A Markov Decision Process Approach for Optimal Data Backup Scheduling

Ruofan Xia¹, Fumio Machida², and Kishor Trivedi¹
¹Department of Electrical and Computer Engineering, Duke University, Durham, USA
{ruofan.xia, ktrivedi}@duke.edu
²Knowledge Discovery Research Laboratories, NEC Corporation, Kawasaki, Japan
f-machida@ab.jp.nec.com

Abstract—The explosive growth of data generation and increasing reliance of business analysis on massive data make data loss more damaging than ever before. Nowadays many organizations start relying on cloud services for keeping their valuable data. It is a critical issue for cloud service provider to protect the data for individual users securely and effectively. To protect the data in a system with multiple data sources, backup schedule plays an important role for achieving the desired level of data protection while minimizing the impact on system operation. In this paper we investigate the use of Markov Decision Process (MDP) to guide the scheduling of data backup operation and propose a framework to automatically generate an MDP instance from system specifications and data protection requirements. We then demonstrate the benefits of the MDP approach.

Keywords—data backup, scheduling, Markov decision process

I. INTRODUCTION

Today businesses are increasingly relying on data analytics to provide insight into their operation and guide decision-making processes. Due to the increase in the business users of cloud servers, maintaining their data properly becomes a task of great importance for cloud service providers. Unfortunately, data can be lost due to various reasons [1]. Files may be corrupted by unintentional modification or malicious behaviors. Data may become inaccessible due to failures of storage system components. Natural disasters may destroy the physical infrastructure which stores important data. Impact from such losses ranges from reduced productivity and capability to carry out normal operation, to significant financial losses, damaged business reputation and customer confidence [2]. Thus cloud service providers need adequate protection for users’ important data against such eventualities. While some cloud service providers may offer replication of the user data across geographically distributed sites, data backup are still necessary for users requiring data archive. Moreover, since data replication itself does not tolerate to data loss caused by operational errors or malicious activities, it is important to take backup of the data in a periodic manner. The frequency of data backup could depend on the users’ business requirements to specify the acceptable damage due to data loss. Two frequently used requirement terms are Recovery Point Objective (RPO) and Recovery Time Objective (RTO) [1]. For a data source, RPO defines a time point in system history so that a recovery procedure must be able to recover the system status at that epoch. On the other hand, RTO specifies the maximum length of the recovery period that is acceptable. These requirements deal with the data loss and downtime aspects of data source failures respectively, and serve as part of the design and operational guidelines for the data sources.

Backup scheduling plays an important role in meeting such requirements under administrative constraints, but it is typically a non-trivial task. The reason for this is two-fold. First, data backup may incur cost associated with downtime or performance overhead. The execution of data backup may affect normal system operation, by suspending data access (for purposes such as maintaining data consistency) or by consuming system resources and affecting production workload execution. Second, data backup must take into consideration various factors in system operation, such as data priorities and access methods.

In this paper we investigate the application of Markov decision process (MDP) [3] to data backup scheduling. We present a framework that translates a system specification, consisting of different data sets and their required levels of protection, along with the amount of resource available for backup execution, into an MDP instance. The framework takes into consideration the failure/recovery behaviors of system components and provides optimal backup operation scheduling so as to minimize downtime. We organize the paper as follows. Section II provides a short introduction to MDP. Section III details the type of system that we investigate. Section IV presents the framework to translate the system scenario into an MDP instance. Section V provides numerical results from our investigation and relevant discussions. Section VI discusses related work. Finally, Section VII concludes the paper with directions for future work.

II. A BRIEF INTRODUCTION TO MDP

The Markov Decision Process (MDP) [3] is a state-based optimization formalism consisting of the following components:

1) A set of states, \( S \), that describe the status of the system being modeled.
2) A set of actions \( A(s) \), \( s \in S \) that specifies the ways the system may evolve from the state \( s \). The actions form the “control variables” of the optimization problem.
3) Decision points, which are the time points at which an action can be chosen from those available based on the current state \( s \).
4) A transition probability function \( P(s' | s, a) \) which defines the transition probability into a given new state \( s' \) at the next decision point, conditioned on the state \( s \) and the action a chosen at the current decision point.
5) A reward function \( R(s' | s, a) \) which defines the (expected) reward that will be generated when the system goes into
a new state $s'$ at the next decision point, conditioned on the state $s$ and the action $a$ at the current decision point.

