Improving scalability and reliability of MPI-agnostic transparent checkpointing for production workloads at NERSC

Prashant Singh Chouhan
Northeastern University
Boston, USA
chouhan.p@northeastern.edu

Harsh Khetawat
North Carolina State University
Raleigh, USA
hkhetaw@ncsu.edu

Neil Resnik
Northeastern University
Boston, USA
resnik.n@northeastern.edu

Twinkle Jain
Northeastern University
Boston, USA
jain.t@northeastern.edu

Rohan Garg
Nutanix, Inc.
Seattle, USA
rohan.garg@nutanix.com

Gene Cooperman
Northeastern University
Boston, USA
gene@ccs.neu.edu

Rebecca Hartman–Baker
Lawrence Berkeley Nat. Lab.
Berkeley, USA
rjhartmanbaker@lbl.gov

Zhengji Zhao
Lawrence Berkeley Nat. Lab.
Berkeley, USA
zzhao@lbl.gov

***Abstract*—Checkpoint/restart (C/R) provides fault-tolerant computing capability, enables long running applications, and provides scheduling flexibility for computing centers to support diverse workloads with different priority. It is therefore vital to get transparent C/R capability working at NERSC. MANA [1], a transparent checkpointing tool, has been selected due to its MPI-agnostic and network-agnostic approach. However, originally written as a proof-of-concept code, MANA was not ready to use with NERSC's diverse production workloads, which are dominated by MPI and hybrid MPI+OpenMP applications. In this talk, we present ongoing work at NERSC to enable MANA for NERSC's production workloads, including fixing bugs that were exposed by the top applications at NERSC, adding new features to address system changes, evaluating C/R overhead at scale, etc. The lessons learned from making MANA production-ready for HPC applications will be useful for C/R tool developers, supercomputing centers and HPC end users alike.**

***Index Terms*—transparent checkpointing, MANA, DMTCP, split-process, production workloads, supercomputing**

Transparent checkpointing for HPC has been discussed and developed at least since the 1990s [2], [3]. Yet, it is not in common use at the level of supercomputing. This work describes an effort to make transparent checkpointing available for HPC production workloads: first for users at NERSC [4], and then as a demonstration of best practices for other supercomputing sites to benefit from.

The work is based on the use of MANA for MPI [1], which in turn is based on the DMTCP package for transparent checkpointing [5]. MANA employs a new split-process model, first described in 2019. While that initial prototype demonstrates the practicality and advantages of the split-process approach, it has not yet been made production-ready for supercomputing.

In order to place MANA in context, the evolution of transparent checkpointing toward support for supercomputing is briefly summarized in the figure below. ("Checkpointing" or "checkpoint-restart" will often be abbreviated to "C/R" in the rest of this article.)

A Short History of Checkpointing for MPI
Early: Application-specific checkpointing is widely used
1990s: Single-computer checkpointing [2], [3]
2005: BLCR: Berkeley Laboratory C/R [6]
2009: Interconnect Agnostic C/R for Open MPI [7]
2014: Transparent C/R for InfiniBand (below MPI) [8]
2016: Petascale-level transparent checkpointing [9]
2019: MANA: MPI-Agnostic, Network-Agnostic [1]
2020: Production-ready MANA (this work)

The "$M \times N$" problem is the problem of supporting M possible variants of MPI and N possible variants of network. Supercomputing centers generally procure new systems every 3 to 5 years, which may use completely different MPI or network architectures. Therefore any transparent checkpointing solution must solve the "$M \times N$" problem to ensure portability from one generation of machine to the next. The MANA architecture was particularly promising in this regard.

MANA solves the "$M \times N$" problem through a *split-process* approach. The memory regions of the MPI application are tagged as *upper-half* regions, and the MPI, network and other system libraries are tagged as *lower-half*. Only the upper half is checkpointed. On restart, a trivial MPI application is created, thus instantiating the lower half. Each MPI rank of this trivial application then restores the upper-half memory regions of that same rank. For details on DMTCP [5] and MANA [1], see the relevant citations.

