Run-time Performance Optimization and Job Management in a Data Protection Solution

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

Harald Burose, Subramaniam Venkata Kalambur,
Bernhard Kappler, Kuttiraja Veeranan
Hewlett-Packard Co.,Technology Solutions Group
{harald.burose,kvs,bernhard.kappler,kuttiraja}@hp.com

***Abstract*— The amount of stored data in enterprise Data Centers quadruples every 18 months. This trend presents a serious challenge for backup management and sets new requirements for performance efficiency of traditional backup and archival tools. In this work, we discuss potential performance shortcomings of the existing backup solutions. During a backup session a predefined set of objects (client filesystems) should be backed up. Traditionally, no information on the expected duration and throughput requirements of different backup jobs is provided. This may lead to an inefficient job schedule and the increased backup session time. We analyze historic data on backup processing from eight backup servers in HP Labs, and introduce two additional metrics associated with each backup job, called *job duration* and *job throughput*. Our goal is to use this additional information for automated design of a backup schedule that minimizes the overall completion time for a given set of backup jobs. This problem can be formulated as a resource constrained scheduling problem which is known to be NP-complete. Instead, we propose an efficient heuristics for building an optimized job schedule, called FlexLBF. The new job schedule provides a significant reduction in the backup time (up to 50%) and reduced resource usage (up to 2-3 times). Moreover, we design a simulation-based tool that aims to automate parameter tuning for avoiding manual configuration by system administrators while helping them to achieve nearly optimal performance.**

I. Introduction

No modern business can risk a data loss. The explosion of electronic documents, new company-wide regulations and document retention rules require IT departments to rethink their information management and data protection strategies. System administrators face multiple challenges for effectively and timely backing up the vast amounts of data stored throughout the enterprise. Most backup and restore operations involve many manual processes, they are people- and labor-intensive. Existing shortcomings will only be aggravated by continuing double-digit growth rates of data. For storage organizations facing this dramatic growth of data, the backup and data recovery processing remains a primary struggle point. The analysis show that 60% to 70% of the effort associated with storage management is related to backup/recovery [18]. Processing the ever increasing amounts of data while meeting the timing constraints of backup windows requires more efficient resource allocation, optimized job scheduling, and run-time management of the existing backup infrastructure before new resources have to be added.

HP Data Protector (DP) is HP's enterprise backup offering. During a backup session a predefined set of objects (client filesystems) should be backed up. Traditionally, there is no additional information about these objects such as the expected duration and/or throughput requirements, and the jobs are scheduled in the random order. In our earlier paper [7], we showed that the random job schedule may lead to inefficient backup processing and an increased backup time. Therefore, we proposed a new backup job scheduling, called LBF (longest backup first), which takes advantage of a historic information about the object backup processing time for optimizing the overall backup time.

Typically, a backup tool has a configuration parameter which defines a level of concurrency, i.e., the number of concurrent processes (called disk agents) which can backup different objects in parallel to the same tape drive. One of the unsolved problems in our previous work [7] was automating the parameter setting of concurrent disk agents per tape drive that optimizes the drive throughput. The number of concurrent agents is constant during the session independent on the aggregate throughput of the backed up objects. In this work, we revisit the traditional backup tool architecture, and raise the question whether a constant number of concurrent disk agents per tape drive in the backup session is a "right" decision.

For each backup job, we explicitly introduce two additional metrics, called *job duration* and *job throughput* which are computed using historic data. Our analysis of backup jobs from eight backup servers at HP Labs reveals that the past measurements of backup time and throughput of the same object are quite stable over time, and therefore can be used for predicting performance characteristics of these jobs during future backups. The optimized scheduling of backup jobs can be formulated as a resource constrained scheduling or "bin-packing" problem [20] where a set of N jobs should be scheduled on M machines with given capacities. Each job J is defined by a pair of attributes *(length, width)*. At any time, each machine can process multiple jobs in parallel but the total width of these jobs can not exceed the capacity of the machine. The objective functions is to find a schedule that minimizes the processing makespan or the overall completion time for a given set of jobs. However, as shown in [20] this problem is *NP*-complete even for $M = 1$.

As an alternative solution to the classic optimization problem, we propose a heuristic-based job scheduling algorithm, called FlexLBF, where both the job duration and job throughput are taken into account. Under this algorithm, we dynamically vary the number of concurrent objects assigned for processing per tape drive in order to optimize both the overall backup time and the tape drive utilization during the backup session. To evaluate the benefits of a new job schedule, we use a workload collected from eight backup servers at HP Labs. There are significant time savings achieved under new FlexLBF scheduling: a 20%-50% backup time reduction compared to the already optimized backup time under the LBF scheduler proposed in [7]. Moreover, by using an adaptive number of concurrent agents over time, the FlexLBF scheduler

is able to provide significant resource savings: each workload from the HP Labs backup servers could be processed by using 1-2 tape drives instead of a traditional solution that uses 4 tape drives (while reducing the overall backup time). The released drives can be used for processing additional workloads to further improve the run-time performance of the DP solution.

Finally, we design a simulation-based tool for system administrators to automate parameter tuning and perform useful "what-if" analysis to support their capacity planning and performance optimization efforts. The remainder of the paper presents our results in more detail.