As its name suggests, MDP possesses the Markov property in the sense that the system evolution beyond a decision point depends only on the system state and the action chosen at that point. The theory of MDP indicates that it is sufficient to locate a stationary policy to achieve optimality [3], meaning that there is no need to consider the past history when making a decision about which action to perform in a given state. Thus the goal of solving an MDP is to generate a mapping from the states to their actions, i.e. a stationary policy, so that in a given state an action is known to be the optimal choice to control system evolution from that point. Once a stationary policy is obtained, the transitions (and their probabilities) from each state become fixed, and the MDP becomes a (discrete-time) Markov chain. The optimality in an MDP is in the sense of maximizing or minimizing some reward criteria during the evolution of the system. There are several different reward criteria: the expected total reward, the expected discounted reward, and the expected time-averaged reward [3]. The expected total reward criterion is most commonly applied to finite-horizon MDPs, where the number of decision points is finite. By contrast, the expected discounted reward and expected time-averaged reward criteria can be applied to infinite-horizon MDPs, where the number of decision points is infinite (but typically countable). Of these two, the expected discounted reward is easier to apply (and consequently more widely used) than the expected time-averaged reward criteria. Therefore, in this paper we focus on the expected discounted reward criteria.

III. SYSTEM SCENARIO

Consider a cloud system maintaining multiple data sources (e.g. sets of files that are managed together, databases, etc.) concurrently. In normal operation, the data sources accumulate data changes which should be stored securely by the system. However, data sources may experience failures (caused by, e.g., logical errors in the program managing the data or failure of hardware pieces hosting data) during operation which lead to loss of the accumulated data change. In addition to such losses, data source failures and the subsequent recovery also introduce additional downtime. In order to protect against data loss, there are periodic backup points during system operation at which the data sources may take backups. We assume the data sources accumulate data, fail and recover independently, and that there is no common failure mode that affects multiple data sources at the same time. We also assume that a data source is unavailable for accepting data change during a backup and recovery.

We assume that the set of concurrently executing backup operations are determined at each backup point, and thus we call these points the ‘decision points’, while the period between two neighboring decision points is the ‘decision interval’. The ‘decision’ may be interpreted as whether to execute backup for a given data source. Thus at each decision point, a data source may execute backup or continue normal operation. It may also happen that a data source is recovering from a previous failure at the beginning of the backup window. In that case we assume: 1) the data source can still be scheduled for backup; 2) the data source will recover before the next decision point, and 3) if the data source is scheduled for backup, the backup will execute after the recovery and complete before the next decision point. As a simplification we assume backup operations do not fail, and when a backup is complete all the data processed is no longer subject to loss. An illustration of the system with two data sources is given in Fig. 1, where the arrows represent the backup choice (whether to backup, what type to perform) for each data source at the decision points.

If a data source executes backup, it can be a full backup or a partial (incremental) backup. A full backup processes all the data and produces a copy from which alone the data source can be recovered (to the status when the backup was made), while a partial backup covers the data changes since the last backup and would need the nearest full backup in the past, in addition to all the partial backups in between, to restore data to the moment when the partial backup was made. Thus in our context a completed full backup removes all the accumulated partial backups since the latter are no longer needed for recovery. Execution of a backup also requires some system resource (e.g. network bandwidth to transfer the processed data), and there may not be enough resource to accommodate concurrent backup execution for all data sources. This fact may require some data sources to backup more/less frequently, and serves as both the primary connection among the data sources (recall that we assume they behave largely independently of each other) and one of the factors that make scheduling nontrivial. In this work we do not distinguish between different types of resources and combine them into a single abstraction. Regarding the time it takes to perform a backup or a recovery, we assume that, for a given data source, a full backup takes more time than a partial backup. For recovery, we assume each recovery of a data source takes firstly a certain amount of time to restore the previous full backup copy, and then some additional time to restore the copies of partial backups that have been taken since that full backup, up to the moment of the current failure. The second time quantity is assumed to be proportional to the number of partial backups involved, and may be zero if the closest backup in the past was a full backup.

As described earlier, each data source has requirements for protection in the form of RPO and RTO. In our context, the RPO defines the maximum number of backups that a data source may consecutively skip, while the RTO defines how many partial backups may be accumulated by a data source before a full backup should take place. We assume that the operational goal of the system is to minimize the average data source downtime while satisfying the RPO/RTO requirements of all the data sources.
Because data sources may fail, the scheduling should also take into account their failure and recovery behaviors. Since the focus of the paper is on the MDP-based framework, we assume a simple availability model (for each data source, since we consider them to behave largely independently) which consists of an up state ("U"), a backup execution state ("B") and a down state ("D"), as shown in Fig. 2.

