Enhancing and Optimizing a Data Protection Solution

Ludmila Cherkasova, Roger Lau
Hewlett-Packard Labs
Palo Alto, CA 94304, USA
{lucy.cherkasova, roger.lau}@hp.com

Harald Burose, Bernhard Kappler
Hewlett-Packard,Technology Solutions Group
Schickardstrasse 25, Boeblingen, Germany
{harald.burose,bernhard.kappler}@hp.com

***Abstract*— Analyzing and managing large amounts of unstructured information is a high priority task for many companies. For implementing content management solutions, companies need a comprehensive view of their unstructured data. In order to provide a new level of intelligence and control over data resident within the enterprise, one needs to build a chain of tools and automated processes that enable the evaluation, analysis, and visibility into information assets and their dynamics during the information life-cycle. We propose a novel framework to utilize the existing backup infrastructure by integrating additional content analysis routines and extracting already available filesystem metadata over time. This is used to perform data analysis and trending required for adding performance optimization and self-management capabilities to backup and information management tasks.**

Backup management faces serious challenges on its own: processing ever increasing amount of data while meeting the timing constraints of backup windows could require adaptive changes in backup scheduling routines. We revisit a traditional backup job scheduling and demonstrate that random job scheduling may lead to inefficient backup processing and an increased backup time. In this work, we use a historic information about the object backup processing time and suggest an additional job scheduling, and automated parameter tuning which may significantly optimize the overall backup time. Under this scheduling, called LBF, the longest backups (the objects with longest backup time) are scheduled first. We evaluate the performance benefits of the introduced scheduling using a realistic workload collected from the seven backup servers at HP Labs. Significant reduction of the backup time (up to 30%) and improved quality of service can be achieved under the proposed job assignment policy.

I. Introduction

Unstructured data is the largest and fastest growing portion of most enterprise's assets, often representing 70% to 80% of online data. As companies implement enterprise-wide content management (such as information classification and enterprise search), and the volume of data in the enterprises continues to increase, establishing a data management strategy requires an insight into data trends that is currently unavailable. While many CIOs are concerned solely with issues of information management, such as classification and search, IT departments must grapple with the difficult questions of data protection, resource allocation, and management. Examples include:

- How many different types of data do we have and how quickly are they growing?
- What are the most popular data types (and the applications that generate them)?
- How much new data is added/modified/deleted over time?
- What are the rates of growth? What are the overall storage requirements and trends over time?

Currently, enterprises have little visibility into the historical patterns and behaviors of their data, making it difficult (if not impossible) to identify these trends. Yet, answering these questions becomes essential for designing and configuring the resources and application constraints that IT professionals deal with: application consolidation, resource allocation, storage sizing, quota management, backup system configuration, and so forth. System administrators struggle to manage and scale their backup infrastructure to protect their data and increase data availability. Backup management faces serious challenges: processing the ever increasing amount of data while meeting the timing constraints of backup windows requires adaptive changes in scheduling incremental and full backups during 52 weeks a year. Analysis of the backup data, data dynamics, and trends is useful to optimize the backup management itself. Efficiency of solving capacity planning problems, optimizing scheduling and resource allocation tasks significantly depends on our ability to analyze historic data and extract trends. Answering data trending questions requires a historical summary of metadata that reflects sufficient aspects and details of the data over time. Although we are currently building a research prototype to collect metadata from all of the information across enterprise [18], this information can also be gathered from existing sources, such as enterprise backups.

HP Data Protector is HP's enterprise backup offering. Currently, Data Protector maintains course-grained metadata of each backup period which can be used to provide data points for initial metadata analysis and trending. Furthermore, upcoming versions of Data Protector will include a robust data analysis pipeline, providing hooks for more complex data analysis and metadata collection. The backup infrastructure is a promising place to integrate additional content analysis routines as well as to extract already available filesystem metadata and perform data analysis using historical filesystem snapshots. Having visibility into the location of files in the corporate network and machines will be very useful in legal cases. Using the backup catalogue to prove which user had access at which time to a file and who had a copy on their desktop or laptop computer will become more important to know. Understanding the type of data located on a computer could also be used as a trigger of further investigating data on

a specific computer and performing additional content analysis of the data. Analyzing the metadata allows the identification of a suspect system and triggering a restore of old data to a temporary (potentially even virtual system) for the analytics to run without the employee noticing.

We revisit a traditional backup job scheduling and demonstrate that random job scheduling traditionally used in the backup tools may lead to inefficient backup processing and an increased backup time. In this work, we use a historic information about the object backup processing time and suggest an additional job scheduling and automated parameter tuning which may significantly optimize the overall backup time. Under this schedule, called LBF, the longest backups (the objects with longest backup time) are scheduled first.

In our performance study, we use a realistic workload collected from seven backup servers at HP Labs. There are significant time savings (40 min-212 min) achieved under the new job scheduling for all seven backup servers under study. The reduction of the backup time (5%-30%) depends on the size distribution of objects the backup server is responsible for. When a backup server has a significant portion of objects with a long processing time, the proposed new job scheduling is especially efficient and provides significant backup time reduction.

Typically, a backup tool has a configuration parameter which defines a level of concurrency, i.e., a number of concurrent processes (called disk agents) which can backup different objects in parallel to the tape drives. The drawback of such an approach is that the data streams from different objects are interleaved on the tape, and when data from a particular object needs to be restored there is a higher restoration time for its retrieval such data compared with a continuous, non-interleaved data stream written by a single disk agent. Using a workload analysis of 7 backup servers in HP Labs, we observe that the overall backup time is often limited by the longest job duration, which can not be further improved. In such situations, a tool can use a reduced number of disk agents to avoid additional data interleaving (or recommend a decreased number of tape drives) while still optimizing the overall backup time. In this work, we provide a variety of additional algorithms which help in automating system administrator efforts and optimizing the backup tool performance.

The remainder of the paper is organized as follows. Section II presents our framework for assessing dynamics of enterprise information assets and provides a workload analysis of the backup servers under study. Section III outlines a traditional backup tool architecture, it explains potential performance inefficiencies of the current approach, and provides a formal problem definition. A new optimized backup scheduling is introduced in Section IV accompanied by a performance study to evaluate its performance benefits. An algorithm for creating balanced backup groups is introduced in Section V. Section VI proposes an approach to minimize data streams interleaving while optimizing the backup time. A review of related work is presented in Section VII. Finally, Section VIII draws conclusions.

