Mathematical Model of Data Backup and Recovery

Karel Burda

The Faculty of Electrical Engineering and Communication
Brno University of Technology, Brno, Czech Republic

Summary
The data backup and data recovery are well established disciplines within the computer science already. Despite this fact, however, there is currently no sufficiently general mathematical model for quantification of backup strategies. In this paper, such mathematical model is derived. The model introduced is based on the assumption that the total size of the data is constant. Using this model, we can quantitatively evaluate the properties of different types of backup strategies. For the full, differential and incremental backup strategies, formulae are derived for the calculation of the average total backup size and the average data recovery size in dependence on the probability of the change of data units. The quantitative relations obtained are discussed.

Key words:
Data backup, data recovery, mathematical model.

1. Introduction

Data are one of the most valuable assets of persons and organizations. However, we can lose these data due to technical failures, human errors, etc. This is the reason why data are copied into backup data repositories (so-called data backup) in order to have the possibility of reconstructing these data when the data are lost in the main data repository (so-called data recovery).

With regard to mathematical models of backup processes, A. Frisch states that "studies performing numerical and quantitative modeling backup processes are surprisingly rare" (see [1], p. 234). The models published so far allow only computations of full and incremental backups (e.g. [2], p. 4) and the quantification of the Amanda backup scheme ([3], p. 745-750 or [1], p. 225-229). Z. Kurmas and L. Chervenak have published a comparison of some backup strategies [4], however, their evaluation of backup strategies is based on simulations and therefore no analytic results are presented in this paper. Other research papers are oriented on backup frequency and timing (e.g. [5] or [6]) and also do not provide any suitable results for the evaluation of general backup strategies. Therefore, we can state that there is currently no sufficiently general mathematical model for quantification of data backup and recovery, despite the great importance of both of these procedures. This paper introduces a mathematical model of data backup which allows a quantitative comparison of different backup methods.

Manuscript received July 5, 2014
Manuscript revised July 20, 2014

In the next section, the basic terms are formulated and the process of data backup and recovery is formalized. The third section is dedicated to the schemes of data recovery and the fourth section deals with the atomic backup (backups based on snapshots of data units). In the fifth section, a mathematical model for the quantification of backups is derived and in the sixth section, this model is applied to different data backup methods and the results obtained are discussed.

2. Data backup formalization

The terminology of the data backup is not unified and many authors "are using the basic terms in different and often conflicting ways" [7]. Therefore, we specify and formalize basic terms at first.

Data D are a structure of symbols, which represents some information. In the data repository, these data are arranged into certain elementary (i.e. further indivisible) sequences of symbols (e.g. sectors of the hard disk). We refer to an elementary sequence d of symbols as a data unit and denote the x-th data unit as d_x. In the course of time, the data unit d_x can correspond to different sequences of symbols, which we call versions of the data unit d_x. The data unit d_x, which does not represent any information, is the so-called empty data unit. We will formally denote such a data unit as $d_x = -$. The model is based on the assumption that the total size of the data D is constant and that the size of the metadata (overhead of data units) is negligible.

We have already stated that the symbol sequence of the data unit d_x varies with time t. From the backup viewpoint, the important events are changing the data units and taking the backups. To simplify description, we assume that the events mentioned can not come simultaneously and changing the data units and taking the backups are instantaneous, i.e. the duration of these events is equal to zero. When the data unit d_x has been changed at the time t_i, we denote this version of the data unit as $d_x(t_i)$. This version is valid in the time interval $\langle t_i, t_j)$, where t_j is the instant of the next change of the data unit d_x. The quantity $d_x(t_i)$ can also represent a version of the data unit which is valid at the time t_i. In our model, we assume that if $j > i$, then $t_j > t_i$.

We will denote the data at the instant t_i as $D(t_i)$ and assume that these data consist of n data units $d_1(t_i)$ to $d_n(t_i)$. Then, we can formally represent the data as an array of n data units:

$$D(t_i) = \left[d_1(t_i), d_2(t_i), \ldots, d_n(t_i)\right] = \left[d_x(t_i)\right]_{x=1}^{n}. \quad (1)$$

The basis of any arbitrary type of backup is the full backup. The full backup $F(t_i)$ is a record of all data units which were valid at the time t_i, i.e.:

$$F(t_i) = D(t_i). \quad (2)$$

Full backups are large, because they also contain data units which have remained unchanged since the previous backup. Therefore, these backups are combined with partial backups, which contain only data which have been changed compared to some previous backup.

Let $D(t_j)$ be the data at the instant t_j and let $D(t_i)$ be the data at the instant t_i. Denote by $b_x(t_i, t_j)$ the change of the data unit d_x at the instant t_j in contrast to the instant t_i. It holds for this change:

$$b_x(t_i, t_j) = \begin{cases} -, \text{ when } d_x \text{ has been deleted,} \\ 0, \text{ when } d_x(t_i) = d_x(t_j), \\ d_x(t_j), \text{ otherwise.} \end{cases} \quad (3)$$

The first line expresses the situation when the data unit d_x has been deleted, the second line represents the situation when there has been no change in the data unit d_x and the third line corresponds to the situation when the data unit d_x has been either created or modified.

Then, the partial backup $P(t_i, t_j)$ is a total of changes of all data units within the interval $(t_i, t_j\rangle$:

$$P(t_i, t_j) = \left[b_x(t_i, t_j)\right]_{x=1}^{n}. \quad (4)$$

It generally holds that $b_x(t_i, t_j) \neq b_x(t_j, t_i)$ and therefore $P(t_i, t_j) \neq P(t_j, t_i)$. For example, when $D(t_1) = [d_1(t_1), d_2(t_1), d_3(t_1)]$ and $D(t_2) = [d_1(t_1), d_2(t_2), -]$, then $P(t_1, t_2) = [0, d_2(t_2), -]$ but $P(t_2, t_1) = [0, d_2(t_1), d_3(t_1)]$.

