] BoringSSL. https://boringssl.googlesource.com/boringssl, 2018.