DP+IP = Design of Efficient Backup Scheduling

Ludmila Cherkasova, Alex Zhang, Xiaozhou Li
Hewlett-Packard Labs
Palo Alto, CA 94304, USA
{lucy.cherkasova, alex.zhang, xiaozhou.li}@hp.com

Abstract*— Many industries experience an explosion in digital content. This explosion of electronic documents, along with new regulations and document retention rules, sets new requirements for performance efficiency of traditional data protection and archival tools. 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 a suboptimal job schedule that results in the increased backup session time. In this work, we characterize each backup job via two metrics, called *job duration* and *job throughput*. These metrics are derived from collected historic information about backup jobs during previous backup sessions. Our goal is to automate the 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 where a set of n jobs should be scheduled on m machines with given capacities. We provide an integer programming (IP) formulation of this problem and use available IP-solvers for finding an optimized schedule, called *bin-packing* schedule. Performance benefits of the new bin-packing schedule are evaluated via a broad variety of realistic experiments using backup processing data from six backup servers in HP Labs. The new bin-packing job schedule significantly optimizes the backup session time (20%-60% of backup time reduction). HP Data Protector (DP) is HP's enterprise backup offering and it can directly benefit from the designed technique. Moreover, significantly reduced backup session times guarantee an improved resource/power usage of the overall backup solution.

I. INTRODUCTION

One of continuous challenges for IT departments is effectively backing up and protecting the vast amounts of data stored throughout the enterprise. It is a daily struggle. The explosion of digital content, along with new compliance and document retention rules, requires more efficient data protection and archival tools. Current data protection shortcomings and challenges will only be exacerbated by continuing double-digit growth rates of data. Backup and restore operations still involve many manual processes, they are staff and labor intensive. Current systems should be significantly optimized and automated to timely handle growing volumes of data. Reliable and efficient backup/recovery processing remains a primary pain point for most storage organizations. The estimates are that 60% to 70% of the effort associated with storage management is related to backup/recovery [20].

HP Data Protector (DP) is HP's enterprise backup offering. Currently, Data Protector maintains course-grained metadata for each backup period which can be used to provide data points for deriving metadata analysis and trending. In our earlier paper [7], we analyzed a traditional backup job scheduling and demonstrated that the random job scheduling traditionally used in backup tools 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 aims to optimize the overall backup time. This scheduler takes advantage of a historic information about the object backup processing time available from the previous backups and uses this information for job scheduling in the upcoming backup.

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 tape drives. 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 tape drive throughput.

In this work, we take a different approach to the problem. We characterize each backup job via two metrics, called *job duration* and *job throughput*. These metrics are derived from collected historic information about backup jobs during previous backup sessions. Our goal is to automate the 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 where a set of n jobs should be scheduled on m machines with given capacities. However, as shown in [22] this problem is *NP*-complete even for $m = 1$. We provide a general integer programming (IP) formulation of the backup job scheduling problem for multiple tape drives configuration. Then we design an improved and more compact IP formulation for the case of a single drive configuration. We use available IP-solvers (e.g., CPLEX) for finding an optimized schedule, called a *bin-packing* job schedule.

In our performance study, we use a realistic workload collected from six backup servers at HP Labs. There are significant time savings achieved under the new bin-packing job scheduling: a 20%-60% backup time reduction compared to the already optimized backup time under the LBF scheduler proposed in [7]. Moreover, significantly reduced backup session times result in the improved resource/power usage and price/performance ratio of the overall backup solution.

The traditional challenge in applying the integer programming technique is finding a solution in a reasonable time (for being useful in practice). The solution time for integer programming models is notoriously difficult to predict. We observed that the solution time is very bimodal for all the experiments that we have performed: either the optimal solution is found very quickly (within 10 sec-1.5 min), or it takes a few hours to produce a good result. In order to understand the performance benefits, efficiency, and limitations of the designed approach we have created a broad spectrum of different realistic workloads using the combined workload from six HP Labs backup servers. We identified a metric which can be derived from a given workload and the backup tool configuration parameters. This metric correlates very well with a solution time and therefore can be useful in its prediction. These findings highlight an interesting and promising approach for a future work: how to tune the backup tool configuration parameters or change a workload profile to satisfy the identified metric and to provide a quick solution of corresponding IP model for building an efficient bin-packing job schedule.

