



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

Ludmila Cherkasova, Roger Lau, Harald Burose, Subramaniam Venkata Kalambur, Bernhard Kappler, Kuttiraja Veeranan

HP Laboratories  
HPL-2010-72

### **Keyword(s):**

data Protector, backup management, job scheduling, performance evaluation, automated parameter tuning

### **Abstract:**

The amount of stored data in enterprise Data Centers quadruples every 18 months. This trend presents a serious challenge for backup management: one either needs to continuously scale the backup infrastructure or to significantly improve the performance and efficiency of existing backup tools. In this work, we discuss potential performance shortcomings of the traditional backup solutions. 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 design 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. As an efficient heuristic for the classic optimization problem, we propose a novel job scheduling algorithm, called FlexLBF. The scheduler utilizes extracted information from historic data and provides a significant reduction in the backup time (up to 50%), improved quality of service, and reduced resource usage (up to 2-3 times). Moreover, the proposed framework automates parameter tuning to avoid manual configuration by system administrators while helping them to achieve nearly optimal performance.

External Posting Date: May 4, 2011 [Fulltext]

Approved for External Publication

Internal Posting Date: June 21, 2010 [Fulltext]

To be published in Proceedings of the 11th IFIP/IEEE Symposium on Integrated Management (IM'2011), Dublin, Ireland, May 23-27, 2011.

© Copyright 2010 Hewlett-Packard Development Company, L.P.

# 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: one either needs to continuously scale the backup infrastructure or to significantly improve the performance and efficiency of existing backup tools. In this work, we discuss potential performance shortcomings of the traditional backup solutions. 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 design 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. As an efficient heuristic for the classic optimization problem, we propose a novel job scheduling algorithm, called FlexLBF. The scheduler utilizes extracted information from historic data and provides a significant reduction in the backup time (up to 50%), improved quality of service, and reduced resource usage (up to 2-3 times). Moreover, the proposed framework automates parameter tuning to avoid manual configuration by system administrators while helping them to achieve nearly optimal performance.

## I. INTRODUCTION

Establishing a data management strategy becomes a high priority for many companies when they face the challenge of dealing with data growth and gaining visibility in the dynamics and evolution of their information assets. No modern business can risk a data loss. System administrators struggle to manage and scale their backup infrastructure to protect their data and increase data availability. Processing the ever increasing amounts of data while meeting the timing constraints of backup windows requires efficient resource allocation in the existing infrastructure before it needs to be scaled up and a new capacity has to be added. HP Data Protector is HP’s enterprise backup offering. For each backed up object (which represents a mount point or a filesystem) there is a recorded monitoring information on the total number of transferred bytes and the elapsed backup processing time. The question is whether the analysis of this historic data can help in understanding the potential inefficiency of existing backup processes and can be used for optimizing the run-time performance of the future backups.

A traditional backup tool has a configuration parameter which defines a level of concurrency, i.e., a fixed number of concurrent processes (called disk agents) which can backup

different objects in parallel to the tape drives. The number of concurrent agents is constant during the session independent on the aggregate throughput of the backed up objects (their throughputs might be quite diverse). In this work, we revisit the traditional backup tool architecture, and raise the question whether a fixed number of concurrent disk agents per tape drive in the backup session is a “right” decision.

During a backup session a predefined set of objects (client filesystems) should be backed up. Traditionally, there is no information on the expected duration and throughput requirements of different backup jobs, and they are scheduled in the random order. In this work, we introduce two additional metrics associated with each backup job, called *job duration* and *job throughput* which are computed using historic data. Our analysis of backup jobs from eight backup servers at HP Labs reveal that the past measurements of backup time and throughput of the same object are quite stable over time. Therefore, this historic information can be used to optimize the backup schedule with an objective to minimize the backup session duration. The optimized scheduling of backup jobs can be formulated as a resource constrained scheduling problem [21] 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 function 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 [21] this problem is NP-complete even for  $M = 1$ .

