Checkpoint/Restart Vision and Strategies for NERSC's Production Workloads

Zhengji Zhao and Rebecca Hartman–Baker
NERSC User Engagement Group
{zzhao,rjhartmanbaker}@lbl.gov

***Abstract*—As a primary approach to fault-tolerant computing, Checkpoint/Restart (C/R) improves scientific productivity for users, provides scheduling flexibility for computing centers, and protects against system failures. While both application-specific (or application-level) and transparent C/R are used in practice, we are interested in transparent checkpointing, which is vital for system-level checkpointing. Developing and maintaining transparent C/R tools for HPC applications, however, is labor intensive and highly complex due to ever-changing HPC systems and diverse production workloads. Existing C/R tools are often research-oriented, so there is a gap to close before they can be used reliably with production workloads, especially on cutting-edge HPC systems. In this position paper, we present our journey to prepare a production-ready MPI-Agnostic Network-Agnostic (MANA) transparent checkpointing tool for NERSC, and share our vision and strategies to bring transparent C/R capabilities to NERSC's production workloads on current and future systems.**

***Index Terms*—checkpoint/restart, MANA, DMTCP, HPC, production workloads, preemption**

I. Introduction

As a primary approach to fault-tolerant computing, Checkpoint/Restart (C/R) is essential to a wide range of the HPC community. C/R enables long-running jobs for users, and improves their scientific productivity by minimizing compute-time loss due to system failures and improving queue turnaround by utilizing the backfill opportunities with shorter jobs. C/R makes it possible for users to migrate a computation from one system to another, and also enables replay debugging, which is crucial for efficient debugging, especially at large scales, by checkpointing a hard-to-reproduce and/or expensive computation. From computing centers' perspective, C/R provides some protection against system failures (including, for example, PG&E's Public Safety Power Shutoffs), makes it easier to schedule large jobs, increases system utilization, and enables scheduling flexibility to maintain the systems and support diverse workloads with different priorities. For example, the DOE SC's experimental and observational facilities often require real-time access to computing resources to generate immediate feedback for follow-up experiments. These kinds of real-time workloads will continue to increase in the near future at NERSC, meaning we may soon need the capability to preempt partial- or full-system jobs when these demanding real-time jobs enter our systems. It is therefore vital to get C/R capability working at NERSC.

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.

Both application-specific and transparent checkpointing can be used in practice. Many applications at NERSC have some level of internal C/R support, which can be efficient. However, due to the lack of standard entry points for checkpoint and restart operations, application-level checkpointing usually requires user intervention to restart the application, and is difficult to use for system-level checkpointing. How is a batch system supposed to know how to restart a specific application? There are also restrictions for when applications can checkpoint, resulting in limited and inflexible C/R capabilities in most applications. For example, the application may need to complete an iteration or reach the end of a phase in the program before checkpointing can take place. In addition, application-specific C/R imposes additional code development and maintenance burdens on application developers, who would prefer to focus their effort on science and often cannot afford the additional cost of C/R code development, further limiting C/R capabilities available in applications.

In contrast, transparent checkpointing does not require code changes in applications, and is capable of stopping and resuming at any point of execution. The batch system can simply restart the pre-terminated jobs from the most recent successful checkpoint files, requiring no application-specific information. These capabilities are requirements for system-level checkpointing. Therefore, we focus our efforts on transparent checkpointing[1].

Making a transparent-to-users C/R tool capable of supporting HPC production workloads is challenging. It is labor intensive and highly complex primarily due to the ever-changing HPC system landscape and diverse, evolving production workloads. Many checkpointing packages developed in the past, including the most influential Berkeley Lab Checkpoint/Restart (BLCR) [1], are no longer maintained. As pointed out by Cooperman [2], among the 19 checkpointing packages from before 2010, which were listed in the website, https://checkpointing.org/, only two are still active. (One of the two is DMTCP [3] which we have selected to work with.) As a result, for many years no viable C/R tools were available for production use on cutting-edge HPC systems. Even now, existing C/R tools are often research-oriented and meant to demonstrate promising C/R capabilities on specific platforms. There is thus a large gap to close before these tools

[1]In some cases, though, it may be necessary to combine transparent and application-specific checkpointing to achieve optimal C/R performance. See IV-B.

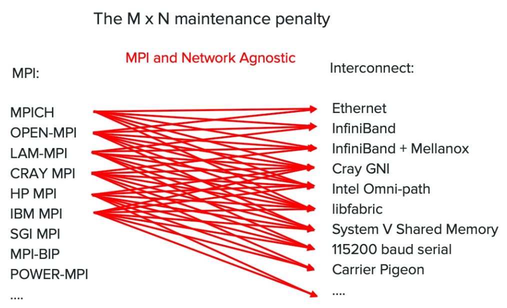

Fig. 1. The $M \times N$ maintenance penalty of C/R tools. The figure was modified from Rohan Garg's HPDC'19 talk slides [4].

can reliably provide C/R capabilities across a wide range of different applications and cutting-edge HPC systems.