When we know the data $D(t_i)$ and the partial backup $P(t_i, t_j)$, then we can recover $D(t_j)$, i.e. data at the instant t_j. Let us define an operation within which the data $D(t_j)$ are recovered from the data $D(t_i)$ and the partial backup $P(t_i, t_j)$. We refer to this operation as a data revision. Formally, we define the data revision in the following way:

$$D(t_i) \triangleright P(t_i, t_j) = D(t_j), \quad (5)$$

where the revision of each data unit is given by the formula:

$$d_x(t_j) = d_x(t_i) \triangleright b_x(t_i, t_j) = \\ = \begin{cases} -, \text{ when } b_x(t_i, t_j) = -, \\ d_x(t_i), \text{ when } b_x(t_i, t_j) = 0, \\ b_x(t_i, t_j), \text{ otherwise.} \end{cases} \quad (6)$$

In the case of the first line, the backup contains the information that the data unit d_x has been deleted in the course of the time interval $(t_i, t_j\rangle$. For this data unit, the result of data revision is an empty unit. In the case of the second line, the backup contains the information that there has been no change in the data unit d_x in the given interval. For this data unit, the result of data revision is the version of the data unit at the instant t_i. The third line corresponds to the situation when the backup contains the changed version of the data unit d_x. In this case, the result of data revision is the changed version of the data unit.

When it holds for the data revision in (5) that $t_i < t_j$, then we will speak about forward data recovery. In this case, we obtain by data revision a newer version, $D(t_j)$, from the older version, $D(t_i)$. When it holds that $t_i > t_j$, then we will speak about backward data recovery. In this case, we obtain by data revision the older version, $D(t_j)$, from the newer version, $D(t_i)$. For our examples of backups, it holds that:

$$D(t_2) = D(t_1) \triangleright P(t_1, t_2) = \left[d_1(t_1), d_2(t_1), d_3(t_1)\right] \triangleright \left[0, d_2(t_2), -\right] = \\ = \left[d_1(t_1) \triangleright 0, d_2(t_1) \triangleright d_2(t_2), d_3(t_1) \triangleright -\right] = \left[d_1(t_1), d_2(t_2), -\right]$$

and

$$D(t_1) = D(t_2) \triangleright P(t_2, t_1) = \left[d_1(t_1), d_2(t_2), -\right] \triangleright \left[0, d_2(t_1), d_3(t_1)\right] = \\ = \left[d_1(t_1) \triangleright 0, d_2(t_2) \triangleright d_2(t_1), - \triangleright d_3(t_1)\right] = \left[d_1(t_1), d_2(t_1), d_3(t_1)\right].$$

When we do not need to distinguish between the full backup $F(t_i)$ and the partial backup $P(t_j, t_i)$, then we will generally denote these backups as $B(t_i)$.

We will classify partial backups into interval and atomic backups. An interval backup $I(t_i, t_j)$ is a partial backup $P(t_i, t_j)$ which contains the latest version of the data units that have been changed in the interval $(t_i, t_j\rangle$. This means that if any data unit is changed several times in the interval $(t_i, t_j\rangle$, then the given interval backup will contain only the version of the data unit which was valid at the instant t_j. From this, it follows that we can recover the state of data only for the instants of taking backups. This type of backup is historically the oldest and these backups have a smaller size.

The atomic backup is a modern method which has been made possible by the so-called snapshot technique (e.g. [8]). With this technique, when a new version of a data unit is written, the older version of the given data unit (the so-called snapshot) is retained in the repository. In the case of atomic backup, all snapshots are backed up and therefore we can recover the data into the state at an arbitrary instant. A drawback of this type of backup is the large size of the aggregate of all snapshots. Before we explain the atomic backups in more detail, we will first discuss the data recovery schemes.

3. Data recovery schemes

The scheme of data recovery defines the procedure of data reconstruction from the backups acquired. In this paper, we will express data recovery schemes by a graph. The nodes i and j of such a graph represent the backups $B(t_i)$ and $B(t_j)$ and the oriented edge (i, j) expresses that we must first recover the data from the backup $B(t_i)$ and subsequently revise these data by using the data from the backup $B(t_j)$. We will call this relation between backups as a backup succession. In this case, we will call the backup $B(t_i)$ a reference backup and the backup $B(t_j)$ a subsequent backup.

An example of the recovery scheme for five consecutive backups $B(t_1)$ to $B(t_5)$ is given in Fig. 1. The backup $B(t_1)$, i.e. node 1, is a full backup which contains all data at the instant t_1. Simultaneously, this backup is the reference backup for the partial backups $B(t_2)$ and $B(t_4)$. These backups are reference backups for the partial backups $B(t_3)$ and $B(t_5)$. The interval backups $B(t_2)$, $B(t_3)$ and $B(t_5)$ contain changes of data units compared to the nearest previous backup. The interval backup $B(t_4)$ contains the latest version of the data units which were changed within the interval $(t_1, t_4\rangle$.

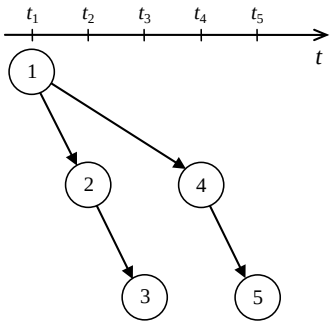

Fig. 1: An example of a recovery scheme.

From Fig. 1, it is apparent that if we want to recover data at the instant t_3, we must first write the full backup $B(t_1)$ into the main repository, then we must revise these data by the backup $B(t_2)$, and finally, we must still revise the data obtained by using the backup $B(t_3)$.

