I/O Deduplication

Abstract

- Duplication of data in systems is becoming commonplace

- We introduce I/O deduplication, a technique that leverages duplication for improving I/O performance.

- I/O deduplication has the potential to both reduce the mechanical delays during I/O operations as well as eliminate I/O operations altogether.

- Evaluation of a prototype implementation using a variety of workloads including A,B,C, revealed that ...

- We believe that ...

1 Introduction

Data duplication in storage systems is becoming increasingly common. The elimination of such duplication for improving storage space utilization is an active area of research []. Indeed, eliminating most duplicate content is inevitable (for cost-effectiveness) in capacity-sensitive applications (e.g., data backup). On the other hand, in systems where the amount of duplicate content is not overwhelming, eliminating duplicate content is not a key concern. Examples of such systems are more common than one may imagine. Some of these include email servers where mailing-lists, circulated attachments, and SPAM can equally contribute to the creation of duplicate content, virtualized environments where virtual machines may run similar software and thus create co-located duplicate content across their virtual disks, and file and versioning control servers of collaborative groups that often store copies of the same documents, executables, sources etc.

Taking a contrary view on data duplication, we explore the premise that *intrinsic* data duplication in systems can be utilized to improve I/O performance. Here, we refer to intrinsic (or application/user generated) duplication as opposed to forced (system generated) duplication such as in a RAID 1 duplicated storage system, a member of a well-known class of systems trade-off capacity for performance [**?**].

We conducted a preliminary trace-based analysis of intrinsic duplication in both production and non-production systems. Our analysis revealed **[[fill in based on section 2. two metrics of our interest: *content similarity distribution* (content-similarity / distance distribution for similar content as % of storage volume), and *content reuse-distance distribution*.**]].

In this paper, we present and evaluate techniques that utilize data duplication for optimizing I/O operations. These techniques either optimize disk head movement or eliminate I/O operations altogether in a bid to minimize duplicate I/O activity within the storage system. We refer to effect of these optimizations collectively as *I/O deduplication*.

I/O deduplication comprises three key techniques: (i) *content based caching* that uses the popularity of "data content" rather than "data location" of I/O accesses in making caching decisions, (ii) *dynamic replica retrieval* that upon a cache miss, dynamically chooses target replica to retrive that minimizes disk head movement , and (iii) *optional replica updates* that dynamically chooses between updating the target location (as requested) for duplicate content and registering persistent copy-on-write metadata information.

We evaluated a Linux implementation of the I/O deduplication techniques both individually and collectively for a variety of workloads including an email server, a virtual machine monitor running production servers, and a simulated research group desktop virtual machine server. Our evaluation aims at demonstrating the benefits and overheads of I/O deduplication. Particularly, **[[summarize performance results for various workloads. We also measured the memory and CPU overheads. —- summarize overhead.]]**

In the rest of this paper, we make the case for I/O deduplication (§ **??**), elaborate on a specific design and implementation (§ **??**), perform a detailed evaluation of improvements and overhead for various workloads (§ **??**), discuss related research (§ **??**), and finally, present conclusions and suggest directions for future work (§ **??**).

2 Conclusion

3 Motivation

In this section we analyze the repetition of I/O requests from two points of view: static and dynamic. Static refers to the fact that some blocks might be copied over the disk.

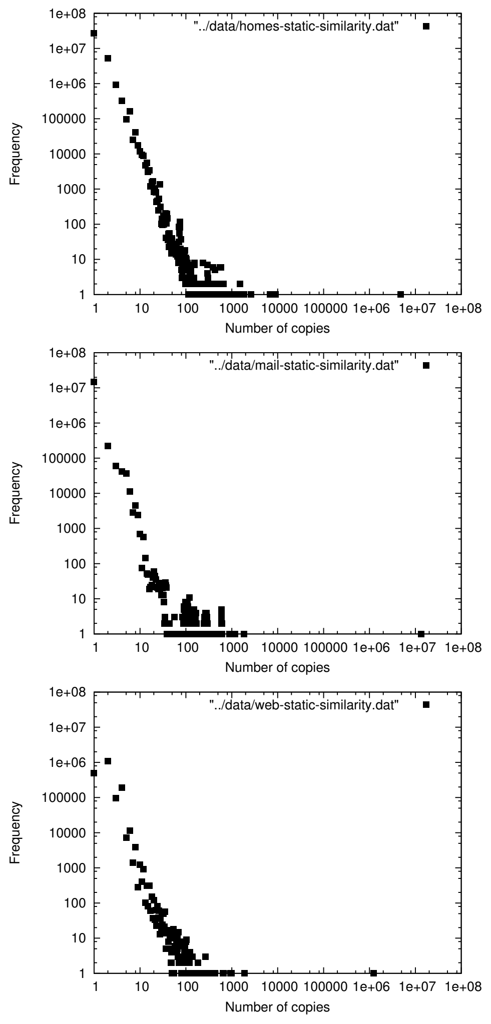

Figure 1: Distribution of the number of copies for the a. homes, the b. mail and the c. web workloads. The point on the far right represents the zero block with a high number of copies.

