Spring 5-2004

Availability in IBPro, a Personal Video Recorder Built on Faulty Storage Components

John Allen Garrison
University of Tennessee - Knoxville

UNIVERSITY HONORS PROGRAM

SENIOR PROJECT - APPROVAL

Name: John Garrison

College: Engineering Department: Electrical & Computer

Faculty Mentor: Jim Plank

PROJECT TITLE: Availability in IBP vd, a Personal Video Recorder Built on Faulty Storage Components

I have reviewed this completed senior honors thesis with this student and certify that it is a project commensurate with honors level undergraduate research in this field.

Signed: _____, Faculty Mentor

Date: 5-6-04

Comments (Optional):

Availability in IBPvo, a Personal Video Recorder Built on Faulty Storage Components

John Garrison Scott Atchley James S. Plank

Logistical Computing and Internetworking Laboratory
Computer Science Department, University of Tennessee
{jgarriso, atchley, plank}@cs.utk.edu

Abstract

This paper addresses the availability of storage in IBPvo, a video recording application built on faulty storage components provided by the Network Storage Stack. IBPvo uses the Network Storage Stack to store, distribute, and manage the large files generated by recording television. We examined how IBPvo uses this storage over a 47.5-day period and found that while IBPvo experiences failures and loss of data, this loss is relatively small.

1. Introduction

Logistical Networking is the application of end-to-end approaches to storage within the network. As the need for large-scale distributed storage in web based, distributed, and peer-to-peer applications rises, a solution is needed to handle storage in this environment [1]. Logistical Networking and the Network Storage Stack provide this.

The Internet Backplane Protocol (IBP) is the foundation of Logistical Networking [2]. IBP provides server software that handles the insertion of storage into the network as well as client software that enables clients to allocate and use the storage. The exNode is a data structure for aggregation, analogous to the inode in Unix. Instead of aggregating disk blocks on a single disk volume to compose a file like an inode, an exNode aggregates IBP byte-arrays to compose a logical entity like a file [3].

The Logistical Runtime System (LoRS) is a set of generic tools that allows users to create, manipulate, and use the "files" on the network. The LoRS tools are:

lors_upload:	Store a data into a wide area network file.
lors_download:	Retrieve a network file from the wide area.
lors_augment:	Add replicas to a network file.
lors_trim:	Subtract replicas form a network file.
lors_refresh:	Extend the expiration times of storage allocations.

IBPvo is an application that uses the Network Storage Stack to provide a video recording system and then deliver exNodes of recorded files to users. Users program recordings via a web page where they set parameters such as what channel to record, length of recording, bitrate of the recording, and others. After recording the files, IBPvo

uses **lors_upload** to store the file in the network. **Lors_upload** uses the option -l zip= 37996 to upload the file to servers near zip code 37996 (Knoxville, TN). The primary reason for this is that the recorder is in Knoxville on the University of Tennessee Campus and an upload can be performed quickly there. Next it uses **lors_augment** to create two additional copies in the network. The first copy uses -l airport= TYS to create copies near the Knoxville, TN airport. The second copy goes to a location selected by the user when setting up the recording. IBPvo then e-mails the exNode to the user.

Once data is in the network, IBPvo does regular maintenance on the exNodes until the user deletes the exNode. Maintenance occurs twice daily, at noon and midnight. In a maintenance cycle, IBPvo calls various LoRS functions on all the current exNodes. IBPvo first calls **lors_refresh** to extend the expiration time of the storage. IBPvo then inspects all the files currently being stored. IBPvo calls **lors_trim** to remove any pieces that either could not be refreshed, or that are unavailable from the exNode. Then it calls **lors_augment** to replace any pieces that have been trimmed. **Lors_augment** is called with the option -l "zip= 37996", which means that new copies are made on servers close to the 37996 zip code (Knoxville, TN).

2.1. Data Set

The data in the logs comes from a 47.5-day period starting at noon on January 1, 2004. At the beginning of the logs, there were files present in the system. For the purposes of this analysis, we disregarded these files. The primary reason for this is that we do not know when the files were uploaded into the system and that these files are statistically different from the rest of the files that were uploaded beginning on January 1. The most notable difference is the propensity of these files to have fatal failures on the first day, as is the case in 28 of the 43 files in the system on January 1. The IBPvo system uploads 269 files over the course of the logs. Many of these files do not survive until the end of the logs due in part to errors, but mostly from deletion by the user. At the end of the logs there are 154 files in the system. Figures 1 and 2 show a relatively linear increase in both files and pieces respectively in the logs. This is consistent with what one would expect from regular use of the IBPvo system as people upload files faster than they delete them.

