DORA: Exploring a Dynamic File Assignment Strategy with Replication

Jonathan Tjioe, Renata Widjaja, and Abraham Lee
Computer Science Department
San Diego State University
San Diego, CA 92182

1. Introduction

The problem of managing and distributing files to maximize disk performance has been a popular topic of many discussions [1][2][3][4][5]. There are several effective static algorithms that have addressed this issue such as the *static round robin* (SOR) algorithm. SOR has been proven to produce better response time than other static algorithms such as Greedy, Sort Partition (SP), and Hybrid Partition (HP) [1]. SOR is unique compared to the other static algorithms because it provides considerable performance improvements even if the workload assumption, which says that there is an inverse correlation between file size and its popularity (small files are more popular than large files), does not hold [1]. However, as its name states, it is a static algorithm, and its functionality is limited by the assumption that the file and their access patterns do not change over time. In reality, however, this workload assumption is not accurate for all cases. We, therefore, propose a new dynamic algorithm called *dynamic round robin with replication (DORA)*. The main characteristic of DORA is that it takes into account the dynamic nature of file or data access patterns to uniquely adapt to changing users' demand. In addition, DORA utilizes file replication to further maximize response time and file throughput; these will be further explained in the project details section.

This proposal is organized as follows. Section 1 provides a brief introduction. In Section 2, the motivating factors prompting this research are detailed. Section 3 summarizes the project at a high level perspective. Section 4.1 begins by describing the overall architecture and implementation details of DORA. Section 4.2 discusses several challenges that must be addressed when implementing DORA with replication. Sections 4.3 and 4.4 outline specific deliverables and the schedule of the project. Finally, Section 5 reiterates the functionality of DORA and how it compares to other dynamic algorithms such as cool vanilla (C-V) [11] and simple cost minimization (CM) [11].

2. Motivation

Fast response time is a technology factor that end-users are accustomed to. In a world of distributed applications and web pages that grow increasingly more bandwidth intensive, considerable research has been done to improve methods which can lead to providing instantaneous response to the impatient end-user. Oftentimes, the physical disk is the bottleneck to providing timely response to users' requests. As a result, much of today's research centers on efficiently managing the assignment of file and disk scheduling. Some examples of these research areas are RAID architecture that focuses on data striping, data replication, and data mirroring to achieve high data throughput and high data reliability [6][7]. Substantial research has also been done to reduce disk head latencies associated with moving the head to the physical location of the data on disk [2][3]. For example, SOR has contributed a successful static file management algorithm which operates by first sorting the files according to their size and then allocating them to homogeneous disks in a round robin manner such that the popularity (heat) of files are distributed equally across the disks [1]. Moreover, several published dynamic algorithms such as C-V, and CM have also made significant contributions to this research field. C-V and CM both work by constantly monitoring the heat imbalance between disks in order to distribute them evenly across the disks. In addition the CM algorithm implements methods to find an optimized location for each newly created file; therefore, the cost from the need to reorganize files can be minimized [11].

The DORA algorithm furthers the research in the data management algorithms by dynamically

assigning files, while still providing fast access to those files. Let us take the example of a web-server application, Amazon.com, where prospective buyers can search for a book title and the server will respond in a matter of seconds. Sellers can add new product which will be stored in the database, or modify an existing post, thereby changing the contents and ultimately the size of the file. In those kinds of scenarios, the assumption is that there are algorithms in place to manage the thousands of files in such a way such that users can receive fast response time to requests they have issued to the server.

In such a dynamic scenario mentioned in the above example, static algorithms have a common drawback: they do not take into account the changing popularity (heat) of files. Therefore, the advantage of DORA over static algorithms is it has the ability to dynamically adapt to the heat of files by constantly monitoring and periodically rearranging the files in such as way that file throughput can be maximized without compromising the response time. Additionally, DORA will be designed to handle file reorganization while the system remains online and continues to serve users' request.

3. Project Summary

The following techniques will be implemented in DORA to improve the response time performance in a dynamic environment: file replication [10], file-size variance minimization [5], and garbage collection for files. In addition, simulations will corroborate that performance can be improved by considering the heat variance of files and replication method for hot files. Also, a performance comparison with other dynamic algorithms such as C-V and CM will be shown over varying workloads to simulate the robustness of the DORA algorithm. The simulation results will show that DORA will be able to improve the response time regardless of files' sizes or the changing heat of files.

4. Project Details

4.1 Architecture and Environment

Matlab software will be used for all simulations in this project. For benchmark comparison purposes, several other dynamic algorithms will be simulated against DORA: C-V, and CM.

There will be ten homogeneous physical hard disks used in the simulation; however, this will not be a RAID configuration.

