# Optimizing QoS, Performance, and Power Efficiency of Backup Services

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

*Abstract*—**For most businesses, backup is a daily operation that needs to reliably protect diverse digital assets distributed across the enterprise. Efficiently processing ever increasing amounts of data residing on multiple desktops, servers, laptops, etc., and providing dynamic recovery capabilities becomes a high priority task for many IT departments. Driven by the advances in cloud computing and Software as a Service (SaaS) delivery model, IT departments are transitioning from providing highly customized services to offering on demand services which can be requested and canceled instantly. Backup service providers must be able to react efficiently to on-demand requests and cannot afford labor intensive resource planning and manual adjustments of schedules. 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. In this work, we compare the outcome of the integer programming formulation with a heuristic-based job scheduling algorithm, called FlexLBF. The FlexLBF schedule produces close to optimal results (reducing backup time 20%-60%) while carrying no additional computing overhead and scaling well to efficiently process large datasets compared to the IP-based solution. Moreover, FlexLBF can be easily analyzed in a simulation environment to further tune a backup server configuration for achieving given performance objectives while saving power (up to additional 50% in our experiments). It helps to avoid guess-based configuration efforts by system administrators and significantly increase the quality and reliability of implemented solutions.**

## I. Introduction

Today's modern businesses cannot afford a risk of data loss. According to Faulkner Information Services, 50% of businesses that lose their data due to disasters go out of business within 24 months, and, according to the US Bureau of Labor, 93% are out of business within five years [4]. The explosion of digital content, along with new compliance and document retention rules, set new requirements for performance efficiency of traditional data protection and archival tools. As a result, IT departments are taking on an ever-greater role in designing and implementing regulatory compliance procedures and systems. However, backup and restore operations still involve many manual steps and processes, they are time consuming, staff and labor intensive. Current data protection shortcomings are exacerbated by continuing double-digit growth rates of data. The progress and attention to the technology of backup over the last decade as a whole has barely kept up when compared to the incredible gains in the technologies associated with computer and storage platforms. Current systems and processes should be significantly optimized and automated to timely handle growing volumes of data. Reliable and efficient backup and 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 [15].

HP Data Protector (DP) is HP's enterprise backup offering. The goal of server backup and data restore operations is to ensure that a business can recover from varying degrees of failure: from the loss of individual files to a disaster affecting an entire system. During a backup session a predefined, large set of distributed 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. To overcome this inefficiency, 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 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 where a set of $n$ jobs should be scheduled on $m$ machines with given capacities. In our earlier work [2], we provide an integer programming (IP) formulation of this problem and use available IP-solvers for finding an optimized schedule, called *bin-packing* schedule. The new bin-packing job schedule significantly optimizes the overall backup session time (it demonstrates the range of 20%-60% of backup time reduction). However, the solution time for integer programming models is notoriously difficult to predict. We observe that the IP-solver runtime is very bimodal for all the experiments that we performed: either the optimal solution is found relatively quickly or it takes a few hours to produce a good result. As an alternative solution to the classic optimization problem, we propose a heuristic-based job scheduling algorithm, called FlexLBF [3]. Under this algorithm, the longest backup jobs are scheduled first. In addition, by using an adaptive number of concurrent disk agents over time, we dynamically vary the number of concurrent objects assigned for processing per tape drive. This allows to optimize the tape drive throughput and achieve a *significantly lower backup time*, and as a result, *significantly reduced power usage*. The question is how close the proposed heuristic to the nearly-optimal IP-based schedule?

This paper makes the *following contributions:*

**Performance comparison of two solutions.** We compare the efficiency of the new FlexLBF job schedule with the IP-based, bin-packing schedule. For the evaluation we use a workload collected from six backup servers at HP Labs. The outcome is quite interesting: the FlexLBF heuristic produces results that are close to optimal (within 3%-10% of bin-packing schedule results) while carrying no additional computing time (which can range from tens of minutes to a few hours) required by the off-line nature of the IP-based solution.

**Scalability evaluation of two solutions.** A typical problem of IP-based solutions is their limited scalability. To evaluate the scalability properties of both approaches, we create an *aggregate* workload by combining workloads from all six backup servers. Our results show that for this size dataset the backup processing time with FlexLBF schedule is up to 25% better than with bin-packing schedule. In other words, the IP-based approach does not scale well for the large datasets: the produced schedule is still far away from the optimal solution (even after 2 hours of compute time).

978-3-901882-70-8 ©2015 IFIP

**Configuration and QoS optimization of backup systems.** We design and prototype a powerful simulation environment to assist system administrators who have concerns about growing data volumes, longer data retention periods, and increased power costs (15-20% per year) that ultimately impact the data protection cost. Our simulator suggests a system configuration that *minimizes the number of actively used backup servers* and therefore improves their resource and power usage while meeting the timing constraints of backup windows and satisfying data restore rates. There are interesting consequences of introducing the job scheduling component and workload characterization in the backup tool architecture. Since the duration and throughput requirements of different backup jobs become known over time and FlexLBF algorithm defines a predictable order for job processing by the backup system, this enables a useful and efficient simulation of the backup session under different system configurations. The designed simulation environment helps to perform useful "what-if" analysis and to avoid dangerous guess-based configuration efforts by system administrators. These automated configuration procedures significantly increase performance, reliability, and quality of implemented solutions. The remainder of the paper presents our results in more detail.