Now, we can explain the recovery schemes more precisely. In this paper, we focus on the milestone, differential and incremental recovery schemes because these schemes are the most widely used. The above recovery schemes are associated with the full, differential and incremental backup strategies [9]. For our example of five backups, the aforesaid schemes are illustrated in Fig. 2 to Fig. 4. Fig. 2 illustrates the milestone recovery scheme. This scheme consists of full backups (the so-called milestones) only, i.e. $B(t_i) = F(t_i)$. A drawback of this recovery scheme is the large size of backups (a large capacity of backup repositories is required), but the data recovery is very simple. The data $D(t_i)$ are simple to recover by writing the full backup $F(t_i)$ into the main repository, i.e.:

$$D(t_i) = F(t_i). \tag{7}$$

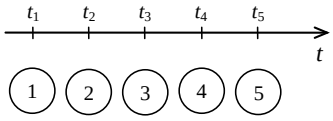

Fig. 2: An example of the milestone recovery scheme.

Other schemes are based on a combination of the full backup and partials backups. We will refer to these schemes as reference recovery schemes. The differential and incremental recovery schemes are the most frequently used. The differential recovery scheme is illustrated in Fig. 3. In this type of recovery scheme, the first backup $B(t_1)$ is the full backup and all the other backups are interval backups. The reference backup for each interval backup is the first backup. Formally, we can express this scheme as $B(t_1) = F(t_1)$ and $B(t_i) = I(t_1, t_i)$ where i = 2 to 5. The advantage over the previous scheme is the smaller size of the aggregate of backups but the recovery of $D(t_i)$ for $i > 1$ is more complicated. In this case, we first write the full backup $F(t_1)$ into the main repository at first and then we revise these data by the backup $I(t_1, t_i)$. Formally, we can express the data recovery according to the differential recovery scheme as:

$$D(t_i) = \begin{cases} F(t_1), \text{for } i = 1, \\ F(t_1) \triangleright I(t_1, t_i), \text{otherwise.} \end{cases} \tag{8}$$

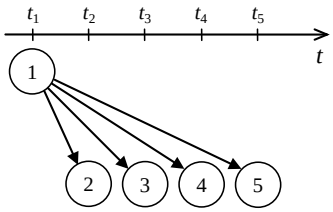

Fig.3: An example of the differential recovery scheme.

Fig. 4 illustrates the incremental recovery scheme. In this type of recovery scheme, the first backup $B(t_1)$ is again the full backup and all the other backups are interval backups. However, the reference backup for each interval backup is the nearest previous backup. Formally, we can express this scheme as $B(t_1) = F(t_1)$ and $B(t_i) = I(t_{i-1}, t_i)$ where i = 2 to 5. The advantage over the two previous schemes is the smaller size of the aggregate of backups; however, the recovery of $D(t_i)$ for $i > 1$ is the most complicated on the average. This is because data from the backup $B(t_1)$ must be sequentially revised by all backups

$B(t_2)$ to $B(t_i)$. Formally, we can express the data recovery according to the incremental recovery scheme as:

$$D(t_i) = \begin{cases} F(t_1), \text{ for } i = 1, \\ F(t_1) \triangleright I(t_1, t_2) \triangleright \ldots \triangleright I(t_{i-1}, t_i), \text{ otherwise.} \end{cases} \quad (9)$$

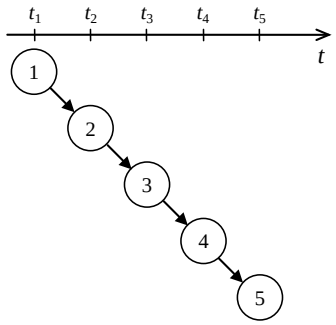

Fig. 4: An example of the incremental recovery scheme.

All the reference schemes described are forward data recovery schemes, i.e. we can recover the data $D(t_j)$ from the initial full backup $F(t_i)$, where $t_j > t_i$. By analogy, there are also backward data recovery schemes. In these cases, the reference full backup contains the newest data and the partial backups contain older versions of data units. If need be, we can return to a certain former state of the data. The representatives of backup systems with backward data recovery are some backup systems with data mirroring. Data mirroring (e.g. [10]) is based on that each change of any data unit in the main repository is practically instantly executed in the backup repository too. Therefore, the data in both repositories are identical and the backup repository can be used in the case of the failure of the main repository as a full-blown replacement of the impaired repository. When we back up older versions of data units, then from the valid data $D(t_i)$, we can restore data at the instant t_j, where $t_j < t_i$.

4. Atomic backup

Now, we can return to the description of the atomic backup. In the case of atomic backup, when a new version of the data unit is written, the older version of this data unit (the so-called snapshot) is retained in the repository. A backup program looks up and backs up snapshots which have not been backed up yet. By using the time data, which are kept about each snapshot, we can arrange the versions of all data units according to their validity in the time. This sequence allows us to recover data at an arbitrary instant. From the backup viewpoint, we can consider each snapshot as an individual backup. Then we can consider the sequence of snapshots in time as a sequence of individual backups, which are organized in the incremental recovery scheme. Each atomic backup always contains a single version of a single data unit only. When the data unit d_y is changed at the instant t_i, then we can formally express the corresponding atomic backup $A(t_j, t_i)$ as:

$$A(t_j, t_i) = \left[b_x(t_j, t_i) \right]_{x=1}^{n}, \quad (10)$$

where

$$b_x(t_j, t_i) = \begin{cases} d_y(t_i), \text{ for } x = y, \\ 0, \text{ otherwise.} \end{cases} \quad (11)$$

Of course, the operation of data revision also holds for atomic backups.

We will illustrate the atomic backup on a scenario according to Table 1 and Fig. 5.