II. Framework for Assessing Dynamics of Enterprise Information Assets

To provide a "first-approximation" summary of unstructured information assets and their trends over time in the enterprise, we exploit historical data available from backup databases. Companies now store months to years of backup metadata online, in order to provide the ability to quickly find and recover files when they are needed. In this way, companies' backups already capture business-critical data and their evolution over time. However, backup tools and utilities are not designed to support file reporting and their metadata analysis, classification, aggregation, and trending over time. We believe that there is an excellent opportunity to fill this void by extending the backup tools to provide valuable data and metadata analysis services in an automated, representative, and fast manner.

Conceptually, backup tool functionality is built around the backup session and the objects (mount points or filesystems) that are backed up during the session. Currently, there is no direct and simple way to retrieve only the file metadata for the entire filesystem and to perform a specific analysis of these data over time. To extract the snapshot of filesystem metadata at a particular moment of time, one needs to perform a sequence of steps to retrieve a filesystem catalogue. This procedure might take several hours for a large collection of files (e.g., 1,000,000 files or more), which makes the whole process very inefficient (time- and space-wise) when such snapshots need to be extracted for analysis, building statistics and trends over longer durations of 6-18 months. The existing data structures in the backup DB do not support this large-scale metadata retrieval and data analysis.

We propose creating representative, complementary filesystem *metadata snapshots* during the backup sessions. The format of such snapshots is specifically designed to derive detailed statistics and trends, and to perform an efficient analysis to answer all of the questions raised in the previous section. Furthermore, these snapshots can be further folded into a very compact *filesystem metadata summary* which uniquely reflects filesystem evolution and dynamics over time.

Under this approach, there is an entry with metadata for each file, link, or directory. This data includes file's permissions, owner, group, ACLs, size, date, time, and a file name (with a full path). The entries are sorted in by the file name. Additionally, there is a field in each entry that represents a timestamp of the backup, denoted as the *backup_id*.

When the next snapshot is generated for the same filesystem (say, a week later), we combine them in a single summary in the following way. We merge both tables and sort them in the alphabetic order by the file name. In such a way, if a file did not change between the backups, then there are two identical entries in the table side-by-side (they only differ by their *backup_id*s). In a similar way, we can see which files got modified, or deleted, or newly introduced to the system.

For each full consecutive backup, we create a set of *metrics* that reflect the quantitative differences and dynamics of the system over time (the same characterization is used for large

file subgroups of interest: office files, text files, PowerPoint files, executables, etc):

- *Total* – the overall number of files N_t and their aggregate size S_t;
- *Cold* – the number of unchanged (cold) files N_c and their aggregate size S_c;
- *Modified* – the number of modified file N_m and their aggregate size S_m;
- *New* – the number of newly added files N_n and their aggregate size S_n;
- *Deleted* – the number of deleted files N_d and their aggregate size S_d.

This set of metrics serve as a *filesystem signature*. It represents the system size, both the number of files and their storage requirements, and efficiently reflects the system dynamics over time. The introduced filesystem signature presents the fraction of the files which are stable and do not change between the backups, as well as the *churn* in the system which is characterized by the percentage of files that are modified, added, or deleted in the system. This data collected over time is used in a *regression model* for trending analysis.

Instead of keeping multiple copies of file metadata in different backup snapshots, we create a compact summary that is representative of the overall filesystem and all its files over time. The summary contains the latest file metadata (for each file) and a set of additional fields that represent file dynamics:

- *introduction* date – the *backup_id* of the earliest snapshot that has the file;
- *counter of modifications* over time – this counter is set to 0 when a file is introduced to the system, and gets increased each time a modification to the earlier recorded metadata is observed,
- *modification* date – the *backup_id* that corresponds to the *last modification* date (initialized to *empty*),
- *deletion* date – the *backup_id* that corresponds to the date when the file was *deleted* (initialized to *empty*).

This *filesystem metadata summary* is compact and detailed at the same time: it characterizes (conveniently and uniquely) the filesystem history and evolution over time.

We've implemented a prototype of our filesystem metadata analysis module on the top of the HP Data Protector 6.0 tool [12]. Using seven backup servers in HP Labs, we performed a detailed metadata and trend analysis of the backed up filesystems over 1.5 years.

There were 570 objects[1] in the set. We classified them using the following three groups:

- *Small objects: with less than 5,000 files.* Typically (but not always), the small objects are system files. The object size (in bytes) in this category is quite diverse: from 20 KB to 587 GB. This stresses the importance of having both dimensions in the filesystem characterization: number of files and their storage requirements.
- *Medium objects: with more than 5,000 files but less than 100,000 files.* The object size in this group was in a broad range from 202 MB to 1.2 TB.
- *Large objects: with more than 100,000 files.* The largest object in the set had 4.5 million files. The object size in this category was in a range 665 MB - 750 GB.

Table I summarizes the object characterization in the analyzed collection. While each object group constitutes about one third of the overall collection, the impact of the large objects is clearly dominant: the large objects are responsible for 93.2% of all the files and 66% of all the bytes in the collection.

Object Type	% of All the Objects	% of All the Files	% of All the Bytes
Small	41%	0.3%	8%
Medium	31%	6.5%	26%
Large	28%	93.2%	66%

TABLE I
OBJECT CHARACTERIZATION IN THE ANALYZED COLLECTION.

When we compared weekly (full) backups aiming to characterize dynamics of the collection under study, we found that almost 50% of all the objects did not change between the backups. For the remaining objects the modifications are relatively small: for 95% of the objects the cold files are dominant and they constitute 90-99% of the files and the rates of modified and newly added files are in the range 1-10%. By grouping the filesystems in *cold*, *slow-changing*, and *dynamic* collections, one can optimize how often full versus incremental backups are performed for different type collections.

Figure 1 shows the profiles of the analyzed 570 objects over time. There are two lines in the figure: first line represents the object profile in the beginning of February 2009, while the second line reflects the March profile. Figure 1 (a) shows the number of files per object (sorted in increasing order) on a logarithmic scale. Figure 1 (b) presents the size of the objects in MBytes (sorted in increasing order) on a logarithmic scale. Both lines are very close reflecting a very gradual change in the studied file collection over time.