The remainder of the paper is organized as follows. Section II outlines a traditional backup tool architecture and explains potential performance shortcomings of the traditional approach. The LBF scheduling is described in Section III. A general integer programming formulation of backup job scheduling is provided in Section IV-A. An improved IP formulation for a single drive configuration in is described Section IV-B. The performance study evaluating potential benefits and limitations of the designed bin-packing schedule is presented in Section V. Section VI describes a review of related work. Finally, Section VII draws conclusions.

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]

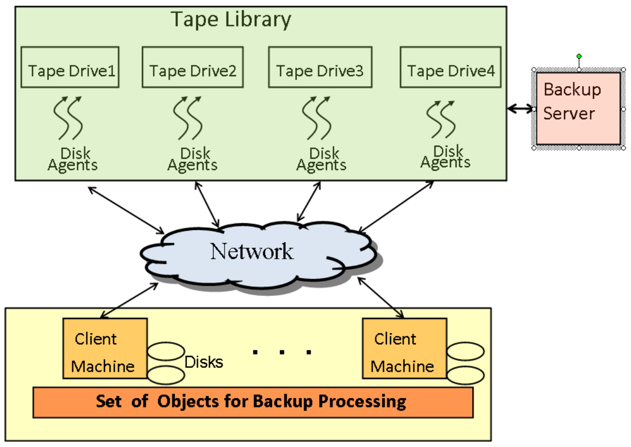


Fig. 1. Traditional Architecture of a Backup Tool with a Tape Library.

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 disk agents, which can backup different objects in parallel to the tape drive. This is done because a single data stream typically cannot fully utilize the capacity/bandwidth of the backup tape drive due to slow client machines (a typical throughput of a client system is 10-20 MB/s). A system administrator can configure a high number of 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 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.

[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.

There are a few potential problems with a traditional backup solution which may cause inefficient backup processing.

- 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.
- When configuring the backup tool, a 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 this leads to a higher restoration time for retrieving such data. 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.

In this work, we aim to revisit a traditional backup job scheduling and configuration issues in order to investigate potential opportunities and benefits of building a tailored backup job schedule from the available historical information on the workload profile.

III. Background: LBF Scheduling

In this section, we briefly describe the LBF job schedule originally proposed in [7], since our goal is to investigate the potential performance improvements and changes to the latest available and already optimized backup job scheduler.

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

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

where Dur_j denotes the backup duration of object O_j, and

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

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

- $DiskAgent_i$ – a counter of available disk agents for tape drive $Tape_i$; and
- $TapeProcTime_i$ – a counter of overall processing time assigned to tape drive $Tape_i$.

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

$$DiskAgent_i = k,$$

$$TapeProcTime_i = 0.$$

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

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

i.e., the tape drive $Tape_r$ has the smallest assigned processing time, and it still has an available disk agent for processing the object O_j.

Then object O_j is assigned for processing to the tape drive $Tape_r$, and the running counters of this tape drive are updated as follows:

$$TapeProcTime_r \Leftarrow TapeProcTime_r + Dur_j,$$

$$DiskAgent_r \Leftarrow DiskAgent_r - 1.$$

Intuitively, under this algorithm, we assign the longest jobs for processing first. In addition, we suggest the job assignment to concurrent disk agents in such a way that it balances the overall amount of processing time assigned to different tape drives. Once the objects were assigned to the available disk agents, the backup processing can start. When a disk agent at a tape drive $Tape_r$ completes the backup of the assigned object, the running counter of this tape drive is updated as follows:

$$DiskAgent_r \Leftarrow DiskAgent_r + 1.$$

Then the disk agent of this tape drive is assigned the next available object from the $OrdObjList$, and the running counters are updated again, and the backup process continues. Typically, under this schedule each tape drive concurrently processes a constant number of k jobs independent on their aggregate throughput.

IV. Bin-Packing Job Schedule

In this section, we provide an integer programming formulation of the multiple machines resource constrained scheduling problem that aims to minimize the makespan (the overall completion time) of a given set of backup jobs for processing by multiple tape drives. The main challenge of this exercise is to provide a compact problem formulation that can be efficiently solved with traditional IP-solvers (e.g., CPLEX [9]) in a reasonable compute time for being useful in practice.