We begin by analyzing the many challenges in C/R code maintenance. First, for HPC workloads, dominated by MPI and hybrid MPI+OpenMP applications, C/R tools that work across many combinations of MPI implementations and networks are exceedingly difficult to maintain. This is the so-called M (# of MPI implementations) $\times N$ (# of Networks) maintenance penalty [5] (see Figure 1). Since front-edge computing centers introduce new HPC systems every few years, the already numerous MPI implementation/network combinations also constantly change. Additionally, fast-emerging process and memory architectures, which are often heterogeneous and even dynamically configurable, impose significant research and development efforts to enable transparent checkpointing on them. The programming models and languages used to program for new hardware are continuously evolving as well. This fast turnover is extremely difficult for code developers to keep pace with, meaning there are no production-ready transparent C/R tools on cutting-edge HPC systems.

Second, C/R tools that require long-term coordination between MPI library, kernel, and resource manager/scheduler developers have proven unsustainable; this was the primary challenge for the once promising and influential BLCR [1]. Third, transparent C/R tools often interact directly with OS kernels, and frequent OS updates on HPC systems can easily break C/R tools. Fourth, C/R tools incur runtime overheads and impose extra work upon users, which hinders user uptake of the C/R approach. Finally, maintaining C/R codes for production use has low priority among C/R researchers, as it generally does not produce novel research results that can secure funding from stakeholders.

Much of our work so far has focused on addressing these challenges. In Section II, we present the strategies that we have been using to bring the transparent C/R capabilities to NERSC's production workloads. In Section III, we present what we have learned so far, and in Section IV we present our long-term C/R vision and strategies. We conclude the paper with our future work.

II. Our Strategies So Far

NERSC, like other computing centers at DOE laboratories, has a unique role in bringing C/R research to production usage, because our funding is dependent upon the successful operation of an HPC center, not production of novel research results. With this understanding and our own vision for the future, we started our long journey to enable transparent C/R capabilities for NERSC production workloads in mid-2017. We have been using the following three strategies to address the challenges we described above.

A. Selecting DMTCP/MANA as our C/R tool and promoting its development

First, we selected Distributed Multi-Threaded CheckPointing (DMTCP) [3] as our C/R tool. DMTCP lives completely in user space and therefore does not require coordination with MPI, kernel, and batch-system developers, or system administrators. With DMTCP, we can also bring C/R capabilities to NERSC incrementally, which is critical to the project's long-term success.

A breakthrough in the DMTCP implementation in May 2019, called the MPI-Agnostic Network-Agnostic transparent checkpointing (MANA) [5], [6], addressed the critical $M \times N$ maintenance issue. MANA is implemented as a plugin for DMTCP. Based on a novel "split-process" approach, MANA is fully transparent to the underlying MPI, network, and libc library and underlying Linux kernel. In a split process, a single system process contains two programs in its memory address space: the lower half, containing an MPI proxy application with MPI library, libc and other system libraries; and the upper half, containing the original MPI application and data (see Figure 2). MANA tracks which memory regions belong to the upper and lower halves, and achieves MPI agnosticism by checkpointing only the upper-half memory and discarding the lower-half memory at checkpoint, and then reinitializing the MPI library upon restart. MANA achieves network agnosticism by draining MPI messages before checkpointing. To ensure that no checkpointing occurs when some ranks participate in a collective call, MANA prefaces all collective MPI calls with a trivial barrier (See Figure 3). The real collective call happens only when all the ranks enter or exit the trivial barrier. During the real collective calls, checkpointing is disabled, assuring that no messages floating in the network when checkpointing is initiated. MANA was a substantial step forward toward ready-to-use C/R tools on future HPC platforms!

Our collaboration with the DMTCP team began in March 2018, with the initial goal of enabling DMTCP to work reliably with serial/threaded applications running on a single node. As of this writing, in collaboration with the DMTCP/MANA team and NERSC's summer interns we have made MANA work with various workloads of the application that uses the most cycles at NERSC, the VASP [7] materials science code. There are still a few outstanding issues to resolve for VASP, (e.g., high runtime overhead), but the improvements to the

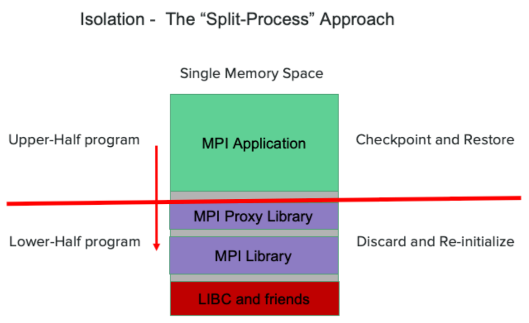

Fig. 2. MANA is based on the "split-process" approach, in which a system process contains two separate programs: the upper half, containing the original MPI application and data; and the lower half, containing MPI library, libc and other system libraries. MPI Agnosticism is achieved by checkpointing only the upper half memory and discarding the lower half, and then reinitializing MPI upon restart. The figure was modified from Rohan Garg's HPDC'19 talk slides [4].