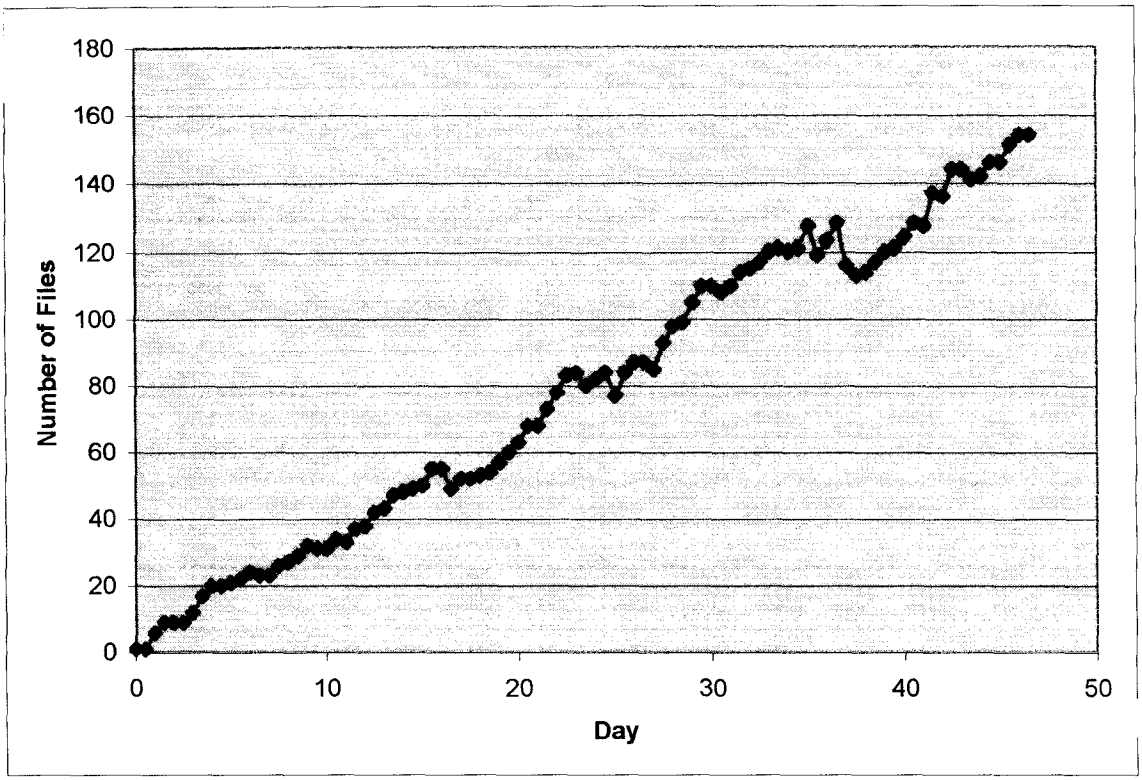

Figure 1: Number of files over time.