As an alternative solution to the classic optimization problem, we propose a job scheduling algorithm, called FlexLBF. Under this algorithm, the *longest backups* are scheduled first and a *flexible number of concurrent jobs* are processed over time. In our performance study, 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].<sup>1</sup> One of the unsolved problems in our previous work [7] was

<sup>1</sup>The LBF scheduler proposed in [7] operates over the traditional backup architecture with a fixed number of concurrent disk agents.

automating the parameter setting of concurrent disk agents per tape drive that optimizes the tape drive throughput. By using an adaptive number of concurrent agents over time, the FlexLBF scheduler solves this problem and is able to provide significant resource savings: the same workload from HP Labs backup servers could be processed using 1-2 tape drives instead of a traditional solution that uses 4 tape drives. The saved tape drives can be used for processing additional workloads to significantly improve the run-time performance of the DP solution. Finally, the proposed framework supports automated parameter tuning in order to eliminate manual configuration and planning efforts by system administrators. 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 filesystem 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.

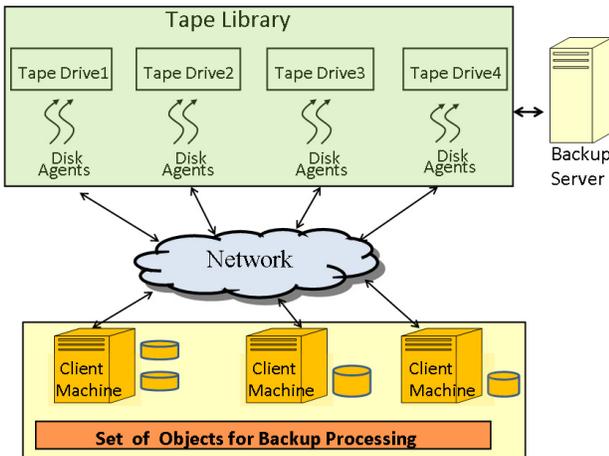


Fig. 1. Traditional Architecture of a Backup Tool with a Tape Library.

Each tape drive has a configuration parameter which defines a concurrency level, i.e., the number of concurrent processes (called disk agents, abbreviated as 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 slow client machines. A system administrator can configure up to 32 DAs per tape drive to enable concurrent backup of 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 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 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 specific 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. The question is whether we could adaptively tune the number of active DAs at each tape drive during the backup session to maximize the system throughput and minimize the backup time.

