SimHEC: Understanding Application Efficiency at Exascales through Simulations

Da Zhang* **and Ioan Raicu*†**
***Department of Computer Science, Illinois Institute of Technology, Chicago, IL**
†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL
dzhang32@hawk.iit.edu, iraicu@cs.iit.edu

Abstract

It is expected that our HEC system will enter exascale era in decade, which is one thousand times of performance as today's system (petascale). In the mean time, many challenges also have been noticed and pointed out, as the size of HEC system increased without some dispensable improving on architecture of today's HEC system, the systems could collapse at exascale, because the functionality would not be able to complete their duties successfully and even break down the whole system. One potential problem is that the MTTF (mean time to failure) of a HEC system will decrease linearly as the system size increases. We will probably have to face the serious situation that our HEC system might be very unreliable and no works could be done successfully, due to too frequent failures. The situation could be worse for HEC with parallel file system, even armed with checkpointing to guarantee its reliability. In this project, we study and explore application efficiency toward exascale, and show that DFS offers a lot better performance by taking its advantage of the local storage speed. We measure the application efficiency in both parallel and distributed file system, scale them from a small system size to exascale with and without checkpointing, and observe the differences between three different workloads: uniform workload with only one job running at anytime, uniform workload with ten jobs running concurrently at anytime, and Intrepid system's 8 month workload from Argonne National Laboratory. All the experiments were simulated in the Java simulator developed in the DataSys laboratory at Illinois Institute of Technology.

1. INTRODUCTION

Exascale computing [3, 6], i.e. 10^{18} FLOPS, is predicted to emerge by the end of 2018 with current trend. Millions of nodes and billions of concurrent data access are expected with the exascale. This degree of computing capability is similar to that of a human brain and will enable the unraveling of significant scientific mysteries and present new challenges and opportunities. The US President made the building of exascale systems a top national priority, stating that it will "dramatically increasing our ability to understand the world around us through simulation and slashing the time needed to design complex products such as therapeutics, advanced materials, and highly-efficient autos and aircraft" [4].

One of most critical challenges for exascale computing is how to maintain the exascale computer reliable. Failures are unavoidable in high end computing (HEC) systems, making the partially-done work useless if no recovery mechanism exists. The reliability of a system is how strong the system is to prevent failures and/or recover after a failure. With millions of nodes and billions of cores and concurrent requests, keeping the entire exascale system reliable is extremely hard.

In order to bring the system back into the last correct state, checkpointing records system's (correct) states periodically. These states need to be stored on the persistent storage, because otherwise they are gone permanently if the system encounters a failure. Checkpointing is a general mechanism to maintain system's reliability, which is independent of any particular system. The fact of dealing with persistent storage implies potentially huge overhead. Therefore, besides other factors like how frequently to save the states, the question of improving the checkpointing degenerates to the question of how to elevate the storage I/O bandwidth.

The state-of-the-art file system for HEC is the parallel file system (e.g. GPFS [5]) which is deployed on the storage servers that are remotely interconnected to the compute nodes. We believe that future HEC systems should be equipped with a distributed filesystem deployed on non-volatile memory on every compute node; every compute node would actively participate in the metadata and data management, leveraging the abundance of computational power many-core processors will have and the many orders of magnitude higher bisection bandwidth in multi-dimensional torus networks as compared to available cost effective bandwidth into remote parallel filesystems.

To answer the question how the current checkpointing mechanism would work for exascale systems among other HEC systems, we built a model to emulate exascale systems, designed and implemented a simulator SimHEC to study its reliability and efficiency. Results show that, unfortunately, current checkpointing mechanism on parallel filesystems is incapable to effectively recover the system from failures. However, it suggests that a distributed filesystem with local persistent storage, e.g. [7], would offer an excellent scalability and aggregate bandwidth, which in turn enables efficient checkpointing at exascale.

2. MODELING THE HEC SYSTEMS

Application Efficiency is defined as the ratio of application up time over the total running time:

$$E = \frac{up\_time}{running\_time} \times 100\%,$$

where *running_time* is the summation of up time, checkpointing time, lost time and rebooting time. **Up time** is when the job is correctly running on the computer. **Checkpointing time** is when the system stores the correct states on persistent storage periodically. **Lost time** measures the time when a failure occurred, the work since the last checkpointing would be lost and needs to be recalculated. **Rebooting time** is simply the time for the system to reboot the node.