![Fig. 2. Availability Model](image)

In Fig. 2 we assume the following firing time distributions for the transitions:

1) The firing time of the failure transition (the thin arrow from "U" to "D") is exponentially distributed.

2) The firing time of the recovery transition (the think arrow from "D" to "U") is general-distributed with bounded supports, and the upper bound is smaller than the time between two backup windows.

3) The firing time of the data backup initiation transition (the hollow arrow from "U" to "B") is deterministic to model data backup schedules.

4) The firing time of the backup completion transition (the thick arrow from "B" to "U") is general-distributed with bounded supports, and the upper bound is smaller than the time between two backup windows.

We assume bounded supports for the backup and recovery completion times because in practice it is generally a priority in system administration to complete backup and recovery within a given timeframe. The above assumptions result in a semi-Markov model for data source availability, and the data source is considered unavailable in both the down and backup states. The failure rate is given and shared by the availability model and the MDP model, while the backup schedule generated by the MDP will be used to determine the parameter values for the non-exponential distributions. Finally the availability model will be solved to yield data source uptime.

IV. FORMULATE BACKUP SCHEDULING PROBLEM AS MDP

We construct the state space of the MDP from the status of the data sources. We denote the status of data source \( i \) as \((X_i, Y_i)\) where \( X_i \) represents the failure status of data source \( i \), \( X_i \) is the number of decision intervals since the last backup of data source \( i \), and \( Y_i \) is the number of partial backups taken since the last full backup. The variable \( X_i \) is either \( U \) or \( D \) according to the availability model in Fig. 2. Note that this variable will not take a value of \( B \), to indicate backup, since that is captured by the actions of MDP as will be clear shortly. The second variable is the number of successive backup points where the data source skips backup. Thus we are assuming that a data source accumulates data changes at a uniform rate so long as it is operational. The maximum values of \( X_i \) and \( Y_i \) are directly connected to RPO, and RTO, the RPO and RTO requirements of data source \( i \). Specifically, RPO, determines how many backups data source \( i \) may skip consecutively, while RTO, specifies the maximum number of accumulated partial backups, since in case of a failure the recovery time is related to the number of partial backups already taken. Thus RPO, and RTO, would set the largest values for \( X_i \) and \( Y_i \).

The action of the MDP is constructed from the backup choices of each data source at a backup point. Each data source normally has three choices: continue normal operation, take a partial backup, or take a full backup. The RPO and RTO requirements dictate whether a given choice is possible under the current status for that data source. For example, if a data source has reached its maximum allowed number of skipped backup then it must perform one, and if it has also reached its maximum number of partial backups then a full backup is the only choice. The possible actions of the MDP in a given state (which summarizes the statuses of all data sources) are then constructed through combination of data source choices, with the additional constraint that the number of data sources executing backup cannot be larger than the amount of system resource.

In a similar fashion, the transition probabilities and rewards of MDP actions are also constructed from the transition probabilities of obtaining entering corresponding new statuses from all data sources, while the reward is the average of those from all data sources. This is a result of the assumption that the data sources behave independently from each other (aside from the resource constraint). For a data source, the reward (or penalty) is the downtime that this source can expect to incur during its operation. The computation of these quantities are described below, where \( X \) is a positive integer, \( Y \) is a nonnegative integer, and the terms \( t \), \( \gamma \) and \( \tau \) are the time variables of different probability density functions.

1) If the current status is \((U, X, Y)\) and no backup is performed, the possible new status are \((D, X', Y)\) and \((U, X+1, Y)\), which correspond, respectively, to the cases of the data source being down or up at the next decision point. The former happens if the data source does not recover from a failure that occurs within the current decision interval, while the latter happens if either there is no failure in the interval, or if there is one failure but the system recovers before the next decision point. The respective probabilities are computed via an exponential distribution with the same rate parameter as in the availability model (e.g. Fig. 2). For the reward, the downtime is the expected time to recovery from a failure. The exact formulas are presented below.