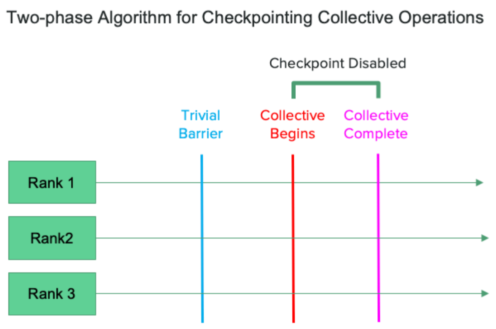

Fig. 3. MANA's two-phase algorithm for checkpointing MPI collective calls. MANA achieves its network agnosticism by draining the network before checkpointing. The figure was modified from Rohan Garg's HPDC'19 talk slides [4].

MANA code from our collaboration was a substantial advancement toward converting MANA to a production-quality tool.

Because MANA can be used reliably with VASP, we have been able to examine what transparent checkpointing can do for VASP. VASP does not scale to a large number of processors due to the nature of its algorithms, so a typical VASP job is run for a long time on a handful of nodes. Nonetheless, VASP usage makes up more than 20% of the computing cycles at NERSC [9]. Like many other applications run at NERSC, VASP has internal C/R support for some of its workloads, such as atomic relaxation and molecular dynamic (MD) simulation jobs. It does not, however, have the same support for other workloads, such as Random Phase Approximation (RPA) jobs, which often have runtimes exceeding 48 hours, the maximum walltime allowed on Cori [8], a Cray XC40 system at NERSC. We had to make special reservations to support these long-running jobs in the past. While enabling long-running jobs is not the only use case of MANA, with MANA we no longer need to make special reservations to support long-running VASP jobs. Even for the atomic relaxation and MD jobs that have internal C/R support, there are multiple advantages to using MANA:

1) MANA can do periodic as well as on-demand checkpointing, and can stop and resume at any point of execution, while the internal checkpointing is available only upon the successful completion of a preset number of iterations. If the job hits the walltime limit before completing the specified number of iterations, VASP will fail to generate checkpoint files.
2) With MANA all output files after multiple checkpoint and restart will be exactly identical to an identical job that runs without C/R. Because MANA tracks the length of each open file at the time of checkpoint, and then at the time of restart, it truncates the file back to the length that it had at the time of checkpoint. With VASP's internal C/R support, users must provide modified input files for each restarted job, and have to back up the output files from each pre-terminated job, otherwise they will be overwritten by the next restarted job. This means the batch system cannot restart a pre-terminated VASP job without user intervention. When the job completes after multiple restarts from the internal support, users have to combine multiple output files when the intermediate results are important.
3) One may argue that managing inputs/outputs to use the internal C/R support could be automated. While it can be done with additional user effort, the process is vulnerable to any random errors that happen on the system at a certain rate. In contrast, transparent checkpointing is resilient to random errors and can always restart from the most recent successful checkpoint files.
4) VASP supports on-demand checkpointing for atomic relaxation or MD jobs, so users can send a checkpoint request to a running job if it cannot complete the preset number of iterations before hitting the walltime limit. Upon receiving the request, VASP completes the current ionic step and writes out the checkpoint file and quits. The next job can start from the checkpoint file. However, it is hard to estimate how long it needs to complete the current ionic step: It could require anywhere from a few seconds to hours depending on where it is in the execution, making it difficult for VASP users to adopt the variable-time job scripts (see next section for more detail) provided by NERSC to automate job resubmissions. In contrast, MANA has a relatively predictable checkpoint overhead, depending on the memory usage and file systems used, making it easy to automate job resubmissions.
5) VASP atomic relaxation and MD jobs can restart with the last updated atomic configuration file when the checkpoint file (WAVECAR) is not present; the result, however, may not be identical to an uninterrupted run (especially for MD jobs). With MANA, bitwise reproducibility can be guaranteed between a job that completes after multiple restarts and an uninterrupted job with identical initial input. MANA transparently saves all states, including random seeds. This makes it an ideal tool to restart chaotic MD simulations, for which

trajectories diverge rapidly with even slight changes in restart data.
6) VASP is a widely used, well maintained commercial code, and has tens of thousands users over the world, but internal C/R support in VASP is limited, indicating that adding internal C/R support in VASP is not trivial and/or may require significant development effort.