Tab. 1: A scenario for the illustration of the atomic backup.

t_i	$d_1(t_i)$	$d_2(t_i)$	$d_3(t_i)$	Backups B
t_1	$d_1(t_1)$	$d_2(t_1)$	$d_3(t_1)$	$B(t_1) = F(t_1) = [d_1(t_1), d_2(t_1), d_3(t_1)]$
t_2	$d_1(t_1)$	$d_2(t_2)$	$d_3(t_1)$	$B(t_2) = A(t_1, t_2) = [0, d_2(t_2), 0]$
t_3	$d_1(t_1)$	$d_2(t_2)$	$d_3(t_3)$	$B(t_3) = A(t_2, t_3) = [0, 0, d_3(t_3)]$
t_4	$d_1(t_1)$	$d_2(t_4)$	$d_3(t_3)$	$B(t_4) = A(t_3, t_4) = [0, d_2(t_4), 0]$
t_5	$d_1(t_1)$	$d_2(t_4)$	$d_3(t_3)$	–

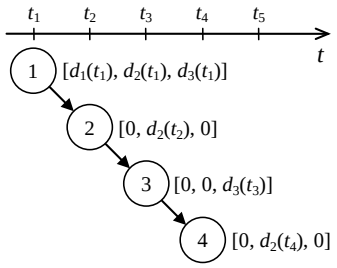

Fig. 5: An example of the atomic backup.

In this scenario, the data consist of three data units d_x, where $x \in \{1, 2, 3\}$. At the instant t_1, three data units $d_1(t_1)$, $d_2(t_1)$ and $d_3(t_1)$ were placed into the repository. The full backup $F(t_1)$ was simultaneously created from these versions of data units, i.e. $B(t_1) = F(t_1) = [d_1(t_1), d_2(t_1), d_3(t_1)]$ (see Fig. 5). At the instant t_2, the data unit d_2 was changed from the version $d_2(t_1)$ to the version $d_2(t_2)$. At the instant t_3, the data unit d_3 was changed from the version $d_3(t_1)$ to the version $d_3(t_3)$ while at the instant t_4, the data unit d_2 was changed from the version $d_2(t_2)$ to the version $d_2(t_4)$. Let us suppose that a backup program is activated at the instant t_5. At this instant, the valid data units are $d_1(t_1)$, $d_2(t_4)$ and $d_3(t_3)$. Also situated in the repository are the snapshots $d_2(t_1)$, $d_2(t_2)$ and $d_3(t_1)$. According to the time data in the metadata of snapshots,

the backup program can arrange these snapshots in time. On this basis, the backup program creates the atomic backups $B(t_2) = A(t_1, t_2) = [0, d_2(t_2), 0]$, $B(t_3) = A(t_2, t_3) = [0, 0, d_3(t_3)]$, and $B(t_4) = A(t_3, t_4) = [0, d_2(t_4), 0]$. The snapshots can then be deleted and the backups $B(t_1)$, $B(t_2)$, $B(t_3)$, and $B(t_4)$ can be utilized to recover the data at an arbitrary instant in the interval $\langle t_1, t_5)$. As regards the last line of Table 1 it should be noted that there was no data change at the instant t_5 and therefore no atomic backup exists at this moment. If the aggregate of atomic backups were overly large, then we could transform the atomic backups into a single interval backup. In this way, we can reduce the total size of backups because the interval backup contains only the latest version of each changed data unit. At the same time, however, we cannot recover data at an arbitrary instant. We can derive the interval backup from atomic backups by using the data revision as follows:

$$I(t_j, t_i) = A(t_j, t_{j+1}) \triangleright A(t_{j+1}, t_{j+2}) \triangleright \ldots \triangleright A(t_{i-1}, t_i). \quad (12)$$

For the above example, we can write that:

$$I(t_1, t_4) = A(t_1, t_2) \triangleright A(t_2, t_3) \triangleright A(t_3, t_4) =$$
$$= [0, d_2(t_2), 0] \triangleright [0, 0, d_3(t_3)] \triangleright [0, d_2(t_4), 0] =$$
$$= [0 \triangleright 0 \triangleright 0, d_2(t_2) \triangleright 0 \triangleright d_2(t_4), 0 \triangleright d_3(t_3) \triangleright 0] =$$
$$= [0, d_2(t_4), d_3(t_3)]$$

We see that our atomic backups contain three non-empty data units but the interval backup consists of two non-empty data units only. We have reduced the size of backups, but now we can only recover the data at the instant t_4.

When we want to have the possibility of recovering data from the past, then we can use a combination of data mirroring and snapshooting. The principle is the following. When the data unit d_x in the main repository is changed from the version $d_x(t_{i-1})$ to the version $d_x(t_i)$, then this change is also executed in the backup repository and the older version $d_x(t_{i-1})$ is backed up as the atomic backup $A(t_i, t_{i-1})$.

Atomic backups arranged according to time enable us to recover any past data. For example, when we need to recover the data $D(t_j)$, then we execute this recovery as:

$$D(t_j) = D(t_i) \triangleright A(t_i, t_{i-1}) \triangleright A(t_{i-1}, t_{i-2}) \triangleright \ldots \triangleright A(t_{j+1}, t_j). \quad (13)$$

An example of the recovery scheme which is based on the combination of the data mirroring and snapshooting is in Fig. 6 and Table 2. This example corresponds to the scenario from Table 1. For this recovery scheme, the valid data $D(t_i)$ of the backup repository are the reference full backup, i.e. $F(t_i) = D(t_i)$. At the instant t_5, it holds that $F(t_5) = [d_1(t_1), d_2(t_4), d_3(t_3)]$. This state is simultaneously the state at the instant of the latest data change, i.e. at the instant t_4. Therefore, we can write that $F(t_5) = D(t_5) = D(t_4) = B(t_4)$. The backup $F(t_5) = B(t_4)$ is the initial reference backup for the incremental recovery scheme with subsequent atomic backups $A(t_4, t_3) = [0, d_2(t_2), 0]$, $A(t_3, t_2) = [0, 0, d_3(t_1)]$, and $A(t_2, t_1) = [0, d_2(t_1), 0]$.

Let us note that this scheme is very similar to the atomic incremental scheme in Fig. 5. The difference between these schemes is their orientation in time. In Fig. 5, we see the forward incremental scheme while Fig. 6 illustrates the backward incremental scheme. We point out that the backward recovery schemes need not be incremental atomic schemes only. As we mentioned above, we can derive arbitrary interval backups from atomic backups by using (12). This enables us to construct arbitrary backward interval schemes.