New status: \((U, X+1, Y)\), if no failure occurs or when the source recovers from one before the next decision point.

\[
\text{Prob}=\text{Prob}_{1} + \int_{0}^{T-U} f(t) g(t) R(T-t) dt + \int_{T-U}^{T} f(t) g(t) R(T-t) dt + \int_{T}^{\infty} f(t) g(t) R(T-t) dt
\]

\[
\text{Prob}_{1} = \frac{e^{-\lambda (T-U)} \gamma (T-U)}{\lambda U_{i}Y_{i} (e^{-\lambda T} - e^{-\lambda U_{i}}) (1 - e^{-\lambda Y_{i}})}
\]

\[
\text{Prob}_{2} = \frac{e^{-\lambda (T-U)} \gamma (T-U)}{\lambda U_{i}Y_{i} (e^{-\lambda T} - e^{-\lambda U_{i}}) (1 - e^{-\lambda Y_{i}})}
\]

\[
\text{Prob}_{3} = \frac{e^{-\lambda (T-U)} \gamma (T-U)}{\lambda U_{i}Y_{i} (e^{-\lambda T} - e^{-\lambda U_{i}}) (1 - e^{-\lambda Y_{i}})}
\]

2) If the current status is \((D, X', Y)\) and no backup is performed, the possible new status are \((D, X', Y')\) which correspond, respectively, to the cases of the data source being up or down at the next decision point. The respective probabilities are computed via an exponential distribution with the same rate parameter as in the availability model (e.g. Fig. 2). For the reward, the downtime is the expected time to recovery from a failure. The exact formulas are presented below.

New status: \((D, X+1, Y)\), if a failure occurs and the source has not recovered by the next decision point.

\[
\text{Prob} = \int_{T-U}^{T} f(t) g(t) R(t) dt + \int_{T}^{\infty} f(t) g(t) dt
\]
\[ E[\text{Prob}] = E[g(t)] + \frac{V_{1} \cdot \lambda_{1}}{2} \quad \text{Prob} = \text{Prob}_1 + \text{Prob}_2 + \text{Prob}_3 \]

In the above equations, \( T \) stands for the length of the decision interval, \( Re \) stands for reward, and \( E(\cdot) \) stands for expectation. \( R(t) \) is the reliability of the data source, i.e. the probability that the data source does not fail within time \( t \). The terms \( f(t) \) and \( g(t) \) stand for the density functions of time-to-failure and time-to-recover, while \( U_r \) and \( L_r \) represent the upper and lower bound of the latter. The term \( \text{Prob}_1 \) is the probability that the data source recovers from a failure within \( T \), and the terms within the two integrations in (1) are to the probabilities that the data source recovers before the next decision point conditioned on different instant of failure occurrence. The integrations then unconditional to yield the total probability of source recovering timely. Equation (2) uses a similar approach, changing the terms within the integrations to be the conditional probabilities that the source does not recover. Finally, the \( \text{Prob} \) terms in (1) and (2) are normalized by their sum to reflect the assumption that at most one failure can occur in an interval.

2) If the current status is \( (D, X, Y) \) and no backup is performed, due to the assumption of at most one failure within a decision interval, there will be no failure before the next decision point. Thus the only possible new status is \( (U, 1, Y) \) with transition probability one, while the expected downtime will both be zero (the amount of time required to recover has been included in the downtime of the backup choice that the data source selected at the previous decision point).

3) If the current status is \( (U, X, Y) \) and a backup (full or partial) is scheduled, there are four potential new statuses, \( (U, I, Y + 1), (D, I, Y + 1), (U, I, 0) \) and \( (D, I, 0) \). The first two correspond to the cases where partial backup is performed, while the latter two occurs when full backup is executed. The first and third cover the cases where the data source is operational at the next decision point, regardless of whether a failure occurs. The other two cases correspond to the situation that the data source has not recovered from a failure by the next decision point. The transition probabilities and expected reward terms are defined as below.

New status: \( (U, I, Y + 1) \) or \( (U, I, 0) \), a partial/full backup is executed and the source is up at the next decision point.

\[ \text{Prob}_1 = \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \left( \frac{U_r}{U_r - L_r} \right) \left( \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \right) \]

\[ \text{Prob}_2 = \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \left( \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \right) \left( \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \right) \]

\[ \text{Prob}_3 = \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \left( \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \right) \left( \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \right) \]

In (3), \( \text{Prob}_1, \text{Prob}_2 \) and \( \text{Prob}_3 \) stand for the probability of no failure after backup, one failure with the source re-covering timely, and one failure with the source not recovering. Most of the other terms in (3) and (4) have identical meanings to those in (1) and (2) earlier. Of the new ones, \( b(t) \) stands for the density function of time-to-backup-completion, while \( U_r \) and \( L_r \) stand for its upper and lower bounds. The method of derivation is also similar to those for (1) and (2), in that (where applicable) we first condition on the duration of backup, then on the occurrence time of failure, and finally on the time of recovery execution. In (3), the first probability terms is the probability of no failure after the backup, while the other two together express the probability of a failure after backup and the later successful recovery of the source. Similarly, (4) computes the probability that a failure occurs from which the source does not recover by the next decision point.