## II. Motivation for Designing New Backup Schedules

HP Data Protector software offers backup and restore functionality specially tailored for enterprise-wide and distributed environments. Data Protector can be used in environments ranging from a single system to thousands of client machines on several sites. It supports backup in heterogeneous environments with clients running on the UNIX or Windows platforms. 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 Fig. 1. [1]

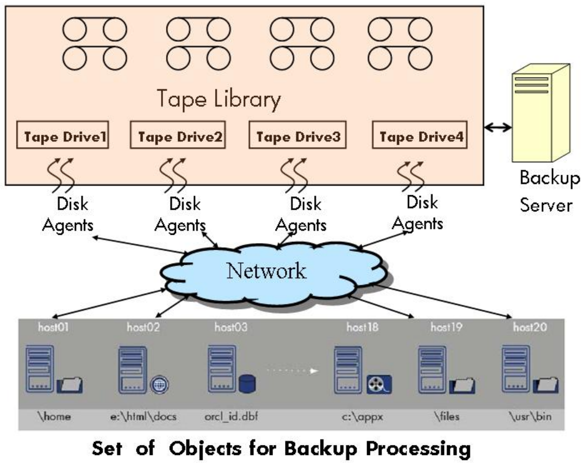

Fig. 1. Traditional Architecture of a Backup Tool with a Tape Library.

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.

Typically, 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, often leading to an order of magnitude difference in observed metrics. However, traditionally, no information on the expected duration and throughput requirements of different backup clients is provided. The data protection tool only knows the name, path, and type of the client machine. The lack of information about the size and speed of the backed up machines may cause inefficient backup processing. Here is a simple example of such inefficiency. Let there be ten objects $O_1, O_2, ..., O_{10}$ in a backup set as shown in Fig. 2 (a). Each backup object is represented by two parameters: its duration (height) and throughput (width). Let the backup tool have a tape drive configured (for simplicity) with 2 concurrent DAs to backup this set, and the drive throughput be 6 (units). If two DAs randomly select objects from the set then the backup session may result in the schedule shown in Fig. 2 (b) that is 28 hours long. Fig. 2 (c) presents a different backup

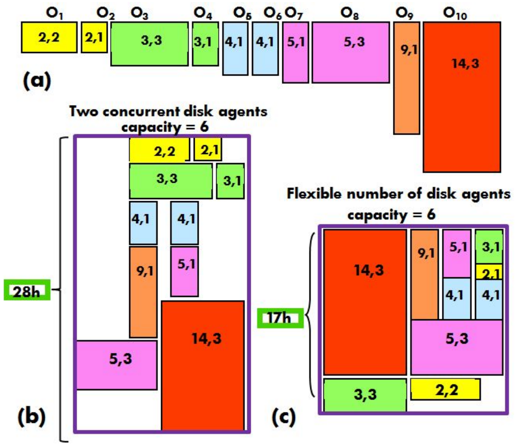
O1
O2
O3
O4
O5
O6
O7
O8
O9
O10
2,2
2,1
3,3
3,1
4,1
4,1
5,1
5,3
9,1
14,3
(a)
Two concurrent disk agents capacity = 6
28h
(b)
Flexible number of disk agents capacity = 6
17h
(c)

Fig. 2. Motivating example.

schedule with a flexible number of concurrent DAs (up to 4). It selects additional objects for backup until a disk drive has enough throughput capacity for handling this new object. Such optimized processing and schedule leads to a backup session of 17 hours, which represents 30% of time reduction.

In this work, we revisit a traditional backup scheduling and configuration issues in order to automate a design of a tailored backup job schedule by using available historical information.

## III. IP-Based 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

[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 [11]. The job schedules designed in the paper will automatically apply to DP with VTL deployment as well.

completion time) of a given set of backup jobs. 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 [5]) in a reasonable compute time for being useful in practice. Below is a very "lean" and optimized IP-based formulation of the problem.

We use the following basic notations:

- $n$ – denotes the number of jobs in the backup set;
- $m$ – denotes the number of tape drives in the backup tool configuration;
- $maxDA$ - 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 $J_i, 1 \le i \le n$ in a given backup set is defined by a pair of attributes $(Dur_i, Tput_i)$, where

- $Dur_i$ is the duration of job $J_i$, and
- $Tput_i$ is the job $J_i$ throughput (i.e., required tape drive throughput or the resource demand of job $J_i$).