A. General Problem Formulation for Multiple Tape Drives

Throughout the paper we use the following basic notations:

- n – denotes the number of jobs in the backup set;
- m – denotes the number of tape drive in the backup tool configuration;

The *scheduling problem* is defined by a given set of n backup jobs that has to be processed by m tape drives with given performance capacities as described below.

In this work, we investigate the backup tool architecture where the tape drives' configuration is specified by the following parameters:

- $maxDA$ - the maximum number of concurrent disk agents configured per tape drive;
- $maxTput$ - the aggregate throughput of the tape drive (each tape library is homogeneous, but there could be different generation tape libraries in the overall set).

Each job $j, 1 \leq j \leq n$ in a given backup set is defined by a pair of attributes (d_j, w_j), where

- d_j is the duration of job j, and
- w_j is the job j throughput (i.e., required tape drive throughput or the resource demand of job j).

At any time, each tape drive can process up to $maxDA$ jobs in parallel but the total "width" of these jobs cannot exceed the capacity of the tape drive $maxTput$. The objective function is to find a schedule that minimizes the processing makespan, i.e., minimizes the overall completion time for a given set of backup jobs.

We define the following variables:

- R_{ij}: a 0/1 variable, indicating whether backup job i is assigned to tape drive j at some point of time.
- Y_{it}: a 0/1 variable, indicating whether job i *starts* its processing at time t.
- Z_{ijt}: a continuous variable (acting as $R_{ij} \cdot Y_{it}$) indicating whether job i is in processing on tape drive j at time t.
- S: the makespan of the entire backup session.

First of all, let us approximate the low bound on the makespan S. The nature of a given backup workload and the backup tool configuration parameters define the following three low bounds on makespan S:

- D_1 represents the duration of the longest backup job in the given set:
$$D_1 = \max_{1 \leq i \leq n} d_i \quad (1)$$
Makespan S (i.e., duration of the entire backup session) cannot be smaller than the longest backup job in the set.
- D_2 is the shortest possible time that would be required to process the entire set of submitted backup jobs at maximum tape drive throughput $maxTput$ (multiplied by the number of tape drives):
$$D_2 = \frac{\sum_{1 \leq i \leq n} d_i \cdot w_i}{m \cdot maxTput} \quad (2)$$
This time represents the ideal processing of "all the bytes" in the given set of backup jobs at the maximum tape drive rate without any other configuration constraints of the backup server. Clearly, makespan S cannot be smaller than the "ideal" processing time of the backup set.
- D_3 is the shortest possible time that would be necessary to process the entire set of submitted backup jobs while using the maximum possible number $maxDA$ of concurrent disk agents at all tape drives (this computation approximates the processing time for the case when $maxDA$ parameter is a constraint that limits backup processing):
$$D_3 = \frac{\sum_{1 \leq i \leq n} d_i}{m \cdot maxDA} \quad (3)$$
It is apparent that makespan S cannot be smaller than D_3 that reflects the ideal processing time of the backup set with $maxDA$ of concurrent disk agents.

In the IP formulation, we use estimates for lower and upper bounds of makespan S computed in the following way:

$$M_{low} = \lceil max(D_1, D_2, D_3) \rceil \tag{4}$$

$$M_{up} = \lceil max(D_1, D_2, D_3)/0.95 \rceil \tag{5}$$

Comment: M_{low} is indeed a lower bound on makespan S since it cannot be smaller than D_1, D_2 or D_3. However, M_{up} is a possible approximation of the upper bound on makespan S, and our current "guess"-estimate might be incorrect. The optimal solution does not depend on M_{up} in a direct way: as long as $S \leq M_{up}$ it leads to a feasible solution. If this guess makes the problem infeasible, we can always repeat the computation for $M_{up} = \lceil max(D_1, D_2, D_3)/0.9 \rceil$ or $M_{up} = \lceil max(D_1, D_2, D_3)/0.85 \rceil$, etc, until the problem is feasible. If we choose M_{up} too large, then we create a higher complexity problem by introducing a higher number of equations and variables. However, if we choose M_{up} too small, then the problem could be made infeasible. Our computational experience with multiple traces we have used in the case study, indicates that $M_{up} = \lceil max(D_1, D_2, D_3)/0.95 \rceil$ is a good starting guess.