We believe that the results of the proposed workload analysis can address a broader spectrum of performance and data management optimizations. The automated analysis of the filesystem "life span" (e.g., identifying filesystems and information sources that became "cold") is useful for automated file migration in multi-tier storage systems. The analyzed file metadata and their summaries provide useful views about managed enterprise information assets, their dynamics and trends based on available file attributes and organization structure. We are experimenting on how to better present these statistics and summaries, visualize and organize trends and "highlights", as well as to provide convenient interface for additional data mining. We envision a few storage-related and information-management applications which will consume this data.

III. TRADITIONAL BACKUP TOOL AND ITS POTENTIAL PERFORMANCE INEFFICIENCIES

The functionality of a backup tool is built around a backup session and the objects (mount points or filesystems of the

[1]In this paper, we use the terms filesystem and object, interchangeably.

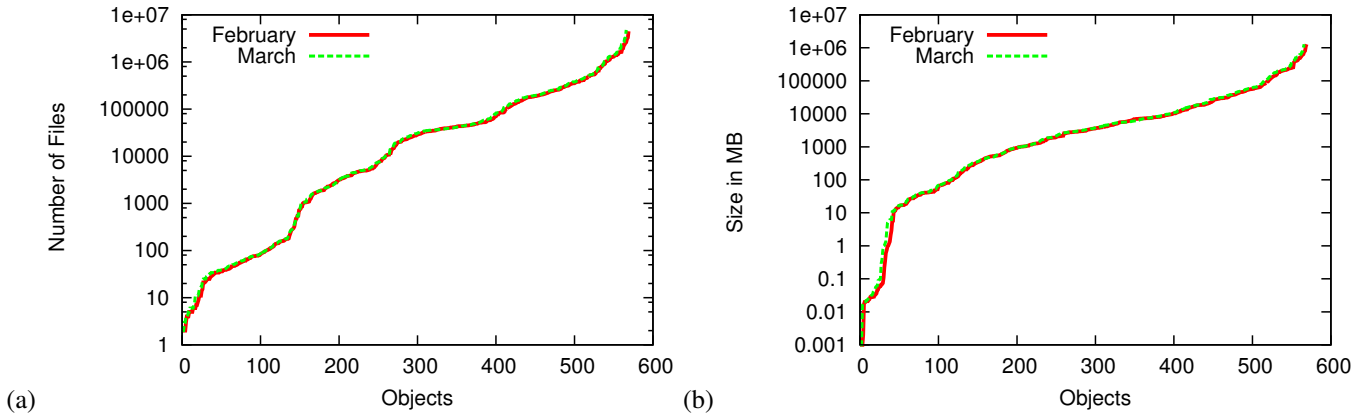

Fig. 1. Object Characterization in the Analyzed Collection during February and March of 2009: (a) number of files per object; (b) object size in MBytes.

client machines) that are backed up during the session. The traditional architecture of a backup tool which uses a tape library is shown in Figure 2.

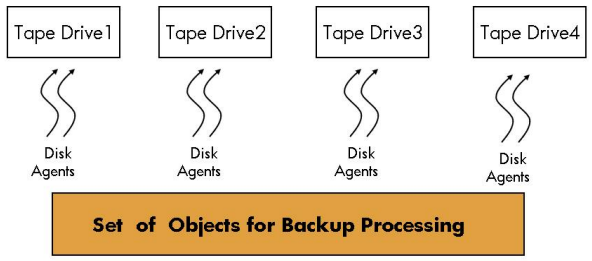

Fig. 2. Traditional Architecture of a Backup Tool with a Tape Library.

Typically, there are 4 to 6 tape drives (each solution comes with a fixed number of tape drives; it is not a parameter). Each such tape drive has a configuration parameter which defines a concurrency level, i.e., a number of concurrent processes (called disk agents) which can backup different objects in parallel to the tape drives. Traditionally, this is done because a single data stream generated by a disk agent copying data from a single object can not fully utilize the capacity/bandwidth of the backup tape drive due to slow client machines. To optimize the total backup throughput, a system administrator can configure up to 32 disk agents for each tape drive to enable concurrent data streams from different objects at the same time. The drawback of such an approach is that the data streams from 32 different objects are interleaved on the tape, and when the data of a particular object needs to be restored there is a higher restoration time for retrieving such data compared with a continuous, non-interleaved data stream written by a single disk agent.[2]

When a group of N objects is assigned to be processed by the backup tool, there is no way to define a sequence or order in which these objects are processed by the tool. Typically, any available disk agent may be assigned for processing to any object from the set, and the objects (which might represent different mount points of the same client machine) can be written to different tape drives. Figure 2 shows a typical configuration of a backup tool for processing a set of objects. There is no way to define an order in which the objects are processed by concurrent disk agents to the different tape drives. Potentially, this may lead to inefficient backup processing and an increased backup time.

Here is a simple example of such inefficiency. Let there be ten objects $O_1, O_2, ..., O_{10}$, in a backup set, and let the backup tool have four tape drives each configured with 2 concurrent disk agents as shown in Figure 2, i.e., with eight disk agents in the system. Let these objects take approximately the following times for their backup processing: $T_1 = T_2 = 4$ hours, $T_3 = T_4 = 5$ hours, $T_5 = T_6 = 6$ hours, $T_7 = T_8 = T_9 = 7$ hours, and $T_{10} = 10$ hours. If the disk agents randomly select the following eight objects, $O_1, O_2, O_3, ..., O_7, O_8$, for initial backup processing then objects O_9 and O_{10} will be processed after the backup of O_1 and O_2 are completed (since backup of O_1 and O_2 take the shortest time of 4 hours), and the disk agents which became available will then process O_9 and O_{10}. In this case, the overall backup time for the entire group will be 14 hours as shown in Figure 3 (a).

Clearly, the optimal scheduling for this group is to process the following eight objects instead: $O_3, O_4, ..., O_{10}$ first, and when processing of O_3 and O_4 is completed after 5 hours, the corresponding disk agents will backup the remaining objects O_1 and O_2. If the object processing follows this new ordering schema then the overall backup time is 10 hours for the entire group as shown in Figure 3 (b).

While we demonstrated the backup tool behaving inefficiently with a simple example, the traditional enterprise environment might have hundreds of objects for backup processing, and it will be beneficial to automate the object scheduling process. In the next section, we introduce an additional job scheduler in the backup solution which aims to optimize the

[2]In the HP Labs environment, each instance of the backup tool has 4 tape drives. Based on the results of performance tuning, each tape drive is configured with 4 disk agents. From the performed experiments, further increasing the number of disk agents per tape drive does not improve the backup throughput while it does introduce redundant data interleaving.