II. Traditional Filesystem Backup Tool

The functionality of a backup tool is built around a backup session and the objects (mount points or filesystems of the 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 1. [1]

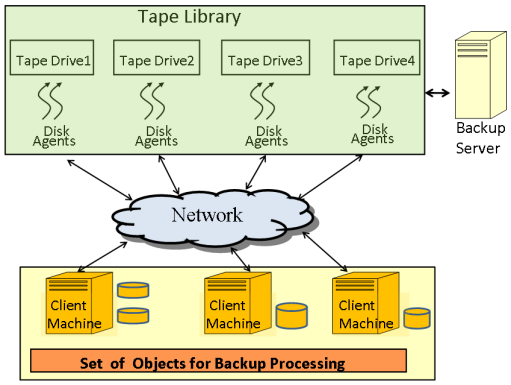

Fig. 1. Traditional Architecture of a Backup Tool with a Tape Library.

The software processes, called disk agents, abbreviated as DAs, are associated with each tape drive. Each disk agent is responsible for backing up a single object at a time.

Each tape drive has a configuration parameter which defines a concurrency level, i.e., the number of concurrent DAs which can backup different objects in parallel to the tape drive. This is done because a single data stream typically can not fully utilize the capacity/bandwidth of the backup tape drive due to slower client machines. A system administrator can configure a high number of DAs per tape drive to enable backup of different objects in parallel. The drawback of such an approach is that the data streams from many 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 data stream written by a single disk agent. There is a significant diversity of the client machines and compute servers (as well as the amount of data stored at these machines) in today's enterprise environments. This diversity impacts the backup duration and its throughput. There are two potential problems with a traditional backup solution which may cause inefficient backup processing.

Job scheduling inefficiency: when a group of N objects is assigned to be processed by the backup tool, there is no way to enforce an order in which these objects should be processed by the tool. If a large (or slow) object with a long backup time is selected significantly later in the backup session this leads to an inefficient schedule and an increased overall backup time.

Fixed, constant number of disk agents inefficiency: when configuring the tool, a system administrator is torn between two orthogonal goals: 1) optimizing the backup throughput by enabling a higher number of concurrent DAs, 2) optimizing the data restore time by avoiding excessive data interleaving (i.e., limiting the number of concurrent DAs). In other words, on one hand, a system administrator should figure out the number of concurrent disk agents that are able to utilize the capacity/bandwidth of the backup tape drive. On the other hand, the system administrator should not over-estimate the required number of concurrent DAs because the data streams from these concurrent agents 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. Moreover, when the aggregate throughput of concurrent streams exceeds the specified tape drive throughput, it may increase the overall backup time instead of decreasing it. Often the backup time of a large object dominates the overall backup time. Too many concurrent data streams written at the same time to the tape drive might decrease the effective throughput of each stream, and therefore, unintentionally increase the backup time of large objects and result in the overall backup time increase.

III. FlexLBF Scheduling to Optimize the Overall Backup Time and Resource Usage

In this section, we further motivate the design of a new scheduler that uses additional information about backup jobs that is extracted from historic data. We present the analysis of backup jobs from eight backup servers at HP Labs. It reveals that the past measurements of backup time and throughput of the same object are quite stable over time, and therefore can be successfully used in such a schedule.

A. Extracting Historic Backup Information

Typically, backup tools record useful monitoring information about the performed backups. In this work, we pursue the efficient management of full backups, i.e., when the data of the entire object is processed during a backup session. [2] For each processed backup job, there is recorded information about the total number of transferred bytes, and the elapsed *backup processing time*. We introduce an additional metric, called *job throughput*, that characterizes the average throughput (MB/s) achieved for the job during the backup session. This metric is defined as follows:

$$job\_throughput = \frac{job\_transferred\_bytes}{job\_processing\_time}$$

Thus any backup job can be characterized by two metrics:

- job processing time;
- job throughput.

[1] HP Data Protector provides the integration with Virtual Tape Libraries (VTL) by emulating the drives of a physical tape library while storing the backup images to disk [14]. The job schedules designed in the paper will automatically apply to DP with VTL deployment as well.

[2] The same approach can be applied to job scheduling in the incremental backups. However, the performance benefits are smaller because incremental backups are shorter and lighter in nature, since they only process modified and newly added files.

