RI 11012 08 Jun 2011 Computer Science

IBM Research Report

RSCMap: Resiliency Planning in Storage Clouds

Vimmi Jaiswal, Aritra Sen, Akshat Verma
vimmi.jaiswal@gmail.com, aritrsen@in.ibm.com,
akshatverma@in.ibm.com

IBM Research Division
IBM Research - India
4-Block C, ISID Campus, Vasant Kunj
New Delhi - 110070. India.

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted is accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties). Copies may be requested from IBM T.J. Watson Research Center, Publications, P.O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home

RSCMap: Resiliency Planning in Storage Clouds

Vimmi Jaiswal[2], Aritra Sen[1], and Akshat Verma[1]

[1] IBM Research - India
[2] Independent

Abstract. Storage clouds use economies of scale to host data for diverse enterprises. However, enterprises differ in the requirements for their data. In this work, we investigate the problem of resiliency or disaster recovery (DR) planning in a storage cloud. The resiliency requirements vary greatly between different enterprises and also between different datasets for the same enterprise. We present in this paper Resilient Storage Cloud Map (RSCMap), a generic cost-minimizing optimization framework for disaster recovery planning, where the cost function may be tailored to meet diverse objectives. We present fast algorithms that come up with a minimum cost DR plan, while meeting all the DR requirements associated with all the datasets hosted on the storage cloud. Our algorithms have strong theoretical properties: 2 factor approximation for bandwidth minimization and fixed parameter constant approximation for the general cost minimization problem. We perform a comprehensive experimental evaluation of RSCMap using models for a wide varity of replication solutions and show that RSCMap outperforms existing resiliency planning approaches.

1 Introduction

Infrastructure as a Service (IaaS) clouds have rapidly emerged as a popular IT delivery model to reduce costs and improve resource utilization in data centers. Clouds allow diverse end users to request resources on the fly and release the resources, when the demand is met. Private clouds treat individual divisions in a large enterprise as end users and use a cloud model to standardize and consolidate delivery of IT infrastructure across the enterprise.

Data governance and Reliability have emerged as the key challenges in the adoption of clouds by enterprises. The always-online 24/7 business model of enterprises today has led to a situation, where downtime leads to direct business loss and customer erosion. Unforeseen events and disasters (hurricanes, earthquakes, terrorist attacks) may bring down the IT infrastructure of an enterprise and one of the key differentiators between a business that survives a disaster and one that shuts down is their ability to bring up their business processes, after a disaster [5]. Further, enterprises must comply with many regulations that require data governance. By moving the data into the cloud, enterprises lose the ability to govern their own data set and rely on the service providers to guarantee the safety of their data.

The unique selling point of cloud computing has been a low cost delivery model, which necessitates multi-tenancy or sharing of resources between end users and standardized offerings. Multi-tenancy implies that end users with varying data governance or reliability needs are co-hosted on the cloud. Cloud providers would increasingly need to provide disaster recovery support for individual customers in line with their governance and reliability requirements.

The crux of a disaster recovery plan is data replication [7]; where application data is replicated on same or a different site. Data replication technologies differ in (a) the time taken to restore data (RTO) (b) the amount of updates lost before disaster in terms of seconds or minutes (RPO) (c) the impact on application performance due to data replication and (d) the maximum distance that they can replicate the data. Resilient data replication technologies with quick recovery are very expensive, and in some cases require expensive network infrastructure as well [3, 4]. Hence, applying a common replication technology for all data in the cloud is infeasible and a replication technology needs to be selected as per user requirements.

The licensing models of various replication technologies are complicated; differing based on the amount of data being replicated, the amount of unique updates being made by the applications, the number of users, and even the number of servers. High performance storage controllers from many vendors provide synchronous, asynchronous replication as well as point-in-time copy technologies [7]. Enterprise-level databases and high performance parallel filesystems (e.g., GPFS) also provide replication techniques that can be used for disaster recovery [8]. The diversity amongst the capabilities and costs of replication technologies makes disaster recovery planning a complex problem.