This work is an effort to improve the scalability and reliability of MANA. Although the prototype version was functional, it was nowhere close to supporting diverse production workloads running at all scales. This work has largely increased the scalability, reliability and usability of the software, and

has also added new features that are required for the code to function after system upgrades. It can now reliably C/R a couple of top applications at NERSC. Further, the performance overhead of the software was evaluated over various file systems available on Cori [10], to prepare for MANA deployment at large scale.

NERSC Production Workloads

NERSC is the primary HPC center for US DOE Office of Science, supporting more than 8,000 users and 900 projects from all scientific fields. Tens of thousands of different application binaries, which are from MPI or MPI+OpenMP codes dominantly, run at NERSC each year; jobs run at all scales — from single node to full machine.

Checkpoint/restart provides fault-tolerant computing capability, enables long running applications, and provides scheduling flexibility to support diverse workloads with different priority levels, e.g., making space for high-priority, real-time workloads by preempting low-priority jobs. It is therefore vital to get transparent C/R capability working at NERSC.

While it is a daunting task to transparently checkpoint and restart a huge number of applications run at NERSC at the system level, this task can be broken into small, incremental steps, prioritizing top applications. As shown in Figure 1 the top 20 applications account for about 70% of NERSC's Cori [10] computing cycles. If we can get MANA to work reliably with these top applications, then potentially about 70% of the system resources can be preempted to support the high priority, real-time workloads.

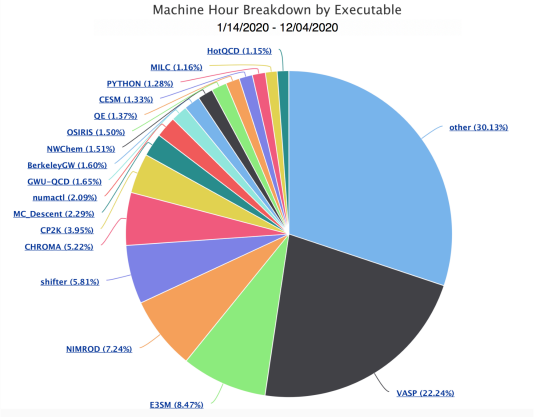

Fig. 1. Application Usage at NERSC in 2020

To get MANA to work with NERSC's production workloads at all scales, we needed to address a few challenges. How production-ready and scalable is this approach, and how much overhead does it create? Under this high-level concept, there are minute details that need to be taken care of while implementing MANA in any checkpoint software.

Software Design Issues at Small Scale

When implementing MANA for NERSC workloads, we observed issues of varying levels of severity at different scales. We began debugging at small scales, where we faced primarily bugs related to synchronization, memory-tagging, and other implementation issues for split processes. To resolve these issues, we instrumented the code to add rank-to-node and process-id mapping for debugging.

Some of the bugs were related to TCP sockets, network delays, missing locks, descriptor conflicts, and lost messages. The DMTCP coordinator connects to each rank via a TCP connection. Network congestion on the production machine at times caused packet losses and disconnects. The TCP KeepAlive option was added to solve this problem. Network delays due to quiescence of the Cray GNI network reconfiguring itself brought additional bugs to the surface. And a few race conditions were seen when the data structures were left in an inconsistent state due to missing locks. The descriptor conflicts would occur upon restart: the upper half opens a file descriptor before checkpoint, and upon restart the lower half opens the same file descriptor number for its internal use. During restart, the lower half then restores the upper half application, creating a file descriptor conflict. We resolved this contention by tagging and reserving file descriptors for each half — upper and lower. And to ensure that no in-transit MPI messages are lost due to checkpointing, we delayed the final checkpoint until the count of total bytes sent and received was equal.

Further, the original MANA assumed that addresses of certain system memory regions were fixed. When the operating system on Cori was upgraded, these assumptions were no longer true, resulting in some memory-region overlaps. To resolve this issue, we used the MMAP_FIXED_NOREPLACE option with mmap to dynamically determine free memory space each time the application executes.

Software Design Issues at Large Scale

At large scale, we started to see new memory corruption, network failure, argument length limit, and disk space errors. We also began to see startup time performance issues with our dynamically linked MANA/DMTCP executables, as static linking is preferred at scale.