Tab.2: A scenario illustrating the backward atomic backup.

t_i	$d_1(t_i)$	$d_2(t_i)$	$d_3(t_i)$	Backups B
t_1	$d_1(t_1)$	$d_2(t_1)$	$d_3(t_1)$	–
t_2	$d_1(t_1)$	$d_2(t_2)$	$d_3(t_1)$	$B(t_1) = A(t_2, t_1) = [0, d_2(t_1), 0]$
t_3	$d_1(t_1)$	$d_2(t_2)$	$d_3(t_3)$	$B(t_2) = A(t_3, t_2) = [0, 0, d_3(t_1)]$
t_4	$d_1(t_1)$	$d_2(t_4)$	$d_3(t_3)$	$B(t_3) = A(t_4, t_3) = [0, d_2(t_2), 0]$
t_5	$d_1(t_1)$	$d_2(t_4)$	$d_3(t_3)$	$D(t_5) = B(t_4) = [d_1(t_1), d_2(t_4), d_3(t_3)]$

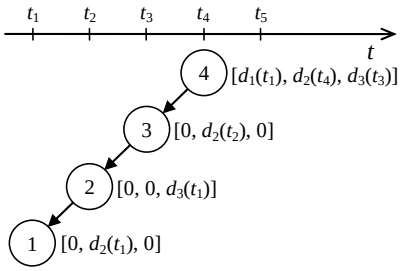

Fig. 6: An example of the backward atomic backup.

Now, we summarize the terminology introduced in this paper. We will classify the backups as follows:

- full,
- partial,
  - interval,
  - atomic.

We will sort the recovery schemes as follows:

- milestone,
- reference,
  - forward,
  - backward.

We will classify the reference recovery schemes as follows:

- differential,
- incremental,

- combined.

In the next section, we introduce a mathematical model for a quantitative evaluation of backups and recovery schemes.

5. Mathematical model

In this section, we deal with a mathematical model which enables us to quantify the size of different backups. This enables us to evaluate the properties of different data recovery schemes. We derive the model for the forward recovery schemes, but the formulas obtained are also valid for the backwards recovery schemes. In the model, we assume that all data units d_x consist of the same number of symbols. We will call this number of symbols the size of the data unit and denote it by $|d|$. We pessimistically suppose that there are no empty data units in the main repository, i.e. all n data units always contain certain data. Then, for the total amount of the data $|D|$ stored in the main repository, it holds that $|D| = n \cdot |d|$. The same formula evidently holds for the size of the full backup:

$$|F(t_i)| = |D|. \tag{14}$$

Now, let us solve the size of partial backups. Let us suppose that the $(i-1)$-th change of the data unit d_x occurred at the instant t_{i-1} and the i-th change of this data unit occurred at the instant t_i. Denote the time between these consecutive changes by u_i, i.e. $u_i = (t_i - t_{i-1})$. In the model, the times u_i are represented by the random variable U whose averaged value is Δ. We assume that this averaged value is the same for all data units. Then, the variable $\lambda = 1/\Delta$ represents the rate of changes of each data unit. We further assume that the probability distribution of the random variable U is an exponential distribution. The cumulative distribution function G of this distribution is:

$$G(u) = \Pr(U \le u) = 1 - e^{-\lambda u}. \tag{15}$$

Now, we will represent mathematically the interval backups. To this purpose, we will use the quantities in Fig. 7. In the Figure, the instants t_1, t_2, t_3, and t_4 are indicated. The instants t_1 and t_4 are instants at which a certain data unit was changed. The instants t_2 and t_3 are instants at which the data backup was executed. We note that these backups need not be adjacent. The quantity $\tau = (t_3 - t_2)$ is the time between these backups, the variable $u = (t_4 - t_1)$ is the time between two consecutive changes of the data unit (i.e. a realization of the random variable U), and the quantity $r = (t_2 - t_1)$ is the time between the change of the data unit and the following backup.

Now, we are interested in the probability $Q(\tau)$, which is the probability that when the data unit has not been again changed until the first backup (i.e. until t_2), then this data unit is not changed until the instant of the next observed backup (i.e. until $t_3 = t_2 + \tau$). We can formally express this probability as a conditional probability:

$$Q(\tau) = \Pr(U > r + \tau \,|\, U > r). \tag{16}$$

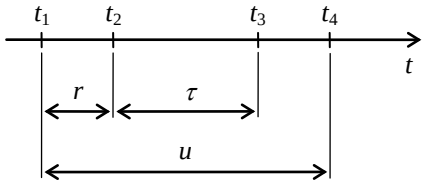

Fig.7: Quantities for the derivation of the mathematical model.

The exponential distribution is a distribution without memory and therefore it holds (e.g. [11], p. 40):

$$\Pr(U > r + \tau \,|\, U > r) = \Pr(U > \tau). \tag{17}$$

It follows from (17) that when the data unit is not changed in the time interval r (the condition $U > r$), then the probability that this data unit does not change in the time interval $(r + \tau)$ is the same as the probability that this change does not occur in the time interval τ. Then we have:

$$\begin{aligned} Q(\tau) &= \Pr(U > r + \tau \,|\, U > r) = \Pr(U > \tau) = \\ &= 1 - \Pr(U \le \tau) = 1 - G(\tau) = e^{-\lambda\tau}. \end{aligned} \tag{18}$$

The complementary probability $P(\tau) = 1 - Q(\tau)$ is, of course, a probability that the data unit is changed in the time interval τ. The data D consist of n data units and therefore the interval backup $I(t_i - \tau, t_i)$ on average consists of $n \cdot P(\tau)$ changed data units. Then the average size $|I(t_i - \tau, t_i)|$ of this backup is:

$$|I(t_i - \tau, t_i)| = |d| \cdot n \cdot P(\tau) = |D| \cdot \left(1 - e^{-\lambda\tau}\right). \tag{19}$$

This formula enables computing the average size of each subsequent interval backup $B(t_i) = I(t_i - \tau, t_i)$, which is spaced the time interval τ from its reference backup $B(t_i - \tau)$.