The integer programming formulation is defined as follows:

- A job is processed by exactly one tape drive (total n equations):

$$\sum_{j=1}^{m} R_{ij} = 1, \quad \forall i \tag{6}$$

- Each job must start backup processing at some period before $t = M_{up} - d_i + 1$:

$$\sum_{t=1}^{M_{up}-d_i+1} Y_{it} = 1, \quad \forall i \tag{7}$$

- The jobs that are processed concurrently by tape drive j have to satisfy the tape drive capacity constraint (at any point of time t), i.e., the jobs' aggregate throughput requirements cannot exceed tape drive maximum throughput (total $m \cdot M_{up}$ inequalities):

$$\sum_{i=1}^{n} w_i \cdot \left(\sum_{t'=t-d_i+1}^{t} Z_{ijt'} \right) \leq maxTput, \quad \forall j, t \tag{8}$$

- Maximum of $maxDA$ concurrent jobs can be assigned to tape drive j at any point of time t:

$$\sum_{i=1}^{n} \left(\sum_{t'=t-d_i+1}^{t} Z_{ijt'} \right) \leq maxDA, \quad \forall j, t \tag{9}$$

- Each job finishes the backup processing within time duration S, i.e., formally defining S as a makespan of the backup session. Below, we optimize the number of inequalities by considering only jobs i that were in processing at time $t \geq M_{low}$ (total $n \cdot (M_{up} - M_{low})$ inequalities):

$$t \cdot \left(\sum_{t'=t-d_i+1}^{t} Y_{i,t'} \right) \leq S, \quad \forall i, t : t \geq M_{low} \tag{10}$$

- Linking Z_{ijt} to binary variables R_{ij} and Y_{it} (total $n \cdot m \cdot M_{up}$ inequalities):

$$Z_{ijt} \geq R_{ij} + Y_{it} - 1, \quad \forall i, j, t \tag{11}$$

- Non-negativity requirements:

$$R_{ij} = 0/1; \quad Y_{it} = 0/1; \quad Z_{ijt} \geq 0 \tag{12}$$

One of the traditional integer programming solvers, e.g. CPLEX [9], can be used for finding a feasible solution. Once an optimized job scheduling is provided by the solver, the backup jobs can be ordered by the assigned "start" timestamps, and then the Data Protector tool can schedule these jobs in the advised order. We will call this schedule as a *bin-packing* schedule. Our goal for a performance study is to evaluate the performance benefits of the bin-packing schedule compared to the earlier proposed LBF schedule. The additional goal is to understand the approach complexity and its runtime performance as a function of workload properties and backup tool configuration parameters.

B. Improved Formulation for a Single Tape Drive

Often, system administrators manually create the so-called backup groups, which are assigned to different tape drives for processing. This helps in controlling the number of tapes that are used for different mount points of the same client machine (i.e., avoiding that different filesystems of the client machine might be written to different tapes). This situation can be especially annoying for smaller client machines when the backed up client data are spread across multiple tapes. In case of a backup group, a given set of backup jobs (a specified backup group) is assigned for processing to a particular tape drive.

If we have n jobs ($i = 1, 2, \dots, n$) and a single tape drive for backup processing then the IP formulation can be significantly simplified as shown below.

We define the following variables:

- Y_{it}: a 0/1 variable, indicating indicate whether job i *starts* its run at time t.
- S: the makespan of the entire backup session.

In the formulation below, we use a lower and upper bounds of makespan S (M_{low} and M_{up} respectively) that are computed similarly as in the previous section IV-A, see equations 4, 5.

Assuming that job i will need to finish by period $t = M_{up}$, then job i will need to start no later than $t = M_{up} - d_i + 1$. Now let us pick any period t: we will know whether job i is running in period t if and only if $\left(\sum_{t'=t-d_i+1}^{t} Y_{i,t'} \right) = 1$.