We found that the MPI library (in the lower half) can create new memory regions for message exchange at runtime. These lower-half regions may overlap with upper-half regions, eventually leading to memory corruption. MANA converts blocking MPI calls (e.g., MPI_Send) to non-blocking MPI calls (e.g., MPI_Isend); without sufficient care, this subtle difference in calls can change the semantics of an application. The Slurm `srun` command uses a network packet containing the list of arguments it was passed, to send commands to its worker processes. Due to the limit on packet sizes, `srun` was unable to pass all checkpoint file names to its workers, leading to a crash. We resolved this by changing the way we provide the file names. Applications with a large memory footprint may fail to checkpoint if there is insufficient storage space for the checkpoint image; a system warning is needed.

For best startup performance at scale, it is recommended to broadcast a statically linked executable to all nodes. DMTCP currently does not support static linking, but we plan to use

the `--wrap=symbol` flag of the Linux linker to interpose on important functions needed by MANA.

Checkpoint Overhead Evaluations

As part of ongoing work, we evaluated MANA's checkpoint overhead using multiple applications on different file systems. Figure 2 shows the checkpointing time for Gromacs [11] for runs ranging from four ranks to 64 ranks with eight OpenMP threads per task using the ADH benchmark [12]. The aggregate memory use is shown in blue and average time for checkpoint is shown in purple and green, for Burst Buffers and the Lustre file system (CSCRATCH), respectively. Preliminary results show that performance on the Burst Buffers is superior to that on the CSCRATCH and also scales better.

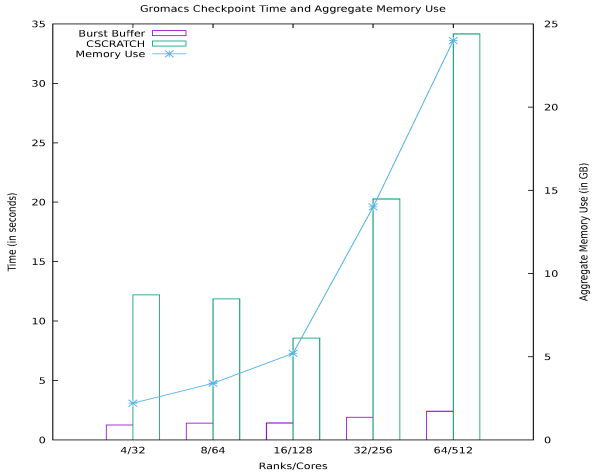

Fig. 2. Checkpoint time of Gromacs with MANA on Burst Buffers and the Lustre file system CSCRATCH on Cori

HPCG [13] displays behavior similar to Gromacs, with checkpoint time for Burst Buffers at 30 seconds and CSCRATCH at over 600 seconds for 512 ranks with eight OpenMP threads per task. The aggregate memory used was 5.8 TB. The speedup for Burst Buffers over CSCRATCH on restart was more modest at about 2.5 times whereas the speedup for checkpointing was more than 20 times.

Lessons Learned

Research codes are usually written quickly to demonstrate a proof of principle. While they show promising features, there is a gap to close before they can be used reliably in production. While debugging MANA and fixing bugs, we had to redesign a few parts of MANA. We learned that instrumenting research codes as follows would have greatly accelerated the conversion to production codes:

1) Greater emphasis on runtime annotations. For example, an annotated table of all memory regions, along with dynamic runtime checks, would help catch bugs early in the development phase.
2) Clearer semantic specifications for each code unit. This will enable better unit testing.
3) Improved design for atomic data structures even for single-threaded code. Each data structure should include a field "CHANGES_PENDING", which would act as a lock.
4) Better attention to warnings and error messages from the beginning. This would help diagnose issues quickly.

Current status of MANA adoption in production workloads

Currently we have enabled MANA with NERSC's top application, VASP [14], which is a widely used materials science code and represents more than 20% of computing cycles at NERSC (Figure 1). VASP is written in Fortran 90 and parallelized with MPI (version 5) or MPI+OpenMP (version 6). VASP jobs usually run with a smaller number of nodes for a long time. While some features implemented in VASP, such as atomic relaxations, have internal C/R support, some other features, such as Random Phase Approximation (RPA), have no such support. The RPA jobs can run for much longer than 48 hours, the max walltime allowed on Cori. In the past we had to make special reservations for these jobs, now they can run on Cori by checkpointing/restarting with MANA. MANA has been tested with various representative VASP workloads and is ready for production deployment.