The downside of running VASP with transparent checkpointing tool is the additional runtime overhead, putting aside the C/R overhead that would be incurred in both application-level and transparent checkpointing. MANA developers have devised algorithms and protocols to minimize the runtime overhead (to be implemented); however, ultimately the question remains: how much overhead can we tolerate in exchange for the benefits of C/R?

B. Promoting C/R uptake among NERSC users

Second, in parallel with the MANA code development work, we developed variable-time job scripts to automate preempted job submissions, rolled out queue policies and configurations to incentivize C/R usage, and provided user training to increase the uptake of C/R. Variable-time job scripts [10], [11] can automatically split a long-running job into multiple shorter ones that self-resubmit until the job completes. This was a key step towards promoting user uptake of the C/R approach, making it easier for users to work with preempted jobs with or without external C/R tools. The "flex" QOS [12] was configured with a substantial charging discount for variable-time jobs to incentivize C/R usage. This queue offers a 75% charging discount in exchange for users' flexibility about job walltime. A flex job must request a minimum walltime of at most two hours, and can use up to 256 Cori KNL nodes for up to 48 hours. The jobs in the flex queue create additional opportunities for the Slurm scheduler to perform backfill and increase machine utilization. In 2020, the "flex" queue was used by over 150 distinct users, consuming nearly 3% of all Cori KNL cycles, enabling NERSC to achieve more than 90% utilization on Cori. More than half of these "flex" users ran their jobs with variable-time job scripts. We hosted multiple user training sessions [13]–[19] to help NERSC users adopt variable-time job scripts and the C/R approach in their production workloads with or without DMTCP/MANA. We have also provided extended user documentation on C/R [20], and have started providing one-on-one NERSC appointment service [21] on checkpointing for users as of June, 2021.

C. Building an active and strong C/R community

Our third strategy was to build an active and strong C/R community to promote production-ready C/R tools development. An active community will ensure the long-term sustainability of the C/R approaches. We hosted the First International Symposium on Checkpointing for Supercomputing [22] in February, 2021, which attracted more than 250 participants, and we will be hosting our second International Symposium on Checkpointing for Supercomputing [23] in November, 2021, in conjunction with SC21.

III. Lessons learned

Despite our extensive efforts promoting MANA for users, adoption is lower than we had hoped. The bottleneck was the delivery of production-quality MANA.

A. Substantial gap to close for production-ready MANA

Enabling transparent checkpointing for production workloads has proven extremely difficult and slow. There is a substantial gap between proof-of-concept MANA code and a production-quality tool. Many issues that did not appear for the demonstration in the MANA paper must now be addressed explicitly. First, MANA lacked support for hugepages memory or static linking, but on Cori the hugepages memory is enabled by default and static linking is needed for applications that run at large scales. Adding support for hugepages and static linking was therefore necessary for MANA to be used with production workloads. Second, MANA works by wrapping MPI APIs to ensure consistent states across MPI tasks when checkpointing, and also to achieve its MPI and network agnosticism, but MANA had only a small set of commonly used MPI APIs wrapped for demonstration. Given the large number of applications run at NERSC, many missing MPI wrapper functions need to be implemented in MANA, which often requires non-trivial effort (e.g., the one-sided MPI APIs). Finally, MANA is implemented as a plugin to DMTCP, adding additional options to the DMTCP commands. This made the commands that launch and restart target applications under MANA quite long, therefore user-friendly interface scripts were added to make it easy to use.

Numerous bugs have been exposed and fixed [24] when testing MANA with VASP production workloads, and some of the bugs show up only after multiple checkpoints/restarts. These bugs ranged from local logical errors in the implementation, to algorithmic design flaws that required redesigning and improving, as well as the lack of proper treatment for some libraries that interact with target applications, e.g., the XPMEM and PMI libraries, as well as some special variables to support Fortran bindings in the Cray MPICH library. Some of the memory corruption and rare race condition bugs were very evasive, requiring substantial effort for developers to fix them.

One such example is a memory corruption bug (glibc error message: double free or corruption) that was exposed with one of the VASP workloads. The developers identified that the corruption occurred when VASP was deallocating a buffer at restart, but it was difficult to debug due to the special structure of MANA and the additional complications added by the differences between MPI's Fortran and C interfaces. First, the developers tried using existing memory debugging tools, including Clang's address sanitizer, electric fence, and glibc's mcheck function. However, these debugging tools ended up not being helpful. Clang's address sanitizer and glibc's mcheck function provided too many false positives during restart and prevented VASP from continuing to run, because the memory-restoring process of MANA confused these memory debugging tools. Electric fence was able to work with MANA but dramatically slowed down the program and changed C/R schedules so that the bug was not reproducible. Then, the developers implemented a small memory debugging tool, libmalloccheck. This library can be loaded with LD_PRELOAD and interpose the malloc and free functions to add more debugging information to each allocation and deallocation. The tool showed that no overrun or underrun in the corrupted memory area was found, implicating glibc's internal data structure for allocation as the cause of the corruption. The developers studied the source code of the malloc and free functions in glibc, and found a broken memory arena allocated inside glibc, which led to a memory inconsistency problem when MANA was restoring anonymous hugepages segments at restart. After resolving the hugepages issue, the memory corruption error was finally fixed.