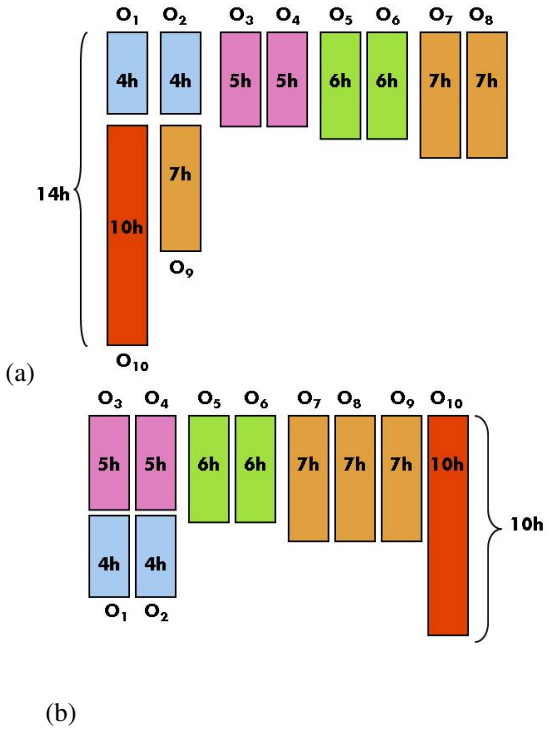

Fig. 3. The backup job assignment from the job list to concurrent processes at the tape drives: (a) random, (b) optimized.

overall backup time and helps to avoid manual configuration efforts by system administrators who try to achieve the same performance goal.

IV. LBF Scheduling to Optimize the Overall Backup Time

Typically, backup tools record useful monitoring information about the performed backups. In this work, we are mostly interested in the efficient management of full backups (i.e., when the data of the entire object is processed during a backup) and not incremental backups, which only process modified and newly added files from the object and which are typically short and light in nature.

For each backed up object, there is recorded information on the number of processed files, the total number of transferred bytes and the elapsed backup processing time. In our solution, we use historic information on duration for processing of backup jobs (the jobs which were successfully completed). One of the main questions is whether past measurements of backup processing time are good predictors of the future processing time, and whether they can be used for backup job assignment and scheduling processes.

Our historic data analysis shows that while different objects might have very different backup processing time, the processing time of the same object is quite stable over time because of gradual changes in the object size, as was shown in the previous section. Figure 4 presents historic snapshots of backup job durations from the three backup servers at HP Labs. Each figure shows reported job durations (sorted in increasing order using a logarithmic scale) of five consecutive, full weekly backups performed in February - March, 2009.

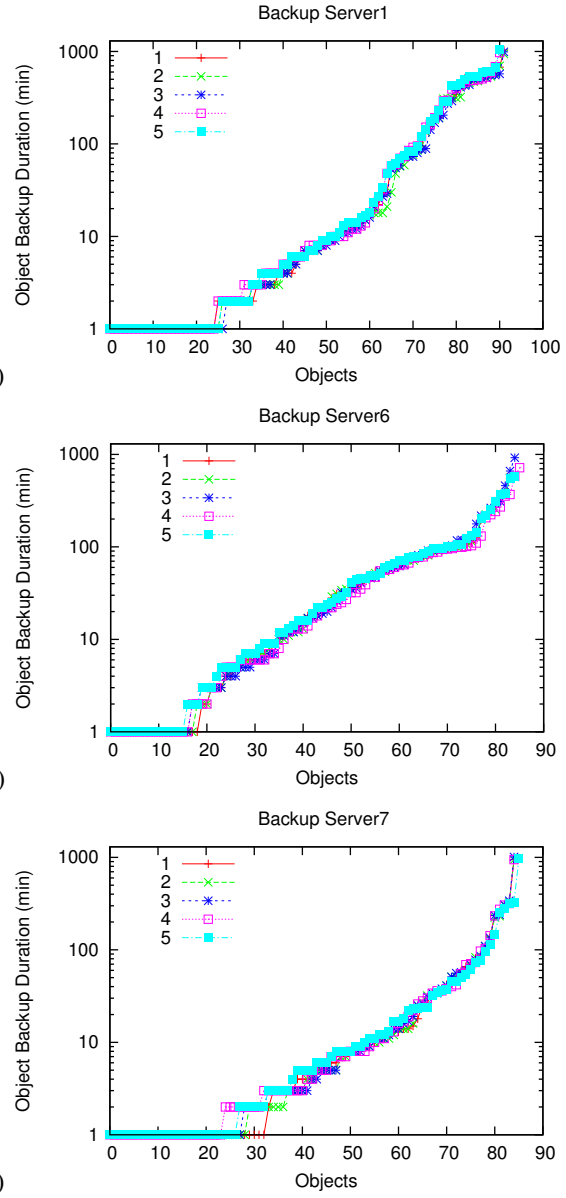

Fig. 4. Historic snapshots of the backup job durations from the five consecutive, full weekly backups performed in February - March, 2009: (a) Server1; (b) Server6; and (c) Server7.

First of all, there is a significant diversity in durations of the backup jobs as shown in Figure 4. This data is consistent with the analysis of object sizes in the previous section. There are three orders of magnitude in object processing time: some objects take only 1 min for backup while other larger objects take 10-14 hours for their backup. Another interesting observation is that there a significant number of "long" backup jobs. Figures 4 (a) and (b) show that about 25% of all the jobs performed by these backup servers are in the range of 2-14 hours. Intuitively, we expect that the additional job scheduling will be useful for these servers. However, the job duration distribution shown in Figure 4 (c) is quite different:

90% of the backup jobs processed by this server are short (less than 2 hours), there are only a few jobs in the range of 4-5 hours, and a single job that requires about 13 hours for its backup processing. This server might have limited opportunities for performance optimization via backup job scheduling compared to the two backup servers with profiles shown in Figures 4 (a) and (b).

Finally, there is a lot of stability in the historic shapshots shown in Figure 4. All the five lines of each graph are very close to each other, meaning that there is a good predictability of job processing duration over time. This monitoring data supports the use of historic information for future job scheduling.

In this work, we assume that the system administrator already uses tuned system parameters, such as the number of concurrent disk agents per tape drive, to optimize the server backup throughput. Our goal is to minimize the overall backup time processing through additional job scheduling in the backup system.

