A Load Balancing Tool Based on Mining Access Patterns for Distributed File System Servers

Alexandra Glagoleva
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
aglagol@almaden.ibm.com

Archana Sathaye
Department of Mathematics and Computer Science
San Jose State University
San Jose CA 95192
sathaye@mathcs.sjsu.edu

Abstract

In this paper we present a new web based Distributed File System server management tool to perform load balancing across multiple servers. The Distributed File System from Distributed Computing Environment (DCE DFS) is a collection of many file systems mounted onto a single virtual file system space with a single namespace. The tool is based on rule-based data mining techniques and graph analysis algorithms. The data mining procedures identify DFS file access patterns and the graph analysis and statistical information relocates the filesets between different file servers. We demonstrate our tool on data collected for five months on DFS servers in a production environment. Experiments with this data show that our load balancing tool is useful to file system administrators to monitor, evaluate DFS state and to make intelligent decisions about file system transfers in order to balance the access request load on "read-write" filesets across DFS servers.

1. Introduction

Load balancing for distributed systems represents mapping or remapping of work to different processors with the intent of assigning each processor an equal amount of work. The heaviest use of load balancing techniques is found in the domain of distributed systems. However, most of the work is done on computational tasks and not in the storage systems area [4]. Distributed File System technology from Distributed Computing Environment (DCE DFS) [15] developed at IBM Transarc Lab provides a user with the ability to store and access data at remote sites, similar to the techniques used with Network File System (NFS) [16]. DCE DFS extends the view of a local, and therefore limited in size, file system to a distributed file system of almost unlimited size. DCE DFS is a collection of several file systems mounted onto a single virtual file system space with a single namespace. The end user has direct access to all files in this distributed file system without knowing where the physical files reside. Placing file systems onto different servers in order to provide the optimal service for the end users, as well as optimize the use of available resources, is *load balancing of DFS file servers*.

Load balancing of data is already more efficient in DFS than in standard nondistributed file systems. One reason is the use of replication, which allows for "read-only" filesets to be replicated on multiple machines. Requests for files from frequently used "read-only" filesets are then spread across different machines, preventing any one of them from becoming overburdened with data requests. Fileset characteristics in DCE DFS are also beneficial for handling load balancing tasks. DCE DFS filesets are typically smaller than standard UNIX filesets. DCE aggregates can accommodate multiple filesets for flexible disk usage, and filesets can be moved between aggregates on different file server machines completely under cover without a user ever knowing about such moves.

The goal of this paper is to present a new system administrator's tool for managing "read-write" filesets across DFS file servers, therefore balancing access requests load on them. Our tool employs data mining techniques and graph theory algorithms to accomplish the desired results of improved work load distribution on DFS files servers. The data mining procedures generate association rules identifying inherent file access patterns, while graph analysis help relocation decisions and recommend fileset transfers.

The Mined Access Patterns DFS (MAPDFS) tool extends and improves load balancing techniques currently present in DFS by augmenting them with an improved management of "read-write" filesets (in addition to "read-only" filesets). MAP DFS tool is designed to make intelligent decisions on mapping "read-

0-7695-1435-9/02 $17.00 (c) 2002 IEEE

write" filesets to multiple DFS file servers. The tool was tested on real access request trace data collected at IBM Almaden Research
Center site in San Jose over a period of five months beginning December, 1999 until May, 2000.

2. Related Work and Background

In this Section, we give a brief overview of DCE DFS organization and architecture, some existing load balancing techniques, data mining approaches and, graph analysis.

2.1. Distributed File Service

DCE DFS is a distributed client-server application, built on the underlying DCE services. It takes full advantage of both the DCE services (remote procedure calls, security and directory services) and the distributed computing model itself. DFS manages information in the form of file systems. DFS data units are organized as follows (shown progressively from smallest to largest):

- *Files and Directories:* A file is a unit of user data. Files can be organized into directories. Directories include files and other directories as part of a hierarchical tree structure.
- *Filesets:* A fileset is the smallest unit of DFS administration. A fileset is a subtree of files and directories, no larger than a disk or partition. A fileset is a convenient grouping of files for administrative purposes, e.g. files pertaining to a particular project can be grouped on the same fileset.
- *Aggregates:* An aggregate is a unit of disk storage, similar to a disk partition. It is also a unit of fileset exporting. Aggregates can contain one or more filesets.
- *Servers:* A collection of aggregates resides on a DFS server.
- *Cell:* All DFS servers together constitute a DFS cell.