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 $S$, i.e., minimizes the overall completion time $S$ for a given set of backup jobs. First of all, let us approximate the lower bound on the makespan $S$ that is defined as the maximum of the following three estimates:

- Makespan $S$ cannot be smaller than the longest backup job in the set (denoted as $D_1$):
$$D_1 = \max_{1 \le i \le n} Dur_i$$
- Makespan $S$ cannot be smaller than the shortest possible time that would be required for processing the entire set of submitted backup jobs at maximum tape drive throughput $maxTput$ by $m$ tape drives (denoted as $D_2$):
$$D_2 = \frac{\sum_{1 \le i \le n} Dur_i \cdot Tput_i}{m \cdot maxTput}$$
- Makespan $S$ cannot be smaller than 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 (denoted as $D_3$):
$$D_3 = \frac{\sum_{1 \le i \le n} Dur_i}{m \cdot maxDA}$$

In the IP formulation, we use estimates for lower and upper bounds of makespan $S$ computed in the following way:
$$M_{low} = \lceil max(D_1, D_2, D_3) \rceil$$
$$M_{up} = \lceil max(D_1, D_2, D_3)/0.95 \rceil$$
$M_{low}$ 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 $S \le M_{up}$ 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. On one hand, if we choose $M_{up}$ too large, then we create a higher complexity problem by introducing a higher number of equations and variables. On the other hand, 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.

We define the following variables:

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

The integer programming formulation is defined as follows:

- A job is processed by exactly one tape drive:
$$\sum_{j=1}^{m} R_{ij} = 1, \quad \forall i : 1 \le i \le n$$
- Each job must start backup processing at some period before $t = M_{up} - Dur_i + 1$:
$$\sum_{t=1}^{M_{up}-Dur_i+1} Y_{it} = 1, \quad \forall i : 1 \le i \le n$$
- The jobs that are processed concurrently by tape drive $j$ have to satisfy its throughput constraint (at any point of time $t$), i.e., the jobs' aggregate throughput requirements cannot exceed tape drive maximum throughput:
$$\sum_{i=1}^{n} Tput_i \cdot \left( \sum_{t'=t-Dur_i+1}^{t} Z_{ijt'} \right) \le maxTput, \quad \forall j, t$$
- Maximum of $maxDA$ concurrent jobs can be assigned to tape drive $j$ at any point of time $t$:
$$\sum_{i=1}^{n} \left( \sum_{t'=t-Dur_i+1}^{t} Z_{ijt'} \right) \le maxDA, \quad \forall j, t$$
- 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 $t \ge M_{low}$:
$$t \cdot \left( \sum_{t'=t-Dur_i+1}^{t} Y_{i,t'} \right) \le S, \quad \forall i, t : t \ge M_{low}$$
- Linking $Z_{ijt}$ to binary variables $R_{ij}$ and $Y_{it}$:
$$Z_{ijt} \ge R_{ij} + Y_{it} - 1, \quad \forall i, j, t$$
- Non-negativity requirements:
$$R_{ij} = 0/1; \quad Y_{it} = 0/1; \quad Z_{ijt} \ge 0$$

One of the traditional integer programming solvers, e.g. CPLEX [5], can be used for finding a feasible solution. Once an optimized job scheduling is provided by the solver, Data Protector can schedule these jobs in the advised order. We will call this schedule as a *bin-packing* schedule. Our goal is to compare the performance of the two scheduling approaches: the IP-based bin-packing schedule and the heuristic-based FlexLBF algorithm described in the next section.

## IV. FlexLBF Scheduling Algorithm

Let us consider a backup tool with $m$ tape drives: $Tape_1, ..., Tape_m$. Under a traditional architecture, there is a configuration parameter $K$ that defines a fixed number of concurrent disk agents (DAs) that backup different objects in parallel to tape drives. In this work, we investigate the backup tool architecture where tape drives may 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:

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

Each job $J_i$ in the future backup session is represented by a tuple: $(O_i, Dur_i, Tput_i)$, where

- $O_i$ is the name of the object;
- $Dur_i$ denotes the backup duration of object $O_i$ observed from the previous full backup, and
- $Tput_i$ denotes the throughput of object $O_i$ computed as a mean of the last $l$ throughput measurements. [2]

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

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

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

The FlexLBF scheduler operates as follows.

Let $J_i = (O_i, Dur_i, Tput_i)$ be the top object in $OrdObjList$. Let tape drive $Tape_j$ have an available disk agent and

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

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

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

$$TapeAggTput_j + Tput_i \leq maxTput.$$

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

$$ActDA_j \Leftarrow ActDA_j + 1,$$

$$TapeAggTput_j \Leftarrow TapeAggTput_j + Tput_i$$

[2]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.