4) If the current status is \( (D, X, Y) \) and a backup (full or partial) is scheduled, then the new status would be either \( (U, I, Y + 1) \) or \( (U, I, 0) \) depending on the type of backup. Due to the assumption of at most one failure, the second variable cannot be zero at the next decision point. In both cases, the probability would be one and expected downtime is the expected backup execution time, as below.

\[ \text{Prob}_1 = \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \left( \frac{1}{2} \left( \frac{U_r}{U_r - L_r} \right) \right) \]

Algorithm to construct an MDP instance for a given system:

1) Construct the MDP state-space from the combination of statuses of all data sources.

2) Generate a reachability graph from the states, where a state can reach another state if there is a valid action (i.e. one that does not exceed resource constraint) that will cause the system to transition from the former to the latter in the case of no failure occurrence.

3) Prune the state space: do a depth-first search through the reachability graph. Any state that cannot reach itself is a state on a path leading to RPO and/or RTO violation. All such states are removed from the MDP state-space.

4) Construct the transition and reward matrices of the MDP. Check each state for possible actions and decide the
next-states and corresponding transition probabilities and rewards, according to the formulas in (1)–(5).

A simple MDP example, consisting of one data source with an RPO of 2 and RTO of 2 to illustrate the approach above, is shown in Fig. 3. The values for transition probabilities and rewards, as well as a few states that are not part of the normal backup cycle (e.g. (U,0,0)), are omitted for clarity. Notice that in some states the set of actions becomes restricted. For example, in (U,2,0) a backup must be taken, while in (U,2,2) only full backup is possible.

The solution of the MDP provides an optimal policy in the form of the action to choose for each data source, based on their status at a decision point, so as to minimize the average downtime from all data sources.

V. QUANTITATIVE INVESTIGATION

We consider a system with four data sources each described with an availability model as in Fig. 2. The data sources have different parameter values and requirements, and we compare the expected average annual downtime resulting from some heuristic backup schedules and the optimal one from the MDP. We describe the parameters as presented in Table I, where U(a, b) represents the uniform distribution between a and b. All time quantities in the table are in the unit of hours. These values are chosen based on the assumptions on the relative magnitudes of various time quantities made earlier.

<table>
<thead>
<tr>
<th>Table I. Data Source Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Time to complete a partial backup</td>
</tr>
<tr>
<td>Time to complete a full backup</td>
</tr>
<tr>
<td>Time to recover from a partial backup</td>
</tr>
<tr>
<td>Time to recover from a full backup</td>
</tr>
<tr>
<td>Mean time to failure (10^6)</td>
</tr>
</tbody>
</table>

Our primary comparison is between several heuristic backup schedules and the policy from the MDP when given different data source RPO/RTO requirements. The heuristic backup schedules used in the comparison are shown in Table II.

<table>
<thead>
<tr>
<th>Table II. Heuristic Backup Schedules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
</tr>
<tr>
<td>h1</td>
</tr>
<tr>
<td>h2</td>
</tr>
<tr>
<td>h3</td>
</tr>
<tr>
<td>h4</td>
</tr>
</tbody>
</table>

The first set of comparison, that takes place in situations where the system resources allow concurrent backup execution for all sources, is presented in Table III.

<table>
<thead>
<tr>
<th>Table III. Average Downtime With Concurrent Backup</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPO/RTO requirements</td>
</tr>
<tr>
<td>(P1,T1), (P2,T2), (P3,T3), (P4,T4)</td>
</tr>
<tr>
<td>(1, 3), (5, 2), (4, 3), (2, 1)</td>
</tr>
<tr>
<td>(1, 3), (5, 3), (5, 1), (3, 4)</td>
</tr>
</tbody>
</table>

In Table III, each row lists the downtime produced by different schedules under a particular set of values for the data source RPO and RTO requirements. As the results shows, the MDP-based approach can generate better schedules, in terms of downtime, than the common heuristics. Note that these downtime values are high due to the fact that we restrict our attention to traditional, offline backups. The investigation of more advanced data protection techniques in the context of the MDP framework is part of our future work.