We ignore the DFS aggregate abstraction because it adds no value to the file access pattern analysis and final decision making process on fileset transfers between DFS file servers.

2.2. Load Balancing

Load balancing techniques are generally widely employed in the domain of distributed systems. However, most of their applications work on redistributing the work load between multiple processors to speed up computational tasks and not in the storage systems area.

Among different existing load balancing strategies are *bidding* [13], *drafting* [9], *random* [5] and *gradient* [8] load balancing algorithms. *Bidding* and *drafting* techniques are based on two main approaches to load balancing: sender initiated and receiver initiated. In sender initiated strategies an overloaded processor initiates the balancing process [13]. In receiver initiated algorithms the lightly loaded processors attempt to "draft" more work from those heavily loaded [9]. Gopinath et. al. [6], presents a hybrid approach that combines both bidding and drafting algorithms to improve general response time and communication overhead. Applying either bidding or drafting techniques to distributed file systems in a straight forward has several drawbacks. First, overloaded servers may select the same lightly loaded server, thus creating an overloaded situation on it with subsequent over migration of filesets and thrashing. If different file servers send out bid or draft requests without evaluating access patterns of a potential receiving server first, than the successful results of balancing the workload are temporary, leading to possible constant moving of filesets back and forth between the same servers. In the *random* strategy [5], the sender randomly selects the destination among under loaded targets, and in the *gradient* [8] approach the balancing is performed between the immediate neighbors to maintain an even work distribution. Both these approaches share the same major drawback –thrashing – due to lack of informed and efficient strategy for choosing the target nodes in the distributed system. Identifying intelligent patterns of mapping workload to process/store units, is an improvement to the above mentioned drawbacks of load balancing in design of the MAPDFS tool.

2.3 Data Mining Technique

Data mining problems are usually divided into three main categories [12]: classification, sequence and association. The first one - classification - partitions data into disjoint groups. Second type - sequence - delivers the expected sequence of the studied items. Association rules techniques determine correlations within a set of items. Practical solutions to all three categories are mapped onto a unified framework in Agrawal et al [1]. The problem of finding association rules among items was first formally presented by Agrawal et. al., [2]. Later, various algorithms were proposed as solutions to this problem including [3],[7]. These algorithms follow

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

a similar approach to finding association rules by first identifying "large" itemsets, and further generating association rules only from "large" itemsets. The number of transactions that contain a specific itemset determines "support" of that itemset among the overall data. A user inputs a minimum threshold number for the confidence and support levels respectively, which are used later to produce association rules of specific granularity. An itemset is labeled "large" if its support is greater or equal to the user defined threshold. Otherwise, an itemset is put into a "small" itemset category and ignored

MAPDFS implements the *Effective Hash-Based Algorithm* proposed by Park et.al., for mining association rules to produce access pattern rules, [10]. The task of determining "large" itemsets from a huge number of "candidate" itemsets in early iterations is the crucial factor for the overall data mining performance. The *Effective Hash-Based Algorithm* addresses this performance bottleneck by trimming the transaction database size of "candidate" itemsets early and, therefore significantly reducing computational time at all later stages of iterations.

2.4. Graph Analysis

Many real world problems of practical interest can be modeled as graph based component finding and coloring problems. To find connected components of a graph we need to traverse it. Traversing a graph means visiting all of its vertices in some systematic order. Depth First Search (DFS) and Breadth First Search (BFS) are two well known traversal techniques. We chose to implement Depth First Search (DFS) as a way of traversing graphs to identify connected components in our application.

The general graph coloring problem involves forming a graph with nodes representing items of special interest to the application. In our case, it is the filesets on the DFS server that are connected through non transient patterns of mined association rules. An edge on the graph, represents each file server and connects two incompatible" items. The coloring problem lies in assigning a color to each item so that every vertex in the incompatible pair is assigned a different color. Graph coloring problem is formally NP-hard for general graphs and is therefore intractable in the worst case, a number of heuristic techniques have been developed in order to solve it for most practical purposes. Our tool implements a heuristic algorithm proposed in Rosen [11].

