www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 7 Issue 7 July 2018, Page No. 24106-24108
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v7i7.01

Using CRIU with HPC Containers: Field Experience

Alexey Vasyukov[1], Katerina Beklemysheva[2]

[1]Moscow Institute of Physics and Technology,
Institutsky lane 9, Dolgoprudny, 141701, Russia
[2] Moscow Institute of Physics and Technology,
Institutsky lane 9, Dolgoprudny, 141701, Russia

Abstract:
The paper provides an overview of current authors' field experience of using CRIU for checkpoints and restores of computing containers with HPC applications. Docker and Singularity containers are considered. Reasonable results are obtained for locally running applications. Problems associated with providing the same features for distributed MPI applications are discussed, and possible solutions are outlined with the regard to current state of development of CRIU, Docker, Singularity, Open MPI and competing technologies.

Keywords: criu, containers, docker, singularity, mpi, cloud computing**.**

1. Problem statement

When using computational scientific applications on high-performance clusters, one of the actual problems is that it is often difficult to predict in advance how much resources will require a specific calculation and, accordingly, how long it will take. At the same time, on large clusters for each specific calculation task the execution time is known to be limited. If the calculation does not fit within the allotted time, it is forcibly stopped by the cluster resource scheduler. Obviously, in the case of such a situation, it is highly desirable in the next allocated time quantum not to start the calculation anew, but to continue it from the same logical moment at which it was stopped. Despite all the importance of this problem, an ideal solution for it does not exist yet; the research is ongoing in this area. This work contributes to the subject area, describing a potential system based on CRIU technology [1], and also pointing out open problem questions and possible solutions.

2. Related work

In principle, two approaches to this problem are possible: checkpoint at the system level or at the level of the application.

Obviously, if the calculation state is checkpointed at the application level, all possible problems and corner cases can be worked around in the application code - at the relevant times the application saves the consistent state of the calculation data and stores it on the disk, and if necessary can restore its state in memory from this dump. This approach is quite time consuming, since it requires the implementation of all logic at the application code level. Nevertheless, computational applications for which this feature is crucial use this approach, as it is reliable and versatile.

Implementing checkpoints at the system level seems to be a potentially more correct approach; it allows implementing universal mechanisms, integrating checkpoint and restore operations into cluster resource schedulers. In this area, the most known are two solutions - DMTCP [2] and BLCR [3].

To date, CRIU technology looks like a promising new approach to this problem. The essential characteristics of CRIU, which make it a very interesting one:

- It's the only solution to date that does not require any special preparation of the runtime environment. CRIU runs on a regular Linux kernel, does not require rebuilding applications, does not require a special procedure for launching

applications to checkpoint them later.

- The solution was originally intended to be used in conjunction with compute containers, such as Docker and (potentially) Singularity. Using containers as computational units becomes a common trend, but legacy systems are designed without taking them into account.

3. Basic CRIU workflow for computational applications

Simple cases are covered by CRIU out of the box. We used a system based on Fedora 26 for testing and High Performance Linpack (HPL) [4] as a standard test computational application. Test procedure is as simple as follows.

Start HPL: /usr/lib64/mpich/bin/xhpl_mpich > /tpm/hpl.log

Find PID of HPL process: ps aux | grep [x]hpl

Create checkpoint for the PID: criu dump --shell-job -t 9256

Review system processes, ensure that HPL is not running. After that restore HPL process from checkpoint: criu restore --shell-job

Review HPL log and ensure that the process was restored and the computation completed successfully:

```
----------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=
0.0025926 ..... PASSED
================================================

Finished      1 tests with the following results:
              1 tests completed and passed residual
checks,
              0 tests completed and failed residual
checks,
              0 tests skipped because of illegal input
values.
----------------------------------------------------------------
End of Tests.
================================================
```

A similar procedure for the application in the Docker container is performed using the CRIU documentation [5] and does not cause any problems. For Singularity, the mechanism of container integration with CRIU has not yet been worked out completely [6], but Singularity developers are positive about the prospects of this technology, so we can expect that in the next versions of Singularity there will be a regular support for CRIU.

4. Using CRIU with distributed applications

This area is much more problematic. To work with distributed scientific applications, the principal thing is the ability to checkpoint and restores applications that use MPI for the interaction of processes on different computing nodes. It has been repeatedly noted that the use of CRIU in conjunction with MPI is quite a complex task, which has not been solved to the end today [7], [8].

Saving the state of the application working with MPI consistently requires a lot of steps and cannot be performed completely by third-party tools. Modern high-speed networks used for MPI on clusters offload a significant part of network exchange operations logic to the level of network equipment (network cards and switches). In this regard, any technology that considers MPI process as a black box cannot correctly save its state.

The only reasonable way out of this situation is the close integration of CR technology (CRIU in this case) with the messaging environment itself. This is the way BLCR and DMTCP have gone and implemented integration with Open MPI.

. In this case, the processing logic for the distributed application is as follows:

- The messaging environment stops the network activity and waits for the normal closing of all connections. After the completion of this procedure, the application is in a condition suitable for creating a checkpoint.
- The messaging environment calls the CRIU plugin to save the state of the application, since only the messaging environment can reliably determine the point at which the application is already usable for checkpointing.
- When the distributed application is restored, its state in memory is restored by the means of CRIU.

- After the application is restored, the network context is recreated and network operation continues.

This logic is quite difficult to implement. Work in this direction was started [9], but today it is far from over.

5. Conclusion

To date, the use of CRIU allows transparent checkpoint and restore for computing applications running in Docker containers. It can be expected that in the near future the same option will be supported for Singularity containers. For distributed applications based on MPI, a similar technology is still under active development. These applications can be checkpointed and restored now using DMTCP or BLCR, which require special preparation of the environment and do not support computational containers for the moment.

Acknowledgments

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

References

[1] CRIU Project [Online]. Available https://criu.org/ [Accessed: June. 20, 2018].

[2] DMTCP Project [Online]. Available http://dmtcp.sourceforge.net/ [Accessed: June. 20, 2018].

[3] BLCR Project [Online]. Available https://upc-bugs.lbl.gov/blcr/doc/html/ [Accessed: June. 20, 2018].

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

[5] CRIU documentation on Docker integration [Online]. Available https://criu.org/Docker [Accessed: June. 20, 2018].

[6] Singularity and CRIU integration discussion [Online]. Available https://github.com/singularityware/singularity/issues/468 [Accessed: June. 20, 2018].

[7] A. Reber and P. Vaterlein, "Checkpoint/Restore in User-Space with Open MPI," In Proceedings of SInCom, 2014.

[8] M. Rodríguez-Pascual, J.A. Moríñigo, R. Mayo-García, "Checkpoint/restart in Slurm: current status and new developments," In Proceedings of SLUG, 2016.

[9] Initial CRIU support in Open MPI [Online]. Available https://github.com/open-mpi/ompi/tree/master/opal/mca/crs/criu [Accessed: June. 20, 2018].

Author Profile

Alexey Vasyukov received the B.S. and M.S. degrees in Applied Mathematics from MIPT in 2007 and 2009, respectively. Received Ph.D. in 2012, the dissertation was devoted to numerical modeling of complex wave patterns.

Katerina Beklemysheva received the B.S. and M.S. degrees in Applied Mathematics from MIPT in 2008 and 2010, respectively. Received Ph.D. in 2013, the dissertation was devoted to mathematical modeling of complex contact conditions.