Let us assume that backups are executed periodically with the time interval T. Then it holds that the time interval between two different backups $\tau = k \cdot T$, where $k = 1, 2, 3, \ldots$.

Now, let us define the quantity q:

$$q = Q(T) = e^{-\lambda T}. \tag{20}$$

The quantity q is the probability that the data unit does not change in the time interval T. The quantity $p = 1 - q$ is the probability that the data unit changes in the time interval T. For this quantity, it holds:

$$p = 1 - q = 1 - e^{-\lambda T}. \tag{21}$$

Then the average size of the interval backup $I(t_i - k \cdot T, t_i)$ is:

$$|I(t_i - k \cdot T, t_i)| = |D| \cdot \left(1 - e^{-\lambda k T}\right) = |D| \cdot \left(1 - q^k\right). \tag{22}$$

This formula enables computing the average size of each subsequent interval backup $B(t_i) = I(t_i - k{\cdot}T, t_i)$, which is spaced the time interval $k{\cdot}T$ from its reference backup $B(t_i - k{\cdot}T)$.

Now, we determine the total average size S of the aggregate of atomic backups from the time interval T. We have already introduced the notation according to which the rate of changes of each data unit is λ and the number of data units is n. Then, the total number of changes $N = n{\cdot}\lambda{\cdot}T$. Each such change is recorded in one atomic backup of $|d|$ symbols in size. Therefore, the total average size $|S(t_i - T, t_i)|$ of the aggregate of atomic backups from the time interval T is:

$$\left|S(t_i - T, t_i)\right| = \left|d\right| \cdot n \cdot \lambda \cdot T = \left|D\right| \cdot \lambda \cdot T = \left|D\right| \cdot \ln \frac{1}{1-p}. \quad (23)$$

We note that the inversion of equation (21) was used to express $|S(t_i - T, t_i)|$ in the latest term.

By using formulas (14), (22), and (23), we can quantify the average size of different types of backup. Now, we can quantitatively analyze the different types of recovery scheme.

6. Discussion

We will illustrate the utilization of our model on recovery schemes in Figs 1 to 4. The milestone scheme is in Fig. 2, the differential scheme in Fig. 3, the incremental scheme in Fig. 4, and the combined scheme in Fig. 1. We note that the schemes introduced are comparable since all these schemes consist of $M = 5$ backups. We will restrict ourselves to forward schemes only, but the results obtained, of course, also hold for the equivalent backward recovery schemes. With all interval recovery schemes, we assume that the backups are executed with the period T. The size of the data $D(t_i)$ and the backups $B(t_i)$ will be denoted $|D(t_i)|$ and $|B(t_i)|$, respectively.

For a comparison of the schemes, we will utilize two parameters. The first parameter is the average total backup size C. We compute the value of this parameter by summing the average sizes of all backups in the given scheme, i.e.:

$$C = \sum_{i=1}^{M} \left|B(t_i)\right|. \quad (24)$$

The parameter C represents the storage space demands of the given recovery scheme and therefore we try to minimize its value. A higher value of the average total backup size means the backup repository has a larger storage capacity and therefore a higher cost too.

The average recovery time is the next criterion for evaluating the recovery schemes. If we assume a constant rate of writing data from the backup repository into the main repository, then the average recovery time depends on the average size of backups that we must write into the main repository in order to recover the required data $D(t_i)$.

However, to recover different data, i.e. $D(t_1)$ to $D(t_M)$, we must write into the repository the different backups of different sizes. For example, in the recovery scheme according to Fig. 1, we need to write only the full backup $B(t_1)$ to recover the data $D(t_1)$. However, to recover the data $D(t_5)$, we need to write into the repository the backups $B(t_1)$, $B(t_4)$, and $B(t_5)$. We will call the set of backups needed for recovering the data $D(t_i)$ the recovery data and denote their average size by R_i. Now, let us return to the recovery schemes. We recall that the node i represents the backup at the instant t_i. We will call the node of a full backup the root. Let us denote by V_i the set of nodes in the path from the root to the node i. In our example, the path for the node 1 is given by the node 1 itself and therefore $V_1 = \{1\}$. In the case of the node 5, the path is given by the sequence of nodes 1-4-5 and therefore $V_5 = \{1, 4, 5\}$. Then, for R_i it holds:

$$R_i = \sum_{k \in V_i} \left|B(t_k)\right|. \quad (25)$$

For example, according to Fig. 1, $R_5 = |B(t_1)| + |B(t_4)| + |B(t_5)|$. By using (25), we can compute the size of the data for recovering any state of $D(t_i)$. We will call the average value of all quantities R_1 to R_M the average data recovery size and denote it by R. Formally, we can write:

$$R = \frac{1}{M} \cdot \sum_{i=1}^{M} R_i. \quad (26)$$

The average recovery time depends in direct proportion on the quantity R and therefore we try to minimize the value of R. In such a case, the data recovery will be the fastest on average.

