Future Generation Computer Systems 37 (2014) 178–190

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Towards minimizing disk I/O contention: A partitioned file assignment approach

Bin Dong [a,b,*], Xiuqiao Li [a], Limin Xiao [a], Li Ruan [a]

[a] School of Computer Science and Engineering, Beihang University, Beijing 100191, China

[b] Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA

HIGHLIGHTS

- A new file allocation metric is explored and evaluated.
- We design a novel static heuristic partitioned file assignment algorithm, MinCPP and its dynamic version DMinCPP.
- We present the definition and construction procedure of the disk I/O contention probability.
- We prove that the file assignment of the MinCPP owns a minimum disk I/O contention probability among all the file allocations.
- We compare the MinCPP and DMinCPP with the round-robin file assignment strategy which is widely used.

ARTICLE INFO

Article history:
Received 15 April 2012
Received in revised form
28 October 2013
Accepted 3 December 2013
Available online 18 December 2013

Keywords:
Parallel I/O system
Partitioned file
File assignment algorithm
I/O contention probability
Distributed resource management

ABSTRACT

One problem with data-intensive computing facilitating is how to effectively manage massive amounts of data stored in a parallel I/O system. The file assignment method plays a significant role in data management. However, in the context of a parallel I/O system, most existing file assignment approaches share the following two limitations. First, most existing methods are designed for a non-partitioned file, while the file in a parallel I/O system is generally partitioned to provide aggregated bandwidth. Second, the file allocation metric, e.g. service time, of most existing methods is difficult to determine in practice, and also these metrics only reflect the static property of the file. In this paper, a new metric, namely file access density is proposed to capture the dynamic property of file access, i.e. disk contention property. Based on file access density definition, this paper introduces a new static file assignment algorithm named MinCPP and its dynamic version DMinCPP, both of which aim at minimizing the disk contention property. Furthermore MinCPP and DMinCPP take the file partition property into consideration by trying to allocate the partitions belonging to the same file onto different disks. By assuming file request arrival follows the Poisson process, we prove the effectiveness of the proposed schemes both analytically and experimentally. The MinCPP presented in this study can be applied to reorganize the files stored in a large-scale parallel I/O system and the DMinCPP can be integrated into file systems which dynamically allocate files in a batch.

1. Introduction

Managing massive amounts of data is a big challenge in data-intensive computing today [1]. The volumes of data which need to be stored have grown rapidly in the recent past [2] but the rate at which the program can read/write data from/to a single disk is increasing much more slowly [3,4]. The data I/O (input/output) speed is the performance bottleneck of dedicated scientific applications, grid applications and diverse enterprize applications when they read or write files [5,6]. On the other hand, the parallel I/O system [7–11] built from a large number of cheap hard disks can offer a high data transfer rate via partitioning files among multiple hard disks and then permitting the accesses from applications to these data executing concurrently [12–14]. A parallel I/O system is widely deployed in a data center to support grid computing [15], cloud computing [16], high performance computing [17], etc.

To fully deliver the performance of the parallel I/O system, files should be carefully assigned onto the available disks before they are accessed [18,11]. For one thing, the response time of the file requests can be reduced through proper file allocation [19]. The response time of file request is important for the applications, such as the web server and online transaction processing. For another, file assignment has a significant impact on the load balancing [20]. With the parallel I/O system continuously expanding in size, an even load distribution among disks can fully reap the benefits of the I/O system parallelism.

The optimal file allocation problem of the parallel I/O system has been extensively investigated in the previous literature [21,22].

* Corresponding author. Tel.: +1 5107798060.
E-mail addresses: Bdong@cse.buaa.edu.cn, dbin@lbl.gov (B. Dong).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.022

The optimization models can accurately describe the file allocation problem, while it is NP-complete and can be transformed to other resource allocation problems [21]. The heuristic file assignment algorithms, on the other hand, have low computational complexity and therefore are a practical file allocation approach. Well-known heuristic file allocation algorithms include SP [19], HP [19], SOR [23], BAS [20], BASB [20], OSP [24] and OSPOline [24]. Typically, these heuristic file assignment algorithms try to optimize the mean response time of file requests either through balancing the load among disks or via minimizing the service time variance at each disk. However, most of these existing heuristic file assignment algorithms share the following two limitations.

First, these heuristic file assignment algorithms are specially designed for non-partitioned files. However, in most parallel I/O systems based on the Grid File System [25], HDFS [10], Luster [9], Ceph [26], etc., a file is always partitioned into multiple subfiles. Assigning these subfiles onto multiple disks permit the data belonging to one file to be accessed concurrently and therefore to deliver high speed aggregated I/O speed [12,7–9]. Thus, the file partition is one important property that the file assignment strategy should take into consideration.