## III. FLEXLBF SCHEDULING TO OPTIMIZE THE OVERALL BACKUP TIME AND RESOURCE USAGE

### 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. 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.

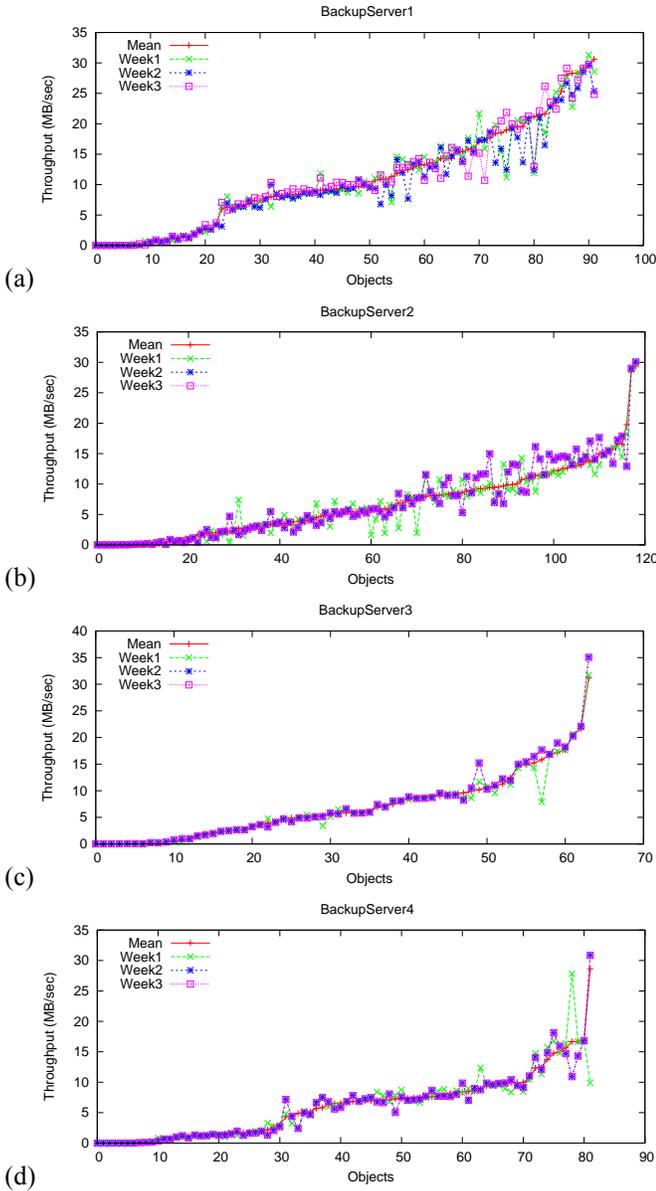


Fig. 2. Historic snapshots of the job average throughput from the three consecutive, full weekly backups: (a) Server1; (b) Server2; (c) Server3, and (d) Server4.

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:

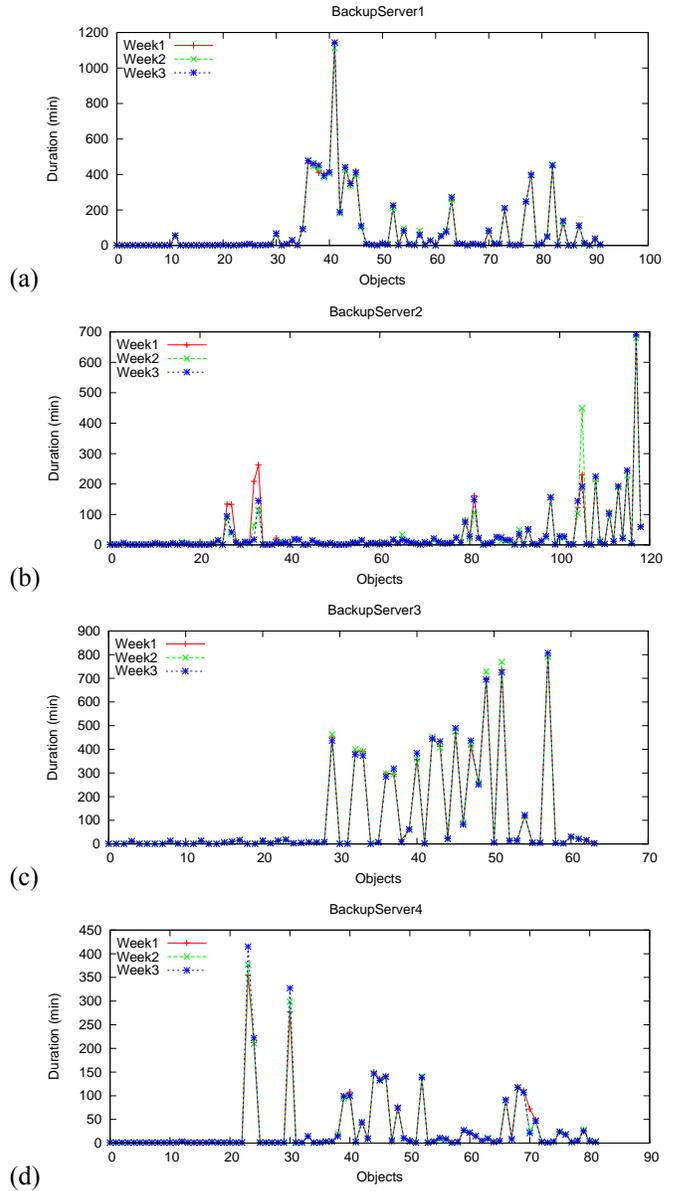


Fig. 3. Historic snapshots of the job duration from the three consecutive, full weekly backups: (a) Server1; (b) Server2; (c) Server3, and (d) Server4.

(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 do not be 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 define 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. FlexLBF Scheduling Algorithm

Let us consider a backup tool with  $N$  tape drives:  $Tape_1, \dots, Tape_N$ . 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.<sup>2</sup>

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:

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

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

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 specific 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.

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 “fit 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 4 summarizes the FlexLBF algorithm.