Optimal Checkpointing Interval is the optimal checkpointing interval as modeled in [1]:

$$OPT = \sqrt{2\delta(M+R)} - \delta,$$

where δ is the checkpointing time, M is the system mean-time-to-failure (MTTF) and R is the rebooting time of a job.

Memory Per Node is modeled as the following based on the specifications of IBM BlueGene/P. When the system has fewer than 64K nodes, each node has 2GB memory. For larger systems, the per-node memory is calculated (in GB) as

$$2 \cdot \frac{\#nodes}{64K}.$$

We have two different models of **Storage Bandwidth** for parallel filesystems (PFS) and distributed filesystems (DFS), respectively, since they have completely different architectures. We assume PFS is the state-of-the-art parallel filesystem used in production today, e.g. GPFS [5], whose bandwidth (in GB/sec) is modeled as

$$BW_{PFS} = \frac{\#nodes}{1000}.$$

And for DFS, it is a hypothetical new storage architecture for exascale. There are no real implementations of a DFS that can scale to exascale, but this study should be a good motivator towards investing resources to the realization of DFS at exascale. The bandwidth of DFS in our simulation has the following bandwidth

$$BW_{DFS} = \#nodes \cdot (\log \#nodes)^2.$$

These equations are based on our empirical observations on the IBM BlueGene/P supercomputer.

For rebooting time, DFS has a constant time of 85 seconds because each node is independent to other nodes. For PFS, the rebooting time (in seconds) is calculated as the following:

$$\lceil 0.0254 \cdot \#nodes + 55.296 \rceil,$$

which is also based on the empirical data of the IBM BlueGene/P supercomputer. The above formulae indicate that DFS has a linear scalability of checkpointing bandwidth, whereas PFS only scales sub-linearly. The sub-linearity of PFS checkpoint bandwidth would prevent it from working effectively for exascale systems.

3. PRELIMINARY RESULTS

Experiments can be categorized into three major types. We first compare SimHEC results to existing valid results with the same parameters and workload to verify SimHEC. Then variant workloads are dispatched on SimHEC to study the effectiveness and efficiency of checkpointing at different scales of HEC systems. Lastly, we apply SimHEC on a 8-month log of an IBM BlueGene/P supercomputer, and emulate the checkpointing at exascale. Metrics *Uptime*, *Check*, *Boot* and *Lost* refer to the definitions of **up time**, **checkpointing time**, **rebooting time** and **lost time**, respectively.

We found that local persistent storage would be dramatically helpful to leverage data locality in the context of traditional distributed filesystems. Our study shows that local storage would be one of the key points to succeed in maintaining the reliability for exascale computers. The results are coincident with the findings in [2], where a hybrid of local/global checkpointing mechanism was proposed for the projected exascale system.

REFERENCES

[1] J. Daly. A model for predicting the optimum checkpoint interval for restart dumps. In *ICCS*, pages 3–12, Berlin, Heidelberg, 2003.

[2] X. Dong, Y. Xie, N. Muralimanohar, and N. P. Jouppi. Hybrid checkpointing using emerging nonvolatile memories for future exascale systems. *ACM Trans. Archit. Code Optim.*, 8(2):6:1–6:29, June 2011.

[3] M. A. Heroux. Software challenges for extreme scale computing: Going from petascale to exascale systems. *Int. J. High Perform. Comput. Appl.*, 23(4):437–439, Nov. 2009.

[4] B. Obama. A strategy for american innovation: Driving towards sustainable growth and quality jobs. National Economic Council, 2009.

[5] F. Schmuck and R. Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. FAST '02, Berkeley, CA, USA, 2002. USENIX Association.

[6] J. Torrellas. Architectures for extreme-scale computing. *Computer*, 42(11):28–35, Nov. 2009.

[7] D. Zhao and I. Raicu. Distributed File Systems for Exascale Computing. In *Doctoral Research, Supercomputing '12*, Salt Lake City, UT, 2012.