We have also enabled MANA for Gromacs, a widely used molecular dynamics simulation code, for which we had to fix multiple bugs exposed only by production workloads at scale. Gromacs is written in C++ and parallelized with MPI+OpenMP. While Gromacs has internal C/R support, a benefit of MANA is that a Gromacs computation can be checkpointed at any point in its execution and resumed to generate exactly the same results as an uninterrupted run.

Currently we are in the process of enabling MANA on the rest of the top applications at NERSC using the real use cases provided by NERSC users.

Future Work

The future work includes: getting MANA to work with more applications, reducing the checkpoint overhead for large-scale applications, deploying a preempt queue for real-time workloads, and enabling MANA for NERSC's next pre-exascale computer, Perlmutter [15], an NVIDIA GPU system.

ACKNOWLEDGEMENT

The authors would like to thank Steve Leak and Chris Samuel at NERSC for valuable discussions and help. We would also like to thank the reviewers for their valuable comments and feedback. This work was supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231. This work was partially supported by National Science Foundation Grant OAC-1740218 and a grant from Intel Corporation.

REFERENCES

[1] R. Garg, G. Price, and G. Cooperman, "MANA for MPI: MPI-agnostic network-agnostic transparent checkpointing," in *Proc. of the 28th Int. Symp. on High-Performance Parallel and Distributed Computing*, pp. 49–60, ACM, 2019.

[2] M. Litzkow, T. Tannenbaum, J. Basney, and M. Livny, "Checkpoint and migration of UNIX processes in the Condor distributed processing system," technical report 1346, University of Wisconsin, Madison, Wisconsin, April 1997.

[3] A. Barak and A. Shiloh, "The MOSIX cluster operating system for distributed computing on Linux clusters, multi-clusters and clouds: A white paper," tech. rep., Citeseer, 2013.

[4] "NERSC, the primary scientific computing facility for the Office of Science in the U.S. Department of Energy." https://nersc.gov/.

[5] J. Ansel, K. Arya, and G. Cooperman, "DMTCP: Transparent checkpointing for cluster computations and the desktop," in *2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS'09)*, (Rome, Italy), pp. 1–12, IEEE, 2009.

[6] P. H. Hargrove and J. C. Duell, "Berkeley Lab Checkpoint/Restart (BLCR) for Linux clusters," *Journal of Physics: Conference Series*, vol. 46, no. 1, p. 494, 2006.

[7] J. Hursey, T. I. Mattox, and A. Lumsdaine, "Interconnect agnostic checkpoint/restart in Open MPI," in *Proc. of 18th ACM Int. Symp. on High Performance Distributed Computing*, pp. 49–58, 2009.

[8] J. Cao, G. Kerr, K. Arya, and G. Cooperman, "Transparent checkpoint-restart over InfiniBand," in *ACM Symposium on High Performance Parallel and and Distributed Computing (HPDC'14)*, ACM Press, 2014.

[9] J. Cao, K. Arya, R. Garg, S. Matott, D. K. Panda, H. Subramoni, J. Vienne, and G. Cooperman, "System-level scalable checkpoint-restart for petascale computing," in *22nd IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS'16)*, pp. 932–941, IEEE Press, 2016.

[10] "Cori, a Cray XC40 system at NERSC." https://docs.nersc.gov/systems/cori/.

[11] "Gromacs." http://www.gromacs.org/.

[12] "Gromacs ADH benchmark." ftp://ftp.gromacs.org/pub/benchmarks/ADH_bench_systems.tar.gz.

[13] "HPCG." https://www.hpcg-benchmark.org/.

[14] "VASP." https://www.vasp.at/.

[15] "Perlmutter, a Cray Cascade system at NERSC." https://www.nersc.gov/systems/perlmutter/.