Otherwise, job $J_i$ 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 throughput (or "width"). When the earlier scheduled job $J_k$ is completed at the tape drive $Tape_l$, the occupied resources are released and the running counters of this tape drive are updated as follows:

$$ActDA_l \Leftarrow ActDA_l - 1,$$

$$TapeAggTput_l \Leftarrow TapeAggTput_l - Tput_k.$$

The pseudo-code in Fig. 3 summarizes the algorithm.

```
Assigning resources to a job
For top job $J_i = (O_i, Dur_i, Tput_i)$ in $OrdObjList$ do
  if (!$Blocked$ & $\exists ActDA_j < maxDA$)
      $TapeAggTput_j = \min_{ActDA_i < maxDA}(TapeAggTput_i)$
      if ($TapeAggTput_j + Tput_i \leq maxTput$)
          assign job$J_i$ for backup processing to $Tape_j$
          $ActDA_j \Leftarrow ActDA_j + 1$
          $TapeAggTput_j \Leftarrow TapeAggTput_j + Tput_i$
          remove job $J_j$ from $OrdObjList$
      else   // not enough resources for processing job $J_i$
        $Blocked \Leftarrow 1$   //job $J_i$ 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 $Tape_l$
    $ActDA_l \Leftarrow ActDA_l - 1$
    $TapeAggTput_l \Leftarrow TapeAggTput_l - Tput_l$
    $Blocked \Leftarrow 0$
```

Fig. 3. The FlexLBF algorithm.

## V. Comparing Bin-packing and FlexLBF Schedules

**Workload for the evaluation study.** To compare performance of the bin-packing schedule and heuristic-based FlexLBF 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 distributed client machines include Windows and Linux desktops. There is a variety of compute server farms and Hadoop clusters used for various research and advanced prototyping projects. In addition, there is a representative collection of high-end servers with significant amount of stored data.

There were 665 objects[3] in the backup sets under study. Fig. 4 (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 backups take only 1 min while other objects take 10-17 hours. Second, there is a significant number of "long" backup jobs. Fig. 4 (a) shows that 20% of jobs processed by backup servers are in the range of 1-17 hours.

Fig. 4 (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

[3]We use the terms of filesystem, mount point, and object, interchangeably.

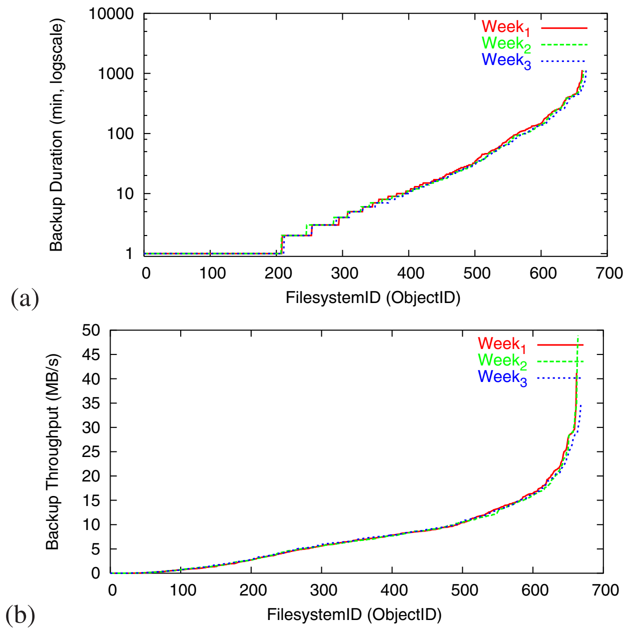

Fig. 4. Workload profile of six HP Labs backup servers: a) backup job duration distribution; b) backup job throughput distribution.

shown in Fig. 4 (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 throughputs. Therefore a fixed number of four concurrent disk agents used by the traditional backup tool would not make the best use of available resources of the tape drive. This observation presents a perfect opportunity for optimizing with new bin-packing and FlexLBF schedules that take the job throughput into account.

**Performance comparison.** For our comparison study, we run a set of experiments where we process workloads (collected from each of the six backup servers under study) with FlexLBF and bin-packing schedules using a single tape drive configured with 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 overall backup session time for processing six workloads (over three weeks) using different schedules: bin-packing and FlexLBF. While significant time savings are achieved across all the six backup servers under bin-packing and FlexLBF schedules compared to the traditional DP approach (not shown here, the savings are **20%-60%** of the session time reduction), the interesting outcome is that the proposed heuristic-based on-line FlexLBF schedule produces *nearly optimal performance results* compared to the formed off-line IP-based bin-packing schedule. The overall backup session durations for Server1, Server2, Server3, Server5, and Server6 are either identical or *less than 3%* apart under the two considered approaches. These results are consistent for three consecutive weeks used in the study, as shown in Table I. The only exception is the result for Server4. The backup sessions for week1 and week2 are 10% longer with FlexLBF compared to the bin-packing schedule. However, the FlexLBF results for week3 are again less than 3% away from the performance of the bin-packing schedule.

| Backup Server | Overall Backup Session Time (min) | | | | | |
|---|---|---|---|---|---|---|
| | week1 | | week2 | | week3 | |
| | *bin-pack* | *FlexLBF* | *bin-pack* | *FlexLBF* | *bin-pack* | *FlexLBF* |
| Server1 | 1260 | 1282 | 1270 | 1286 | 1280 | 1318 |
| Server2 | 688 | 688 | 679 | 702 | 691 | 691 |
| Server3 | 840 | 847 | 850 | 857 | 855 | 862 |
| Server4 | 665 | 732 | 695 | 771 | 695 | 712 |
| Server5 | 251 | 251 | 253 | 253 | 259 | 259 |
| Server6 | 590 | 616 | 540 | 558 | 576 | 642 |

TABLE I. COMPARISON OF BACKUP PROCESSING TIMES UNDER IP-BASED **Bin-Packing** SCHEDULING VS HEURISTIC-BASED **FlexLBF** SCHEDULING.