The optimization algorithms that will be introduced in this and next two sections can also be used in a *simulation mode*. When these algorithms are used in a simulation mode, they help in assessing the potential performance benefits for a given workload. Additionally, a simulation mode is useful for understanding the outcome of different *what–if?* scenarios. Often, a system administrator is interested in estimating the outcome of possible changes to the system. If a subset of client machines is moved from $BackupServer1$ to $BackupServer2$, then what would be new overall backup time for $BackupServer1$? How does this change the overall backup time of $BackupServer2$? We believe that the simulation mode of the proposed algorithms can be useful for different capacity planning and consolidation exercises.

LBF Scheduling Algorithm:

For an upcoming full backup, we use information about the job durations from the previous full backup. At this phase, an ordered list of objects sorted in decreasing order of their backup durations is created:

$$OrderedObjectList = \{(Ob_1, Dur_1), ..., (Ob_n, Dur_n)\}$$

where Dur_j denotes the backup duration of object Ob_j, and

$$Dur_1 \geq Dur_2 \geq Dur_3 \geq ... \geq Dur_n.$$

Let there be N tape drives: $Tape_1, ..., Tape_N$, and each tape drive be configured with k disk agents. We observe the following running counters per each tape drive $Tape_i$:

- $DiskAgent_i$ – a counter of available (not busy) disk agents of tape drive $Tape_i$; and
- $TapeProcTime_i$ – a counter of overall processing time assigned to tape drive $Tape_i$.

For each tape drive $Tape_i$ $(1 \leq i \leq N)$ these counters are initialized as follows:

$$DiskAgent_i = k$$

$$TapeProcTime_i = 0$$

Now, we describe the iteration step of the algorithm. Let (Ob_j, Dur_j) be the top object in the $OrderedObjectList$, and let

$$TapeProcTime_m = \min_{1 \leq i \leq N \& DiskAgent_i > 0} (TapeProcTime_i),$$

i.e., the tape drive $Tape_m$ has the smallest assigned processing time, and it still has an available disk agent that can process the object Ob_j.

Then object Ob_j is assigned for processing to the available disk agent at the tape drive $Tape_m$, and the running counters of this tape drive are updated as follows:

$$TapeProcTime_m \Leftarrow TapeProcTime_m + Dur_j$$

$$DiskAgent_m \Leftarrow DiskAgent_m - 1$$

Intuitively, under this algorithm, we assign the longest jobs to be processed first. In addition, we suggest the job assignment to concurrent disk agents in such a way that it balances the overall amount of processing time assigned to different tape drives.

Once the disk agents are assigned some objects, the backup processing can start. When a disk agent at a tape drive $Tape_m$ completes the backup of the assigned object, the running counter of this tape drive is updated as follows:

$$DiskAgent_m \Leftarrow DiskAgent_m + 1.$$

Then the disk agent of this tape drive is assigned the next available object from the $OrderedObjectList$, and the running counters are updated again, and the backup process continues.

Performance Evaluation of the LBF Scheduling Algorithm:

In our performance study, we use available historic information on the duration of backup jobs collected from seven backup servers at HP Labs. Table II shows the absolute and relative reduction of the overall backup times when the proposed scheduling algorithm is used for the seven backup servers under study. First of all, significant time savings are

Backup Server	Absolute and Relative Reduction of the Overall Backup Time		
	week1	week2	week3
Server1	211 min (20%)	208 min (22%)	185 min (19%)
Server2	78 min (9%)	77 min (9%)	53 min (6%)
Server3	39 min (5%)	39 min (5%)	41 min (5%)
Server4	89 min (25%)	99 min (26%)	96 min (23%)
Server5	203 min (29%)	212 min (30%)	202 min (29%)
Server6	146 min (26%)	145 min (25%)	149 min (21%)
Server7	49 min (5%)	48 min (5%)	49 min (5%)

TABLE II
ABSOLUTE AND RELATIVE REDUCTION OF THE BACKUP TIME UNDER PROPOSED SCHEDULING ALGORITHM ACROSS SEVEN BACKUP SERVERS UNDER STUDY.

achieved across all the seven backup servers when using the new job scheduling. The absolute time savings range from 39 min to 212 min. Moreover, these results are very consistent for the three consecutive weeks used in the study, as shown in Table II.

The relative performance benefits and reduction of the backup time (5%-30%) depends on the size distribution of

objects the backup server is responsible for. When a backup server has a significant portion of objects with a long processing time ((e.g., the ones shown in Figures 4 (a) and (b) for Server1 and Server6) the time savings are significantly higher than in the case of only a few long running jobs (e.g., the distribution shown in Figure 4 (c) for Server7).

Often, the duration of the longest job defines the overall backup time, and therefore it can not be improved further. However, in such situations, one can potentially reduce the number of concurrent disk agents to avoid redundant data stream interleaving while still achieving the same backup time, as will be shown in Section VI.

V. Creating Balanced Backup Groups with BBG Algorithm

Under the traditional approach, when a group of N objects is assigned for backup processing, there is no way to define a sequence or order in which these objects are processed by the tool. Typically, any available disk agent may be assigned for processing to any object from the set, and the objects that represent different mount points of the same client machine might be written to different tapes. This situation can be especially annoying for smaller client machines when the backed up client data are spread across multiple tapes.

Often, system administrators manually create the so-called backup groups, which are assigned to different tape drives for processing in order to control the number of tapes that are used per server. Figure 5 shows four backup groups *Backup1*, *Backup2*, *Backup3*, and *Backup4*, where each group is assigned a single tape drive.

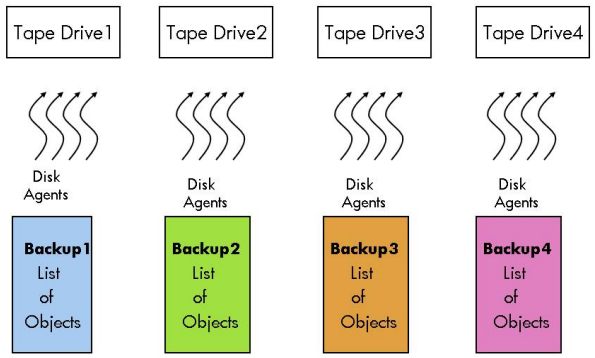

Fig. 5. Backup groups are often created for better control and manageability.