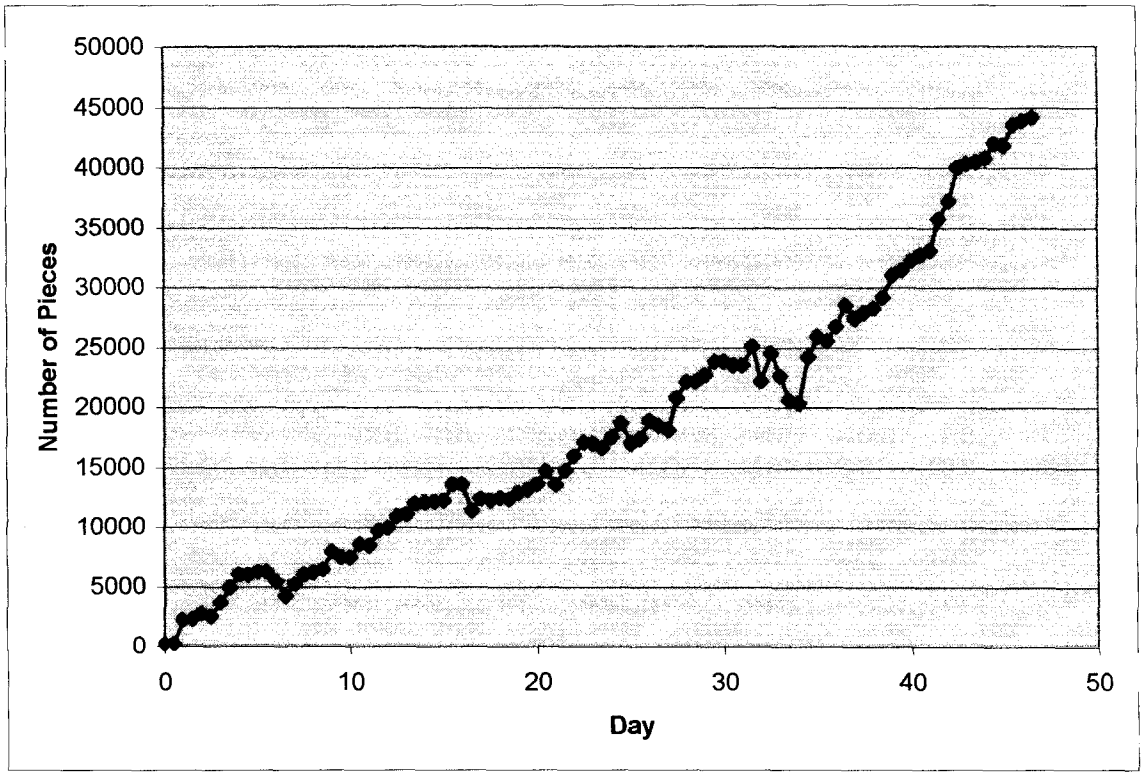

Figure 2: Number of pieces over time.

2.2. Trims

Trims occur when the LoRS tools find that a given piece of a file no longer exists. As would be expected from the increase in number of files and pieces throughout the logs, the number of trims increases as well. The number of trims varies greatly throughout the logs as can bee seen in Figure 3. As few as zero pieces were trimmed quite a few times, and at one point 4086 pieces were trimmed. The size of these swings from very low numbers to high numbers increases over the course of the log as the number of pieces increases. When we examine the number of trims per piece in the system, these swings still produce dramatic changes, but the number of trims per piece never rises above 23% of pieces in the system being trimmed. As Figure 4 clearly shows, the early trims in Figure 3 are much more important than one would suspect by viewing only Figure 3. Due to the fewer number of pieces in the system, these trims amount to a larger percentage of the total system than raw numbers of trims four or more times larger later in the logs.