Overall, these results speak strongly in favor of the proposed heuristic-based FlexLBF scheduler. Note that FlexLBF schedule does not require any special pre-computations: it uses an ordered (by duration) job list. The main disadvantage of the IP-based solution is that it is compute and time expensive. It is difficult to predict the compute time required for generating a good, nearly-optimal solution. While the optimal solutions for Server2 and Server5 had been produced in less than 1.5 minutes, the solution times for Server1, Server3, Server4, and Server6 have required more than 2 hours.

**Scalability study.** To evaluate the scalability of the IP-based approach and FlexLBF heuristic, we've created the *aggregate* workload using workloads from all six backup servers. Workloads of individual backup servers contained 70 - 130 objects. The *aggregate* workload has 665 objects. We have formed the optimized bin-packing schedule for the *aggregate* workload using a backup server configured with 4 tape drives (each drive with the same parameters as considered above). Table II presents the overall backup session time for the *aggregate* workload processed with different schedulers: bin-packing vs. FlexLBF (over three weeks). As we can see, the session time with FlexLBF is up to 25% shorter than with bin-packing schedule. When we used a large-scale workload with 665 objects, the IP-based solution after 2 hours of compute time has produced the job schedule that is still far away from the optimal solution. The IP-based approach *does not scale well* for the large datasets.

| Workload | Backup Session Processing Time (min) | | | | | |
|---|---|---|---|---|---|---|
| | week1 | | week2 | | week3 | |
| | *bin-pack* | *FlexLBF* | *bin-pack* | *FlexLBF* | *bin-pack* | *FlexLBF* |
| *aggregate* | 1620 | 1292 | 1540 | 1464 | 1520 | 1269 |

TABLE II. COMPARISON OF BACKUP PROCESSING TIMES UNDER IP-BASED **Bin-Packing** SCHEDULING VS HEURISTIC-BASED **FlexLBF** SCHEDULING.

In summary, the proposed heuristic-based FlexLBF algorithm is very efficient, produces nearly optimal results, and performs well even for large datasets. Moreover, it enables a simulation framework aimed at system administrators for the automated analysis of different backup management and resource optimization tasks described in the next section.

## VI. OPTIMIZING BACKUP CONFIGURATION AND QOS

The usage of the large-scale enterprise computing environment may fluctuate significantly over time and be impacted by business dynamics, the set of existing applications, and the number of actively used client machines. Therefore, the backup infrastructure tends to be over-provisioned, and often, the same amount of work can be accomplished with a smaller amount of resources. This could additionally *save power and optimize resource usage* without sacrificing the quality of service. Another challenge that system administrators face during backup management is maintaining the duration of the backup window for a set of given backup jobs. We designed a simulation tool that can assist system administrators in evaluating cons and pros of different configuration options for their backup services.

Backup administrators are required to provide the following information to the simulator:

- a description of **backup workload** $W = \{J_1, J_2, ..., J_n\}$, where each backup object $J_i = (O_i, Dur_i, Tput_i)$ is provided with historic information on object duration and object throughput for backup processing.
- a default **backup server configuration** where $K$ - the number of tape drives available in the configuration, $maxTput$ - the maximum throughput of the tape drive; and $maxDA$ - the maximum number of disk agents that can be used concurrently during backup processing.
- a **backup window** $T$ during which the backup service should be performed (the backup window should be larger than the longest backup job in the workload set).

Our simulator is able to evaluate multiple objective functions:

1) the minimal number of tape drives required for processing a given workload with a backup window $T$;
2) the consolidation scenario with a goal of minimizing the overall uptime of active backup servers used for processing with a specified backup window $T$;
3) the load balancing scenario with a goal of minimizing the number of servers in the solution for processing a given workload with a specified backup window $T$ while balancing the load across the participating servers.

The main difference between consolidation and load balancing scenarios is that the consolidation process may lead to an "uneven" object placement across the backup servers in the solution. Let there be N servers in the solution. Since we aim to minimize the overall (aggregate) uptime of these servers while processing a given workload within a specified backup window $T$, the algorithm will try to assign to each of $N-1$ servers the maximum number of jobs that can be processed within $T$. The "last" server will get the remaining fraction of the workload with the session processing time which might be significantly shorter than $T$. Such uneven workload placement will lead to a minimum aggregate uptime of backup servers in the solution, and as a result *may significantly minimize the overall power consumption*. The cumulative uptime of the servers in the solution can be roughly approximated as $T*(N-1)+\Delta$, where $\Delta$ is the session processing time of the "last" server with the remaining fraction of the workload. The consolidation scenario might be especially beneficial when the workload requires two backup servers for processing but only a small fraction of jobs "does not fit" in the backup window when processed by a single backup server. In this case, this small fraction of jobs is assigned and processed by the second backup server, and the aggregate uptime of two servers is $T + \Delta$, where $\Delta \ll T$. If we would balance workload assignment across both servers, the uptime might be much higher and in many cases closer to $2 \cdot T$, and therefore, significantly increasing the power consumption (*practically twice* in this example).