3. MAPDFS – Load Balancing Tool

MAPDFS is tool designed for DFS system administrators to monitor the work load on DFS file servers with respect to the access request work distribution of read-write filesets across the DCE cell. The MAP DFS load balancing process goes through four main steps:

- DFS cell monitoring that reads and statistically analyzes the load on each server;
- Mining association rules from previously collected data;
- Combining and analyzing the first two steps to make recommendations on fileset transfers between servers ;
- Displaying / logging final recommendations for fileset transfers.

The next four subsections go describe the load balancing process step by step.

3.1. Cell State Monitor

The first step in balancing the load on DFS file servers is to take a snapshot of the current state of the DFS cell. Each "read-write" fileset access request number is pulled from raw DFS screen dumps along with the server name and the fileset name, and stored in a file with a timestamp to record the time of the cell's snapshot. After this operation is completed, we can apply statistical analysis to determine which servers are overloaded and which ones are underutilized. A user inputted threshold parameter is used to provide flexibility to the system. We will separate all DFS servers into the following three groups and label them according to their load level: underutilized servers will be called *TO* servers, overloaded servers *FROM* servers, and servers remaining within the user specified threshold *STAY* servers. The total load of access request hits of a file server is the sum of access request numbers of all its filesets within a certain time frame.

One of the MAP DFS input parameters is a *Threshold* value. This parameter is used in the process of separating *TO* servers from *FROM* servers from the "normal" state STAY servers. The Threshold parameter identifies percentage of the margin of freedom that is given to file servers to deviate from the statistically determined "normal" work load level for the current snapshot of the DFS cell. For each DFS cell snapshot we calculate Lowerbound and Upperbound values of the file server hit

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

rates which are used to partition servers into three groups.

UpperBound = "Normal" State + Threshold
LowerBound = "Normal" State – Threshold

TO servers fall below the calculated LowerBound value. *FROM* servers have the number of hits above the UpperBound value. Under ideal conditions all file servers would be marked as *STAY* servers, and therefore require no load balancing.

3.2. Mining Association Rules

In this section we describe the process of identifying a candidate fileset, or a group of filesets to be moved from each of the overloaded *FROM* servers. One logical choice in this case would be to take the heaviest fileset from the *FROM* server and move it away.

This move accomplishes the immediate goal at hand of alleviating the load on this *FROM* server. However, this approach does not produce any long term strategy in managing DFS filesets across the cell, specifically in the way that would minimize the need for frequent fileset transfers back and forth between the same servers in the future.

Our solution to this problem lies in using the knowledge of the past access requests activity to uncover underlying file access patterns and sequences that dominate within each server as well as across the DFS cell. Our data mining procedure in MAPDFS identifies file access patterns and sequences, separates and breaks them apart to produce a schema for mapping filesets to servers in a way that avoids creating interdependencies in access requests between filesets on any one single file server. In other words, our algorithm avoids directing all file access hits of some running process to the same file server instead of spreading the load between multiple peer servers and is thus independent of temporary sporadic bursts of file access activity. Our tool also compensates for inevitable border line and out of bound cases in daily file access request patterns, therefore extracting isolated and short lived spikes of requests for some filesets from the list of final fileset transfer recommendations. For example, the rules for a system with two servers and seven file sets are as follows:

- single server rules forfilesets of the same server:
  Server 1: Fileset_1 → Fileset_2
  Fileset_3 → Fileset_4
  Server 2: Fileset_6 → Fileset_7
- cross server rules for filesets from different servers:

Fileset_1; Server_1 → Fileset_6; Server_2
Fileset_2; Server_1 → Fileset_7; Server_2

3.2. Final Recommendations on Fileset Transfers

This is the final stage in the load balancing process. At this stage we need to combine knowledge gained from the first two steps – statistical analysis of the DFS cell and conclusions drawn from studying mined association rules, and come up with a final decision on what group of filesets are recommended for a transfer from *FROM* servers, and which *TO* or *STAY* servers are currently the best target places.

We combine depth first search, graph coloring and statistical analysis to determine, evaluate and, finally, make a decision on fileset relocation recommendations that would consider as part of the goal minimizing the amount of file transfers within the cell to reduce the administrative time and system overhead.