We can reformulate the single tape drive scheduling problem as follows:

- Each job must start backup processing at some time period before $t = M_{up} - d_i + 1$:

$$\sum_{t=1}^{M_{up}-d_i+1} Y_{it} = 1, \quad \forall i \tag{13}$$

- The jobs that are processed concurrently by the same tape drive have to satisfy a given tape drive capacity constraint, i.e. their combined bandwidth requirements should be less or equal than $maxTput$ (total M_{up} inequalities):

$$\sum_{i=1}^{n} w_i \cdot \left(\sum_{t'=t-d_i+1}^{t} Y_{i,t'} \right) \leq maxTput, \quad \forall t \quad (14)$$

- Maximum of $maxDA$ concurrent jobs can be assigned to the tape drive at any point of time t:

$$\sum_{i=1}^{n} \left(\sum_{t'=t-d_i+1}^{t} Y_{i,t'} \right) \leq maxDA, \quad \forall t \quad (15)$$

- Each job finishes the backup processing within time duration S, i.e., formally defining S as a makespan of the backup session:

$$t \cdot \left(\sum_{t'=t-d_i+1}^{t} Y_{i,t'} \right) \leq S, \quad \forall i, t : t \geq M_{low} \quad (16)$$

Note, that the number of variables, equations and inequalities is significantly reduced compared to the general case of multiple tape drives.

V. Performance Study

To evaluate performance benefits of the new bin-packing schedule and compare its performance with already optimized LBF scheduling, we use data from six backup servers in HP Labs. While HP Labs represent the research organization, its computing infrastructure is a typical instance 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.

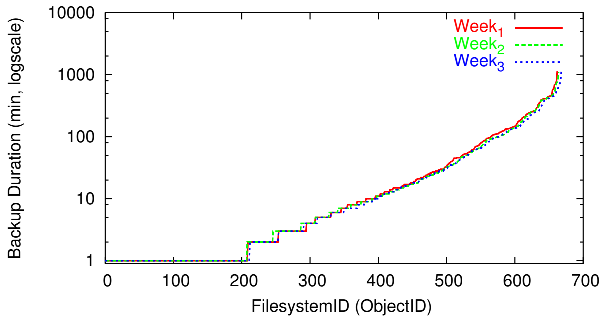
There were 665 objects[2] in the overall backup set under study. Figure 2 (a) shows the object duration distribution in the overall set (sorted in increasing order) for three consecutive, full weekly backups. First of all, there is a significant diversity in durations: some object backups take only 1 min while other objects take 10-17 hours. Second, there is a significant number of "long" backup jobs. Figure 2 (a) shows that about 20% of all the jobs performed by these backup servers are in the range of 1-17 hours.

Figure 2 (b) presents historic snapshots of backup job throughputs in the overall set from six backup servers (sorted in increasing order). There is a significant diversity in observed job throughputs from 0.1 MB/s to 40 MB/s.

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 (b) there is a representative fraction of backup jobs with throughputs above 20 MB/s. This explains why the HP Labs backup configuration is using 4 concurrent disk agents. However, at the same time, there is a significant fraction of backup jobs with much lower observed throughputs. Therefore a fixed number of four concurrent disk agents used by the LBF scheduler and the traditional backup tool would not make the best use of available resources of the tape drive. This observation presents

[2]In this paper, we use the terms filesystem, mount point, and object, interchangeably.

(a)



(b)

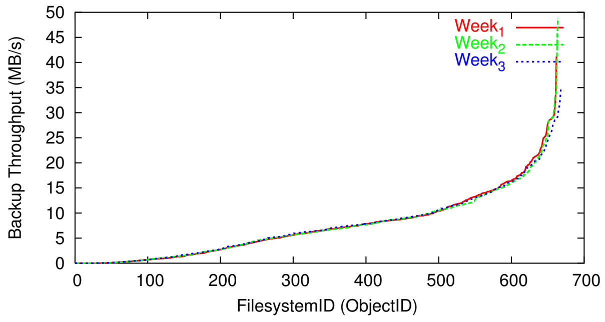


Fig. 2. Workload profile of six HP Labs backup servers: a) backup job duration distribution; b) backup job throughput distribution.

a perfect opportunity for a new bin-packing 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 six backup servers under study) with a new bin-packing 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 session times when the bin-packing schedule is used instead of LBF. The bin-packing schedule is formed with additional information on both job duration and its throughput (observed from the past measurements). This additional information on job throughput is used to schedule a higher number of concurrent backup jobs (when it is appropriate) in order to optimize the tape drive throughput.

Significant time savings are achieved across all the six backup server under bin-packing job scheduling compared to the LBF schedule. The absolute time savings range from 124 min to 928 min (we specially demonstrate the absolute time savings to stress the significance of performance benefits). These results are consistent for three consecutive weeks used in the study, as shown in Table I. The relative performance benefits and reduction in the backup time are 19%-52% and depend on the specifics of workload: the size and throughput distribution of objects the backup server is responsible for.

