Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 67 (2007) 1286–1300
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

General store placement for response time minimization in parallel disks

Akshat Verma, Ashok Anand*

IBM India Research Laboratory, Hauz Khas, New Delhi 110016, India

Received 8 August 2006; received in revised form 24 July 2007; accepted 31 July 2007
Available online 6 September 2007

Abstract

We investigate the placement of N enterprise data-stores (e.g., database tables, application data) across an array of disks with the aim of minimizing the response time averaged over all served requests, while balancing the load evenly across all the disks in the parallel disk array. Incorporating the non-FCFS serving discipline and non-work-conserving nature of disk drives in formulation of the placement problem is difficult and current placement strategies do not take them into account.

We present a novel formulation of the placement problem to incorporate these crucial features and identify the runlength of requests accessing a store as the most important criterion for placing the stores. We use these insights to design a fast (running time of $N \log N$) placement algorithm that is optimal under the assumption that transfer times are small. Further, we develop polynomial-time extensions of the algorithm that minimize response time even if transfer times are large, while balancing the loads across the disks. Comprehensive experimental studies establish the efficacy of the proposed algorithm under a wide variety of workloads with the proposed algorithm reducing the response time for real storage traces by more than a factor of 2 under heterogeneous workload scenarios.

Keywords: Stream placement; Data store placement; Response time minimization; Parallel disks; Storage management; File placement

1. Introduction

Parallel I/O systems have received a lot of attention [11,17,22] in the recent past for their ability to provide fast reliable access, while supporting high transfer rates for dedicated supercomputing applications as well as diverse enterprise applications. Disk arrays partition data across multiple disks in a storage pool and provide concurrent access to multiple applications at the same time. Moreover, a single application with huge data requirements may partition its data into stores and place them across multiple disks and use this parallelism to alleviate the I/O bottleneck to a certain extent. In today's web-service scenario with performance guarantees, throughput is no longer the only performance requirement for applications, and many applications require the average response time of their requests to remain under certain thresholds. Since storage latencies dominate request response times,[1] reducing a request's response time boils down to minimizing the storage latency. The high variances in service times due to the heterogeneous applications served from a disk array, combined with the non-work conserving nature of the disk drives, imply that the response time of the requests of a store is influenced primarily by characteristics of the other stores placed on the same disk.

In this paper, we investigate the placement of logical groups of data, that we term stores (examples of stores include database tables, files owned by a particular user, or data used by an application), across M parallel disks. We denote a sequence of disk requests generated by an application or user as a stream and term the logical datagroup accessed by the stream as a store. The exact question that we answer in this paper is the following:

Given a set of N stores that we need to place on M disks, which stores should be placed together (i.e., on the same disk), such that the response time averaged for all requests is

* Corresponding author. Fax: +91 11 26861555.
E-mail addresses: akshatverma@in.ibm.com (A. Verma), ashokanand@in.ibm.com, ashok.anand@gmail.com (A. Anand).

[1] From year 1990 to 2000, transfer rate has increased by a factor of 10, seek/rotation times by a factor of 2 to 3, whereas processing speed has increased by a factor of 100 [14,15].

0743-7315/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.07.012

minimized. We work under the additional constraint that the load is balanced across all the disks.

The problem also finds applications in web-services [12,3], where user streams are allocated to different web-servers, and each server may manage its own storage. Content distribution and multimedia hosting servers, where disk latencies dominate average response time, are other settings where the placement of stores determine the performance of the service significantly. The current placement strategies strive to load balance the workload on the array of disks without explicitly minimizing the response time. Since mean response time is directly related to variance, such an approach based only on the load (and oblivious of variance) is not sufficient in a heterogeneous shared storage service scenario.

1.1. Related work

Substantial literature exists on the problem of assigning data to disks in a parallel or distributed systems [5,7,11,17,22]. Typically, these algorithms assign the data to disks in a fashion so as to minimize a particular cost function. In a completely general scenario, this cost function may involve all costs: communication costs, storage costs, update costs and queuing costs. Dowdy and Foster [5] show that the above problem is NP-hard even for very simple cost functions and hence viable solutions are heuristic-based. The most important performance measure for many applications are the mean response time and/or system throughput. Hence, a large number of heuristics were developed that aim at optimizing mean response time or total throughput by focusing on the queuing delays, as they dominate over network delays in practice [4,17]. The most common approach for minimizing the queuing delay has been to minimize the utilization on each disk; the above attained by balancing the system load evenly across all disks. A popular technique to balance the load (though indirectly) is by minimizing the total size of files assigned to each disk.

Lee et al. [11] show that balancing the load across all disks is not sufficient to minimize response time and significant reduction in overall response time can be achieved by minimizing the variance in service time on the disks. The strategy used in [11] is to partition the files based on the sizes of the files. Similar partition-based methodologies have been proposed by Harchol-Balter et al. [9] for the task assignment problem where tasks are allocated to servers based on task duration. However, both the task assignment as well as file assignment models are simplistic as compared to the store placement model since they place individual files (or partitions of a file) on a disk (analogously tasks on a server for the task assignment problem) where each file or task has deterministic service time. On the other hand, the service time of a data store is represented by a stochastic distribution and this additional complexity is not captured in any of the above formulations. Garg et al. [7] use a model that captures the stochastic nature of service time while assigning streams to servers in a web-server farm scenario, with the objective of minimizing the average response time.

However, all the above work (including [7]) use queuing theory results based on FCFS scheduling discipline for motivating their algorithms. Since most operating systems no longer use FCFS (linux uses *C-SCAN* and variants), such algorithms are no longer meaningful. Further, an important distinctive feature of the store placement problem is the non-work conserving nature of disk requests (as opposed to server requests, where work, that is measured by service time, is same for same kind of request), where the service time of a request depends on the location of the previous served request. The existing placement strategies as well as task or stream assignment algorithms do not take either of these two critical factors into consideration.

1.2. Contribution

We formulate the problem of placing stores, with associated workload parameters, across parallel disks with the twin objectives of (a) minimizing the overall response time across all disks and (b) balancing the load across the disks, as an integer linear programming (ILP) minimization problem. We transform this NP-hard problem to a set of three sub-problems in a way that an optimal solution to the three sub-problems is an optimal solution to the above minimization problem under certain assumptions and constraints. One of the sub-problem captures the non-work conserving nature of disk workload, whereas the other two capture the queuing behavior. The segregation of the non-work conserving nature of the disk workload from the queuing behavior reduces the complexity of the sub-problems, thus allowing us to solve them under certain realistic assumptions. We define the notion of *Disk-RunLength* (*DRL*) and show that placing stores across the disks based on *DRL* is optimal under the reasonable assumption that transfer times are small. We experimentally demonstrate that our fast *DRL*-based algorithm is able to reduce the average response time significantly, while still meeting the balanced load criterion.

2. Model and preliminaries

Data on an enterprise-wide shared storage service are typically organized into stores. A store is a logically aggregated set of data with associated statistical parameters (e.g., request arrival rate, request size distribution, runlength). A store in a database scenario is a table or a set of associated tables. In a shared file systems, all files belonging to a user may constitute a store. In an I/T production scenario, all source files or all email files may constitute a store. Access to each store is represented as streams, where each stream can be thought of as access from an application or a customer. We aggregate all streams that access a store and represent it as a single stream. Hence, we combine the notions of stores and streams and use it to denote any set of logically grouped requests with associated statistical parameters. We next present the general response time minimization problem and an architecture to estimate the stream parameters and solve this minimization problem.

2.1. Problem formulation and framework

We solve the following QoS minimization problem. Given N streams or stores G_i (and associated parameters) and a set of M