B. *High runtime overheads*

In addition, the MPI and network agnosticism of MANA came at the cost of substantial runtime overhead, which requires further research to resolve. Here the runtime overhead refers to the overhead of running an application under MANA without checkpointing as compared to running the native application. While a negligible runtime overhead was observed with some applications (e.g., HPCG [25], and miniFE [26]), we have seen up to 24% runtime overhead of MANA on Cori KNL (17% on Haswell) when running with VASP, which makes frequent MPI collective calls (thousands of times per second per rank). This high runtime overhead has been attributed to frequent, high-cost context switches between the upper and lower halves. MANA's split-process approach utilizes two different programs in the same memory address space, and the upper (user application) and lower (MPI library) half programs each has their own thread-local storage region, whose variables are referred to using the "FS" register of the x86-64 CPUs. MANA interposes all MPI calls from the user application from the upper half program to call the MPI functions in the lower half. When switching between the upper and lower half programs, the value of the "FS" register must be changed to point to the correct thread-local region, which requires a kernel call to invoke a privileged assembly instruction. Since a kernel call is relatively expensive, when frequently invoked it could result in excessive overhead. A more efficient mechanism for modifying the "FS" register does exist in more recent Linux kernel 5.9, which uses unprivileged assembly instruction (called FSGSBASE instruction set, which allows read/write FS/GSBASE from any privileges). If the FSGSBASE kernel patch [27] could be applied to Cori, which is running Linux kernel 4.12, the high runtime overhead of MANA would be largely reduced, as demonstrated in [5], [28]. However, applying an OS kernel patch is difficult on production systems, and it is very likely that we will not get this patch on Cori before it retires in a couple of years. Fortunately, the MANA developers have made progress on reducing this runtime overhead without the kernel patch, but it required significant effort from them, and the fix is still under development. There are a few other sources (albeit smaller) of runtime overhead, such as the insertion of a trivial barrier (MPI_Barrier) to all MPI collective calls to allow checkpointing to occur only at consistent state; the virtualization of MPI communicators and datatypes that requires a table lookup to map a virtual ID to its real ID; and so on. MANA developers have ideas for algorithms to address all these sources of overhead, such as using the wrapped MPI functions only when checkpointing actually occurs, and implementing a more efficient hash-table lookup algorithm, but all these need time and resources to implement.

C. *Chasing hardware*

MANA has achieved MPI and network agnosticism, which is significant progress towards ready-to-use C/R tools, however, it still needs to deal with new architectures separately. So far we have been working on getting MANA to work on Cori, our Cray XC40 system with Intel Hawell and KNL processors. While MANA's functionality and reliability enhancements for Cori will be useful for our next flagship system, Perlmutter [29], an HPE Cray EX system with both NVIDIA A100 GPUs accelerated nodes and AMD Milan CPU nodes, it will require significant effort to enable MANA on the Perlmutter GPU partition, as MANA does not yet support GPUs. There is a plan to integrate some recent work on transparent checkpointing of CUDA, CRAC [30], into MANA to support Perlmutter GPUs; however, it will require significant time and resources to implement the code integration. It should be noted that in addition to NVIDIA GPUs, a variety of other accelerators, such as AMD GPUs, Intel GPUs, FPGAs, as well as specialized accelerators, are all on the HPC landscape and horizon. It is possible that multiple types of specialized accelerators may be present on a single node on future systems to continue to scale performance beyond Moore's law [31].

Like other HPC centers, NERSC introduces a new cutting-edge system every few years. It is expected that future systems will be more heterogeneous and complex. Can C/R tools ever catch up with emerging architectures and technologies?

IV. Vision and Strategies on Checkpoint/Restart for HPC Production Workloads

To have ready-to-use transparent C/R tools for HPC production workloads on current and future systems, significant efforts are required in both the short term as well as the long term. While anyone in the HPC community can contribute to this goal, computing centers like NERSC have unique and important roles to play to achieve this goal, realizing all the benefits that transparent C/R has to offer, and bringing profound changes to how future systems are designed, operated, and used.