The easy way out that enterprises take today is to select one replication technology to mirror all its data in an ad-hoc manner, independent of its criticality. Such an approach is infeasible in a cloud, which has the goal to provide IT services at a low price point. Hence, a formal approach to resiliency planning is a necessity for emerging clouds. We address the *Plan Composition* problem; which is to design the most cost-efficient disaster recovery plan for an enterprise with differentiated data (in terms of DR requirements) using realistic replication technology models. The input to the *Plan Composition* problem is (i) a set of data containers, each with its own DR requirements and a list of candidate replication solutions that can meet its requirements, (ii) a model of replication technologies and their cost models; the output is a matching of a data container to exactly one of its candidate replication solutions.

1.1 Contribtution

The contribution of our work is two-fold. First, we present a formal framework to study the DR plan composition problem, using realistic models for high performance replication technologies and their costs. Our cost-minimization framework is general enough to capture other optimization objectives (e.g, bandwidth minimization, vendor preference etc) typically employed in DR planning. To the best of our knowledge, this is the first attempt to rigorously study the DR plan composition problem. Secondly, we present an efficient algorithm for the DR plan

composition problem that provides a constant factor approximation for an important sub-class of the problem. Further, we present extensions of the algorithm for the general case with an approximation guarantee bounded by the number of parameters in the cost function, which is typically a constant. Our model and algorithms have been developed in the context of existing DR Planning tools [10, 6, 13, 17].

1.2 Related Work

Disaster Recovery planning has attracted a lot of attention in recent times with work on improved replication mechanisms [2, 9], modeling dependability of a system [12], planning papers [10, 6, 13, 1], plan deployment papers [13] and recovery papers [11, 17]. Recovery modeling has seen a lot of attention recently [11, 17] with Verma *et al.* presenting a rigorous analysis of recovery time and algorithms for fast recovery in a CDP environment in [17].

DR planning, on the other hand, is usually based on hand-coded practises and conservative estimates, that lead to expensive DR plans. Nayak *et al.* present an end-to-end disaster recovery planning and deployment tool whose focus is on the match-making phase and plan deployment [13]; our work is geared towards providing the optimization engine for the *plan composition* phase. Keeton *et al.* [10] tackle the problem of creating a cost-effective disaster recovery plan for a single application site and extend it in [6] to include shared applications. However, both these studies focus on finding cost functions for various technologies and present cost functions that are (a) linear and (b) depend only on the replication technology type (synchronous or asynchronous) and not on the actual replication technology (Global Mirror (GM) vs XRC [7]). Further, the linear nature of the cost model does not take into account price-bundling, which makes the cost functions incapable of capturing many typical planning scenarios. Moreover, [10] do not propose any optimization algorithms and suggests using off-the-shelf algorithms whereas the optimization algorithms used (including the new one in [6]) are based on mixed-integer programming and expensive, making them unsuitable for use in highly interactive disaster recovery planning tools like Ganesha [13]. Hence, earlier planning work [10, 13, 6] only provide a framework for DR planning without efficient algorithms, and this is exactly the deficiency that we address. In this way, our work complements earlier work; by using the match-making methodologies of [13] and the cost-functions proposed in [10, 6] for the optimization framework. Our proposed algorithms provide the last missing piece required in the integrated planning and deployment framework.

2 Model and Problem Formulation

We now present the model for the *Plan Composition Problem*. We start with some definitions.

Definition 1 *Disaster Recovery Service Class (DRSC): A Disaster Recovery Service Class denotes a specific class of service in terms of meeting Disaster Recovery requirements. The DRSCs are named as Gold, Silver, Bronze where*

the classification is based on attributes like RTO, RPO, Application Impact, distance etc.