Next we consider the situations where the available system resource does not permit all data sources to execute backup simultaneously. If there is no resource constraint, the sequence of backups for each data source forms a period defined by its RTO, RPO and other parameter values. However, since the periods of multiple backup sequences may all terminate at the same backup points, such points carry the risk of overloading system resource and violating RPO/RTO requirements. Such a situation may be avoided by shifting the backup periods of some data sources, but this task may not be straightforward for some values of RTO/RPO requirements, and it is generally still difficult to argue about the optimality of the resulting schedule. By comparison, the schedule produced by the MDP in such cases arranges the backup operations of all the data sources in a way, potentially changing the periods of some data sources, so that at any decision point the data sources performing backups will never require resource more than is available to the system (assuming, of course, this is possible in the first place). Furthermore, the theory of MDP guarantees that the schedule thus found is optimal in terms of the reward structure (in our case, in terms of the average downtime).

To illustrate the above point, we will look at a scenario where the amount of system resource does not permit concurrent backup for all data sources. To simplify the presentation, we consider a case of three data sources with the following RPO/RTO requirements: (3, 2), (4, 3), (5, 5). The system has...
only one unit of resource available, meaning that it can only perform one backup at a time. In this case, direct application of the four heuristics described earlier will be problematic: h1 and h2 are obviously not applicable, while the latter two will run into resource violation at some points, e.g. if all three sources start from the status (0, 0), then at time point 12 or 15, among others, the system resource becomes insufficient. In this case, shifting the starting points of the periods will not solve the problem. The only solution is more frequent backups for some sources, but it is difficult to find the optimal way to do this. To see the complexity associated with the optimal arrangement, we compare the results from the following new heuristic and those from the MDP: h4, each data source executes backup every three days, and always prefers partial backup over full backup. It is easy to see that this heuristic yields a feasible schedule, but it is not optimal as the following results show.

<table>
<thead>
<tr>
<th>Schedule</th>
<th>Downtime [hours]</th>
</tr>
</thead>
<tbody>
<tr>
<td>From h4</td>
<td>779.11</td>
</tr>
<tr>
<td>From MDP</td>
<td>425.27</td>
</tr>
</tbody>
</table>

It may be straightforward to understand the suboptimality of the h4 policy, since it calls for more frequent backups for some data sources. On the other hand it is difficult to know what the optimal schedule is without a formal framework such as the MDP-based approach. Although the details is omitted here due to space constraint, we note that the optimal policy from the MDP induces a sequence of periodic status changes for each data source, where the backup choice of a data source is affected by both its parameter values (e.g. the upper/lower bounds), requirements as well as those from other data sources. Such consideration is needed to achieve optimality, but would be difficult to carry out without the MDP-based approach.

VI. RELATED WORK

Due to the importance of data protection, there has been much work devoted to this topic. Some research work focuses on design and implementation of new techniques to improve effectiveness of data backup and other protection techniques, for example [4] and [5] focus on the technique of data de-duplication, while [6] and [7] investigate online backups. Some other works provide overview on general issues and techniques, perhaps in specific contexts, such as [8]. There are also works that focus on modeling and evaluation of data backup techniques and strategies, such as [9] [10] and [11]. Our work here differs from these examples in that we focus on designing of optimal data backup schedule.

There are also works on effective design and/or management of data backup operations. Many of these works utilize some form of optimization. For example, [12] focuses optimization of data backup intervals, [13] considers optimal data placement and level of replication, while [14] investigates the design of a storage solution for a specific context. By comparison, our work focuses on backup scheduling, with the application of MDP which allows a large set of practical scenarios to be modeled and optimized for.

VII. CONCLUSION AND FUTURE WORK

In this paper we presented our work on applying Markov Decision Process (MDP) to data backup scheduling. Our framework allows the translation of several data- and system-related requirements into an MDP instance, so that the solution to the instance provides the optimal schedule that minimizes system downtime while satisfying the requirements. The subsequent numerical investigation and discussion establishes the effectiveness and importance of such formal framework in the context of data backup operation management. In future work, we will improve the scalability of the framework, so that systems of realistic sizes can be modeled.

ACKNOWLEDGMENT

The authors would like to thank Prof. Vidyadhar G. Kulkarni for his comments and feedback on this work.

REFERENCES