A. *Working toward architecture-agnostic C/R*

For the longer term, instead of checkpointing tools chasing hardware, "integrating checkpointing as first class citizens in future systems" [32] is essential. Currently, C/R tools are "forced" to take a software-only approach to checkpoint hardware, because hardware comes without any support for checkpoint and restart operations. If this situation continues, C/R tools will never be able to catch up with the fast changing pace of emerging hardware, especially now as hardware is evolving to be more heterogeneous and complex. If each new piece of hardware were to come with a set of standard APIs that C/R tools could use to query/manipulate the hardware states and manage memory segments on them, this would be important progress toward ready-to-use C/R tools on the future cutting-edge systems. These APIs could tremendously shorten C/R tools development time to checkpoint new hardware (or even enable ready-to-use C/R straight out of the box), and we expect that it would not require a major effort for the hardware vendors, who know their hardware best, to provide them. What we request may simply be a small subset of existing functionalities or tools developed during hardware design and testing phases. Computing centers like NERSC must influence/collaborate with hardware and software vendors to provide C/R capability on future systems via procurement or other avenues, and should contribute to the standardization of the APIs that hardware vendors must provide to support C/R operations. It should be noted that hardware vendors will not provide C/R components on their hardware if their clients, like computing centers, never ask for them. We believe that our efforts will eventually help create a commercial drive among hardware and software vendors, achieving architecture agnostic and ready-to-use C/R tools on future systems.

B. *Handshaking with application-level checkpointing*

So far we have focused on transparent checkpointing, which requires no information from applications. While this approach works well most of the time, it can sometimes result in inefficiencies that could be easily avoided if applications could provide minimal information for external C/R tools. Checkpointing SPAdes workloads, one of the most important production workloads of The Joint Genome Institute at Lawrence Berkeley National Laboratory, using DMTCP is such an example [33]. One of the key characteristics of the SPAdes application is that it creates/uses temporary files of up to a terabyte in size, known as precious files, and later in the execution it deletes them. These files, which may or may not be opened by the processes at all times, co-exist with other output files in the same directory, and SPAdes needs these precious files to properly restart the application. However, DMTCP tracks only the open files at checkpoint time and assumes that all the files associated with a process will be persistent and present at their original paths. To solve this problem, the authors exploited a plugin in DMTCP designed specifically for SPAdes that backs up precious files at the checkpoint and replaces them at restart. Since DMTCP has no feedback from SPAdes about which output file is a precious file and which one is not, it ends up naively saving all the output files at checkpoint time and replacing them at restart time. This increases the size of the files to back up over time, reaching up to a few terabytes, and raises I/O time significantly. This performance penalty could be avoided if SPAdes were somehow "checkpoint-aware" and did not delete these precious files during the course of execution if run under DMTCP. Alternatively, SPAdes could help DMTCP identify the precious files among regular output files so it could save only the precious files instead of all output files. This could be done simply by prefixing/suffixing to identify precious files, or writing all precious files in a specific directory. What DMTCP needs from SPAdes is minimal and non-invasive, and prevents extremely inefficient checkpointing.

Like hardware vendors providing the APIs for their hardware for C/R tools, application developers could provide "some" information about their applications for C/R tools to work properly and efficiently. Computing centers should contribute to creating this minimum requirement for applications, and influence the many application developers with whom they interact regularly to adopt this practice, which would require only minimal effort from application developers. Additionally computing centers should also influence application developers to develop more MPI standard-compliant codes, to minimize unexpected issues when using C/R tools.

C. *Funding support for development/maintenance of C/R tools for production use*

MANA's MPI and network agnosticism is essential for ready-to-use C/R tools on future HPC systems. However, because MANA originated as a research code, there is a significant gap to close before it becomes a production-ready tool. This is a common issue with C/R research codes. Relying primarily on PhD students/research institutions, whose main focus is on novel research, to deliver production-ready code has proven ineffective. We believe that a contractor or commercial software company is the best party to deliver a production-ready C/R tool and to provide on-going support for production use. Computing centers should partner with each other and work with relevant funding agencies to create funding opportunities or provide funding support for C/R tools maintenance and enhancement efforts, and also seek collaboration with the C/R researchers/tools developers (e.g., DMTCP/MANA team) and contractors/software companies to create production-ready tools that meet the C/R needs for HPC production workloads.

In the longer term, we propose creating a consortium for C/R efforts, modeled after the HPSS collaboration[2], to develop a C/R standard, support development of C/R code, and provide direction and strategy. The consortium would be composed of member HPC centers, HPC hardware and software vendors, C/R researchers and code developers, and other interested parties. Each partner would provide developer time and/or representatives on executive and technical steering committees. We believe that no single organization has the experience and resources to tackle all the challenges emerging from fast-changing HPC technologies and production workloads, so a broad collaboration across the HPC community is essential to continue to meet the C/R needs of the HPC community. DOE

[2]HPSS Collaboration: https://www.hpss-collaboration.org/collaboration.shtml

computing centers should take the initiative, and collaborate with commercial entities who are willing to pioneer the C/R market in HPC to build the base for the C/R consortium.