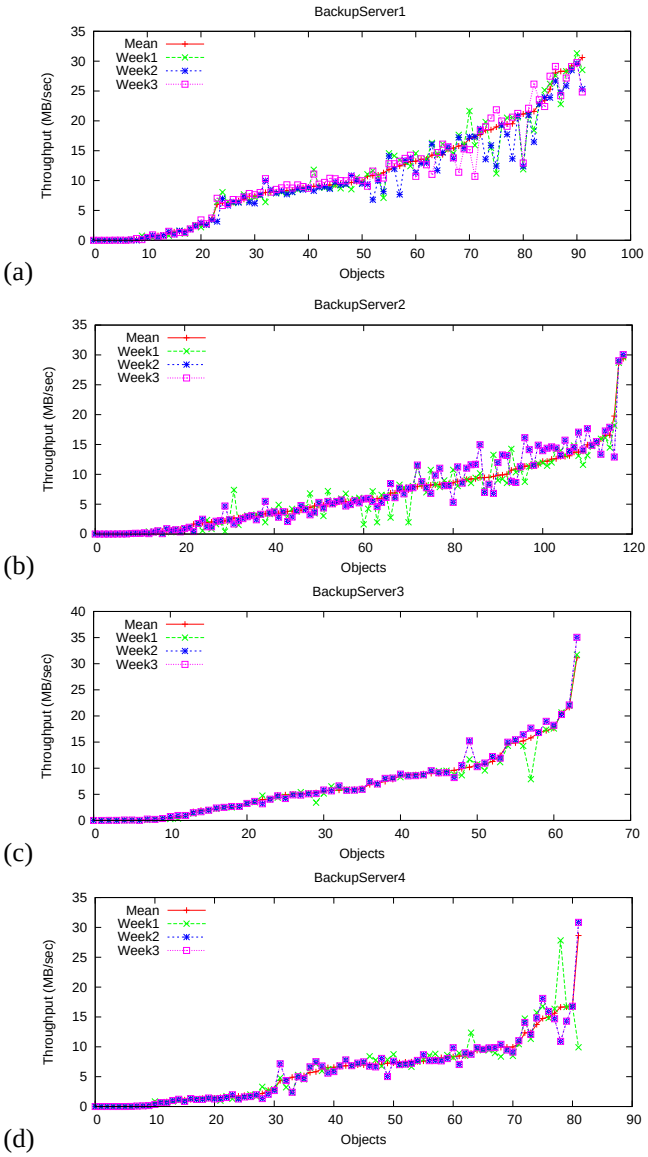

Fig. 2. Historic snapshots of the job average throughput from the three consecutive, full weekly backups: (a) Server1; (b) Server2; (c) Server3, and (d) Server4.

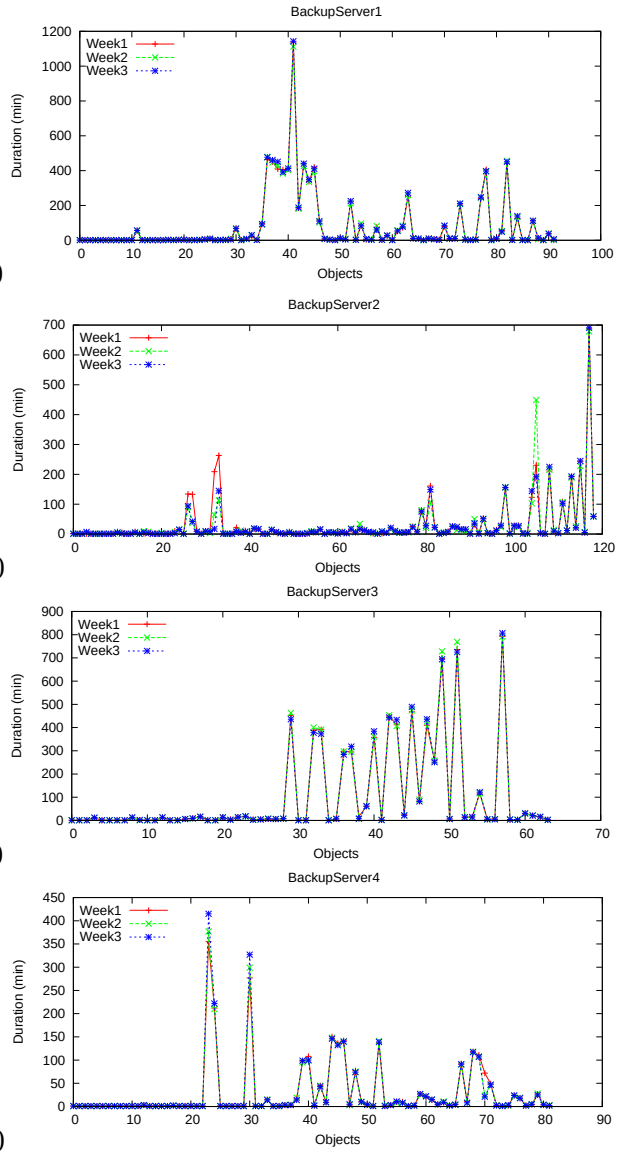

Fig. 3. Historic snapshots of the job duration from the three consecutive, full weekly backups: (a) Server1; (b) Server2; (c) Server3, and (d) Server4.

In this work, we analyze historic data from eight backup servers at HP Labs. First, we need to answer a question whether past measurements of backup processing time and job average throughput are good predictors of the future backup requirements, and whether these past measurements can be used for backup job assignment and scheduling in the future sessions. Figure 2 presents historic snapshots of backup job throughputs from four (out of eight) backup servers at HP Labs. Each figure shows job throughputs (sorted in increasing order) for three consecutive, full weekly backups, and the fourth line corresponds to the mean job throughput for the observed three weeks. We can make the following observations: *(i)* the job throughput of the same object is quite stable over time (especially when compared to the mean throughput over the same time); *(ii)* there is a significant diversity in observed job throughputs: from 0.1 MB/s to 35 MB/s.

These observations are interesting and deserve additional explanations. The networking infrastructure in current data centers and enterprise environments is powerful enough for not being a bottleneck for backup processing. The throughput rate of the stream between the client machine and the backup server (more exactly, the assigned tape drive) is mainly defined by the I/O throughput of the client machine and is less impacted by the network.

Figure 3 presents historic snapshots of backup job durations for the same time period. The job (object) number in Figure 3 is the same as in Figure 2. First of all, the three lines are very close to each other: the backup duration of the same object is quite stable over time (due to gradual changes in the object size). There is a significant diversity in durations: some backups take only 1 min while other backups take 10-16 hours. There is a high percent of "long" jobs: about 25% of all the jobs performed by these backup servers are in the range of 1-16 hours.

We aim to establish whether there is a correlation between job throughputs and job durations, and whether shorter jobs might have lower throughputs, and vice versa, longer jobs

have higher throughputs? The measured elapsed backup time includes a variety of additional overheads such as the overhead of starting a disk agent, connecting to a given client machine, scanning the object metadata, etc. The job throughput metric is derived by dividing the transferred bytes per elapsed backup time. The intuition is that for shorter jobs the overhead might dominate the backup time, and hence could lead to a much lower job throughput. We can see that typically shorter jobs do have low throughputs. However, at the same time, as it is apparent from Figure 2 and Figure 3, there are quite a few short jobs (less than 10 min) with high throughputs, and there are quite a few long jobs (longer than 1 hour) with comparatively low throughputs.