Non-partitioned files of varying sizes and loads corresponding to variable heat distributions will be used to convincingly show that DORA significantly performs better than other dynamic algorithms by adding replication for extremely hot files as well as adding the ability to dynamically recalibrate file assignment according to changing users' demand.

The DORA algorithm will function as follows:

1. Sort files according to size in ascending order in an array I.
2. Compute the average disk utilization of all the files' heat.
3. If the heat of a file is greater than or equal to a designated threshold heat then replicate the hot file across multiple disks.
4. If the heat of a file is not greater to a designated threshold heat write the file to each disk, one at a time, as in SOR.
5. Listen for, accept, and serve users' requests.
6. Record all file accesses while serving requests.
7. At the end of each epoch (1 hour), examine the heat of each file and repeat steps 1-6.

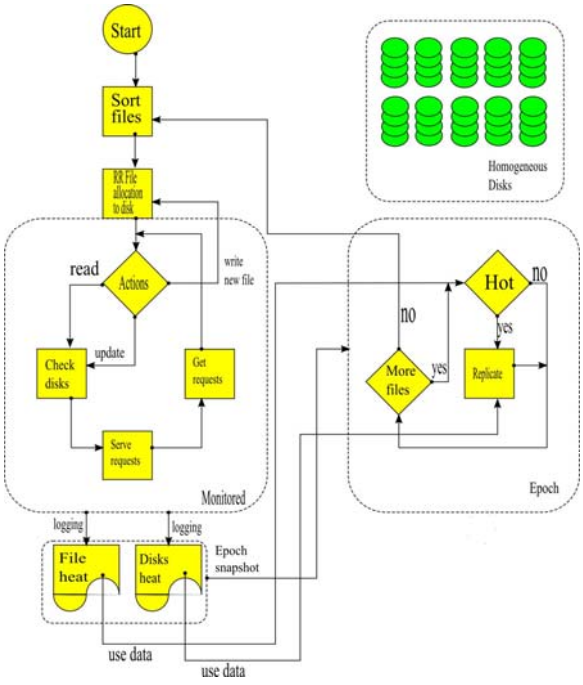

Figure 1. DORA algorithm flowchart.

Figure 1 above shows the flowchart of the DORA algorithm which should assist in explaining how it behaves in real world scenarios.

Let the following variable representations hold:

- *I* represents an array with the set of all files that will be used to service requests coming from users.
- A *file i* is the *ith* file in the Array *I*.

- *hi* is the "heat" of *file i*, where "heat" is the access frequency of a file.
- *HEAT_HOT* is the threshold heat value of a file, at which the decision will be made to replicate the file across multiple physical disks.
- *DISK_NUM* is the total amount of simulated hard disks.
- *FILE_NUM* is the total amount of files that users can request service to.
- *EPOCH_NUM* is the total amount of epochs to evaluate over.

Also, a snippet of DORA pseudo-code is shown in Figure 2 below to give a better explanation of its functionality.

```
while(EPOCH_NUM > 0) {
  sortByAscendingFileSize(I);
  ave_utilization = sum(total heat of
  files)/DISK_NUM
  for(i = 0; i <= FILE_NUM; i++) {
       if(hi >= HEAT_HOT) {
            replicate(file i);
            jump NEXT_FILE;
       }
       else write_SOR(file i);
  NEXT_FILE:
          };
  EPOCH_NUM--;
//Listen for, accept, and serve user requests
};
```

Figure 2. DORA pseudo-code.

Measurements that will be observed in this research include:

Access time: Measurement of how long it takes to access each file.

Access rate: Measurement of request frequency for each file. This will be used to determine which files are extremely hot and should be replicated.

Heat of each disk: Measurement of the disk's workload. This is proportionally related to how many hot files are on the disk.

Response time: Measurement of how long a request takes to complete.

Throughput: Measurement of how many requests can be served over a period of time.

4.2 Implementation Issues and Challenges

Several challenges arise from the dynamic nature of DORA and from the use of file replication.

Creating replicas. While load balancing can be performed effectively by making copies of hot files to multiple disks, creating the replicated files on multiple disks will introduce additional overheads. Cost versus Benefit trade offs must be examined before replicating a file.

Updating replicas. Synchronization issues must be handled. For example, if a user request updates a replicated file, all replicas and the original file must be updated to ensure the latest copy of the file is being used to serve user requests.

Deleting replicas. File access patterns will change overtime, which eventually will result in both hot and cold files taking up space on each disk. Cold files that were once hot were replicated across multiple disks at some point in time in the past. The cold replicas must be removed, while leaving the original file intact. This requires a garbage collection mechanism to conserve space and efficiently manage files.

