J. Parallel Distrib. Comput. 100 (2017) 16–29

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Optimizing checkpoint data placement with guaranteed burst buffer endurance in large-scale hierarchical storage systems

Lipeng Wan [a,*], Qing Cao [a], Feiyi Wang [b], Sarp Oral [b]

[a] University of Tennessee, Knoxville, United States
[b] Oak Ridge National Laboratory, United States

HIGHLIGHTS

- A thorough analysis of both failure patterns and runtime characteristics of HPC systems.
- A new checkpoint placement model for optimizing large-scale hierarchical storage systems' usage.
- A novel adaptive algorithm that can dynamically optimize the checkpoint placement.

ARTICLE INFO

Article history:
Received 5 May 2016
Received in revised form
25 August 2016
Accepted 2 October 2016
Available online 14 October 2016

Keywords:
Fault tolerance
Checkpoint
Hierarchical storage systems
Burst buffer
Solid-state drive

ABSTRACT

Non-volatile devices, such as SSDs, will be an integral part of the deepening storage hierarchy on large-scale HPC systems. These devices can be on the compute nodes as part of a distributed burst buffer service or they can be external. Wherever they are located in the hierarchy, one critical design issue is the SSD endurance under the write-heavy workloads, such as the checkpoint I/O for scientific applications. For these environments, it is widely assumed that checkpoint operations can occur once every 60 min and for each checkpoint step as much as half of the system memory can be written out. Unfortunately, for large-scale HPC applications, the burst buffer SSDs can be worn out much more quickly given the extensive amount of data written at every checkpoint step. One possible solution is to control the amount of data written by reducing the checkpoint frequency. However, a direct effect caused by reduced checkpoint frequency is the increased vulnerability window of system failures and therefore potentially wasted computation time, especially for large-scale compute jobs.

In this paper, we propose a new checkpoint placement optimization model which collaboratively utilizes both the burst buffer and the parallel file system to store the checkpoints, with design goals of maximizing computation efficiency while guaranteeing the SSD endurance requirements. Moreover, we present an adaptive algorithm which can dynamically adjust the checkpoint placement based on the system's dynamic runtime characteristics and continuously optimize the burst buffer utilization. The evaluation results show that by using our adaptive checkpoint placement algorithm we can guarantee the burst buffer endurance with at most 5% performance degradation per application and less than 3% for the entire system.

1. Introduction

Large-scale high performance computing (HPC) systems usually support running tens of scientific simulations on hundreds of thousands of compute nodes simultaneously. Due to the scale of both hardware and software components involved, failures are common and a fact of life in large-scale HPC systems' daily operation. Most scalable scientific applications cope with potential failures using some form of defensive programming technique—by periodically exporting their execution state and intermediary results as a "checkpoint" to a persistent storage. In the event of failures, they will be able to continue the execution (restart) without repeating previous computation.

Checkpoints generated by scientific applications are written to the parallel file systems (PFS) which are usually built using traditional storage servers and spinning disk drivers for balanced cost, performance, and capacity. Parallel file systems provide an efficient data access mechanism between various computation resources over high-performance storage area networks. However,

* Corresponding author.
E-mail addresses: lwan1@vols.utk.edu (L. Wan), cao@utk.edu (Q. Cao), fwang2@ornl.gov (F. Wang), oralhs@ornl.gov (S. Oral).

http://dx.doi.org/10.1016/j.jpdc.2016.10.002
0743-7315/

given the frequency of the checkpoints and the amount of data written at each checkpoint step, the total checkpoint size written in an applications runtime can be daunting. Trying to absorb such large-scale checkpoint I/O with traditional parallel file systems can be cost-prohibitive. On the other hand, studies have shown that PFS has been underutilized in the sense that it operates in much lower bandwidth spectrum most of the time which is nowhere near the peak [14]. In order to resolve the dichotomy, the concept of "burst buffer" was recently proposed and has been designed and prototyped in some large-scale HPC systems [14,22,2,39,23]. The basic idea behind the "burst buffer" is that we can build an intermediate hardware and I/O middleware layer between compute nodes and parallel file systems to better handle I/O workloads from scientific applications by utilizing flash or SSD. The checkpoint data from scientific applications will be temporarily written into the burst buffer layer first and then drained to the underlying parallel file systems asynchronously. Since SSDs can provide much higher read and write bandwidth than hard disk drives, with the help of burst buffer layer, the I/O performance of scientific applications will be improved significantly, which also means the checkpointing can be done faster and more CPU time can be saved for computation.