Definition 2 *Data Container: A Data Container D_i is any set of logically grouped data that has identical DR requirements. One can think of a data container to correspond to all files of a particular user (/home/akshat etc) or of a particular type (temp files, mpeg files) that have the same criticality. For a data container D_i, s_i denotes the space (in terms of GBytes) and w_i denotes the write workload seen by the data container.*

Definition 3 *Failure Type: Failure type denotes the kind of failure F_k that a Disaster Recovery Service Class should cater to. The failure types we consider include Virus corruption, Device failure, Site failure, Subsystem failure, and LSS failure.*

Definition 4 *DR Profile: A DR Profile matches a DR Service Class and a Failure type to a data container, i.e., A DRProfile captures the DR needs for any given data container.*

Definition 5 *Replication Solution: A Replication Solution R_j is any technology that can provides a specific DR Protection for a given failure. Hence, any Replication Solution has tuples of failure class and DRSC parameters. To take an example, the IBM Global Mirror solution provides an RTO of 30 mins and an RPO of 3 seconds for site failure.*

Disaster recovery planning essentially consists of two phases [13]. The first phase is what we term as the matching phase. In the matching phase, we match the requirements of a data container to replication solutions. Hence, for each data container D_i, we create a set of solutions RS_i that meet the DR requirements of D_i. In the second phase called the *plan composition* phase, we select one replication solution for each data container such that the overall cost of deploying the solutions is minimized. This is followed by actual deployment of the computed plan. The cost-minimizing plan composition is the focus of this work. For more details on the matching problem and the plan deployment problem, the reader is referred to [13].

2.1 The DR Cost Minimization Framework

Consider the problem of designing a storage provision plan for a set of data containers, each of which has certain service requirements. Every data container belongs to a storage service class (SSC), where each SSC has performance, availability and 1 or more Disaster Recovery (DR) Profiles associated with it. We focus only on the Disaster Recovery Service Classes associated with the DR Profiles as we restrict ourselves only to creating a DR Plan. Every data container is associated with multiple replication solutions that meet its DR requirements (as a result of the matching step) and its workload information (space s_i, read throughput r_i and write throughput w_i). Each replication solution may specify

a storage controller with its configuration, network support (local and remote), and replication support (technology and licenses). Every replication solution also has an associated cost metric, which is a function of the space of data protected by the replication solution and its characteristics (write rate, read rate, users etc). We use the terms s_i, r_i, w_i to denote the space, read throughput and write throughput of data container D_i and the terms s_j, r_j, w_j to denote the total space, read throughput and write throughput of all data containers protected by replication solution R_j. Further, each replication solution transforms the traffic of the data containers protected by it and this is captured using a bandwidth transformation factor $B_{i,j}$.

Hence, the input to the *Plan Composition* problem is a set, where each entry in the set specifies a data container, a list of eligible solutions that would meet all the DR requirements of the data container and the predicted workload for the data container. The output of the problem is a detailed provisioning plan, which maps a data container to exactly one of its eligible solution, and strives to minimize the cost of the overall solution. Formally, we have:
Given (i) N Data Containers D_i with solution sets RS_i, Space s_i, Read rate r_i, Write rate w_i,
(ii) A set of M replication solutions $R_j = \cup(RS_1, ..., RS_N)$ with cost functions $Cost_{R_j} = Cost_j = C(s_j, w_j, ...)$ and a set of Bandwidth transformation functions $B_{i,j}$, find a mapping $x_{i,j}$ from Data Container D_i to Replication Solution R_j to $minimize \quad C_P$ where

$$C_P = \sum_{\exists i s.t. x_{i,j}=1} C_j(s_j, w_j) \qquad x_{i,j} = 1 \;\; if \;\; D_i \text{ uses } R_j \tag{1}$$

$$s.t. \forall i, j, x_{i,j} = 1 \Rightarrow R_j \in RS_i, \qquad \forall i, \sum_{j=1}^{M} x_{i,j} = 1 \text{ and} \tag{2}$$

$$\forall j, w_j = \sum_{i=1}^{N} x_{i,j} B_{i,j}(w_i) \quad r_j = \sum_{i=1}^{N} x_{i,j} r_i \tag{3}$$

It is easy to see that the *Plan Composition Problem* is NP-hard even for the simple case where cost is a step function (bin-packing can be reduced to it). Further, cost functions in many cases are non-differentiable and hence even LP-rounding techniques are not feasible. Further, Disaster Recovery Planning tools are highly interactive [13] and require fast algorithms for what-if analysis, making LP-based solutions too expensive to be useful. We thus restrict ourselves to finding fast approximation algorithms for the problem.