<sup>2</sup>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.

```

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 job  $J_i$  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. 4. The FlexLBF algorithm.

### C. Background: LBF Scheduling

In this section, we briefly describe the LBF job scheduler introduced in [7]. This scheduler augments the traditional backup solution 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:  $OrdObjList = \{(O_1, Dur_1), \dots, (O_n, Dur_n)\}$  where  $Dur_1 \geq Dur_2 \geq Dur_3 \geq \dots \geq Dur_n$ .

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 5.

## IV. PERFORMANCE STUDY

In our performance study, we use historic information of filesystem backups collected from eight backup servers at HP Labs. 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 quite a few long jobs with throughputs above 20 MB/s. This explains why the backup tool configuration was using 4 concurrent disk agents.

```

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. 5. The LBF algorithm.

In order to demonstrate the efficiency of the new FlexLBF scheduler, we compare the same workload processing under different schedulers using a single tape drive. To set a baseline for the comparison, we first process given workloads using random and LBF scheduling and the traditional tool architecture configured with a single tape drive and a fixed number of four concurrent disk agents per tape. Table I shows the absolute and relative reduction in the overall backup times when the proposed LBF scheduling algorithm is used instead of the traditional (random) job scheduler for eight backup servers under study.

Backup Server	Absolute and Relative Reduction of the Overall Backup Time		
	week1	week2	week3
Server1	671 min (26%)	640 min (25%)	657 min (25%)
Server2	415 min (29%)	371 min (29%)	444 min (34%)
Server3	314 min (15%)	302 min (15%)	289 min (14%)
Server4	164 min (18%)	168 min (18%)	192 min (20%)
Server5	121 min (20%)	139 min (23%)	126 min (21%)
Server6	271 min (19%)	298 min (19%)	310 min (20%)
Server7	118 min (24%)	135 min (26%)	141 min (25%)
Server8	277 min (25%)	249 min (24%)	213 min (23%)

TABLE I

ABSOLUTE AND RELATIVE REDUCTION OF THE BACKUP TIME: **traditional** (RANDOM) JOB SCHEDULING VS. **LBF** JOB SCHEDULING.

First of all, significant time savings are achieved across all the eight backup servers when using the LBF job scheduling compared to the traditional one. The absolute time savings range from 121 min to 415 min. These results are consistent for the three consecutive weeks used in the study, as shown in Table I. The relative performance benefit and reduction in the backup time (14%-34%) depends on the size distribution of objects the backup server is responsible for.

Next, we compare the outcome of backup processing under the LBF scheduler versus the new FlexLBF scheduler for eight backup servers under study. The backup servers under FlexLBF scheduler are configured with a single tape drive and the following parameters:

- $maxDA = 10$ , i.e., no more than 10 concurrent disk

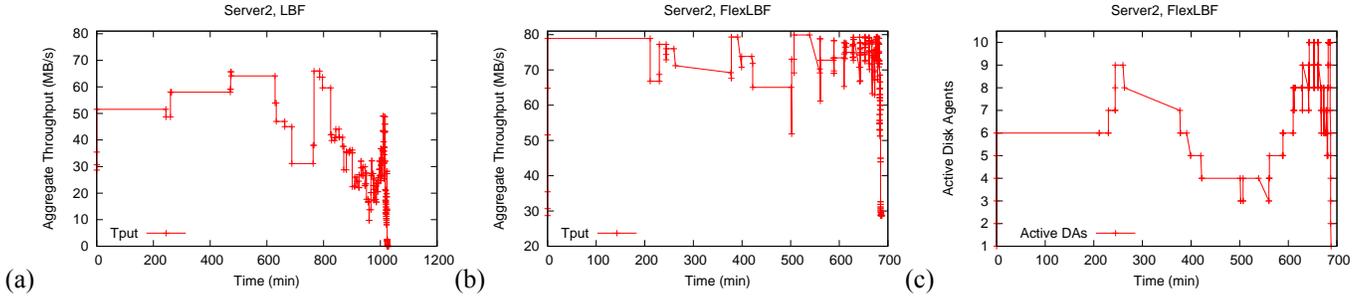


Fig. 6. *Server2*: (a) The aggregate throughput over time under LBF scheduler; (b) The aggregate throughput over time under FlexLBF scheduler; (c) The number of active disk agents over time under FlexLBF scheduler.