Dynamic refers to the movement of similar content.

We start by defining these concepts:

Then calculate them over a set of workloads, described in table **??**.

3.1 Static similarity

3.2 Dynamic similarity

The importance of these graphs is that there are blocks, after a long reuse distance that cannot be captured by sector addressed caches. This is most notorious on figure **??**

The importance of these graphs is that there are blocks, after a long reuse distance that cannot be captured by sector addressed caches. This is most notorious on figure **??**

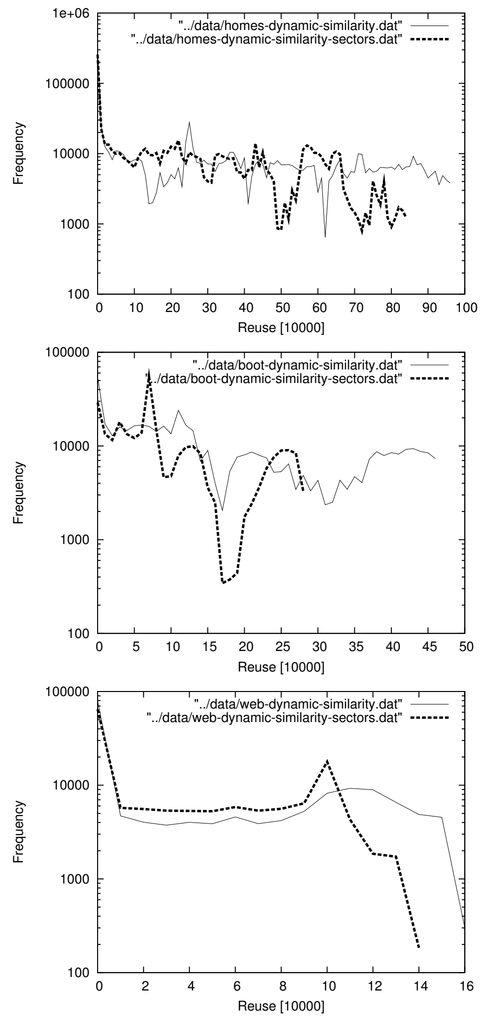

Figure 2: Distribution of reuse distance for the a. homes, b. boot and c. web workloads. Content reuse distance is shown with a solid line and sector reuse distance with a dotted line.

Name	Description
mail	application, number of users, etc etc TODO
web	TODO
homes	4 home partitions being used by researchers using Ubuntu Desktop. The traces were taken for 2 days.
boot	Simultaneous booting of two virtual machines using VMWare Server and file based disks. Both virtual machines are Ubuntu servers 8.04. Note that the virtual disks blocks are not guaranteed to be 4K aligned.

Table 1: Description of the workloads analyzed.

name	blocks	Unique blocks	Static similarity	Dynamic similarity
mail	29298535	15195268	0.49	-
web	5242880	1910287	0.63	47127
homes	49570806	33959841	0.31	306988
boot	-	-	0.31	171517

Table 2: static similarity, total number of used blocks, unique used blocks, and duplicate used blocks (i.e. number of blocks with at least one copy) for the a. mail, b. web and c. homes workloads. Used blocks are those marked as used by the file system.‘

4 Design

References

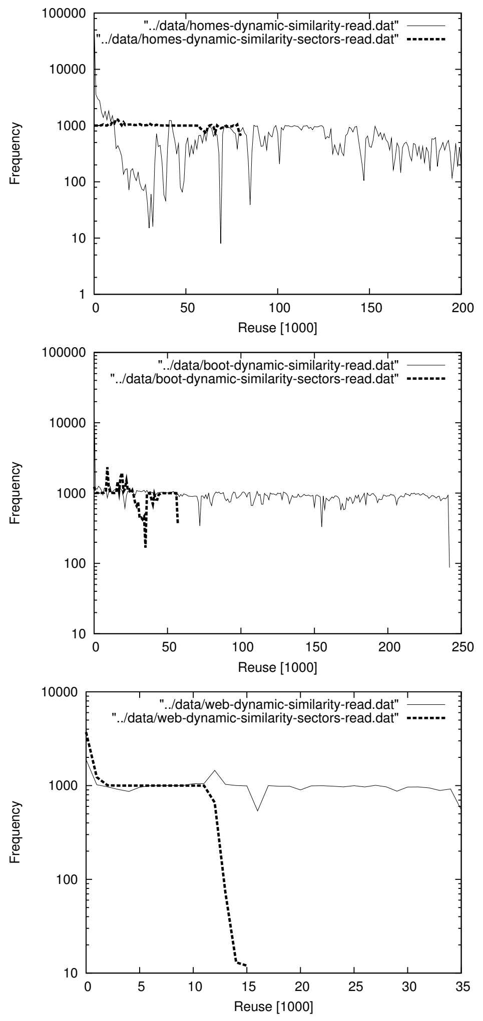

Figure 3: Distribution of read reuse distance for the a. homes, b. boot and c. web workloads. Content reuse distance is showh with a solid line and sector reuse distance with a dotted line.