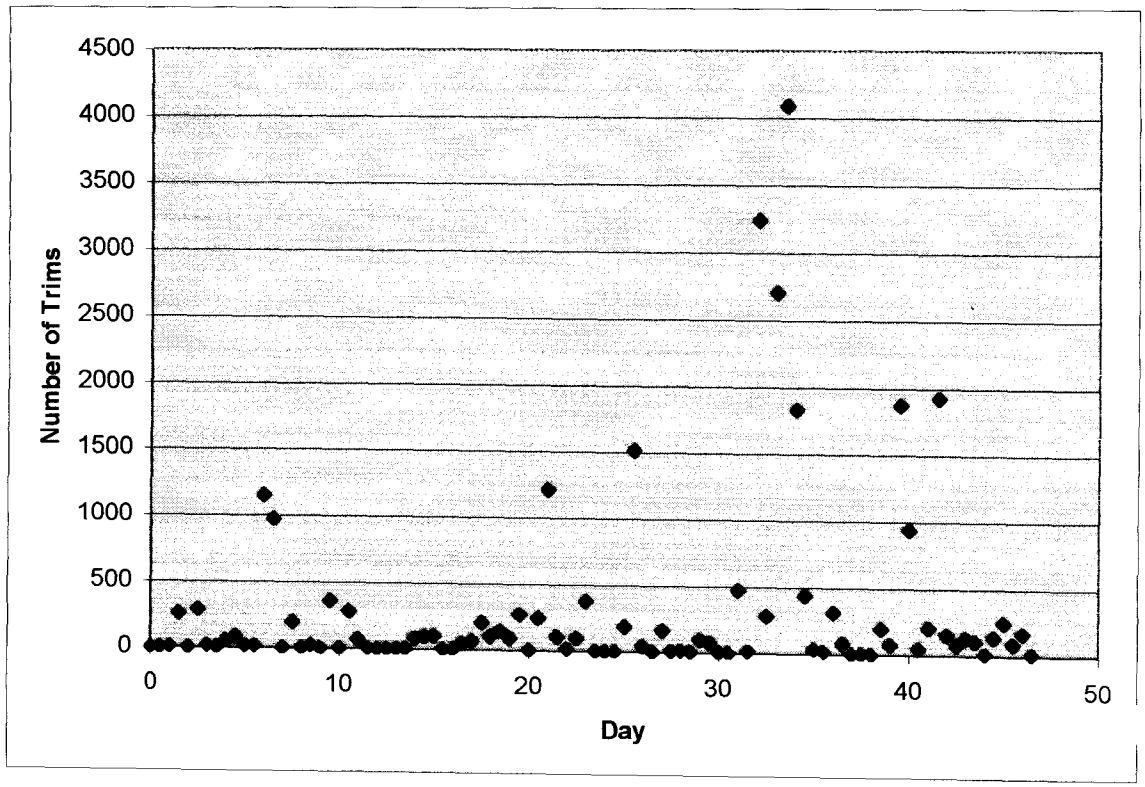

Figure 3: Number of trims over time.

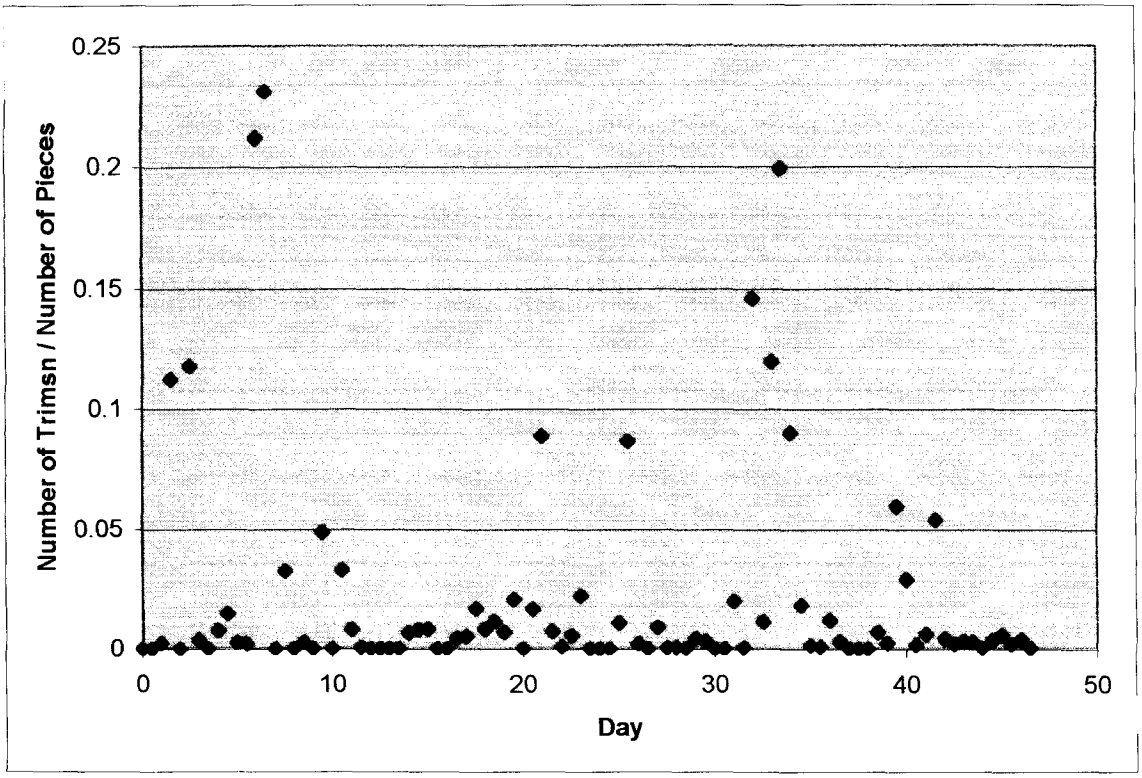

Figure 4: Number of trims per piece.