Backup Server	Absolute and Relative Reduction of the Overall Backup Time		
	week1	week2	week3
Server1	665 min (35%)	651 min (34%)	675 min (35%)
Server2	340 min (33%)	212 min (24%)	163 min (19%)
Server3	922 min (52%)	928 min (52%)	920 min (52%)
Server4	520 min (44%)	552 min (44%)	534 min (43%)
Server5	126 min (33%)	124 min (33%)	165 min (39%)
Server6	231 min (28%)	190 min (26%)	234 min (29%)

TABLE I
ABSOLUTE AND RELATIVE REDUCTION OF THE BACKUP TIME: **LBF** SCHEDULING VS NEW **bin-packing** SCHEDULING.

The performance results of the proposed approach are very promising. The CPLEX solver execution time had a wide range across six servers: around 2 hours for generating a solution for 1st, 3d, 4th and 6th servers, while only 5 sec-1.5 min for generating a solution for 2nd and 5th servers. The number of jobs in the backup sets under study was in the range 70-130. However, the solution time does not directly correlate with the size of the backup sets. For example, the 2nd server had 130 jobs in its backup set (i.e., it was the largest backup set), but CPLEX produced the optimal solution in 1.5 min.

The demonstrated bin-packing schedule results are for a single tape drive model which has been formulated in a significantly more compact and efficient way compared to a multi-tape drive IP formulation. We could not use the collected HP Labs backup sets for performance comparison of the bin-packing and LBF schedules in the original four tape drive configuration because the proposed bin-packing 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 and 5th servers are optimal and cannot be improved.

In order to understand the performance benefits, efficiency, and limitations of the designed IP 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 six HP Labs backup servers as a base (the set consisted of 665 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 configurations (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 3 shows the relative reduction of the backup session makespan under the generated bin-packing schedule compared to backup processing under the LBF schedule. There are two sets of results shown in the graph:

- the first set represents performance benefits of bin-packing schedule over LBF schedule for a single tape drive. 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 3 shows performance savings are very significant for all backup sets: the makespan reduction with bin-packing schedule compared to LBF schedule is consistently high: there is 40% to 59% decrease in the backup processing time.
- the second set represents performance benefits of bin-packing schedule over LBF schedule for multi-drive configurations: we experimented with 2,3 and 4 tape drives in the configuration. These results represent 32 experiments. Again, the bin-packing schedule significantly outperforms the LBF schedule, and only in a few cases, when the makespan is explicitly bounded by the duration of the longest job – both bin-packing and LBF schedule produce similar outcome. Since the amount of makespan reduction depends on the size of a backup set, the duration of the longest job, and the backup tool configuration used in the experiments, we decided to present consolidated results for all the 32 experiments rather than splitting them for different sub-cases.

The traditional challenge in applying the integer programming technique is the solution time. The problem that we are trying to solve is known to be NP-complete even for a single-drive configuration. The models presented in Sections IV-A, IV-B were the outcome of several iterations and parameter tuning to achieve the most effective and compact problem formulation that may produce good practical results. One of the main questions we kept in mind was whether a good solution can be found for larger backup sets and multi-drive configurations in reasonable time. The solution time for integer programming models is notoriously difficult to predict.

Figure 4 shows the solution time for finding an optimized bin-packing schedule., Note, that Y-axes use logscale. The solution time is very bimodal: either the optimal solution is found very quickly (within 10 sec-1.5 min), or it takes a few hours to produce the result. In spite that a single drive formulation is simplified and more compact compared to the multi-drive model, the solution time of almost all single drive experiments (except two) is in the range of a few hours. To explain this phenomena we've considered multiple characteristics of the model generated for each experiment: the number of jobs, the number of tape drives, the number of equations (constraints), the number of variables (including binary and continuous), the size of the time bucket (in minutes), the number of buckets in the model, the duration of the longest job, etc.

As a result of this analysis, we observed a strong correlation between the reported solution time and the relationship of the two low bounds D_1 and D_2 for the makespan (see equations 1, 2 in Section IV-A).