The backup group approach could provide a better manageability iff the created groups are well balanced and take approximately the same time for processing. However, even if manually created groups are well balanced, there is no way to define an order in which the objects are processed within the groups by concurrent disk agents. Potentially, this approach may still lead to an inefficient backup processing and an increased backup time. We apply ideas similar to those described in previous section to automate the creation of **b**alanced **b**ackup **g**roups with the so-called BBG algorithm by using a historic information about the backup time of different objects representing different mount points of the same client machines. This helps to avoid manual configuration efforts by system administrators trying to achieve the same goal. We also introduce an additional job scheduling within each backup group for optimizing the overall backup time.

BBG Algorithm:

First, we create an ordered list of client machines[3] (each might have multiple objects for backup processing) sorted in decreasing order of their backup durations from the previous full backup:

$$OrderedServerList = \{(S_1, Dur_1), ..., (S_n, Dur_n)\}$$

where Dur_r denotes the backup duration of client server S_r, and

$$Dur_1 \geq Dur_2 \geq Dur_3 \geq ... \geq Dur_n.$$

Note, that if server S_r comprised of multiple objects (Ob_r^j, Dur_r^j), $(1 \leq j \leq M_r)$ then

$$Dur_r = \sum_{1 \leq j \leq M_r} Dur_r^j$$

Let there be N tape drives: $Tape_1, Tape_2, ..., Tape_N$.

Let $TapeProcTime_i$ be a running counter of the overall processing time assigned to tape drive $Tape_i$ $(1 \leq i \leq N)$.

The server assignment to different backup groups is defined by the following iteration step of the algorithm.

Let (S_r, Dur_r) be the top server in the $OrderedServerList$, and let

$$TapeProcTime_m = \min_{1 \leq i \leq N} (TapeProcTime_i),$$

i.e., tape drive $Tape_m$ has the smallest assigned processing time.

Then S_r is assigned for processing to the backup group at the tape drive $Tape_m$, and the running counter of this tape drive is updated as follows:

$$TapeProcTime_m \Leftarrow TapeProcTime_m + Dur_r$$

After that the next server from the ordered list $OrderedServerList$ is considered for the assignment.

After all the servers from the list are assigned to backup groups, there is a second stage of the algorithm for object scheduling within the created backup groups.

To avoid processing inefficiency within the backup group, we create an ordered list of objects for this backup group (sorted in decreasing order of their job durations) and schedule them in a similar way as described in previous Section IV.

The proposed algorithm automates the creation of balanced backup groups to limit the number of tapes the client data are written to, while also achieving a significant backup time reduction. The performance results with seven HP Labs backup servers are identical to ones shown in Table II (we omit repeating this table here). This outcome is very interesting: the algorithm was able to create the four balanced groups such that all the smaller client servers are written to a single tape, while still achieving the same backup time reduction as in the case without the grouping constraints.

[3]This approach might be not useful or needed for large servers. In this case, different mount points of large servers are treated as different client machines.

VI. Minimizing Data Interleaving and Tape Drives for a Given Workload

The LBF scheduling algorithm introduced in Section IV has a number of concurrent disk agents per tape drive as the algorithm parameter. However, a fixed number of concurrent disk agents per tape drive might be not a very useful requirement. On the contrary, it might lead to additional interleaving of data streams on the tape. Here is a simple example of this situation. Let there be twenty objects $O_1, O_2, ..., O_{20}$ in the backup set, and let the backup tool have four tape drives each configured with 2 concurrent disk agents as shown in Figure 2. Let these objects take approximately the following time for their backup processing: $T_1 = T_2 = ... = T_{18} = 1$ hour, $T_{19} = 9$ hours, $T_{20} = 10$ hours. If we use the new scheduling algorithm proposed in Section IV then it will enforce the job scheduling as shown in Figure 6 (a).

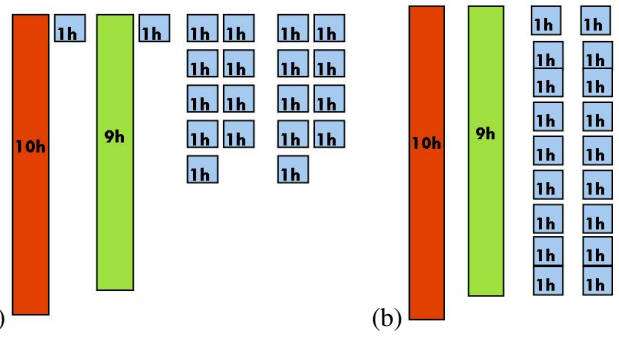

Fig. 6. The job assignment in the backup tool: (a) according to the new LBF scheduling algorithm with a fixed number (two) of concurrent disk agents per each of the four tape drives; and (b) according to the optimized scheduling algorithm with tuned, minimum number of disk agents.

However, as shown in Figure 6 (a), the overall backup time is defined by the duration of the longest job, $T_{20} = 10$ hours, and most of the disk agents in the backup system are idle at least half of the time. Figure 6 (b) shows that all these jobs can be scheduled without interleaving, while completing the overall backup in the same amount of time. If some interleaving is desirable for achieving a higher tape drive throughput then the same idea can be used for minimizing the number of tape drives used for processing a given workload. Freed resources can be used for some dedicated tasks like database backup, etc.

In this section, we introduce a useful analysis routine that determines the minimum required number of **d**isk **a**gents (DA) in the backup system for optimizing the overall backup time. This routine (called DA-analysis) aims to automate tool configuration with either a minimum number of disk agents or a minimum number of tape drives in the backup system required for efficient processing of a given workload.

DA-Analysis Algorithm:

The idea behind this algorithm is to first simulate the achievable backup processing time under the default system parameters. Then we repeat the simulation cycle for estimating the backup processing time under a decreased number of disk agents in the system. We stop the simulation once a decreased number of disk agents in the system leads to a worse system performance, i.e., an increased backup processing time for a given workload.

In the simulation, we use information about the job durations from the previous full backup. At this phase, an ordered list of objects sorted in decreasing order of their backup durations is created:

$$OrderedObjectList = \{(Ob_1, Dur_1), ..., (Ob_n, Dur_n)\},$$

where Dur_j denotes the backup duration of object Ob_j, and

$$Dur_1 \geq Dur_2 \geq Dur_3 \geq ... \geq Dur_n.$$

Let there be N tape drives: $Tape_1$, $Tape_2$, ..., $Tape_N$, and each tape drive be originally configured with k default disk agents. Hence,

$$NumGr = N \times k$$

defines a default value of overall number of disk agents available in the system with a default configuration.

First, we simulate the backup processing by assigning the backup jobs from the ordered list $OrderedObjectList$ to $Group_1, ..., Group_{NumGr}$ according to the LBF scheduling algorithm. The object assignment is simulated using the following iteration step.