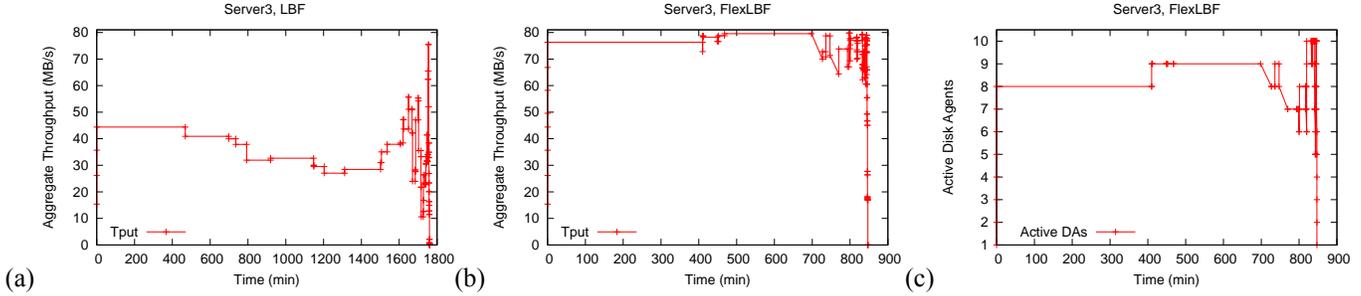


Fig. 7. *Server3*: (a) The aggregate throughput over time under LBF scheduler; (b) The aggregate throughput over time under FlexLBF scheduler; (c) The number of active disk agents over time under FlexLBF scheduler.

agents can be used per tape drive;

- $maxT_{put} = 80 MB/s$ , i.e., the aggregate throughput of the assigned concurrent objects per tape drive should not exceed 80 MB/s.

Table II shows the absolute and relative reduction in the overall backup times when the FlexLBF scheduling algorithm is used instead of LBF. Under FlexLBF scheduling, each job is characterized by its duration and its throughput (observed from the past measurements). 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 II

ABSOLUTE AND RELATIVE REDUCTION OF THE BACKUP TIME: **LBF** JOB SCHEDULING VS. NEW **FlexLBF** JOB SCHEDULING.

Table II shows further significant reduction in the overall backup times 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.<sup>3</sup> 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 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

<sup>3</sup>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.

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 specific 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.

The number of jobs in the backup sets of the eight servers under study was in the range 70-130. We could not use the collected HP Labs backup sets for performance comparison in the original four tape drive configuration because the proposed FlexLBF schedule is capable of processing given backup sets with a single tape drive configuration in a nearly optimal time, e.g., the generated job schedules for the 2nd, 4th, 5th and 7th servers are optimal and cannot be improved.

In order to understand the performance benefits, efficiency, and limitations of the designed approach for multi-tape drive configurations we have created a diverse spectrum of realistic workloads in the following way. Using the overall set of backup jobs from the eight HP Labs backup servers as a base (the set consisted of more than 700 jobs), we have created different backup set “samples” of a given size. In such a way, we have generated multiple different backup sets with 100, 200, 300, and 400 jobs. In particular, we’ve created four “samples” of each size. Thus we had 16 different backup sets of different size but with representative characteristics of real workloads. We used generated backup sets with 100 and 200 jobs for evaluating 1 and 2 tape drive configuration (these workloads are still relatively small and typically lead to optimal solutions in a 2-tape drive configuration) We used the backup sets with 300 and 400 jobs for evaluating a full spectrum of 1-4 tape drive configurations

Figure 8 shows the relative reduction of the backup session makespan under the generated FlexLBF schedule compared to backup processing under the LBF schedule. There are four sets (lines) of results shown in the graph.

- the first two lines represent performance benefit of the FlexLBF schedule over LBF schedule for 1 and 2 tape drive configurations. These results are obtained from simulating the backup processing of 16 different backup sets: 4 x 100 jobs, 4 x 200 jobs, 4 x 300 jobs, and 4 x 400 jobs. As Figure 8 shows performance savings are very significant for most of backup sets: for a single tape

drive the makespan reduction with the FlexLBF schedule compared to LBF schedule is consistently high: there is 40% to 55% decrease in the backup processing time. When we use two tape drives (i.e., double the amount of resources used), in a few cases, both FlexLBF and LBF schedule produce near-optimal results, but for the remaining cases FlexLBF still significantly outperforms LBF schedule: 30%-50%.