Let us revisit the definition of these bounds:

- D_1 represents the duration of the longest backup job in the given set (clearly, makespan S cannot be smaller than the longest backup job in the set):

$$D_1 = \max_{1 \le i \le n} d_i$$

- D_2 is the shortest possible time that would be required to process the entire set of submitted backup jobs at maximum tape drive throughput $maxTput$:

$$D_2 = \frac{\sum_{1 \le i \le n} d_i \cdot w_i}{m \cdot maxTput}$$

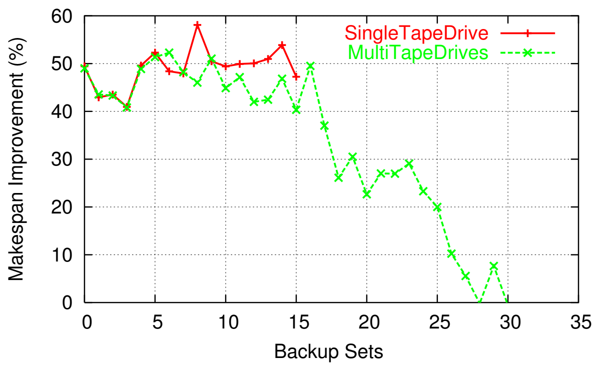


Fig. 3. Relative reduction of makespan under new **bin-packing** scheduling vs **LBF** scheduling for a variety of different backup tool configurations.

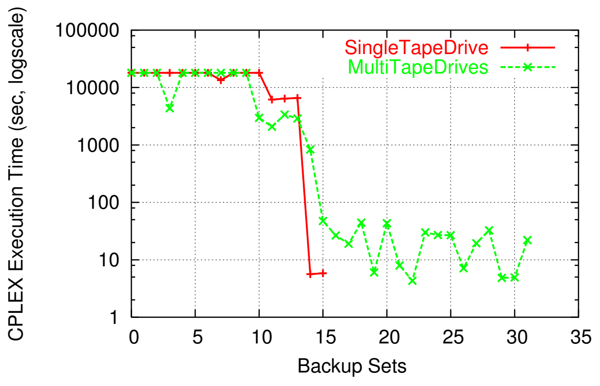


Fig. 4. The solution time for finding an optimized **bin-packing** schedule.

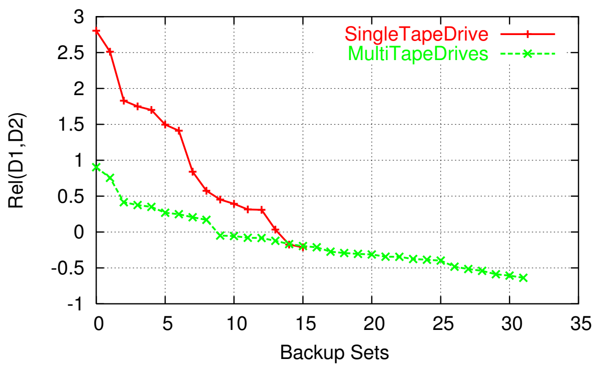


Fig. 5. Metric $Rel(D_1, D_2)$ for different backup sets.

This time represents the ideal processing of "all the bytes" in a given set of backup jobs at the maximum tape drive rate (multiplied by the number of drives) without any other configuration constraints of the backup server. Clearly, makespan S cannot be smaller than the "ideal" processing time of the backup set.

Comment: For the backup tool configuration considered in this paper, the third low bound D_3 was consistently smaller than D_2, i.e., the value of parameter $maxDA = 10$ was not imposing the strong constraint and was not limiting the backup tool performance, and therefore the larger of the other two bounds D_1 and D_2 was defining the lower bound of the makespan. This situation could change for a tool configuration with a small number of concurrent disk agents, and in this case, the relationship between D_1 and D_3 might become critical in a correlation to the solution time.

The relationship between D_1 and D_2 are interesting for intuitive understanding of the "complexity" of the backup job scheduling problem.

When $D_1 \geq D_2$ it means that D_1 defines the lower bound of the makespan and the *ideal* processing of all the jobs at the maximum disk drive rate completes earlier than D_1. In this case, the duration of the longest job strongly impacts the makespan. The difference between D_1 and D_2 determines the size of the "extra room" for making different job scheduling choices (but many of these choices will not increase the overall makespan). Typically, this case means that the solver can quickly find the near-optimal or optimal solution by scheduling the longest job as one of the first jobs, and often the remaining jobs might be scheduled in a flexible way without impacting the schedule makespan.