In summary, there is a lot of stability in the historic snapshots shown in Figures 2 and 3. The lines (representing both job duration and throughput) for different weeks are close to each other, meaning that there is a good predictability of these metrics over time. Therefore, this supports the usefulness of historic measurements for optimizing future job scheduling.

B. Background: LBF Scheduling

In this section, we briefly describe the LBF job scheduler introduced in [7]. This scheduler augments the traditional backup solution that operates with a fixed number K of concurrent DAs, i.e., with a constant number of active DAs per tape drive during a backup session.

We observe the following running counters per tape drive:

- $ActDA_i$ – the number of active (busy) disk agents of tape drive $Tape_i$ (initialized as $ActDA_i = 0$); and
- $TapeProcTime_i$ – the overall processing time of all the objects that have been assigned to $Tape_i$ during the current session (initialized as $TapeProcTime_i = 0$).

Each job J_j in the future backup session is represented by a simple tuple: (O_j, Dur_j), where Dur_j denotes the backup duration of object O_j observed from the previous full backup. The LBF scheduler uses an ordered list of objects sorted in decreasing order of their backup durations: $OrdObjList = \{(O_1, Dur_1), ..., (O_n, Dur_n)\}$ where $Dur_1 \geq Dur_2 \geq Dur_3 \geq ... \geq Dur_n$.

Intuitively, under the LBF algorithm, the longest jobs are processed first. In addition, the job assignment process attempts to load balance the overall amount of processing time assigned to different tape drives. Typically, each tape drive concurrently processes a constant number of K jobs. The pseudo-code of the the LBF algorithm is shown in Figure 4.

Assigning resources to a job
For top job $J_j = (O_j, Dur_j)$ in $OrdObjList$ do
  if (!$Blocked$ AND $\exists ActDA_i < K$)
    $TapeProcTime_m = \min_{ActDA_i<K}(TapeProcTime_i)$
    assign job J_i for backup processing to $Tape_m$
    $ActDA_m \Leftarrow ActDA_m + 1$
    $TapeProcTime_m \Leftarrow TapeProcTime_m + Dur_j$
    remove job J_j from $OrdObjList$
  else // no available disk agents for processing job J_j
    $Blocked \Leftarrow 1$ //job J_j assignment is blocked
                until some earlier job is completed
Releasing resources when a job is completed
If backup processing of job J_k is completed by $Tape_i$
  $ActDA_i \Leftarrow ActDA_i - 1$
  $Blocked \Leftarrow 0$

Fig. 4. The LBF algorithm.

C. FlexLBF Scheduling Algorithm

Let us consider a backup tool with N tape drives: $Tape_1, ..., Tape_N$. Under the traditional architecture, there is a configuration parameter K which defines the fixed concurrency level, i.e., a fixed number of concurrent disk agents (DAs) that can backup different objects in parallel to the tape drives. In this work, we investigate the backup tool architecture where tape drives can have a variable number of concurrent DAs defined by the following parameters:

- $maxDA$ - the limit on the maximum number of concurrent disk agents which can be assigned per tape (one can consider different limits for different tape drives);
- $maxTput$ - the aggregate throughput of the tape drive (each tape library is homogeneous, but there could be different generation tape libraries in the overall set).

We observe the following running counters per tape drive:

- $ActDA_i$ – the number of active (busy) disk agents of tape drive $Tape_i$ (initialized as $ActDA_i = 0$); and
- $TapeAggTput_i$ – the aggregate throughput of the currently assigned objects (jobs) to tape drive $Tape_i$ (initialized as $TapeAggTput_i = 0$).

Each job J_j in the future backup session is represented by a tuple: $(O_j, Dur_j, Tput_j)$, where

- O_j is the name of the object;
- Dur_j denotes the backup duration of object O_j observed from the previous full backup, and
- $Tput_j$ denotes the throughput of object O_j computed as a mean of the last l throughput measurements. [3]

Once we have historic information about all the objects, an ordered list of objects $OrdObjList$ (sorted in decreasing order of their backup durations) is created:

$$OrdObjList = \{(O_1, Dur_1, Tput_1), ..., (O_n, Dur_n, Tput_n)\}$$

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

The FlexLBF scheduler operates as follows.

Let $J_j = (O_j, Dur_j, Tput_j)$ be the top object in $OrdObjList$. Let tape drive $Tape_m$ have an available disk agent and

$$TapeAggTput_m = \min_{ActDA_i<maxDA}(TapeAggTput_i),$$

i.e., $Tape_m$ is among the tape drives with an available disk agent, and $Tape_m$ has the smallest aggregate throughput.

Job J_j is assigned to $Tape_m$ if its assignment does not violate the maximum aggregate throughput specified per tape drive, i.e., if the following condition is true:

$$TapeAggTput_m + Tput_j \leq maxTput.$$

If this condition holds then object O_j is assigned to $Tape_m$, and the tape drive running counters are updated as follows:

$$ActDA_m \Leftarrow ActDA_m + 1,$$
$$TapeAggTput_m \Leftarrow TapeAggTput_m + Tput_j$$

Otherwise, job J_j can not be scheduled at this step, and the assignment process is blocked until some earlier scheduled jobs are completed and the additional resources are released.

[3]Using a mean value of the last l throughput measurements provides a more reliable metric and reduces its variance compared to a throughput metric computed only from the latest backup.