Let $GroupProcTime_i$ $(1 \leq i \leq NumGr)$ be a counter for overall processing time assigned to group $Group_i$, and which is initialized as $GroupProcTime_i = 0$.

Let (Ob_j, Dur_j) be the top object in the $OrderedObjectList$, and let

$$GroupProcTime_m = \min_{1 \leq i \leq N}(GroupProcTime_i),$$

i.e., $Group_m$ has the smallest assigned processing time.

Then object Ob_j is assigned for processing to group $Group_m$, and the running counter of this group is updated as follows:

$$GroupProcTime_m \Leftarrow GroupProcTime_m + Dur_j$$

After the assignment of all the objects from the list is completed, we compute

$$MaxProcTime_{NumGr} = \max_{1 \leq i \leq NumGr}(GroupProcTime_i).$$

The computed time $MaxProcTime_{NumGr}$ defines the overall backup processing time under the default system parameters. Then we decrease number of disk agents in the system:

$$NumGr \Leftarrow NumGr - 1,$$

and repeat the backup processing simulation for the decreased number of groups $NumGr - 1$.

If $MaxProcTime_{NumGr} = MaxProcTime_{NumGr-1}$, then it means that the same backup processing time can be achieved with a decreased number of disk agents in the system. We repeat the DA analysis routine until we find a minimum number of disk agents in the system that guarantees the optimized backup time while avoiding unnecessary interleaving of data streams at the tape drives.

Once the minimum number of disk agents in the system is found, it is used for tuning the default configuration. For example, let the DA analysis routine recommend 9 disk agents total for a backup server with four tape drives. It means, that three tape drives can be configured with 2 disk agents, and the remaining forth tape drive with 3 disk agents.

Note that the same recommendation can be used in a different way. For example, if the DA analysis routine recommends 9 disk agents in total, the system administrator might use this analysis to rather reduce the number of tape drives that are needed for processing a given workload, i.e., only use 3 tape drives, while allocating the fourth tape drive for processing an additional workload (such as a database backup, which typically requires a special setup).

In our performance study, we use available historic information on duration of the backup jobs collected from seven backup servers at HP Labs. Table III shows the minimum number of disk agents in the system configuration that guarantees the optimized backup processing time without sacrificing service quality. Note that the default value of disk agents for backup servers at HP Labs was set to 4 per each tape drive, with 4 tape drives in the original system configuration, i.e., 16 disk agents total in the original backup system.

Backup Server	Required Number of Disk Agents to Avoid Redundant Data Interleaving		
	week1	week2	week3
Server1	10	10	10
Server2	5	6	5
Server3	8	8	8
Server4	9	9	8
Server5	15	14	14
Server6	10	10	8
Server7	10	10	8

TABLE III
REQUIRED MINIMUM NUMBER OF DISK AGENTS IN THE BACKUP SYSTEM WHILE OPTIMIZING OVERALL BACKUP TIME.

The results shown in Table III are quite interesting: for six out of seven backup servers there is a significant reduction in the recommended number of disk agents in the system compared to the default value of 16 disk agents: five servers might operate with 8-10 disk agents in total, while Server2 might be configured with 5-6 disk agents in total. Only for Server5 the default configuration parameters are close to the ones which are required for efficient processing of the given workload. For Server5 our algorithm was able to reduce the number of disk agents only to 14-15 from the original 16 agents. Note that this server had the largest backup time reduction under LBF scheduling algorithm as shown in Table II (with 30% reduction). This means that the workload of this backup server is quite balanced under LBF scheduling and can not be optimized much further.

The proposed DA-analysis routine can also be used for achieving a slightly different performance objective set by a system administrator. For example, suppose the system administrator cares about completing the backup in time T (where T might be longer than the optimal time). Then the question for DA-analysis routine is: what should be a minimum required number of disk agents in the backup system (or respectively a minimum required number of tape drives) to process a given workload within the time T? The proposed DA-analysis algorithm is well-suited to answer this question.

VII. RELATED WORK

Researchers have studied filesystem dynamics using different system characterization while aiming to achieve different performance/application objectives. There were several earlier file-system metadata studies [10], [15], [16]. They include studying file systems of different operating systems: Digital PDP-10 at CMU in 1981 [15], a study of file systems in 46 HP-UX systems at Hewlett-Packard in 1994 [16], and a large-scale study of over ten thousand Windows systems at Microsoft [10]. The mentioned studies involve filesystem snapshots taken at a single time, and static analysis of collected metadata. This is different compared to goals of our study which involves analysis of information evolution and dynamics over time.

There have also been longitudinal studies of file-system metadata, such as 48 file systems on four file servers Harvard over a period of ten months in 1994 [17] and a large-scale study of over sixty thousand Windows systems at Microsoft over five years [1]. Smith and Seltzer' study includes daily filesystem snapshots with the goal of correlating the issues of file layout, space allocation, fragmentation and file system performance over time. The large-scale study performed in [1] had yearly filesystem snapshots (i.e., five snapshots in total) with the goal of understanding general statistics, metadata profiles, file size distribution, changes of application popularity among Windows users, etc., over long period of time. While this study and its findings are very interesting, it does not provide visibility in file changes, file life span, and filesystem evolution over a shorter time period of weeks and months. In our work, we propose to utilize and enhance the existing backup infrastructure and the backup catalogues for understanding the filesystem dynamics, files evolution, and performing trending analysis during the information life-cycle.

The current generation of commercial backup tools [11], [12], [19] provides a variety of different means to system administrators for scheduling designated collections of client machines on a certain time table. However, within the created collection a random job scheduling is used which can lead to inefficient backup processing and increased backup time.

Scheduling of incoming jobs and the assignment of processors to the scheduled jobs has been always an important factor for optimizing the performance of parallel and distributed systems (see a variety of papers on the topic [2]-[9], [14], [20]-[23]). Designing an efficient distributed server system often assumes choosing the "best" task assignment policy for the given model and user requirements. However, the question of "best" job scheduling or task assignment policy is still open for many models. Typically, the choice of the scheduling/assignment algorithm is driven by performance objectives. If the performance goal is to minimize mean response time then the optimal algorithm is to schedule the shortest job first [8], [13]. However, if there is a requirement of

fairness in jobs' processing then *processor-sharing* or *round-robin* scheduling [8], [21] might be preferable. For minimizing the *makespan*, or schedule length, a promising approach is to schedule the longest job first [22], [23]. The usefulness and performance benefits of different approaches critically depend on the system parameters and job distribution. Moreover, in many cases the job processing time is not-known in advance, and should be either approximated or derived from the past experience. In such situations, one needs to justify the accuracy of the approximation or the model that is used to derive the job processing time. In our work, the analysis of the job size distribution as well as the observation on slow and gradual system evolution over time have motivated and led us to the choice of the optimization technique related to the "longest job first" processing.