- the remaining two lines in the graph represent performance benefit of the FlexLBF schedule over LBF schedule for 3-4 drive configuration for 8 different backup sets (4 x 300 jobs and 4 x 400 jobs). Again, the FlexLBF schedule significantly outperforms the LBF schedule in most of the cases: 20%-50% decrease in the backup processing time, and only in a few cases, when the makespan is explicitly bounded by the duration of the longest job – both FlexLBF and LBF schedule produce a similar outcome.

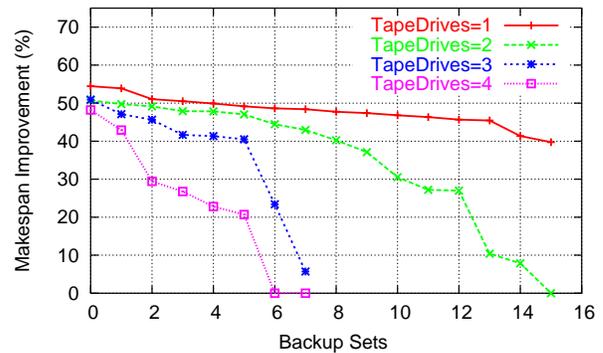


Fig. 8. Relative reduction of makespan under new **bin-packing** scheduling vs **LBF** scheduling for a variety of different backup tool configurations

#### A. Sensitivity Analysis of FlexLBF Parameters

The FlexLBF scheduling algorithm has two parameters: *i*)  $maxDA$  - the limit on the maximum number of concurrent DAs per tape drive; and *ii*)  $maxTput$  - the limit on the aggregate throughput of the assigned concurrent objects per a drive (typically, defined by the tape drive specification). Let us understand how different settings of  $maxDA$  might impact the performance results for a given workload. In our study, we selected *Server2* and *Server3* (without loss of generality) to perform this sensitivity analysis.

Figures 9 a) and b) show the aggregate throughput and the number of active disk agents for *Server2* over time respectively. Each figure has three lines that correspond to different values of  $maxDA=6, 8, 10$ . The observations are interesting. For *Server2* the maximum throughput is achieved with 6 concurrent disk agents in the beginning of the session, and overall, for most of the time during the session the maximum throughput is achievable with 6 concurrent disk agents or less. As a result, the backup time for  $maxDA=6$  is within 7% of the optimal. The performance difference between settings of  $maxDA=8$  and  $maxDA=10$  is negligible, it is

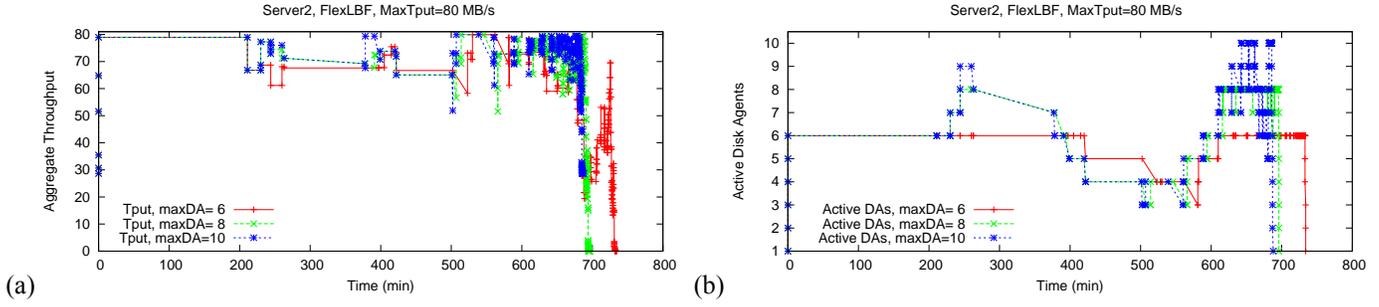


Fig. 9. *Server2*: a) the aggregate throughput; and b) the number of active disk agents over time under FlexLBF scheduling.