Intuitively, under the FlexLBF algorithm, the longest jobs are processed first. Each next object is considered for the assignment to a tape drive with the largest available "space", i.e., to the tape drive: 1) with an available DA; 2) the smallest assigned aggregate throughput (i.e., the largest available "space"), and 3) the condition that the assignment of this new job does not violate the tape drive throughput $maxTput$, i.e., the current job "fits to the available space".

When the earlier scheduled job J_k is completed at the tape drive $Tape_m$, the occupied resources are released and the running counters of this tape drive are updated as follows:

$$ActDA_m \Leftarrow ActDA_m - 1,$$
$$TapeAggTput_m \Leftarrow TapeAggTput_m - Tput_k.$$

The pseudo-code shown in Figure 5 summarizes the FlexLBF algorithm.

Assigning resources to a job
For top job $J_j = (O_j, Dur_j, Tput_j)$ in $OrdObjList$ do
 if (!$Blocked$ AND $\exists ActDA_i < maxDA$)
  $TapeAggTput_m = \min_{ActDA_i < maxDA}(TapeAggTput_i)$
  if ($TapeAggTput_m + Tput_j \leq maxTput$)
   assign jobJ_i for backup processing to $Tape_m$
   $ActDA_m \Leftarrow ActDA_m + 1$
   $TapeAggTput_m \Leftarrow TapeAggTput_m + Tput_j$
   remove job J_j from $OrdObjList$
  else // not enough resources for processing job J_j
   $Blocked \Leftarrow 1$ //job J_j assignment is blocked
    until some earlier job is completed
 else
  $Blocked \Leftarrow 1$
Releasing resources when a job is completed
If backup processing of job J_k is completed by $Tape_i$
 $ActDA_i \Leftarrow ActDA_i - 1$
 $TapeAggTput_i \Leftarrow TapeAggTput_i - Tput_i$
 $Blocked \Leftarrow 0$

Fig. 5. The FlexLBF algorithm.

IV. Performance Study

In our performance study, we use historic information of filesystem backups collected from eight backup servers at HP Labs. While HP Labs represent the research organization, its computing infrastructure is a typical representative of a medium-size enterprise environment. The client machines include a variety of Windows and Linux desktops. In addition, there is a collection of large and powerful servers with significant amount of stored data.

The HP Labs backup servers have 4 tape drives (with maximum data rate of 80 MB/s), each configured with 4 concurrent disk agents. As shown in Figure 2 there are many jobs with throughputs above 20 MB/s. This explains why the backup tool configuration was using 4 concurrent disk agents. However, at the same time, a large fraction of backup jobs have much lower observed throughputs. Therefore the traditional solution with a fixed number of four concurrent disk agents might often leave the tape drives underutilized. This observation presents a perfect opportunity for a new FlexLBF schedule that aims to take the job throughput into account.

To set a base line for a performance comparison, we first process given workloads using LBF scheduling in the traditional tool architecture configured with a single tape drive and a fixed number of four concurrent disk agents per tape.

Then we process the same workloads (from eight backup servers under study) with a new FlexLBF schedule. The backup servers are configured with a single tape drive and the following parameters:

- $maxDA = 10$, i.e., no more than 10 concurrent disk agents can be used per tape drive;
- $maxTput = 80\ MB/s$, i.e., the aggregate throughput of the assigned concurrent objects per tape drive should not exceed 80 MB/s.

Table I shows the absolute and relative reduction in the overall backup times when the FlexLBF scheduling algorithm is used instead of LBF. Under FlexLBF scheduling, the additional information on job throughput is used to dynamically regulate the number of concurrent disk agents that are used for processing to optimize the tape drive throughput.

Backup Server	Absolute and Relative Reduction of the Overall Backup Time		
	week1	week2	week3
Server1	645 min (34%)	642 min (33%)	649 min (33%)
Server2	340 min (33%)	189 min (21%)	163 min (19%)
Server3	915 min (52%)	926 min (52%)	908 min (51%)
Server4	393 min (53%)	370 min (50%)	341 min (45%)
Server5	224 min (47%)	192 min (41%)	211 min (42%)
Server6	453 min (38%)	476 min (38%)	517 min (42%)
Server7	126 min (33%)	124 min (33%)	165 min (39%)
Server8	210 min (26%)	210 min (25%)	168 min (21%)

TABLE I
Absolute and relative reduction of the backup time: **LBF** job scheduling vs. new **FlexLBF** job scheduling.

Table I shows a significant reduction in the overall backup times under FlexLBF across all the eight servers: from 124 min to 926 min (which translates in 21%-53% relative backup time reduction).

Let us look in detail, what contributes to such a significant performance improvement of backup processing under new FlexLBF versus LBF scheduling. Figures 6 a) and 7 a) present the aggregate job throughput under LBF scheduling for $Server2$ and $Server3$ respectively. [4] There are time periods when the aggregate backup throughput reaches 66 MB/s for $Server2$. However, most of the time the backup throughput is significantly lower. The aggregate backup throughput for $Server3$ is even lower on average (see Figure 7 a) while there are short periods when it reaches 76 MB/s. It is apparent that four concurrent disk agents used by LBF scheduler leave the tape drive underutilized most of the time.

Figures 6 b) and 7 b) present the aggregate job throughput under FlexLBF scheduling for $Server2$ and $Server3$ respectively. The achieved backup throughput is much higher for both servers, and approaches 80 MB/s most of the time.