2.2 Disaster Recovery Service Class (DRSC) Model

The Disaster Recovery Service Class notion captures the important attributes of disaster recovery. DRSC is used both to denote the DR requirements of a data container as well as the DR capabilities of a replication solution. A DRSC consists of the following attributes

– Recovery Time Objective (RTO): Recovery Time Objective denotes the downtime after a disaster has occurred and is defined as the maximum time that will be taken to recover data after a disaster.
– Recovery Point Objective (RPO): The Recovery Point objective indicates the staleness of recovered data. It is expressed in seconds and denotes the amount of time by which the recovered data lags behind the lost data.
– Application Impact:Application Impact denotes the latency impact that occurs as a result of deploying the replication solution.
– Resource Impact:A replication solution takes away system resources (CPU cycles, Disk bandwidth) and resource impact captures this aspect of a replication solution.
– Distance:This represents the maximum supported distance between the copies of a replication solution.

2.3 Replication Solution Model

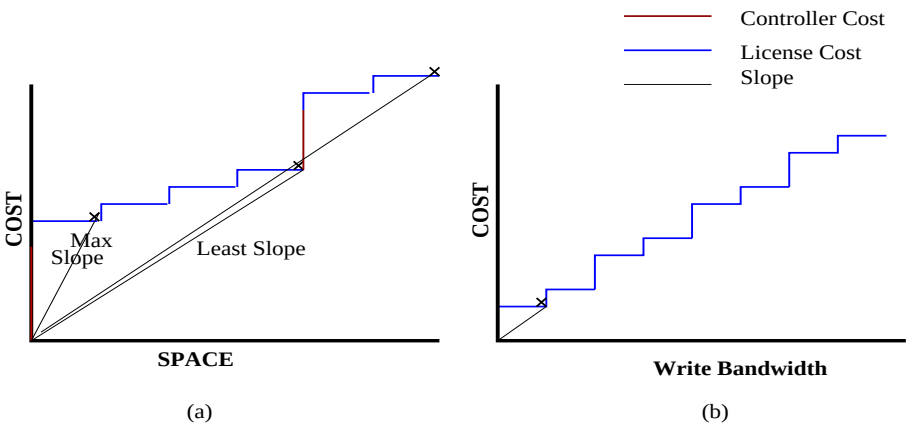


Fig. 1. Cost functions for IBM Flashcopy and DB2 HADR Technology

A Disaster Recovery requirement is met by deploying a replication technology along with its relevant configuration parameters. We use the notion of a replication solution to capture a replication technology along with its configuration. To take an example, the IBM DS8000 controller supports Flashcopy technology where we can configure the Flashcopy as either a Full Flashcopy or an Incremental Flashcopy. We create one replication solution each for both these configuration options as the RTO, RPO and other DRSC values for Flashcopy differ based on whether we are taking a full Flashcopy or an incremental Flashcopy.

Further, replication technologies transform the replication traffic generated and this is captured using the bandwidth transformation function. To take an example, asynchronous replication technologies like IBM Global Copy use techniques like write-coalescing to reduce the number of writes that reduces the network cost for the solution. The Bandwidth transformation function $B_{i,j}$ is used to

capture this reduction in bandwidth. Hence, if a data container D_i with write throughput w_i is protected using the replication solution S_j, then the write bandwidth is given by $B_{i,j}(w_i)$.