Ideally, the burst buffer was designed to absorb all I/O workloads generated by large-scale applications running on supercomputers. However, in reality we may have to limit the amount of data written to the burst buffer if the endurance requirements on SSD devices are to be considered. Specifically, each block in an SSD must be erased before being rewritten and only a finite number of erasures are possible before the bit error of SSD becomes unacceptably high. Moreover, random writes could generate more erasure operations compared to sequential writes and lead to the increase of the so called write amplification [10]. In this paper, we focus our study on how general HPC I/O workloads, where a normal write amplification factor is often assumed, affect the lifetime of a typical SSD device. However, in some scenarios, different models of SSD devices might have different endurance even under the same I/O workload, or the inherent random access pattern of the I/O workloads might make the endurance issue more severe than the general case studied here. Those special scenarios are out of this paper's scope, but will be studied in our future work. As an example, let us assume designing a burst buffer layer for a hypothetical large-scale HPC platform with tens of thousands of compute clients. If the building blocks are typical 256 GB SSDs and if we are targeting a relatively moderate sized burst buffer layer (e.g. 5 PB aggregate capacity), then the total number of SSDs required is about 20,000. According to the datasheet [21], the newest Samsung 850 Pro SSD (256 GB) has a warranty for maximum 150 TB write. If the burst buffer is designed to serve 5 years, the maximum amount of data that can be written to the entire burst buffer per day is 1600 TB. We further assume that the write amplification factor is around 1.3 [10], then the actual allowed write is about 1200 TB per day. On the other hand, some common large-scale scientific applications often produce huge checkpoint data. For example, if half of the compute nodes in ORNL's Titan [18] supercomputer are used to run the CHIMERA application [30], the size of each checkpoint generated at each checkpoint step is almost 160 TB [29]. This is a common case when scientific applications are run at large-scale, since at each checkpoint step half of the data residing in the system memory can be written to the storage as a checkpoint. If several such scientific applications run simultaneously, the total size of the write workloads per day will be much larger than the SSD endurance requirements. Therefore, without constraints, the intensive write workloads produced by large-scale long-running scientific applications through checkpointing could degrade the endurance of SSD devices and the reliability of the burst buffer significantly.

Several techniques and approaches [40,13,11] have been proposed to optimize the endurance of SSD devices under different I/O workloads, particularly the kinds of workloads produced by personal computers, web servers, database systems, etc. However, none of them tackles on SSD endurance issues in HPC environment, because the HPC I/O workloads usually consist of extremely intensive write operations which can quickly wear out the SSD devices even when cutting-edge endurance optimization techniques are used. In fact, the HPC community does not have a full understanding of how to effectively maintain sustainable cost-to-performance and cost-to-capacity ratios for SSD devices under such write-heavy I/O workloads. One possible solution might be replacing the worn-out SSDs often to maintain a given capacity level, however, this solution is not feasible or cost-effective. The system-exclusive burst buffer can be built either by using node-local SSDs (i.e., an SSD device on every compute node) or can be shared (i.e., a set of pool of SSDs serving all compute nodes in a given HPC system). In the node-local case, the number of SSDs required grows linearly with the number of compute nodes. For the shared case, the required number of SSDs will grow linearly with the total memory size to absorb and flush the output data burst. In either case, we end up with thousands or tens of thousands of SSDs for a large-scale HPC system. To maintain the wear-out levels of this number of SSDs in a large-scale HPC facility will require extensive resources (i.e., man power to monitor and physically replace the worn-out devices on regular basis). Also, this approach will incur additional costs of the replaced devices. As an example, a modest size SSD can easily cost a few hundred US dollars today and the replacing just the half of a 5000 SSD population will amount to a few million US dollars. Moreover, this approach requires compute node downtimes and interruptions to replace the worn-out devices from otherwise a "healthy" node (in terms of remaining components, such as CPU and memory), which is also an additional but hidden cost for the total cost of ownership (TCO) of a large-scale HPC system. For all these reasons combined, solely relying on physically replacing worn-out SSDs to maintain a set of required capacity and endurance targets is not cost-effective or practical.

Besides frequently replacing worn-out SSD devices, another possible solution would be reducing the amount of data written to the burst buffer. In [9], the authors proposed a checkpoint interval optimization model for large-scale scientific applications which takes the constraint of burst buffer capacity into consideration. In such model, SSD-based burst buffers of supercomputers are used to absorb all checkpoint data of the scientific applications. Therefore, in order to satisfy the capacity constraint, the model intends to reduce the checkpointing frequency of some write-heavy jobs so that the amount of data written to the burst buffer can be reduced. However, a direct effect caused by such reduction in checkpointing frequency is that the potential wasted computation time due to system failures also increases significantly, especially for large computation jobs.

In this paper, we propose a new checkpoint placement optimization model which collaboratively utilizes both the burst buffer and parallel file system in a large-scale storage system to store the checkpoint data generated by scientific applications. Specifically, our model guarantees the endurance requirements of the SSD-based burst buffer layers without sacrificing too much of the computational efficiency. Moreover, in order to make the model feasible to real HPC systems, we also design an adaptive algorithm which can dynamically adjust the checkpoint placement based on the changing runtime characteristics of the HPC system and continuously optimize the usage of the burst buffer. The evaluation results demonstrate the effectiveness of our checkpoint placement model and adaptive checkpoint placement algorithm. Particularly, using our adaptive checkpoint placement algorithm