However, a system administrator may also pursue a balanced backup job assignment, where the overall workload gets equally partitioned among the servers in the solution. While it can lead to a higher aggregate uptime of these servers, the benefits are that the amount of bytes written by each server is more balanced. The outcome of consolidation and load balancing scenarios does significantly depend on workload characteristics. For some backup workloads, both scenarios might result in a similar outcome. System administrators should evaluate both options before making a final decision.

Below we outline the analysis and simulation routines performed by our simulator to answer the configuration questions described above. In all the scenarios, there is a common first step, where the simulator checks whether the specified backup window $T$ is equal or larger than the duration $T_{lgst}$ of the longest backup job in a given set. If $T \leq T_{lgst}$ then the solution is infeasible.

**1. Determining the minimal number $N$ of tape drives** required for processing a given workload with a specified backup window $T$. First, we simulate the achievable backup processing time $T_{sim}$ under the FlexLBF algorithm with a single tape drive (i.e., $N = 1$) and specified $maxTput$ and $maxDA$ as shown in Figure 5. If $T_{sim} \leq T$ then the objective is achieved and a given workload can be processed by a single tape drive. Otherwise, if $T_{sim} > T$ then we repeat the simulation cycle with the increased number of tape drives, i.e., $N = N+1$. We stop the simulation once the increased number of tape drives leads to the achievable backup processing time within a backup window, i.e., $T_{sim} \leq T$. The simulator outputs two values: the required number of drives $N$ and the expected (simulated) duration of the backup session $T_{sim}$.

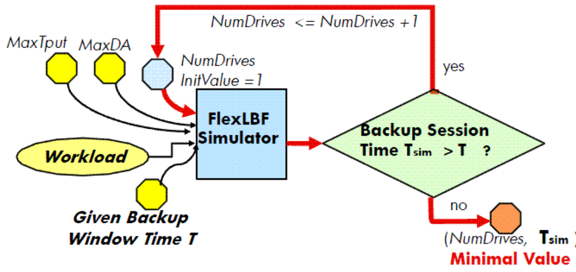

Fig. 5. Simulation routine: determining the minimal number of drives.

**2. Performing a consolidation scenario** with a goal of minimizing the overall uptime of $N$ backup servers required for processing a given set of backup jobs $W$ with a specified backup window $T$.

First, we simulate the achievable backup processing time $T_{sim}$ under the FlexLBF algorithm with a single backup server, where a backup server is equipped with a default number tape drives (i.e., $NumDr = K$) and a specified number of $maxDA$ per drive. If $T_{sim} \leq T$ then the objective is achieved and a given workload $W$ can be processed by a single backup server. Otherwise, if $T_{sim} \geq T$ then we repeat the simulation cycle with the increased number of tape drives, i.e., $NumDr = NumDr+1$. We stop the simulation once the increased number of drives leads to a processing time within a given backup window, i.e., $T_{sim} \leq T$. We sort these tape drives in the order of decreasing processing time per drive and group $K$ drives per a server respectively. The last remaining group may have a number of drives less than $K$. In this case, the simulator replays processing of these jobs by $K$ drives. These jobs are sorted in the list $OrdObjList$ (see Section IV) for simulating their processing by the FlexLBF algorithm.

This way, each server in the solution (except the last one) processes the maximum subset of backup jobs within backup window $T$. The "last" server in the solution may have arbitrary "short" or "long" processing time compared to $T$ (it depends on the workload and this is the reason why the job allocation for consolidation scenario may be uneven.

The simulator outputs: *i)* the number of backup servers required in the consolidation solution, *ii)* the job assignment for each backup server, and *iii)* the backup session time (i.e., backup processing time or uptime) of each server for processing the assigned fraction of the workload.

**3. Performing a load balancing scenario** with a goal of minimizing the number of servers in the solution for processing a given workload with a specified backup window $T$ while balancing the load across the participating servers.

First, we simulate the achievable backup processing time $T_{sim}$ under the FlexLBF algorithm with a single backup server (similar to the simulation under the consolidation scenario). If $T_{sim} \leq T$ then the objective is achieved and a given workload $W$ can be processed by a single backup server.