After we determine which file servers are currently overloaded with work, how do we decide intelligently on fileset relocations? First we check, which related filesets of the overloaded server, are also participants in the access pattern association rules. To better model these fileset interdependencies within each file server, we construct a graph object where filesets identified by data mining association rules become vertices of the graph, and the association rules represent the edges. Next we employ the depth first search graph traversing technique to examine our graph object in order to reveal all its connected components. Each connected component represents a group of transitive relations between rule-present filesets of that server.The final step in making a decision on which fileset or a group of filesets are to be moved from this server to alleviate its access request load. *Vertex coloring* approach was chosen to determine this step of the load balancing process. *Vertex coloring* is an assignment of labels of colors to each vertex of a graph such that no edge connects two identically colored vertices [14].

MAPDFS implements vertex coloring with coloring schemas containing between 2 and 5 colors to simulate the average number of simultaneous active file servers in the DFS cell. The main steps MAPDFS takes to color large graph of filesets are based on the heuristic algorithm Rosen [11] are given below.

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

- Assign "color 1" to the vertex with highest degree.
- Also assign "color 1" to any vertex that is not connected to this vertex.
- Assign "color 2" to the vertex with the next highest degree that is not already colored.
- Also assign "color 2" to any vertex not connected to this vertex and that is not already colored.
- If uncolored vertices remain, assign "color 3" to the uncolored vertex with next highest degree and other uncolored, unconnected vertices.
- Proceed in this manner until all vertices are colored.

The created color schemas for each component on the server are used to make a final selection on filesets to be transferred to currently underused servers to achieve a better load balance across the DFS cell. Transferring groups of filesets by color separates the inherent dependencies between the transferred filesets and hence distributing the spikes of filesets activity across the cell. We describe the details of the algorithm to evaluate the steps in deciding fileset relocation in [17].

The decision on whether a "color" of filesets can be moved to a particular server is made after evaluating answers to the two following questions:

- Does the collection of Cross Server Association Rules contain an entry that connects one of the filesets of this "color" to a fileset on the target server?
- Does the move of the current "color" of filesets push this server beyond the *Threshold* parameter into the *FROM* servers category?

If the answer to any of the two questions is YES, the server under evaluation is passed, and the search for a more suitable target place continues until all choices in the DFS cell are exhausted. Such precise fileset transfer restrictions make sense for obvious reasons in the case of the second test question. In case of the first question, we need not create additional fileset interdependencies on the target file server, because it is the exact problem we are trying to solve on the source server. In order to verify that the new mapping solution of DFS filesets to file servers is indeed accomplishing the task of better work load distribution , MAPDFS computes a new round of statistical analysis of the access request load on file servers after the virtual completion of fileset transfers. The transfers are virtual at first because we want to know the degree of possible improvements in the system, and based on the level of improvement, the tool decides the real need to carry out recommended changes on DFS file servers.

3.4. Presentation and Defined Interfaces

MAPDFS is designed to be used by system administrators to monitor and adjust the load of file access requests on DFS servers. The application takes the following input parameters:

- <input data file>: name of the file containing snapshot of the DFS cell with each line of the form: *fileset_name ; server_name ; number_of_accesses*
- <output file>: name of the file or directory where the output result will be stored.
- <threshold>: number <0 ; 100> without a percent sign.
- <#_of_heavy_fs> :an integer N; used to record N heaviest filesets on the server.
- <rules_file>: name of the file containing data mining association rules

MAP DFS can be run in one of the three ways: (1) a command line process with all the above mentioned parameters supplied as command line arguments; (2)a scheduled job once a day, a week or a month depending on the needs of a particular DFS environment; (3) a web-based interface to manage DFS load balancing tasks remotely through a MAP DFS servlet. The general design of the web interface is shown in Figure 1.