Figures 6 c) and 7 c) present the number of concurrently used disk agents (or concurrently processed backup jobs) under FlexLBF scheduling for $Server2$ and $Server3$ respectively. Note that under the LBF scheduler there is a fixed number of four concurrent agents per tape, and they translate in the straight line of four active disk agents used over time. Therefore, we omit the corresponding figure.

It is interesting to see that for $Server2$ the maximum throughput is achieved with 6 concurrent disk agents in the

[4] Due to space constraints, we show the analysis of throughputs and concurrent disk agents for Server2 and Server3 only. However, the observations are similar for other servers under study.

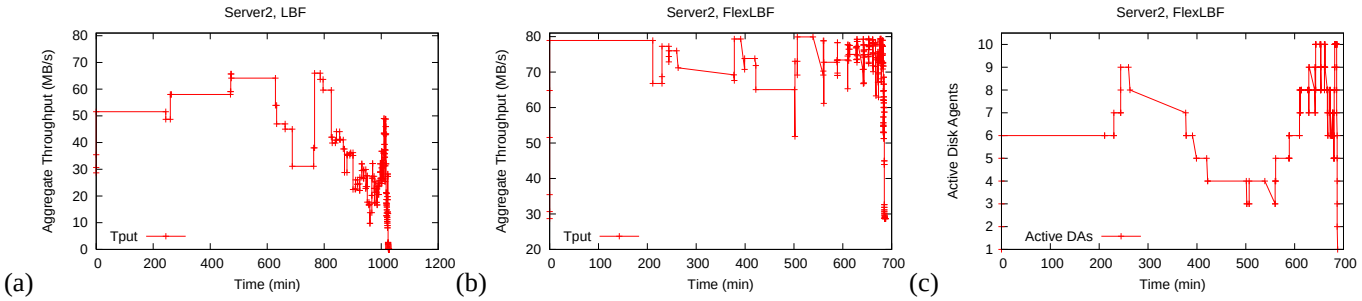

Fig. 6. $Server2$: (a) The aggregate throughput over time under LBF scheduler; (b) The aggregate throughput over time under FlexLBF scheduler; (c) The number of active disk agents over time under FlexLBF scheduler.

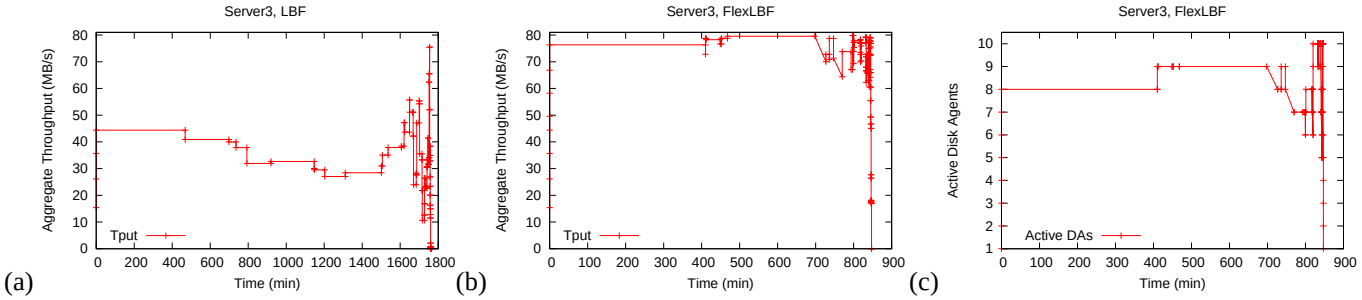

Fig. 7. $Server3$: (a) The aggregate throughput over time under LBF scheduler; (b) The aggregate throughput over time under FlexLBF scheduler; (c) The number of active disk agents over time under FlexLBF scheduler.

beginning of the session as shown in Figure 6 c). Also, as we can see from this figure, there is a time interval between 400 and 600 min where only 3-4 disk agents are active. It means that the scheduled objects had high throughput requirements and the next object in the list could not be scheduled without a violation of the specified limit on the maximum tape drive throughput. At the end of the backup session, all the 10 concurrent disk agents were used for processing.

For $Server3$ the maximum throughput is achieved with 8 concurrent disk agents in the beginning of the session as shown in Figure 7 c). It is apparent that objects backed up by this server have lower throughputs compared to the objects processed by $Server2$. Most of the time $Server3$ uses 8-9 active DAs for backup processing.

This detailed analysis of the number of active DAs over time during a single backup session stresses the difficulty of choosing a single, fixed number of concurrent DAs for efficient backup processing. A fixed number of DAs is always suboptimal in the diverse enterprise environment, and the flexible, adaptive number of concurrent DAs under the FlexLBF scheduler provides a significant advantage for optimizing both backup processing time and the tape drive resource usage.

V. Automated Parameter Tuning

Our goal is to equip system administrators with a useful simulation environment to analyze the potentials of their backup infrastructure and its available capacity before the infrastructure needs to be scaled up and a new capacity has to be added. We designed a set of simulation and analysis routines to identify the range of useful parameter settings and the minimal backup server configuration required for processing a given workload.

The system administrator provides the following inputs to the simulator:

- a given workload, i.e., a set of objects for backup processing with their historical information on object durations and throughputs;
- a default backup server configuration with the number of tape drives $NumDrives$ available in the configuration;
- $maxTput$ - the maximum throughput of the tape drives;
- $maxDA$ - the maximum number of disk agents that can be used concurrently during backup processing. This number should reflect the comfort level of an acceptable data interleaving on the tape that the system administrator is ready to accept. The tool will try to minimize this number to avoid the excessive interleaving if the specified $maxDA$ does not provide additional performance benefits.