When a histogram of the number of trims per piece is generated, an interesting result is observed. In Figure 5, we generated a histogram by grouping values into sets of .5%. The resulting graph shows a large number of low values that quickly drops off as the values increase. In over half the cases, less than .5% of pieces were removed from the system, and less than 1% of pieces were trimmed in over 70% of the cases. This graph resembles an exponential, although we did not do the analysis to reach that conclusion analytically. The mean value of trims per piece is 2.1%.

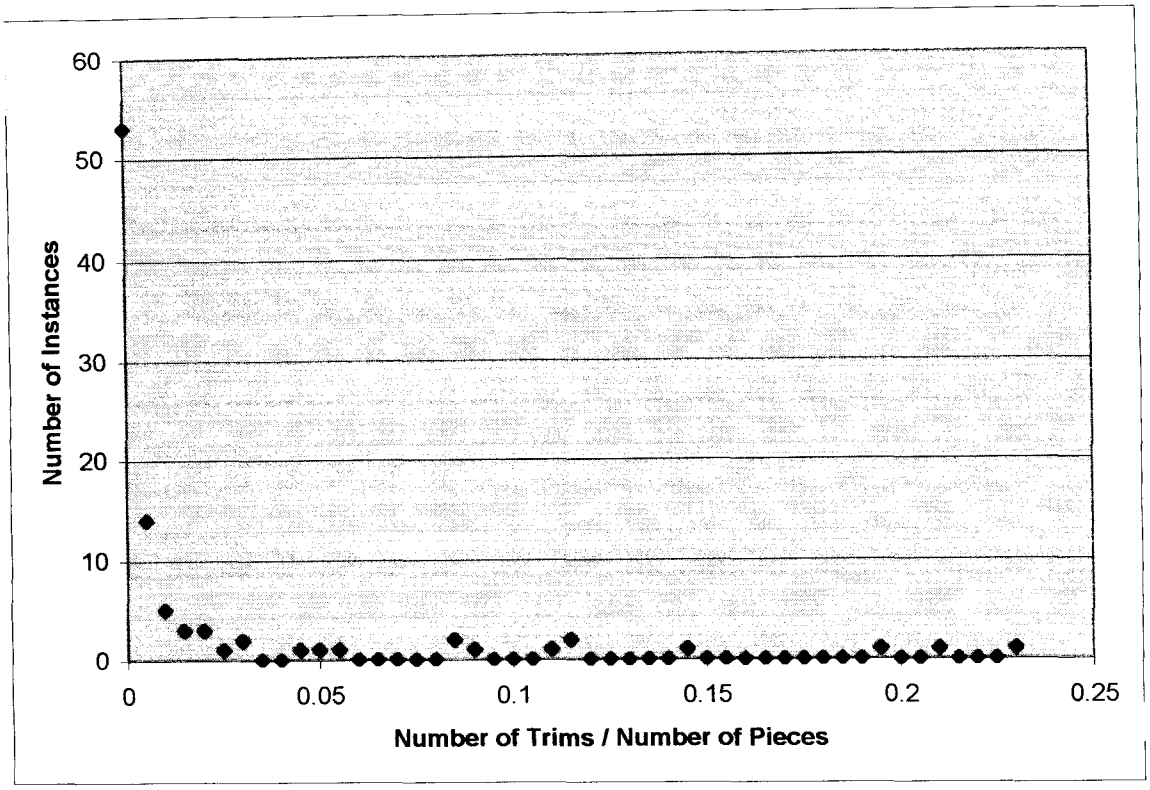

Figure 5: Histogram of trims per piece.

2.3. Errors

There are two error situations that are similar, but the differences in these two situations should be noted. The first situation is a "fatal" error. In this case, a LORS_ERROR –9 has occurred, indicating that among the servers holding the file, there is a piece that is no longer on any server. At this point the file does not exist in its entirety and cannot be recovered. This is the most common failure mode. The second situation is any of a number of "nonfatal" errors. These include all other failures other than LORS_ERROR –9. A few of these errors occur, including timeout errors and errors with IBP such as trying to copy to a depot with a full disk.