D. Reducing C/R overhead

Optimizing I/O for large checkpoint images at all scales is essential for C/R tools to be used in production. In addition to deploying faster file systems such as local storage or burst buffers to store the checkpoint images, utilizing external libraries e.g., [34], in transparent C/R tools themselves is a direction to pursue to reduce C/R read/write overhead. We should also explore hardware/hardware-assisted solutions where applicable, e.g., persistent memory devices [35]–[37], which can be effective in reducing C/R overhead.

E. Changing common HPC system operation practices

C/R tools often interact directly with operating system kernels. It has been difficult to apply OS kernel patches on production systems. Moving forward, computing centers should work with system vendors to establish a standard procedure to make it possible to apply needed critical OS kernel patches to support C/R tools and more.

F. Building a strong C/R community

Building a strong and active C/R community is critical for meeting our C/R needs on current and future systems. C/R researchers and tools developers, practitioners, application developers, and end users as well as hardware and software vendors can all contribute to the development of C/R research and success in production use, motivating the development of usable C/R tools, and harnessing the full benefits of C/R.

G. Critical approach for long-term success

Finally, we would like to stress that it is critical to take an **incremental approach** when working towards bring transparent C/R capabilities to HPC production workloads with ultimate goal of system-level checkpointing.

V. Future Work

So far much of our effort has focused on getting MANA to work with our top application, VASP. In the near term, we will continue to collaborate with the MANA developers to reduce the high runtime overhead of MANA for VASP, which invokes MPI collective calls excessively. When the runtime overhead issue of VASP from using MANA is resolved, we will be able to pre-empt the 20% of Cori node-hours used by VASP jobs to run on-demand real-time workloads. Perhaps this is the most exciting application of MANA from NERSC's perspective. We will also continue enabling MANA on the remaining top applications on Cori (Note that top 20 applications at NERSC account for more than 80% of machine usage!) in an incremental manner and will implement a preemptable queue to create a buffer of nodes deployable for real-time workloads. Of special interest for future work is the goal of optimizing I/O for large checkpoint images by exploring external libraries in MANA, e.g., [34]. We will work on enabling C/R on Perlmutter by integrating CRAC [30], recently developed for transparently checkpointing CUDA, into MANA.

Meanwhile, we will continue to follow the vision and strategies outlined in Section IV to bring transparent C/R capabilities to HPC production workloads on current and future systems.

Acknowledgments

The authors would like to thank Gene Cooperman and his DMTCP/MANA team (Rohan Garg, Twinkle Jain, *et al.*) at Northeastern University, who have been in close collaboration with the authors on bringing transparent C/R capabilities to NERSC production workloads. They would like to thank Gene Cooperman, Eric Roman, Nick Wright, Stephen Leak, Rollin Thomas, and David Skinner at NERSC for providing valuable comments and suggestions on this position paper, and Yao Xu, a PhD student at Northeastern University and a NERSC summer intern, for providing the technical specifics of the memory corruption error debugging. They would also like to thank NERSC interns who have contributed to this project: Yao Xu and Prashant Singh Chouhan from Northeastern University, Harsh Khetawat from North Carolina State University, and Tiffany Connors from Texas State University, and NERSC staff Christopher Samuel and Stephen Leak.

References

[1] Berkeley Lab Checkpoint Restart for Linux (BLCR). [Online]. Available: https://crd.lbl.gov/departments/computer-science/class/research/past-projects/BLCR/.
[2] G. Cooperman, SuperCheck21 keynote. [Online]. Available: https://drive.google.com/file/d/10c8KWAK0cjROyNZ3OEB2N8CyDu6g_1mg/view
[3] DMTCP. [Online]. Available: http://dmtcp.sourceforge.net/.
[4] R. Garg, G. Price, and G. Cooperman, "MANA for MPI: MPI-Agnostic Network-Agnostic Transparent Checkpointing". [Online]. Available: http://dmtcp.sourceforge.net/papers/hpdc19-slides.pdf
[5] R. Garg, G. Price, and G. Cooperman, "MANA for MPI: MPI-Agnostic Network-Agnostic Transparent Checkpointing", Proc. of 28th Int. Symp. on High-Performance Parallel and Distributed Computing (HPDC'19), June 2019, Pages 49-60, https://doi.org/10.1145/3307681.3325962.
[6] R. Garg, G. Price, P. Chouhan, Yao, and G. Cooperman, *et al*, MANA code. [Online]. Available: https://github.com/mpickpt/mana.
[7] VASP. [Online]. Available: https://www.vasp.at/.
[8] Cori, a Cray XC40 system at NERSC. [Online]. Available: https://docs.nersc.gov/systems/cori/.
[9] B. Driscoll, and Z. Zhao, "Automation of NERSC Application Usage Report", Proc. of Seventh Annual Workshop on HPC User Support Tools (HUST 2020), November 11, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9308071
[10] T. Connors, R. Hartman–Baker, Z. Zhao, and S. Leak, Variable-time job scripts. [Online]. Available: https://github.com/NERSC/variable-time-job
[11] Variable-time job scripts. [Online]. Available: https://docs.nersc.gov/jobs/examples/#variable-time-jobs.
[12] The flex QOS for KNL, created in April 2019 [Online]. Available: https://docs.nersc.gov/jobs/policy/#flex
[13] User Training on Checkpointing and Restarting VASP Jobs Using MANA on May 25, 2021. [Online]. Available: https://www.nersc.gov/users/training/events.
[14] User training on MANA, a transparent checkpointing tool, on May 7, 2021. [Online]. Available: https://www.nersc.gov/users/training/events.
[15] Hands-on user training on variable-time job scripts for VASP users, June 2020. [Online]. Available: https://www.nersc.gov/users/training/events.
[16] Hands-on user training on variable-time job scripts, May 2020. [Online]. Available: https://www.nersc.gov/users/training/events.