When $D_1 \leq D_2$ it means that D_2 defines the lower bound of the makespan, and potentially there are many more possible schedules that have different makespan. The larger difference between D_2 and D_1 creates more and more choices for different schedule choices, and the problem becomes much harder to solve.

Let us formally define the relationship between D_1 and D_2 in the following way:

$$Rel(D_1, D_2) = (D_2 - D_1)/D_1$$

Figure 5 shows the computed values for $Rel(D_1, D_2)$ across all the experiments we performed. This new metric $Rel(D_1, D_2)$ has negative values when $D_1 \geq D_2$. The larger negative values are highly correlated with a fast solver runtime and chances of finding the near-optimal solution (as shown by the corresponding runtime values in Figure 4). The metric $Rel(D_1, D_2)$ has positive values when $D_1 \leq D_2$. The positive values of $Rel(D_1, D_2)$ are strongly correlated with a high runtime of the solver as can be seen in Figure 4.

The new $Rel(D_1, D_2)$ metric correlates very well with the solution time of the solver and therefore can be useful in its prediction. An interesting question for a future work is whether we can tune the backup tool configuration parameters or change the workload profile to satisfy the identified metric in order to benefit from the faster solution time.

VI. Related Work

The current generation of commercial backup tools [10], [13], [14], [17], [23] provides a variety of different means to system administrators for scheduling designated collections of client machines on a certain time table. 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 [21]. However, within the created backup groups 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 [1]-[8], [18], [24]-[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. There is a whole body of scheduling research which focuses on minimizing makespan for jobs with precedence constraints [11], [1], [4].

For large-scale heterogeneous distributed systems such as the Grid, job scheduling is one of the main component of resource management. Most work in the Grid-related job scheduling space aims to empirically evaluate scheduling heuristics. Here are a few policies used to improve system utilization and throughput: backfilling [19], adaptive scheduling [16], and task grouping [24].

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 [22] this problem is *NP*-complete even for $m = 1$. 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. While these solutions do not provide the optimal results, they typically identify good feasible solutions that are useful in practice. Our work is another example in this direction, showing that the integer programming approach may indeed be very useful for building the efficient backup scheduling in practice.

VII. Conclusion and Future Work

While there is a growing variety of services and systems that provide efficient filesystem backups over the Internet, 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. Consequently, many organizations have significant amounts of backup data stored on tape, and are interested in improving performance of tape-based data protection solution.

In this paper, we pursue a goal for automated design of a backup schedule that minimizes the overall completion time for a given set of backup jobs. We provide an integer programming (IP) formulation of this problem and use available IP-solvers for finding an optimized schedule, called bin-packing schedule. Performance benefits of the new bin-packing schedule are evaluated via a broad variety of realistic workloads: the new bin-packing job schedule provides 20%-60% of backup time reduction. The same approach can be applied to job scheduling in the incremental backups.

Moreover, we identified a metric which can be derived from a given workload and the backup tool configuration parameters. This metric correlates very well with the solution time of the solver and therefore can be useful in its prediction. An interesting question for a future work is whether we can tune the backup tool configuration parameters or change the workload profile to satisfy the identified metric in order to benefit from the faster solution time for building an efficient bin-packing job schedule.

Acknowledgements: The authors are grateful to Roger Lau for providing access to monitoring data of the backup servers.

References

[1] T. Adam, K. Chandy, and J. Dickson. A comparison of list schedules for parallel processing systems. CACM, 17(12), Dec. 1974.

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

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

[4] R. Brent. The parallel evaluation of general arithmetic expressions. JACM, 21(2), 1974.

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

[9] CPLEX: http://www.cplex.com/

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

[11] R. Graham. Bounds for certain multiprocessor anomalies. Bell Sys Tech J, 45, 1966.

[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] E. Heymann, M. Senar, E. Luque, and M. Livny. Adaptive scheduling for master-worker applications on the computational grid. Proc. of the 1st IEEE/ACM Intl. Workshop on Grid Computing, LNCS 1971, 2000

[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] D. Lifka. The ANL/IBM SP scheduling system. Proc. of JSSPP, LNCS 949, 1995.

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

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

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

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

[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] 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.