The raw number of errors is not an overly useful statistic due to the dramatic increase in the amount of data stored over time. Examining the frequency of errors over time in relation to the number of files does lead to useful results. Since errors are a condition that affects files, we examine errors in relation to files as opposed to pieces as we did with trims. As Figure 6 shows, the number of fatal and non-fatal errors in relation to the number of files follows a general trend. In the beginning of the logs there are large periods without failure as well as dramatic peaks in errors per file. This evens out over time as more files are added. The last case of zero failures occurs on February 5. After that the average number of total errors per file is .021, indicating that 2.1% of files have errors during a set of augment attempts. During this period the average number of fatal errors per file is .016.

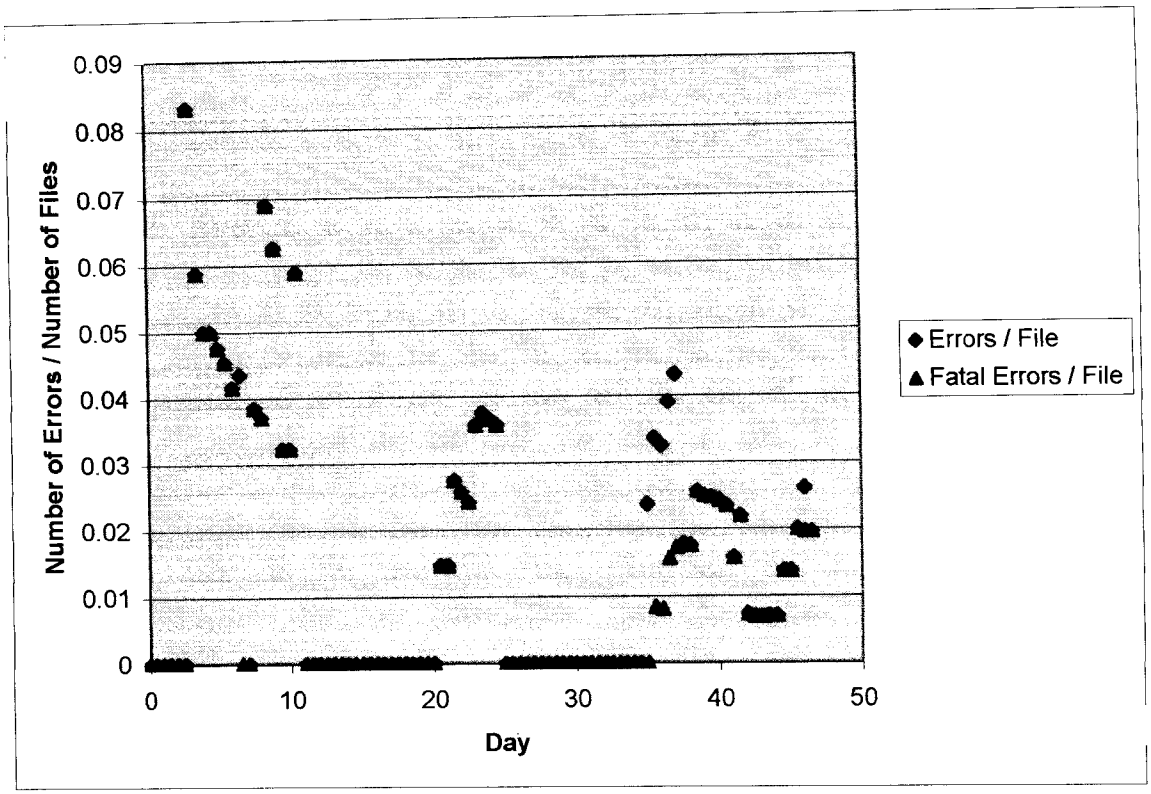

Figure 6: Errors in relation to number of files.

A histogram of the number of fatal errors per file gives a similar result to the histogram of trims per piece. Figure 7 shows the same large number of instances in very low percentages as Figure 5 shows for trims. The difference in Figures 4 and 6, however, indicate that we should not draw the same conclusion from these two similar graphs. The graph of trims per piece is much more evenly distributed than the graph of errors per file. Figure 6 shows extended periods of zero errors and extended periods of errors. In many cases a file that undergoes an error stays in the system, unlike a piece that is trimmed. This helps to account for the extended periods of errors. The extended periods of zero errors and the dramatic shifts to zero errors are harder to explain.