VIII. Conclusion

For implementing content management solutions, companies need a comprehensive view of their unstructured data. The existing commercial backup solutions do not provide any data classification, data analysis, or trending information that helps to improve user visibility to the nature and evolution of backed up information assets. We propose a novel framework to utilize the existing backup infrastructure by integrating additional content analysis routines and extracting already available filesystem metadata over time.

The analyzed file metadata and their summaries provide useful views about managed enterprise information assets, their dynamics and trends. We are experimenting on how to better present these statistics and summaries, visualize and organize trends and "highlights", as well as to provide convenient interface for additional data mining. We envision a few storage-related and information-management applications which will consume this data: e.g., the automated analysis of the filesystem "life span" and identifying filesystems and information sources that became "cold" is useful for automated file migration in multi-tier storage systems and performing a fewer full backups over time.

Backup management faces serious challenges on its own: processing ever increasing amount of data while meeting the timing constraints of backup windows. We revisit a traditional backup job scheduling and demonstrate that random job scheduling traditionally used in the backup tools may lead to inefficient backup processing and an increased backup time. Our workload analysis of seven backup servers at HP Labs shows that historic information on the object backup processing time can be used as a good predictor in optimizing future backup processing. We propose the additional backup job scheduling, called LBF, and automated parameter tuning of the backup configuration which may significantly optimize the overall backup time and quality of service. The optimization algorithms introduced in this paper can also be useful for understanding the outcome of different *what if?* scenarios, capacity planning and consolidation exercises.

A few unsolved problems for future work remain around automating the parameter setting such as the optimal number of concurrent disk agents per tape drive that optimizes the tape drive throughput. Currently, system administrators need to perform a set of manual tests to figure out this parameter for their environment (it depends both on available network bandwidth and servers I/O throughput). The goal is to "learn" the best parameter value by observing the past performance under different conditions. This might require introducing additional measurement support by the DP tool.

Acknowledgments: The authors are grateful for useful comments, suggestions, and discussions provided by our HPL colleagues Joe Tucek, Alistair Veitch, Craig Soules.

References

[1] N. Agrawal, W. Bolosky, J. Douceur, J. Lorch. A five-year study of file-system metadata. Proc. of the 5th Conference on File and Storage Technologies (FAST '07), Feb 2007, San Jose, CA.

[2] G. Blelloch, P. Gibbons, and Y. Matias. Provably efficient scheduling for languages with fine-grained parallelism. JACM, 46(2), 1999.

[3] Robert D. Blumofe and Charles E. Leiserson. Space-efficient scheduling of multithreaded computations. SIAM J Computing, 27(1), 1998.

[4] R. Blumofe and C. Leiserson. Scheduling multithreaded computations by work stealing. JACM, 46(5), September 1999.

[5] C. Chekuri and M. A. Bender. An efficient approximation algorithm for minimizing makespan on uniformly related machines. Proc. of the 6th Conf on Integer Programming & Combinatorial Optimization (IPCO'98), Springer LNCS 1412, 1998.

[6] C. Chekuri and S. Khanna. Approximation algorithms for minimizing average weighted completion time. In " Handbook of Scheduling: Algorithms, Models, and Performance Analysis". CRC Press, 2004.

[7] L. Cherkasova, A. Davis, R. Hodgson, V. Kotov, I. Robinson, T. Rokicki: Components of Congestion Control. Proc. of 8th ACM Symposium on Parallel Algorithms and Architectures, SPAA'96, June, 1996.

[8] L. Cherkasova. Scheduling Strategy to Improve Response Time for Web Applications. Proc. on High Performance Computing and Networking (HPCN'98), LNCS, Springer-Verlag, vol. 1401, April 21-23, 1998.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. Proc. of OSDI'2004, December 2004.

[10] J. Douceur, W. Bolosky. A Large-Scale Study of File-System Contents. Proc. of the 1999 Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), May, 1999.

[11] EMC Backup Advisor. http://www.emc.com/products/detail/software/backup-advisor.htm

[12] HP Data Protector. www.hp.com/go/dataprotector

[13] M. Harchol-Balter, B. Schroeder, N. Bansal, M. Agrawal. Size-based Scheduling to Improve Web Performance. ACM Transactions on Computer Systems (TOCS 2003) , vol. 21, no. 2, May 2003.

[14] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour. On the design and evaluation of job scheduling algorithms. Proc. of JSSPP'99, LNCS 1659, 1999.

[15] M. Satyanarayanan. A study of file sizes and functional lifetimes. Proc. of the 8th ACM Symposium on Operating Systems Principles (SOSP), Pacific Grove, CA, December 1981.

[16] T. Siencnecht, R. Friedrich, J. Martinka, P. Friedenbach. The implications of distributed data in a commercial environment on the design of hierarchical storage management. Performance Evaluation, vol. 20, May, 1994.

[17] K. Smith, M. Seltzer. File layout and file system performance. Technical Report TR-35-94, Harvard University, 1994.

[18] C. Soules, K. Keeton, C. B. Morrey. SCAN-Lite: Enterprise-wide analysis on the cheap. Proc. of EuroSys'2009, Nuremberg, Germany, March 31- April 3, 2009.

[19] Symantec: Veritas NetBackup. http://www.symantec.com/business/netbackup

[20] L. Tan and Z. Tari. Dynamic task assignment in server farms: Better performance by task grouping. Proc. of the Int. Symposium on Computers and Communications (ISCC), July 2002.

[21] A. Wierman, M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. Proc. of SIGMETRICS'03, June 2003.

[22] S-M. Yoo, H.Y. Youn. Largest-Job-First-Scan-All Scheduling Policy for 2D Mesh-Connected Systems. Proc. of the 6th Symposium on the Frontiers of Massively Parallel Computation, 1996.

[23] Y. Zhou, T. Kelly, J. Wiener, and E. Anderson. An extended evaluation of two-phase scheduling methods for animation rendering. Proc. of JSSPP'05, LNCS, Springer, 2005.