Based on the initial inputs from the system administrator, the simulator will produce:

- the minimal number of tape drives required for processing a given workload;
- the optimized number of $maxDA$ for FlexLBF; and
- the estimated overall backup time.

The analysis consists of the following *two* phases.

1. During *the first simulation routine* shown in Figure 8 we simulate the achievable backup processing time under the

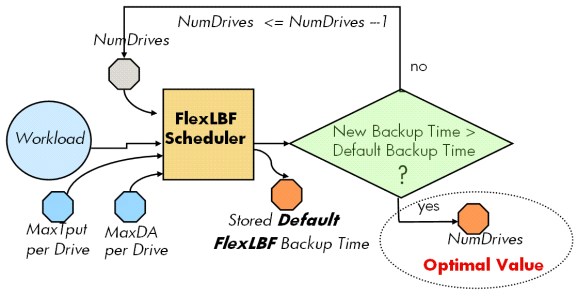

Fig. 8. First simulation routine to minimize a value of $NumDrives$.

FlexLBF algorithm with the default number of tape drives and the specified (by the administrator) number of $maxDA$. This simulated time is called the *default FlexLBF backup time* and it is used as a reference for the best achievable backup time in the full configuration specified by the system administrator.

Then we repeat the simulation cycle for estimating the backup processing time under a decreased number of tape drives in the system. We stop the simulation once a decreased number of tape drives leads to a worse system performance, i.e., an increased backup processing time for a given workload compared to the stored default backup time. In such a way, we first determine the minimal number of tape drives required for a given workload under the FlexLBF scheduler and specified input parameters of $maxTput$ and $maxDA$.

2. During *the second simulation routine* shown in Figure 9 we simulate the achievable backup processing time with the FlexLBF scheduler under a decreased number of $maxDA$. We stop the simulation once a decreased number of $maxDA$ leads to a worse system performance, i.e., when it results in the increased backup processing time for a given workload compared to the stored default backup time.

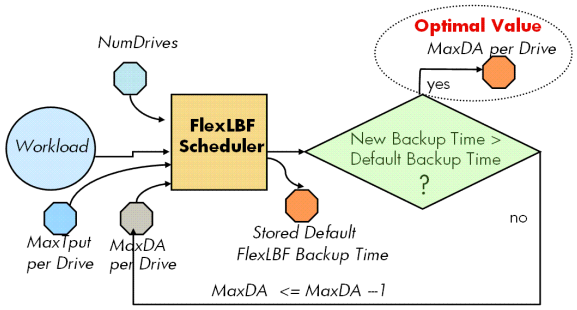

Fig. 9. Second simulation routine to optimize a value of $maxDA$.

Table II shows the tuned configuration parameters across eight HP Labs backup servers with the tape drive target rate of $maxTput$=80 MB/s. For example, for $Server1$ and $Server8$ the simulator shows that the best backup time can be achieved with two tape drives each configured with $MaxDA$=4. Workloads at $Server3$ and $Server6$ can be handled with the best backup time in the configuration with two tape drives and $maxDA$=5.

Backup Server	Configuration Parameters			
	Best Backup Time		Within 15% of Best Time	
	NumDrives	maxDA	NumDrives	maxDA
Server1	2	4	1	7
Server2	1	10	1	6
Server3	2	5	1	9
Server4	1	9	1	8
Server5	1	8	1	7
Server6	2	5	2	4
Server7	1	7	1	6
Server8	2	4	1	5

TABLE II

TUNED CONFIGURATION PARAMETERS ACROSS EIGHT BACKUP SERVERS. THE TARGET DATA RATE OF TAPE DRIVES: $maxTput$=80 MB/S.

For remaining four servers in the study, $Server2$, $Server4$, $Server5$, and $Server7$, their workloads can be processed with a single tape drive and and each of these servers would require a different number of active disk agents. While $Server2$ would benefit from all the 10 concurrent disk agents, $Server7$ would achieve the best backup time with $maxDA$=7. If we would set $maxDA$=10 for $Server7$ then it will just introduce excessive data interleaving with no additional performance benefits. The outlined framework aims to automate parameter tuning and to avoid the potential inefficiencies.

The proposed simulation framework can also be used for achieving a slightly different performance objective set by a system administrator. For example, suppose a system administrator cares about completing the backup in time T (where T might be longer than the optimal time). Then the question for the simulation framework is: what should be a minimum required configuration under the FlexLBF (or LBF) scheduler to process a given workload within time T? The proposed simulation framework is well-suited to answer this question.

In the second half of Table II, we show the required configurations across eight HP Labs backup servers for handling their workloads within 15% of the optimal time. In many cases, there is a significant reduction in the required resources when the backup window requirements are relaxed. Only $Server6$ still would require two tape drives for handling its workload. The remaining servers could meet the required backup time specifications with a single tape drive and different $maxDA$ configurations in a range from 5 to 9 as shown in Table II.

The runtime of the simulator depends on the number of iterations and backup jobs, but for a typical workload of 100 jobs and 3-5 iterations the runtime is 1-2 min, and therefore a system administrator can efficiently use the simulator for understanding the outcome of many different *what–if?* scenarios.

VI. RELATED WORK

Traditionally, magnetic tapes has been used for data backup in enterprise environments. Well-known Unix utilities such as dump, cpio, and tar [19] can write a full filesystem backup as a single data stream to tape. Enterprises might implement different backup policies that define how often the backups are done, whether it is full or incremental backup, and how long these backups are kept. Tools such as AMANDA [1] (built on dump and tar) manages the process of scheduling full and incremental backups from a network of computers and writes these backups to tape as a group.