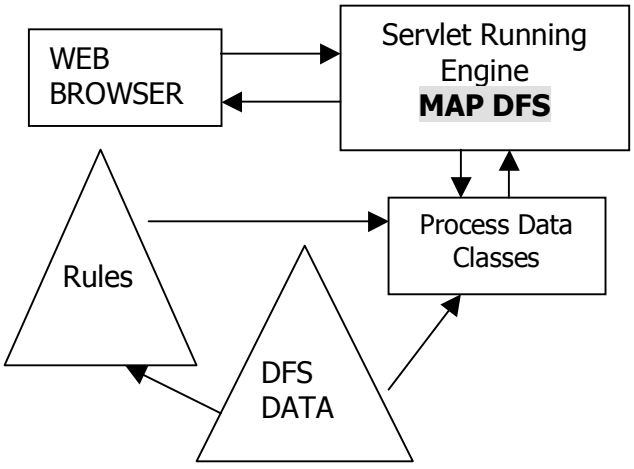

Figure 1: Design of MAPDFS with a Web Interface

4. Experiments and Results

The fileset access data was collected on DFS file servers at IBM Almaden Research Center in San Jose. The data was recorded between December 1999 and April 2000.

The following statistical analysis of the workload distribution on DFS file servers is done by MAPDFS:

- Sample Arithmetic Mean $X = 1/n \Sigma x(i)$, where $i \in 1, n$ - Mean value of hits for servers in the cell.
- LowerBound and UpperBound values for the server access request work load based on the Threshold value. LowerBound = Sample Arithmetic Mean - Threshold and UpperBound = Sample Arithmetic Mean + Threshold
- Mean Absolute Deviation $S = 1/n \Sigma | x(i) - X |$ for the cell, where $i \in 1, n$ and X is Sample Arithmetic Mean.
- Coefficient of variation COV = S / X, where S is Mean Absolute Deviation and X is Sample Arithmetic Mean

In the experiments, *Mean Absolute Deviation* and *Coefficient of Variation* are calculated and recorded before and after the load balancing process. These two statistical measurements evaluate the extent of disparity between studied objects, in our case, between access request loads on different file servers. The *Coefficient of Variation*, being a ratio between the *Mean Absolute Deviation* and the *Sample Arithmetic Mean*, reflects the degree of variability in the workload among DFS servers. Coefficient of variation can be expressed as a quotient: S/X, or as a percentage: $S/X * 100\%$. A zero COV value would indicate no variability at all. *Mean Absolute Deviation* was chosen over *Sample Standard Deviation* because of the nature of the data being used in calculations. A DFS server gets hundreds of millions of hits during each window frame and hence taking the square roots of such large numbers proved to be impractical and error prone.

Multiple sets of access request association rules were produced during the experiments with the data mining procedure. The extensive data gathering period gave us an opportunity to run data mining procedures on different quantities of data in order to verify the best data collection strategy for file access information mining in the future. Significant changes in quantities of initial data yield corresponded shifts in resulting rule sets. Close examination of the collected fileset control data reveals possible tremendous variations in numbers of accesses between filesets. Some filesets are accessed less than 100 times during a two hour window, while others sustain more than 100,000,000 accesses within the same time frame. The high level of "noise" in the data lead to a practical decision to filter out the most infrequently used filesets from future load balancing efforts.

To compare results of mining association rules from different volumes of data, we further separated all the data collected over the five months period into the five categories named R1, R2, R3, R4, R5 listed in the table below. Where Ri represents the rules for data collected for 'i' months.

Table 1: Transaction Data Groups for Rule Mining

	Dec, 1999	Jan, 2000	Feb, 2000	Mar, 2000	Apr, 2000
R1	X				
R2	X	X			
R3	X	X	X		
R4	X	X	X	X	
R5	X	X	X	X	X

Our experiment investigated how the higher volume of transaction data affects the rule mining process. We observed that the number of association rules produced per set decreases as data mining is performed over larger volumes of records and as the minimum data mining support value goes up. This is an obvious result reflecting increasing variations in resident filesets of a server and across the DFS cell over a longer period of time. In fact, mining association rules from data gathered over a five month period produces mostly results with 0 rules in the set. The reason for such results can be explained by the fact that the variations and turn over in the active fileset population residing on file servers is too great for the five month period worth of data to pass successfully through a minimum data mining support level. From the results obtained with the current DFS setup at IBM Almaden Research Center, we conclude that there is no need to collect and mine fileset access data beyond the four months period, as R5 set of rules yields no usable information.