Managing replicas. Records of all replicated files must be kept which include information about the file such as the file identifier, how many replicas are in circulation, and what disks the replicas reside on. Selecting the destination drive and the number of copies remains one of the main focuses of many file management algorithms that use replication. The choice of heuristics significantly affects performance [9].

Response time vs Throughput. Quick response time is the top priority of end-users On the other hand, maximizing throughput is the major concern for web server administrators. While replication of files could improve throughput, the overhead of creating, updating, deleting, and managing replicas could undesirably increase response time. As a result, replication must be carefully implemented so that the overall overhead is tolerable and performance does not suffer significantly.

DORA will address all these issues and dynamically manage file assignment as the access patterns change over time. Therefore, DORA is better suited for real world applications as it can evolve with changing users' demands and can be reconfigured online.

4.3 Deliverables

As mentioned before, DORA will be simulated using Matlab software. For benchmark purposes, several other dynamic algorithms such as C-V and CM will be simulated. They will run over ten non-RAIDed homogeneous physical hard disks, which will also be simulated in Matlab. In addition, the workload will be synthetically generated.

Moreover, this research will accomplish the following: 1.) Simulate a dynamic algorithm that takes into account the changing heat of files, 2.)

Develop a replication model that will address the issues mentioned in Section 4.2 and further improve performance and throughput of users' requests, and 3.) The simulation of DORA in various workload conditions will show that its performance will hold regardless whether or not the workload assumption holds true.

4.4 Timeline

- *Paper reading: (11 Feb 2008 – 11 March 2008)* In this period, extensive reading of the relevant topics and references will be conducted.
- *Simulator construction: (12 March 2008 – 1 April 2008)* During this time a simulator will be developed for DORA. Several other algorithms will be implemented for comparison.
- *Experimental: (2 April 2008 – 29 April 2008)* Using a synthetically generated workload, intensive experiments will be performed on DORA and other related algorithms.
- *Paper writing: (30 April 2008 – 9 May 2008)* Write an IEEE format technical paper to present our research and its result.

5. Conclusions

Fast response time is a technology factor that end-users demand. Considerable research has been performed to find better ways to arrange data such that fast response time can be achieved [1][4][5]. SOR effectively increased file search efficiencies over other published static algorithms. However, DORA is better suited for a more realistic environment where files' popularity is constantly changing.

References

[1] Tao Xie, "SOR: A Static File Assignment Strategy Immune to Workload Characteristic Assumptions in Parallel I/O Systems", *IEEE, ICPP*, 2007.

[2] H. Huang, W. Hung, and K.G. Shin, "FS2: dynamic data replication in free disk space for improving disk performance and energy consumption," *Proc. 12th ACM SOSP*, pp. 263-276, 2005.

[3] W.W. Hsu, A.J. Smith, and H.C. Young, "The automatic improvement of locality in storage systems," *ACM Transactions on Computer Systems*, Vol. 23, Issue 4, pp. 424- 473, 2005.

[4] P. Triantafillou, S. Christodoulakis, and C. Georgiadis, "Optimal data placement on disks: a comprehensive solution for different technologies," *IEEE Trans. Knowledge Data Eng..*, Vol. 12, Issue. 2, pp. 324 - 330, 2000

[5] Lin-Wen Lee, Peter Scheuermann, and Radek Vingralek, "File Assignment in Parallel I/O Systems with Minimal Variance of Service Time," *IEEE TRANSACTIONS ON COMPUTERS*, VOL. 49, NO. 2,, pp. 127 -140, 2000

[6] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, and D.A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, Vol. 26, No.2, pp. 145 -185, 1994.

[7] S.H. Lim, Y.W. Jeong, and K.H. Park, "Interactive Media Server with Media Synchronized RAID Storage System," *ACM*, NOSSDAV '05, June 2005.

[8] R.L. Graham, "Bounds on Multiprocessing Timing Anomalies," *SIAM Journal Applied Math*, Vol. 7, No. 2, pp. 416 – 429, 1969

[9] M. Karlsson and C. Karamanolis, "Choosing replica placement heuristics for wide-area systems," *Proc. 24th Int'l Conf. Distributed Computing Systems*, pp. 350 - 359, 2004.

[10] T. Loukopoulos and I. Ahmad, "Static and adaptive data replication algorithms for fast information access in large distributed systems," *Proc. ICDCS*, pp. 385 - 392, April 2000.

[11] P.Scheuermann, G. Weikum, and P. Zabback, "Dynamic File Allocation in Disk Arrays" *ACM SIGMOD Record*, Vol 20, Issue 2, pp.406-415, 1991