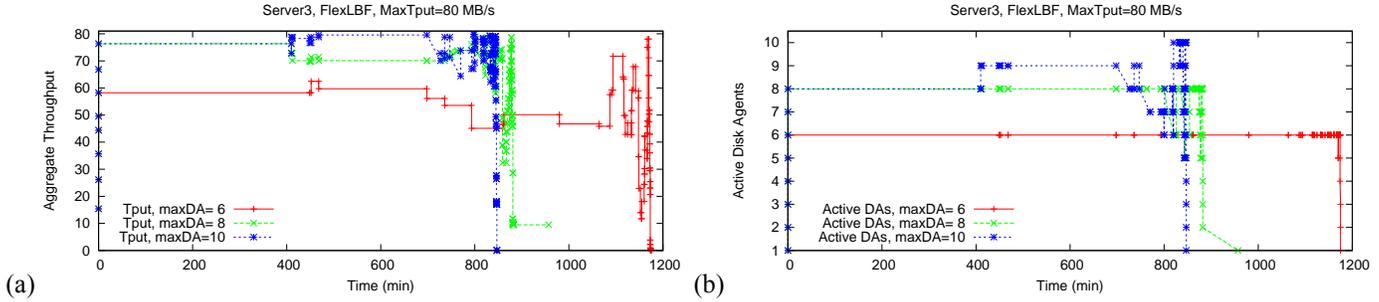


Fig. 10. *Server3*: a) the aggregate throughput; and b) the number of active disk agents over time under FlexLBF scheduling.

around 1%. The setting of  $maxDA=10$  results in the optimal backup time, it is bounded by the duration of the longest backup job in the set, and can not be further improved.

Figures 10 a) and b) show the aggregate throughput and the number of active disk agents for *Server3* over time respectively. For this server, the considered settings of  $maxDA=6, 8, 10$  produce a much larger and more pronounced difference. For  $maxDA=6$ , the achievable tape drive throughput most of the time is approximately 60 MB/s except a short period closer to the end of the session, where the throughput is reaching 70-80 MB/s. For  $maxDA=6$ , the backup time is 40% worse than for  $maxDA=10$ . Using a value of  $maxDA=8$  significantly improves the observed tape drive throughput as well as the overall backup time. Now, the backup time is only 13% worse than for  $maxDA=10$ .

Figures 10 a) and b) show the aggregate throughput and the number of active disk agents for *Server3* over time respectively. Each figure has three lines that correspond to different values of  $maxDA=6, 8, 10$ . The observations are interesting. For  $maxDA=6$ , the achievable tape drive throughput most of the time is approximately 60 MB/s except a short period closer to the end, where the throughput is reaching 70-80 MB/s. Using a value of  $maxDA=8$  significantly improves the observed tape drive throughput as well as the overall backup time. The next value of  $maxDA=10$  still has a positive impact on the performance, while it is less significant. However, using greater values of  $maxDA > 10$  do not change the overall backup time.

It is apparent, that for a given workload and a specific value of  $maxTput$  the impact of different values of  $maxDA$  could be significant. At the same time, typically, there is a

limited range of useful values for  $maxDA$ , i.e., values which have a positive performance impact. Often, a higher value of concurrent DAs might not result in a better performance. It is especially true in the cases, when the overall backup time is limited by the duration of the longest job, and hence can not be further improved. In these situations, using a higher number of concurrent disk agents might lead to excessive data interleaving on the tape without any additional performance benefits.

## 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 specific  $maxDA$  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 11 we simulate the achievable backup processing time under the

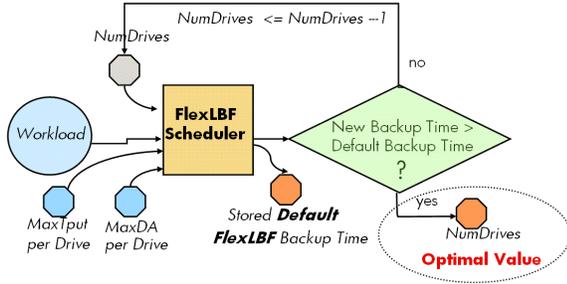


Fig. 11. First simulation routine to minimize a value of  $NumDrives$ .

FlexLBF algorithm with the default number of tape drives and the specific (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 specific 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 specific input parameters of  $maxTput$  and  $maxDA$ .

2. During the second simulation routine shown in Figure 12 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.

Table III 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  $maxDA=5$ .