With falling cost of disk and the explosion of disk capacity, there is a new trend to write backups to disk. Adding disk in a data protection solution uncouples the serial nature of tape from the backup process, it may enable faster backups and can significantly speed up restore operations. Data deduplication became an essential and critical component of disk-to-disk backup systems [17], [10], [22], [27]. Also, there is a growing variety of services and systems that provide efficient filesystem backups over the Internet [24], [11]. However, while disk backup systems provide some advantages over tape, there are still many advantages that are exclusive to tape. The tape-based data protection solution has a lower cost, it consumes much less energy, and provides simple scalability principle (tape-based solution supports capacity extension by the simple addition of more cartridges).

The current generation of commercial backup tools [9], [13], [16], [21] 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]-[8], [23]-[26]). 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], [15]. However, if there is a requirement of fairness in jobs' processing then *processor-sharing* or *round-robin* scheduling [8], [25] might be preferable. For minimizing the *makespan*, i.e., the schedule length, a promising approach is to schedule the longest job first [12], [26]. In [12], an interesting theoretical result is proved, it provides an upper bound of makespan under the longest job first scheduler compared to the time of the optimal strategy in multiprocessor systems.

The usefulness and performance benefits of different scheduling approaches critically depend on the system parameters and job characteristics. In many cases these characteristics are 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 derived approximation. In our work, we carefully justify the choice of backup job characteristics and the stability of their values over time.

Many scheduling problems can be formulated as a resource constrained scheduling problem where a set of n jobs should be scheduled on m machines with given capacities. However, as shown in [20] this problem is *NP*-complete. Recognizing the proven difficulty of solving such scheduling problems, many studies have been undertaken using genetic algorithms, simulated annealing, tabu search, and other integer and linear programming related techniques. As an alternative solution to the classic optimization problem, we propose an efficient heuristic-based algorithm FLexLBF for backup job scheduling.

VII. Conclusion and Future Work

It is fairly clear that in spite of different new offerings in data protection solutions (D2D backup and Internet-hosted backup) the traditional tape-based backup is still a preferred choice in many enterprise environments and the best choice for long-term data backup and data archival. Tape continues to be the most economical solution for long-term storage requirements for the mid-sized data centers. Consequently, many companies have significant amounts of backup data stored on tape, and they are interested in improving performance of tape-based data backup solutions. The algorithms proposed in the paper form a core of novel run-time optimizations in the next major release of Data Protector. They enable a set of new, differentiated features that are currently not available in competing products on the market.

In this paper, we analyzed performance inefficiencies of the backup job scheduling that exists in the traditional backup solution. We proposed the optimized FlexLBF job scheduling with adaptive number of active disk agents for optimizing run-time backup performance. The introduced framework provides a tunable knob to system administrators for achieving multiple QoS objectives: improving resource usage, providing nearly optimal backup latency, and/or optimizing the data restore time. Moreover, we designed a set of simulation and analysis routines to avoid manual configuration and planning efforts by system administrators. The proposed framework automates the backup server configuration and parameter tuning for processing a given workload helping to achieve nearly optimal performance. There are some possible further improvements to FlexLBF. Currently, if the next object can not be scheduled because of its high throughput the algorithm is blocked. We can search through the object list for the object that satisfies current conditions. However, it would lead to a more expensive and complex algorithm. We plan to investigate trade-offs between additional performance benefits and a higher algorithm complexity.

References

[1] AMANDA: The Automated Maryland Automatic Network Disk Archiver. `http://www.amanda.org`

[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, R. Lau, H. Burose, B. Kappler: Enhancing and Optimizing a Data Protection Solution. Proc. of the 17th IEEE/ACM Intl. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'2009), Sept., 2009.

[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, 1998.

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

[10] EMC Corporation. EMC Centera: Content Addressed Storage System, Data Sheet, April 2002.

[11] B. Fitzpatrick. Brackup. http://code.google.com/p/brackup/, http://brad.livejournal.com/tag/brackup.

[12] R. Graham. Bounds on multiprocessing timing anomalies. SIAM J. Appl. Math. vol.17, No.2, March 1969.

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

[14] HP Data Protector Advanced Backup to Disk Integration with Virtual Tape Libraries. White paper. http://h41112.www4.hp.com/promo/imhub/data_protector/backup-to-disk.html

[15] 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.

[16] IBM Tivoli Continuous Data Protection for Files. http://www-142.ibm.com/software/products/us/en/category/tivoli/SWJ10

[17] M. Lillibridge, K. Eshgi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble. Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality. Proc. of FAST 2009, 7th USENIX Conference on File and Storage Technologies, San Francisco, CA, USA, 2009.

[18] Magnetic tape: Whither Thou Goest? White paper. META Delta, Sept.15, 2003.

[19] W. Preston. Backup & Recovery. O'Reily, 2006.

[20] J. Remy. Resource constrained scheduling on multiple machines. Information Processing Letters, Vol. 91, Issue 4, Aug. 2004.

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

[22] S. Quinlan and S. Dorward. Venti: A new approach to archival storage. Proc. of the FAST'2002, First USENIX Conference on File and Storage Technologies, Monterey, CA, USA, 2002.

[23] 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.

[24] M. Vrable, S. Savage and G. Voelker. Cumulus: Filesystem Backup to the Cloud. Proc. of FAST 2009, 7th USENIX Conference on File and Storage Technologies, San Francisco, CA, USA, 2009.

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

[26] 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.

[27] L. You, K. Pollack, and D. Long. Deep Store: An archival storage system architecture. Proc. of the 21st International Conference on Data Engineering (ICDE'05), Tokyo, Japan, April 2005.