Our tool provides system administrators choice between the two load balancing strategies. The strategies correspond to the two existing types of fileset interdependencies. We call them *sporadic* and *stable* dependencies. *Sporadic* fileset dependencies are revealed through mining access request data over relatively short time periods – a day or a week. *Stable*

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

fileset dependencies are the long lasting access patterns that can be identified when we mine data collected over several months and more. *Sporadic* and *Stable* are the two separate, but complementary ways in which load balancing tasks can be performed. Combining the two ways together and carefully manipulating the granularity of rule sets, a system administrator can manage long term persistent filesets on a monthly or even quarterly basis with *Stable* rule sets which incorporate extensive knowledge of persistent data patterns, and further augment that schedule balancing the load of more recent additions to the fileset community through association rules obtained from the last month's, week's or even yesterday's data, reflecting *Sporadic* access patterns.

Statistical analysis of the DFS file servers' state before and after the load balancing work provide quantitative ways to present the final results for the MAP DFS tool. The goal of our tests is to determine the optimal length of time for data collection, which would produce the most efficient set of association rules used for DFS load balancing tasks. The tested rule sets are: R1, R2, R3 and R4. We are not going to consider rule sets R5 for the reasons explained earlier. In the tests, the threshold parameter was set at 25%, a number empirically determined to be optimal considering the average number of online DFS servers and their typical work load, and the data mining support value set at 95%, accounting for high frequency filesets only.

The results of the tests are presented in the charts below. *Mean Absolute Deviation (MAD)* and *Coefficient of Variation (COV)* are calculated and recorded before and after the load balancing process. These two statistical measurements evaluate the extent of disparity between studied objects, in our case, between access request loads on different file servers. The lower the values of *Mean Absolute Deviation* and *Coefficient of Variation* are the more uniform the workload is across DFS file servers. A zero value for the *Coefficient of Variation* would indicate an absolutely equal distribution of work between servers. An average reduction in the cell *Mean Absolute Deviation (MAD)* in millions of access requests after the load balancing process for rule sets and the average percent of decrease in the cell*Coefficient of Variation* (*COV)* obtained from one, two, three and four months of data (R1, R2, R3, R4) is shown in Table 2.

As we can see from the charts, tests showed no further improvements toward a better work load distribution with the application of R4 rule set. Rules R4 are mined from large volumes of data (four months). The resulting associations are the most *Stable* and persistent throughout the DFS cell, but they are also too small in number to carry sufficient access pattern information to be used successfully for load balancing tasks in the high volume environment like a DCE cell. Looking at the rule set R1, we see some improvements in the work load distribution. But, the most efficient results are achieved with an application of rule sets R2 and R3. In case of R2 experiments MAD was reduced on average by 14.2 millions of accesses. The cell COV went down by an average of 20.5 %. In analyzing R3 test results, we observe that the initial fileset access request data recorded in February, 2000 showed good uniformity in the server work load for the larger part of the month and we were able to decrease the cell MAD by 12.4 millions of accesses. The resulting average overall COV improvement with R3 rules set application was 6 %, which came on top of already present good initial work load distribution.

Table 2: Average Decrease in MAD and COV

Rule Sets	MAD in Millions of Access Request	Percent COV
R1	3.7	5
R2	14.2	20.5
R3	12.4	6
R4	0	0

5. Conclusions and Future Work

In this paper we presented a system administrator's tool (MAP DFS) to perform load balancing tasks for Distributed File System (DFS) from Distributed Computing Environment (DCE). Load balancing of data is already more efficient in DFS than in standard nondistributed file systems due to the use of replication, which allows for "read-only" filesets to be replicated on multiple machines. We developed a tool to improve existing load balancing techniques by managing "read-write" filesets in addition to "read-only" filesets, therefore preventing any one machine from becoming overloaded with data requests to any type of filesets in the system. Our implementation employs data mining techniques and graph theory algorithms to accomplish the desired results of improved work load distribution on DFS files servers. The MAP DFS tool was designed to make intelligent decisions on mapping "read-write" filesets with sporadic and stable dependencies to multiple file servers. The tool was tested on real file access request trace data collected at IBM Almaden Research

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

Center in San Jose over a period of five months (December, 1999 - May, 2000).

Our experiments showed that there is no need to collect and mine fileset access data beyond the four month period as it yields no usable information. Statistical analysis of the DFS file servers' state before and after load balancing process with the threshold parameter identifying groups of *TO, FROM* and *STAY* servers set at 25% showed that the most efficient results are achieved with an application of association rules obtained from mining data collected over periods of two to three months. The actual percentage of improvements in cell's Coefficient of Variation when balancing the file server load received from sample test runs varied from 6% to 20.5% depending on the initial uniformity of the work load .