Now, we derive the formulas for C and R for all the schemes considered. In the case of the milestone scheme (Fig. 2), we see that the average total backup size equals the sum of the average sizes of all $M = 5$ full backups, i.e. it holds:

$$C = \sum_{i=1}^{M} \left|F(t_i)\right| = \left|D\right| \cdot M. \quad (27)$$

The average sizes of the recovery data R_i are identical for all $D(t_i)$, where $R_i = |D|$. Therefore, for the average data recovery size R, it simply holds:

$$R = \left|D\right|. \quad (28)$$

In the case of the differential recovery scheme (see Fig. 3), we know that $B(t_1) = F(t_1)$ and $B(t_i) = I(t_1, t_i)$, where $i = 2$ to M. Then for the sizes of single backups, it is valid:

$$\left|B(t_i)\right| = \begin{cases} \left|F(t_1)\right| = \left|D\right|, i = 1, \\ \left|I(t_1, t_i)\right| = \left|D\right| \cdot (1 - q^{i-1}), i = 2,3,...M. \end{cases} \quad (29)$$

By substituting these quantities into (24), (25), and (26), we finally obtain:

$$C = |D| \cdot \left(M - \frac{q - q^M}{1 - q} \right) \tag{30}$$

and

$$R = \frac{|D|}{M} \cdot \left(2 \cdot M - 1 - \frac{q - q^M}{1 - q} \right). \tag{31}$$

In the case of the incremental recovery scheme (see Fig. 4), we know that $B(t_1) = F(t_1)$ and $B(t_i) = I(t_{i-1}, t_i)$, where $i = 2$ to M. Then for the sizes of single backups, it is valid:

$$|B(t_i)| = \begin{cases} |F(t_1)| = |D|, i = 1, \\ |I(t_{i-1}, t_i)| = |D| \cdot (1 - q), i = 2,3,...M. \end{cases} \tag{32}$$

By substituting these quantities into (24), (25), and (26), we finally obtain:

$$C = |D| \cdot [M - (M - 1) \cdot q] \tag{33}$$

and

$$R = \frac{|D|}{2} \cdot [1 + M - (M - 1) \cdot q]. \tag{34}$$

For comparison, we additionally derive the quantities C and R for the combined recovery scheme in Fig. 1. The first backup is full, i.e. $|B(t_1)| = |D|$. The second, third and fifth backups are spaced one time interval T from their reference backups and therefore we can write that $|B(t_2)| = |B(t_3)| = |B(t_5)| = |D| \cdot (1 - q)$. The fourth backup is spaced the time interval $3 \cdot T$ from its reference backup and therefore $|B(t_4)| = |D| \cdot (1 - q^3)$. By substituting these quantities into (24), (25), and (26), we obtain:

$$C = |D| \cdot (5 - 3 \cdot q - q^3) \tag{35}$$

and

$$R = \frac{|D|}{5} \cdot (11 - 4 \cdot q - 2 \cdot q^3). \tag{36}$$

To compare C and R for the schemes considered, we utilize the functions $C = f(p)$ and $R = g(p)$, where p is the probability that the data unit is changed in the time interval T. The graphs of functions $C = f(p)$ in the multiples of the quantity $|D|$ are in Fig. 8 for the all schemes considered. In the case of the milestone recovery scheme, the value of C is the constant $5 \cdot |D|$ or generally $M \cdot |D|$.

Now, let us analyze the reference recovery schemes. For $p = 0$, we can see that $C = |D|$ for all reference schemes. This is due to the fact that there are no changes in the data units. All subsequent backups are empty and the value $C = |D|$ is given by the first (i.e. full) backup. The second common extreme of all reference schemes is the value of C when the rate of data unit changes $\lambda \to \infty$. In this case, all data units are always changed within the interval T and therefore the probability $p = 1$. Then, the total average size C of backups equals the value $M \cdot |D|$, i.e. the storage space demands of all reference schemes are identical with those of the milestone scheme.

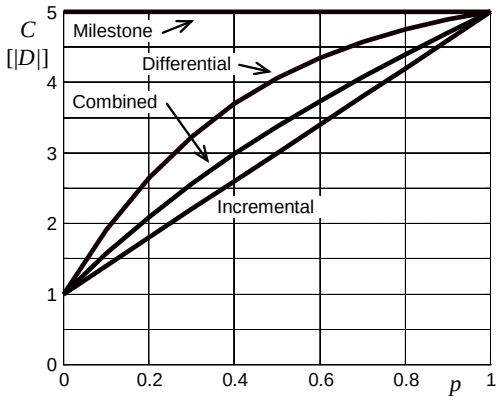

Fig. 8: The dependence of the average total backup size C on the probability p for different recovery schemes.

When we compare all schemes, we can see that there are two extremes. The first extreme is the milestone recovery scheme. For all $p \in (0, 1)$, this scheme has the highest demands on the storage capacity of the backup repository. The second extreme is the incremental recovery scheme where, by contrast, these storage space demands are the lowest. Then, we can take the storage space demands of the milestone recovery scheme as an upper bound:

$$C_{\max} = |D| \cdot M \tag{37}$$

and the storage space demands of the incremental recovery scheme as a lower bound:

$$C_{\min} = |D| \cdot [(M - 1) \cdot p + 1] \tag{38}$$

The storage space demands of the other schemes are between these bounds.

Fig. 9 illustrates the functions $R = g(p)$, i.e. the dependence of the average data recovery size R on the probability p. We can see that the lower bound of R is given by the milestone recovery scheme:

$$R_{\min} = |D|. \tag{39}$$

By contrast, the upper bound of R is given by the incremental recovery scheme:

$$R_{\max} = |D| \cdot \left(\frac{M - 1}{2} \cdot p + 1 \right). \tag{40}$$

From the two figures above, we can see that the milestone and incremental recovery schemes are boundary cases for both the quantity C and the quantity R. The milestone recovery scheme is the worst scheme from the viewpoint of storage space demands, but the best scheme from the viewpoint of the recovery time. In the case of the

incremental recovery scheme, the exact opposite holds. Then, we can take the differential scheme and combined recovery schemes as a compromise between the two extremes above.

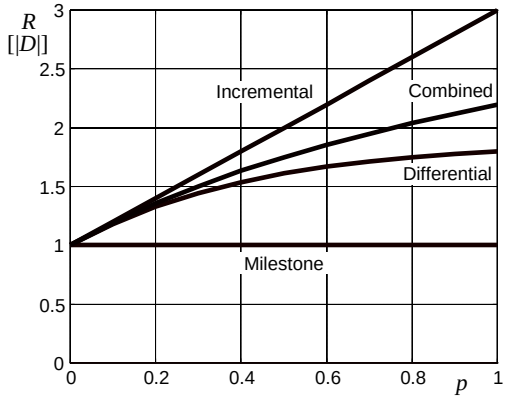

Fig. 9: The dependence of the average data recovery size R on the probability p for different recovery schemes.