Otherwise, if $T_{sim} \geq T$ then we repeat the simulation cycle with the increased number of backup servers, i.e., $N = N + 1$. The simulation follows the usual execution of the FlexLBF algorithm where two backup servers are represented by their aggregate number of tape drives. We stop the simulation once the increased number of backup servers leads to the achievable backup processing time within a backup window, i.e., $T_{sim} \leq T$. This is a balanced solution since each of the $N$ servers gets assigned an approximately equal (balanced) portion of the workload for processing. The simulator outputs *i)* the number of backup servers required in the balanced solution, *ii)* the job assignment for each backup server, and *iii)* the backup session time (i.e., backup processing time or uptime) of each server for processing the assigned fraction of the workload.

**Evaluation case study.** To demonstrate the usefulness of the proposed simulation tool we use the *aggregate* workload (with 665 objects) from all six backup servers under study. Table III shows the minimal number of tape drives required for processing a given workload with a backup window T = 1200 min while using a different number of concurrent DAs per tape drive. We perform the simulation for three consecutive weeks.

| $T$ = 1200 min | Required Number of Drives | | |
|---|---|---|---|
| *maxTput*=80MB/s | week1 | week2 | week3 |
| *maxDA*=4 | 7 | 8 | 7 |
| *maxDA*=6 | 5 | 6 | 5 |
| *maxDA*=8 | 5 | 5 | 5 |
| *maxDA*=10 | 5 | 5 | 5 |

TABLE III. THE MINIMAL NUMBER OF TAPE DRIVES (CONFIGURED WITH DIFFERENT $maxDA$) REQUIRED FOR PROCESSING THE AGGREGATE WORKLOAD.

We can see that with the increased number of configured *maxDA* per drive the number of required drives for processing decreases. However, the value of *maxDA = 10* does not lead to improved performance while introducing the higher degree of data interleaving on tapes. The value of *maxDA = 8* provides an optimal performance. As we also see, if the default server configuration uses 4 tape drives then this aggregate workload would require 2 backup servers (they would contain 8 drives overall) for processing with a backup window T = 1200 min.

Table IV shows the combined uptime of two servers required for processing a given workload with a specified backup window T = 1200 min under two different scenarios: consolidation and load balancing job placement. For two scenarios, we also consider a different number of concurrent DAs per a tape drive in the server configuration. The results are interesting: for a backup server with *maxDA = 4* the outcomes of consolidation and load balancing scenarios are practically the same. As we increase the number of *maxDA* per drive the difference between the two scenarios becomes more pronounced: under the *consolidation* scenario the **power consumption** (expressed

| *NumServers=2* | Combined Uptime (min) | | | | | |
|---|---|---|---|---|---|---|
| | week1 | | week2 | | week3 | |
| | *consol* | *load-bal* | *consol* | *load-bal* | *consol* | *load-bal* |
| *maxDA*=4 | 2086 | 2138 | 2392 | 2392 | 2071 | 2100 |
| *maxDA*=6 | 1456 | 1930 | 1660 | 1900 | 1377 | 1950 |
| *maxDA*=8 | 1330 | 1930 | 1490 | 1900 | 1307 | 1950 |
| *maxDA*=10 | 1293 | 1930 | 1460 | 1900 | 1269 | 1950 |

TABLE IV. COMPARISON OF SERVERS' UPTIME WHILE PROCESSING THE AGGREGATE WORKLOAD UNDER **consolidation** AND **load balancing** SCENARIOS.

as the combined uptime of the backup servers in the solution while processing a given workload) is **up to 50% lower** than under the *load balancing* approach.

The runtime of the simulator depends on the number of iterations and backup jobs: simulation of 100-300 jobs with 3-5 iterations finishes under 1-2 min. Therefore a system administrator can efficiently use the simulator for understanding the outcome of many different *what–if?* scenarios.

**4. Performing QoS job management to support tailored data restore rates.** When planning and scheduling backup sessions, a system administrator has to take into consideration the job's QoS requirements. Typically, there are two parts in QoS requirements: the desirable backup window as well as the desirable data restore rates. The targeted backup window is supported via creating the appropriate backup groups processed with the FlexLBF schedule as discussed earlier. The object restore speed depends on the drive configuration parameter (its concurrency degree) at the backup time of the object. A higher number of concurrent DAs causes a higher degree of data interleaving on tape. This results in higher restoration times when retrieving such data compared to a continuous, non-interleaved data stream written by a single disk agent. Typically, the restore rates are measured using a set of microbenchmarks with different number of concurrent DAs and published in the corresponding product datasheets. The object's QoS requirements are defined using these classes.

To simplify the notation of QoS restore classes, we will use an explicit concurrency degree parameter that corresponds to the required restore rates in the QoS job's description. Therefore, each job $J_i$ is represented by a tuple: $(O_i, Dur_i, Tput_i, Conc_i)$, where $O_i$ is the name of the object, $Dur_i$ denotes the backup duration of object $O_i$ observed from the previous full backup, $Tput_i$ denotes the throughput of object $O_i$ computed from the previous full backups, and $Conc_i$ reflects the concurrency degree parameter that corresponds to the restore rates in the job's description. If an object does not have specific requirements for the restore rates then we can use $\infty$ in the QoS specification (meaning the best effort).