[17] User training on DMTCP for users running serial and threaded applications, November 2019. [Online]. Available: https://www.nersc.gov/users/training/events.

[18] Hands-on user training on variable-time job scripts for VASP users, June 2019. [Online]. Available: https://www.nersc.gov/users/training/events.

[19] Z. Zhao, "Variable-time Jobs", a presentation in the Monthly NERSC User Group (NUG) Webinars, May 2018. [Online]. Available: https://www.nersc.gov/users/NUG/teleconferences/nug-webinar-may-17-2018/.

[20] User documentation on Checkpoint/Restart. [Online]. Available: https://docs.nersc.gov/development/checkpoint-restart/

[21] NERSC weekly email, June 7, 2021. [Online]. Available: https://rest.nersc.gov/REST/announcements/message_text.php?id=4466

[22] First International Symposium on Checkpointing for Supercomputing (SuperCheck21). [Online]. Available: https://supercheck.lbl.gov/archive/supercheck21

[23] Second International Symposium on Checkpointing for Supercomputing (SuperCheck-SC21). [Online]. Available: https://supercheck.lbl.gov

[24] Y. Xu, and Gene Cooperman. [Online] Available: https://drive.google.com/file/d/1qPl68oGZmtLMXkiGZa4871cpUpzakwNu/view?usp=sharing.

[25] HPCG. [Online]. Available: https://www.hpcg-benchmark.org/.

[26] M. Heroux and S. Hammond, "MiniFE: Finite Element Solver". [Online]. Available: https://tinyurl.com/y7hslf65. [Online; accessed Jan 2019].

[27] FSGSBASE kernel patch. [Online]. Available: https://lwn.net/Articles/769355/

[28] Y. Xu, Z. Zhao, and G. Cooperman, "Reducing the Runtime Overhead of Communication-intensive MPI Applications in the MANA Checkpoint Framework", submitted to Euro21.

[29] Perlmutter, an HPE Cray EX system at NERSC. [Online]. Available: https://docs.nersc.gov/systems/perlmutter/.

[30] T. Jain and G. Cooperman, "CRAC: Checkpoint-Restart Architecture for CUDA with Streams and UVM", Proc. of Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC'20), (and earlier technical report at https://arxiv.org/abs/2008.10596)

[31] M. Schulz, D. Kranzlmüller, L. B. Schulz, C. Trinitis, and J. Weidendorfer, "On the Inevitability of Integrated HPC Systems and How they will Change HPC System Operations", International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART 2021). [Online] Available: https://www.isc.lrz.de/fileadmin/ISC/Download/heart-hpc.pdf

[32] M. Schulz, Panel discussion in SuperCheck21. https://drive.google.com/file/d/1N5HSZFwl6paJEBE05DGTL_R5QOIrkmEJ/view

[33] T. Jain, and J. Wang, "Checkpointing SPAdes for Metagenome Assembly: Transparency versus Performance in Production", Proceeding of the First International Symposium on Checkpointing for Supercomputing (SuperCheck21), February 4-5, 2021. [Online]. Available: https://arxiv.org/pdf/2103.03311.pdf

[34] B. Nicolae , A. Moody, E. Gonsiorowski, K. Mohror, and F. Cappello, "VeloC: Towards High Performance Adaptive Asynchronous Checkpointing at Large Scale", The 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS'19), pp. 911–920, Rio de Janeiro, Brazil, 2019. [Online] Available: https://doi.org/10.1109/IPDPS.2019.00099

[35] HPE technical white paper, "Performance Comparative Analysis of MemVerge Memory Machine with REDIS workloads [Online]. Available: https://memverge.com/wp-content/uploads/2020/10.

[36] "Genomic Sequencing Challenges". [Online]. Available: https://memverge.com/big-memory-solutions-genomics/

[37] "Zero-Impact Crash Recovery for Kx Kdb+". [Online]. Available: https://memverge.com/zero-impact-crash-recovery-for-kx-kdb/