The last thing in this discussion is comparing the sizes of the interval and atomic backups. From (22), we know that the average size $|I|$ of the interval backup within the time interval T is:

$$\left|I(t_i - T, t_i)\right| = \left|D\right| \cdot (1-q) = \left|D\right| \cdot p. \qquad (41)$$

From (23), we know that the total average size $|S|$ of all atomic backups from the time interval T is:

$$\left|S(t_i - T, t_i)\right| = \left|D\right| \cdot \ln \frac{1}{1-p}. \qquad (42)$$

Fig. 10 illustrates the dependence of $|S|$ and $|I|$ on the probability p. We can easily prove that it holds that $|S(t_i - T, t_i)|/|I(t_i - T, t_i)|$, i.e. the average size $|I|$ of the interval backup within the time interval T is not greater than the total average size $|S|$ of all atomic backups within the same time interval T. This is due to that the aggregate of atomic backups contains all versions of data units and not only the latest versions.

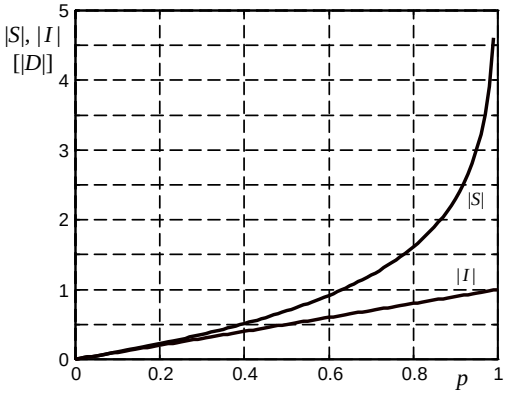

Fig. 10: Dependence of $|S|$ and $|I|$ on the probability p.

From the Figure, we can see that the differences between $|S|$ and $|I|$ are not significant for small values of p. However, these differences are definitely significant for greater values of p. For example, the ratio $|S| / |I|$ equals approximately 1.4 for $p = 0.5$. Then, in the case of the atomic backup, we need a backup repository with a capacity which is 40 percent greater than in the case of the interval backup. In the case of $p \approx 0.8$, the ratio $|S| / |I|$ equals 2, i.e. we need a backup repository with a capacity which is two times greater than in the case of the interval backup. The advantage of the atomic backup, i.e. the possibility of recovering the data at any time instant, is paid for by increased requirements for the storage capacity of the backup repository.

7. Conclusion

In the paper, the terminology and mathematical apparatus for data backup and recovery is extended. The core of the paper is a mathematical model of the data backup and recovery. The mathematical model enables us to compute the average size of an arbitrary backup (eq. 21 and 22) from the probability p that the data unit is changed in the time interval T. It enables us to determine the average total backup size C (eq. 24) for any recovery scheme and also the average data recovery size R (eq. 26). The model proposed allows us to compare different recovery schemes (e.g. Fig. 8 and 9) and extends the theory of the data backup and recovery. The model also allows us to compare interval and atomic backups (see eq. 22, 23 and Fig. 10).

The model introduced is based on the assumptions that the probability p is the same for all data units and that the data unit change is an event which has no influence on the changes of other data units, i.e. these changes are mutually independent events. Further assumptions are that the total size of the data is constant and that no data unit is empty. The assumptions introduced are not general and therefore our next goal is to create a more general model. In any case, however, the model described is suitable for theoretic purposes at least.

References

[1] A. Frisch: System Backup: Methodologies, Algorithms and Efficiency Models. In J. Bergstra, M. Burgess: Handbook of Network and System Administration. Elsevier, Amsterdam 2007.

[2] S. Nelson: Pro Data Backup and Recovery. Apress, New York 2011.

[3] A. Frisch: Essential System Administration. O'Reilly Media, Sebastopol 2002.

[4] Z. Kurmas, A. L. Chervenak: Evaluating Backup Algorithms. In IEEE Symposium on Mass Storage Systems. 2000, pp. 235-242.
[5] M. Burgess, T. Reitan: A risk analysis of disk backup or repository maintenance. Science of Computer Programming. 64 (2007), pp. 312-331. DOI: 10.1016/j.scico.2006.06.003.
[6] C. Qian, Y. Huang, X. Zhao, Toshio Nakagawa: X: Optimal Backup Interval for a Database System with Full and Periodic Incremental Backup. Journal of Computers. 5 (2010), pp. 557-564.
[7] Teradactyl: Backup Terms and Definitions. Teradactyl LLC. Retrieved April 17, 2014 from http://www.teradactyl.com/backup-knowledge/backup-definitions/backup terminology.html
[8] N. Garrimella: Understanding and exploiting snapshot technology for data protection. Part 1: Snapshot technology overview. IBM developerWorks. (April 26, 2006). Retrieved April 17, 2014 from http://www.ibm.com/developerworks/tivoli/library/t-snaptsm1/index.html?ca=dat-
[9] P. de Guise: Enterprise Systems Backup and Recovery. CRC Press, Boca Raton 2008.
[10] M. Liotine: Mission-Critical Network Planning. Artech House, London 2003.
[11] L. Lakatos, L. Szeidl, M. Telek: Introduction to Queueing Systems with Telecommunication Applications. Springer, New York 2013.

Karel Burda received the M.S. and PhD. degrees in Electrical Engineering from the Liptovsky Mikulas Military Academy in 1981 and 1988, respectively. During 1988-2004, he was a lecturer in two military academies. At present, he works at Brno University of Technology. His current research interests include the security of information systems and cryptology.