Available online at www.sciencedirect.com
SCIENCE DIRECT®
Future Generation Computer Systems 22 (2006) 611–619
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

# Adaptive parallel I/O scheduling algorithm for multiprogrammed systems

J.H. Abawajy

*Deakin University, School of Information Technology, Geelong, Vic. 3217, Australia*

Available online 17 November 2005

**Abstract**

As the rate at which disk drives read and write data is improving at a much slower pace than the speed of processors, I/O has risen to become the bottleneck in high-performance computing for many applications. A possible approach to address this problem is to schedule parallel I/O operations explicitly. To this end, we propose two new I/O scheduling algorithms and evaluate the relative performance of the proposed policies against two baseline policies. Simulation results show that the proposed policies outperform the baseline policies.


*Keywords:* I/O scheduling algorithm; Cluster computing; Parallel I/O; Performance analysis

## 1. Introduction

The widespread adoption of cluster computing as a high performance computing platform has seen the growth of data intensive applications such as digital libraries and image repositories. In addition to running these applications on multiple processors, the parallelization of I/O operations and the use of multiple disk drives are required for achieving high system performance. Parallel input/output techniques can help solve this problem by creating multiple data paths between memory and disks, that is, exploiting parallelism in the I/O system. To this end, several parallel file systems (e.g., Parallel Input/OUtput System (PIOUS) [1] and Parallel Virtual File System (PVFS) [2]) based on user-level libraries have been developed to distribute data over several nodes and providing low-level data access mechanisms. Although these approaches have helped reduce the I/O bottlenecks, it is shown that peak performance is rarely attained from these coordinated storage devices [3,4]. As a result, the performance of carefully tuned parallel programs can slow down dramatically when they read or write files in such systems [5].

The gap between the I/O subsystem and processors is expected to increase in future since I/O performance is limited by physical motion. Therefore, it is imperative to find techniques that can improve the access performance of storage devices despite their high latency. Significant improvements in access latency relative to processor and network speeds are unlikely mainly due to physical limitations of I/O devices [6]. As a result,

0167-739X/$ – see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.future.2005.09.008

various methodologies over the past few years have been devised to address the I/O bottleneck problem. Most attention has focused on improving the performance of I/O devices using fairly low-level parallelism in techniques such as disk stripping [7], disk-directed I/O [8], data distribution strategies and data layout strategies [9,10], compiler support [11] and I/O runtime libraries [12]. Prefetching and caching [13,14], are another area of research considered for optimizing I/O performance as well. Several optimizations for reducing data transfer time for parallel I/O have been proposed in the past few years. This class of research are focusing on the *data access strategies* such as *collective I/O* [15] and the two-phase I/O optimization [16], which reduces disk access time by splitting an I/O operation into two phases: inter-processor data exchange through the network, and bulk accesses to the disks. The placement of I/O servers to improve parallel I/O performance on switch-based clusters [17] is another parallel I/O research focus. An important dimension and the one we are interested in is concerned with the effective management of parallel I/O by using appropriate I/O scheduling strategies.

In this paper, we focus on scheduling parallel I/O operations to minimize the completion time of parallel applications on cluster computing environments. We present a set of new Parallel I/O scheduling algorithms for multiprogrammed cluster computing environments running parallel I/O workloads. Parallel I/O scheduling is concerned with scheduling of parallel I/O operations with the goal of minimizing the overall I/O response times. The motivation for studying this problem is that parallel I/O has recently drawn increasing attention as a promising approach to alleviating I/O bottlenecks in cluster computing [2]. As noted in [18], there is a lack of research that investigates the effectiveness of parallel I/O scheduling strategies for multiprogrammed cluster computing environments. In the multi-application environment, the parallel I/O system is a shared resource which usually impacts the I/O performance delivered to the simultaneously running applications. In addition, we believe that scheduling parallel I/O operations will become increasingly attractive and can potentially provide substantial performance benefits. We present simulation results showing that the proposed I/O scheduling algorithms can produce a substantial improvement over previous I/O scheduling algorithms.

The rest of the paper is organized as follows. Section 2 is an overview of the system of interest and related work. Section 3, discusses the proposed Parallel I/O scheduling algorithms. The performance analysis of the proposed scheduling policies is discussed in Section 5. The results and discussions of the experiments are presented in Section 6. The conclusions and future directions are given in Section 7.

## 2. Overview

Parallel I/O is a necessary component of data-intensive applications such as scientific simulations. Parallel hardware storage systems with multiple disks and high-bandwidth switched interconnect are becoming increasingly available to fill this need. In this section, we will present the system model assumed in this paper. The scheduling problem and related work are also briefly described.

### 2.1. System model

Fig. 1 shows the architecture of high performance *cluster computing* system of interest. The system consists of a set of $M$ independent processors, $P = \{P_1, P_2, \ldots, P_M\}$, and a set of $N$ independent disks, $S = \{D_1, D_2, \ldots, D_N\}$, that are connected by a fast interconnection network. Data is stored on the disks in units of blocks; a block is the unit of access from a disk. In each parallel I/O operation a set of up to $N$ blocks, one from each disk, can be accessed. The blocks

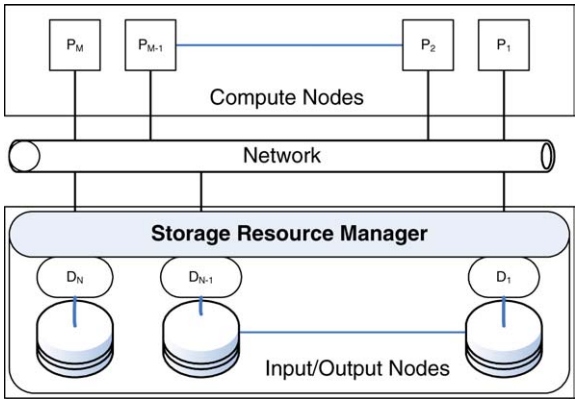


Fig. 1. Architecture of the system.