For remaining four servers in the study, *Server2*, *Server4*, *Server5*, and *Server7*, their workloads can be processed with

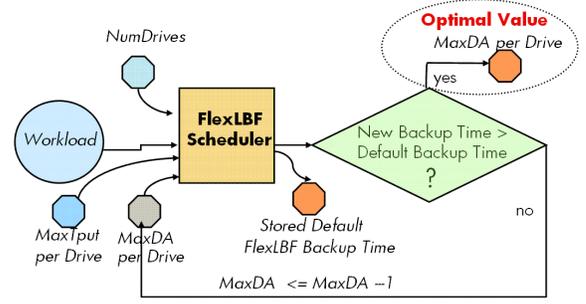


Fig. 12. Second simulation routine to optimize a value of  $maxDA$

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 III

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

a single tape drive 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 7 concurrent disk agents. If we would set  $maxDA$  to 10 for *Server7* – it will 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 III, we show the required configuration 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 specification with a single tape drive and different  $maxDA$  configuration in a range from 5 to 9 as shown in Table III.

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 have been used for data backup in enterprise environments. Well-known Unix utilities such as `dump`, `cpio`, and `tar` [20] 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 [19], [12], [23], [28]. Also, there is a growing variety of services and systems that provide efficient filesystem backups over the Internet [25], [13]. 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). Many enterprises must keep a certain amount of historical data. Consequently, many organizations have significant amounts of backup data stored on tape, and are interested in improving performance of tape-based data protection solution.

The current generation of commercial backup tools [11], [15], [17], [22] 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]-[10], [18], [24]-[29]). 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 [9], [16]. However, if there is a requirement of fairness in jobs’ processing then *processor-sharing* or *round-robin* scheduling [9], [26] might be preferable. For minimizing the

*makespan*, i.e., the schedule length, a promising approach is to schedule the longest job first [27], [29]. In [14], 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 benefit of different scheduling approaches critically depend on the system parameters and job distribution. However, 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 that is used to derive the job processing time. In our work, the analysis of the job size distribution as well as the observation of slow and gradual system evolution over time have led us to the choice of the optimization technique related to the “longest job first” processing.

## 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.

In this paper, we analyzed performance inefficiencies of the backup job scheduling that exists in the traditional backup solution. We proposed a novel 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 benefit and a higher algorithm complexity.

**Acknowledgments:** The authors are grateful to Xiaozhou Li for pointing out Graham’s and Remy’s papers [14], [21] and related discussions.

## REFERENCES

- [1] AMANDA: The Automated Maryland Automatic Network Disk Archiver. <http://www.amanda.org>
- [2] G. Belloch, 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 International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'2009), London, UK, September 21-23, 2009.
- [8] 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.
- [9] 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.
- [10] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. Proc. of OSDI'2004, December 2004.
- [11] EMC Backup Advisor. <http://www.emc.com/products/detail/software/backup-advisor.htm>
- [12] EMC Corporation. EMC Centera: Content Addressed Storage System, Data Sheet, April 2002.
- [13] B. Fitzpatrick. Backup. <http://code.google.com/p/brackup/>, <http://brad.livejournal.com/tag/brackup>.
- [14] R. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* vol.17, No.2, March 1969.
- [15] HP Data Protector. [www.hp.com/go/dataprotector](http://www.hp.com/go/dataprotector)
- [16] 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.
- [17] IBM Tivoli Continuous Data Protection for Files. <http://www-142.ibm.com/software/products/us/en/category/tivoli/SWJ10>
- [18] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour. On the design and evaluation of job scheduling algorithms. Proc. of JSSPP'99, LNCS 1659, 1999.
- [19] 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.
- [20] W. Preston. Backup & Recovery. O'Reilly, 2006.
- [21] J. Remy. Resource constrained scheduling on multiple machines. *Information Processing Letters*, Vol. 91, Issue 4, Aug. 2004.
- [22] Symantec: Veritas NetBackup. <http://www.symantec.com/business/netbackup>
- [23] 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.
- [24] 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.
- [25] M. Vrabie, 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.
- [26] A. Wierman, M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. Proc. of SIGMETRICS'03, June 2003.
- [27] 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.
- [28] 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.
- [29] 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.