For a given workload and a specified 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 on decreasing the backup window. Often, a higher value of concurrent DAs might not result in a better performance. 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. For example, Table III shows that for processing a consolidated workload from the six HP Labs backup servers we would need 5 tape drives with $maxDA = 8$. By increasing the concurrency degree higher, i.e., $maxDA > 8$, we do not get a shorter backup session, but only an increased data interleaving on the tape. Moreover, if there are 4 drives in the server configuration, the *aggregate* workload will require two backup servers anyway, and in this case, one can guarantee much higher restore rates with $maxDA = 4$ while using the same amount of resources (in case, if the power consumption and the servers' uptime are not driving consideration factors).

The proposed simulation environment allows to assess different backup management scenarios and estimate time required for processing the specified QoS job classes while making the best use of the available system resources.

The introduced FlexLBF algorithm can be generalized to handle job's QoS requirements without an additional "hassle" of partitioning jobs into different QoS classes for processing by tape drives configured with the required concurrency degree. To achieve this goal, an additional control on the concurrency degree (required by the job) can be directly introduced in the job admission module of FlexLBF. That is, for each job $J_i$ that needs to be scheduled, the algorithm verifies whether there is a tape drive $Tape_j$ with the number of active DAs less than the required (by the job's QoS) concurrency degree (i.e., $ActDA_j < Conc_i$) and whether the job throughput does not exceed the currently available throughput at $Tape_j$. The proposed FlexLBF enhancement aims to achieve both performance goals: to optimize the session processing time and to respect the individual job's QoS requirements.

## VII. Related Work

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. For minimizing the *makespan*, i.e., the schedule length, a promising approach is to schedule the longest job first [9], [21]. In [9], 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. Our work is another example in this direction, showing that the efficient heuristic may be a good alternative to the integer programming approach for building the efficient backup scheduling in practice.

Traditionally, magnetic tapes has been used for data backup in enterprise environments. Well-known Unix utilities such as dump, cpio, and tar [16] 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. Existing commercial backup tools [6], [10], [11], [13], [18] 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 [16]. However, within the created backup groups a random job scheduling is used which can lead to inefficient backup processing and increased backup time.

Large-scale distributed systems such as Grid exploit job scheduling as the main component of resource management. Grid-related job scheduling aims to empirically evaluate a variety of scheduling heuristics. Here are a few policies used to improve system utilization and throughput: backfilling [14], adaptive scheduling [12], and task grouping [19].

There is a growing number of services and systems that provide efficient filesystem backups over the Internet [20], [7]. Most of the Internet-based backup services do use the random job scheduling. They will benefit from a more efficient job scheduling as well. We believe that the proposed automated management framework is useful for dealing with the dynamics and optimization of these backup services.

## VIII. Conclusion and Future Work

Despite that for most businesses, backup is a daily operations to protect the business critical assets, there are still many inefficiences in the backup processes. In this paper, we revisit a traditional backup job scheduling and discuss inefficiencies due to random job scheduling broadly used in the backup tools. The main reason why random job scheduling is typically used by backup tools is that, traditionally, no information on the expected duration and throughput requirements of different backup jobs is provided. In this work, we explicitly characterize each backup job via two metrics: job duration and job throughput. These metrics are used in the automated design of a backup schedule for minimizing the overall completion time of a given set of backup jobs. We compare the efficiency of the IP-based schedule and the heuristic-based FlexLBF job schedule. Our analysis reveals that the simple FlexLBF heuristic augmented with adaptive number of active DAs is close to optimal while having no additional computing overhead compared to the compute expensive IP-based solution. The simulator based on FlexLBF enables the analysis to minimize the number of actively used backup servers while meeting backup window constraints and satisfying data restore rates. As a result, the proposed approach significantly increases the performance, reliability, and quality of implemented solutions.

## References

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

[2] L. Cherkasova, A. Zhang, X. Li: DP+IP = Design of Efficient Backup Scheduling. Proc. of the 6th International Conference on Network and Service Management (CNSM'2010), Niagara Falls, Canada, 2010.

[3] L. Cherkasova, R. Lau, H. Burose, S. V. Kalambur, B. Kappler, K. Veeranan: Run-time Performance Optimization and Job Management in a Data Protection Solution.Proc. of the 11th IFIP/IEEE Symposium on Integrated Management (IM'2011), Dublin, Ireland, 2011.

[4] P. Chisholm: Is Your Company Prepared to Recover From an IT Disaster? http://www.certmag.com/read.php?in=3310

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

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

[7] B. Fitzpatrick. Brackup. http://code.google.com/p/brackup/

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

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

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

[11] 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

[12] 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.

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

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

[15] Optimizing data protection. White paper.http://www.storagesearch.com/storagetek-art2.pdf

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

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

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

[19] L. Tan and Z. Tari. Dynamic task assignment in server farms: Better performance by task grouping. Proc. of the ISCC, July 2002.

[20] M. Vrable, S. Savage and G. Voelker. Cumulus: Filesystem Backup to the Cloud. Proc. of the FAST'2009.

[21] 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.