Second, the file service time used by most existing heuristic algorithms [19,23,20,24] to guide file allocation tend to be hardware dependent and static. Typically, file service time is estimated from seek time, rotation time and transfer time of the hard disk. In a real production environment, file service time, especially for small files, is difficult to be determine because of cache effect, network delay, and hard disk mechanical property [27]. Moreover, file service time tends to be a static property of the file and therefore is not able to reflect the dynamic file access characteristic, e.g. disk I/O contention. Disk I/O contention plays an important role in the performance optimization of a parallel I/O system [28]. The reason is that the disk I/O contention turns the parallel I/O into the sequential ones [29–31] which results in the underutilization of the disk parallelism [31]. Therefore, minimizing the disk I/O contention probability can further improve the performance of a parallel I/O system.

In this paper, we explore a new file allocation metric, namely file access density, which is independent of the underlying hardware configuration and also can reflect the dynamic file access characteristic to some extent. Based on the file access density, a static heuristic file allocation algorithm named *min*imum I/O *c*ontention *p*robability for *p*artitioned file (MinCPP) is introduced. MinCPP can be applied to reorganize the files stored in a parallel I/O system to minimize disk I/O contention. By applying the same idea in a dynamic environment, we introduce a *d*ynamical version of *MinCPP* (DMinCPP). The DMinCPP allocates files in a batch manner and can be integrated into the file system which may dynamically allocate files in a batch. In particular, we make the following contributions:

1. We introduce and evaluate a new and hardware independent file allocation metric, file access density.
2. We design and analyze MinCPP and DMinCPP, a novel static heuristic partitioned file assignment algorithm aiming at minimized disk contention.
3. Based on the access information of the files, we present the definition construction procedure of the disk I/O contention probability. In addition, we prove that the file assignment of the MinCPP owns the minimum disk I/O contention probability among all the file allocations.
4. Using a synthetic workload, MinCPP and DMinCPP are compared with round-robin, the most popular assignment strategy for partitioned files. MinCPP reduces the response time of file requests and the load variance of disks by 12.11% and 28.59%, respectively, and DMinCPP reduces by 6.25% and 16.14%, respectively, compared to round-robin.

The rest of the article is organized as follows. Section 2 surveys the related work and discusses the motivation. The MinCPP algorithm, its proof and a case study of small group of files are presented in Section 3. The DMinCPP is presented and analyzed in Section 4. Section 5 analyzes the complexity of MinCPP and DMinCPP. In Section 6, we evaluate the proposed algorithms with the synthetic workload. Section 7 concludes this article and presents the future work.

2. Related work and motivation

2.1. Disk I/O contention

To store massive amounts of data generated by data intensive applications, building a large-scale parallel I/O system with cheap hard disks is still the first choice in most data centers. Even though the basic mechanical architecture of a hard disk has stayed almost the same for decades [3], the storage density keeps increasing and the price of hard disks keeps decreasing. Meanwhile, new storage devices such as solid-state drive (SSD) [32] are able to provide high bandwidth and low latency, and they have received attention [33,34]. However, because the high performance of SSD comes with very high cost, it is still not affordable for most data centers.

As a hard disk drive can serve only one request at a time [3], minimizing the disk I/O contention can improve the performance of a parallel I/O system [28]. Minimizing the disk I/O contention can fully reap the benefits of the parallelism of multiple disks [29–31]. Moreover, as a single request involves the disk seek time, the rotational time, and the data transfer time [3], minimizing the disk contention can reduce the disk seek and rotational times, thereby optimizing the performance of a parallel I/O system.

The files stored in almost all mainstream parallel I/O systems are always allocated onto the available disks in a round-robin manner [12,7–9]. In such a case, the subfiles belonging to one file tend to be evenly distributed among disks. Such a type of file allocation may be optimal for one file which it is accessed in its entirety, but it may degrade the performance of the whole system as interference, e.g. disk I/O contention, among different files exists. For example, when the load of one disk is low but the load of other disks located in a parallel I/O system is very high, allocating more subfiles of a file onto these disks with high load may result in more disk contention on this disk.

Hence, the disk I/O contention is one of the important factors which the file assignment algorithm should take into consideration. Hence, from the view of disk I/O contention, we aim at designing a new file assignment scheme which can improve the performance of a parallel I/O system.

2.2. File assignment algorithm

The optimal file assignment problems are surveyed and summarized by Lawrence W. Dowdy in 1982 [21]. Based on the objective functions or the performance metrics used in the different models, the author divides the optimal file assignment models into different categories. Although these optimal models can accurately describe the file allocation problem, finding the solution to these models is difficult. Actually, the file allocation problem is an NP-complete problem and can be expressed as another resource allocation problem [21]. In 2007, Akshat Verma [22] employed the divide-and-conquer method to split the object function of the optimal file allocation model into the three sub problems: Rotational Delay problem, Seek Time problem, and Transfer Delay problem. Assuming the transfer time is small, the author states that the algorithm with $n \log(n)$ complexity can resolve the file allocation problem.

The heuristic file allocation algorithm, on the other hand, has low computational complexity and therefore has become an effective and practical file allocation approach. Based on the