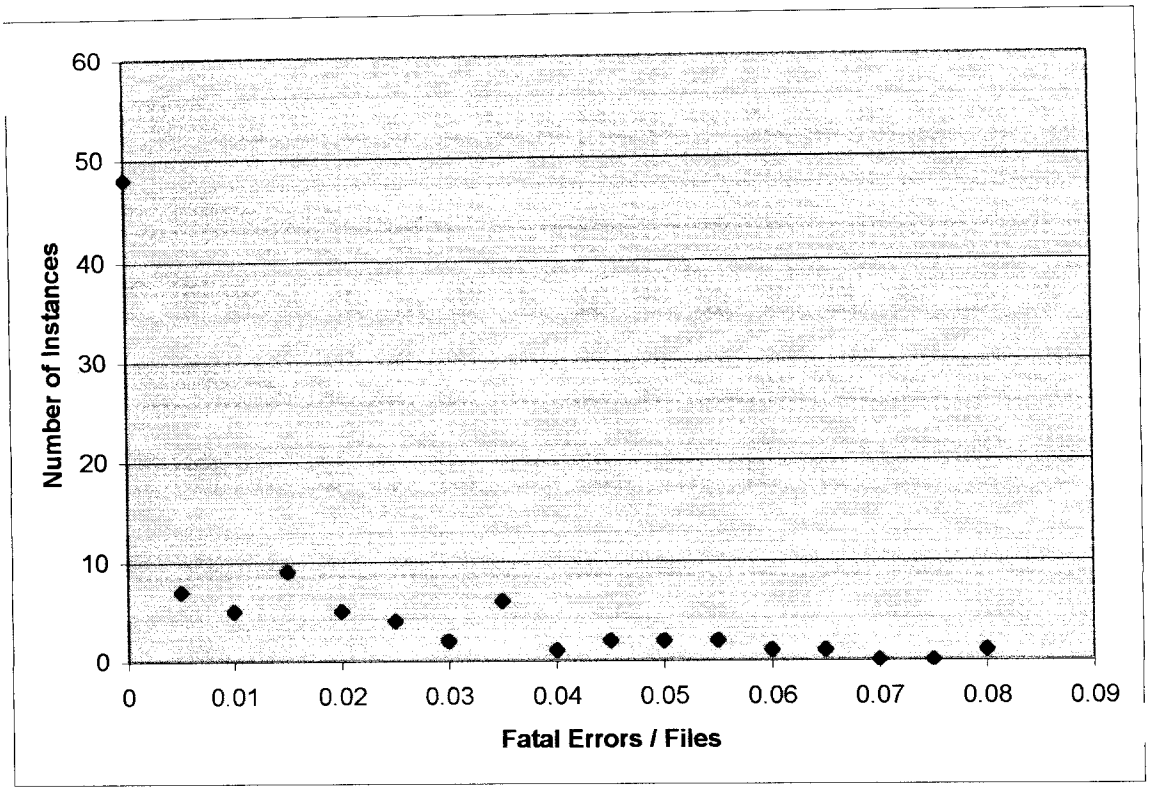

Figure 7: Histogram of fatal errors per file.

3. Conclusion

Throughout the logs, the failure rate of files is relatively consistent if one accounts for the change in data size. In the beginning, before the number of files has increased sufficiently, any error's impact on the total is significant. Once enough data is being stored, the system stabilizes to an error rate of 2.1% with a fatal error rate of 1.5%. The rate of trims varies significantly throughout the logs, and has a correlation coefficient of only .06 with the error rate. In the future, we will use this data to determine optimal or near-optimal replica placement strategies, such as employing more replicas, or some sort of erasure encoding [4].

References

[1] M. Beck, T. Moore, and J. S. Plank. An end-to-end approach to globally scalable network storage. In *ACM SIGCOMM'02*, Pittsburgh, August 2002.

[2] J. S. Plank, A. Bassi, M. Beck, T. Moore, D. M. Swany, and R. Wolski. Managing data storage in the network. *IEEE Internet Computing, 5(5): 50-58, September/October 2001.*

[3] S. Atchley, S. Soltesz, J. S. Plank, M. Beck, and T. Moore.

Fault-tolerance in the network storage stack. In *IEEE Workshop on Fault-Tolerant Par. and Dist. Sys.*, April 2002.

[4] J. S. Plank, and M. G. Thomason. A Practical Analysis of Low-Density Parity-Check Erasure Codes for Wide-Area Storage Applications. In *DSN-2004,* Florence, Italy, June 2004.