Future work could examine the following issues. The final percentage of improvements on the fileset access request distribution could be increased in the future by setting an additional threshold parameter in fileset transfer validation procedures. A set of "cross server" rules was used during validation tests to determine if a fileset that is a candidate for a move carries any dependencies to the filesets residing on the target server. If the answer is true, the candidate fileset was passed. Setting a threshold for the number of rules involving a candidate fileset and a target server in a "cross server" rules set will allow to validate transfers for filesets that fall below the threshold value, therefore increasing the final number of fileset moves. Also, the threshold parameters used in the application could be changed to adjuste dynamically to the present conditions on the DFS servers, lowering or raising their values depending on the current number of online file servers and their work load distribution.

5. References

[1] R. Agrawal, T. Imielinski, and A. Swami, "Database Mining: A Performance Perspective", *IEEE Transactions on knowledge and Data Engineering, Special Issue on Learning and Discovery in Knowledge-Based Databases,* 5(6), December 1993.

[2] R.Agrawal, R.Srikant,"Fast Algorithms for Mining Association Rules", *In Proc. of the 20th International Conference on Very Large Databases*, Santiago, Chile, September 1994

[3] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases", *Proceedings: ACM SIGMOD International Conference on Management of Data*, Washington D.C. May 1993, 207-216

[4] Pallab Dasgupta, A.K. Majumder, and P. Bhattacharya, "V_THR: An Adaptive Load Balancing Algorithm", *Journal of Parallel and Distributed Computing* 42, 1997, 101-108.

[5] D.L.Eager, E.D.Lazowska, J. Zahorian, "Dynamic Load Sharing in Homogeneous Distributed Systems", *Technical Report, University of Washington*, October 1984.

[6] Prabha Gopinath and Rajiv Gupta, "A Hybrid Approach to Load Balancing in Distributed Systems", *Proceedings: Symposium on Experiences with Distributed and Multiprocessor Systems (SEDMS II)*, USENIX, 133-148.

[7] M. Houstma, A.Swami, "Set Oriented Mining of Association Rules", Technical Report, IBM Almaden Research Center, San Jose, California, October 1993.

[8] F.C.H. Lin, R.M.Keller, "Gradient Model: A Demand Driven Load Balancing Scheme", *In Proc. Of International Conference on Distributed Computing Systems*, 1986, pages 329 – 336.

[9] L.M. Ni, C.-W. Xu, T.B. Gendreau, "A Distributed Drafting Algorithm for Load Balancing", *IEEE Transactions on Software Engineering*, SE-11(10),October 1985,1153-1161.

[10] Jong Soo Park, Ming-Syran Chen, Phillip S.Yu, "An effective hash-based algorithm for mining association rules", *In Proc. Of the ACM-SIGMOD Conference on Management Data,* San Jose, California, May 1995.

[11] Kenneth Rosen, *Discrete Mathematics and Its Applications*, McGraw Hill Higher Education, 1998.

[12] Yucel Saygin, Ozgur Ulusoy, "Exploring Data Mining Techniques For Broadcasting Data in Mobile Computing Environments*", Bilkent Univ.*, Turkey

[13] J.A.Stankovic, I.S.Sindhu, "An Adaptive Bidding Algorithm for Processes, Clusters and Distributed Groups", *In Proc. Of 4th Int. Conference on Distributed Computing Systems, San Francisco, CA*, , IEEE, May 1984 ,pages 49-59.

[14] G.Tinhofer, E.Mayer,H.Holtemeier, M.Syslo, R.Albrecht, *Computational Graph Theory*, Springer-Verlag Wien New York, 1990

[15] An Overview of DFS http://www.transarc.com/Library/documentation/dce/1.1/dfs_admin_gd_1.html.

[16] An Overview of NFS, http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/aixbman/commadmn/nfs_intro.htm.

[17] Alexandra Glagoleva and Archana Sathaye," Load Balancing Distributed File System Servers: A Rule Based Approach" , *Proceedings of 13th International Conference on System Research, Information and Cybernetics*, Baden-Baden Germany, July 30, 2001.