Finally, the cost of using a replication solution depends on various parameters like the amount of data being protected (licensing costs are based on space) or the amount of write bandwidth (network cost depends on inter-site bandwidth). These are captured using a cost function, that is specific to each replication solution. These dependencies are not linear for many technologies. To take an example, the cost for an IBM Flashcopy for protecting X TB of data would first require one to buy an IBM storage controller that supports Flashcopy and then buy Flashcopy licenses for X TB of data. Hence, as shown in Fig. 1 there is a huge starting cost to protect even a small amount of data, and the incremental cost is low, till we hit the capacity of the controller. Once the capacity of the controller is exceeded, one may have the option of adding expansion slots, which also incur significant cost. Once all the expansion slots are taken, one has no choice but to buy another controller. On the other hand, protecting data using DB2 replication may require a more smooth cost function, as cost is primarily dependent on the size of the transaction logs, which is related to the write bandwidth.

2.4 Modeling Various Objectives in the Cost Formulation

The Cost Minimization framework for the Plan Composition problem naturally covers the Disaster Recovery Planning setting where the objective of the plan is to provide the required level of disaster recovery to each of the data containers at the minimum possible cost. However, in many practical scenarios, cost may not be the optimizing criterion for planning.

A key restriction that disaster recovery consultants face, is in adding new sites for the purpose of disaster recovery. In most cases, one has to work with existing sites and existing network links. In these scenarios, setting up more sites or more links between the sites is not an option. The objective of the planning process in such a scenario is to provide a DR plan that minimizes the inter-site bandwidth. Also, in many cases, enterprises have bias towards specific vendors. These biases could be a result of trust with the vendor or familiarity. Similarly, enterprises may be biased towards specific replication technologies because of reliability or ease of management. In order for the optimization framework to find wide acceptance, it should be rich enough to capture all these diverse objectives.

Our proposed Cost Minimization framework is flexible enough to capture all these settings. The bandwidth minimization problem is captured by using the bandwidth transformation function B_j for a replication solution R_j, in place of the cost function C_j. To capture vendor preferences, the cost functions for technologies are weighted by a preference function and the same framework proposes solutions, while taking vendor preferences into account. A similar approach is used to capture technology preferences (e.g., DB-level replication is preferred over Block-level replication). Hence, our generic cost minimization framework is rich enough to capture most optimization criteria used in practice.

3 Model Assumptions

We now use insights from the practical setting of the *Plan Composition* problem to simplify the problem. The reader may observe that all the restricted versions of the problem we consider that take into account the practical constraints of the problem are also NP-hard. We first make the simplifying assumption that the amount of data to be protected by any DR Service Class is more than the space taken by a single instance of any replication solution. One may observe that this is true for any enterprise that requires DR protection. Further, one can take very large-sized replication technologies into account separately by listing them and comparing the best solution for a service class obtained using small-sized technologies with the cheapest of the large-sized technologies.

We now investigate more intricate properties about the replication solution sets RS_i that are typically true in real deployments.

3.1 Pure Subset Replication Set Property

In order to solve the *Plan Composition* problem, the first observation we use is that a replication solution that can meet the *Gold* service class for a given failure type F_k would also meet *Silver* or *Bronze* service class for failure F_k. Hence, any data container that requires a low level DR protection can use all the replication solutions that provide protection for that or any higher class for the given failure type. We capture this real-life constraint in the *Pure Subset Replication Set Property*, which we define next.

Definition 6 *Pure Subset Replication Set: A Plan Composition problem is said to satisfy the Pure Subset Replication Set Property if*

$$\forall D_i, D_j \quad RS_i \cap RS_j \neq \phi \quad \Rightarrow \quad RS_i \subseteq RS_j \;\; or \;\; RS_j \subseteq RS_i \tag{4}$$

The *Pure Subset* property captures the fact that any two replication solutions either provide different functionality (catering to different kinds of failure) or they have a total order amongst themselves. We will later use this property to establish bounds on the goodness of proposed algorithms.

3.2 Traffic Independent Bandwidth Transformation

Replication technologies use various techniques to reduce the replication bandwidth generated by any given data container. The savings are obtained by techniques like compression and write-coalescing. In many cases, the savings in bandwidth depends entirely on the replication technology being used and not on the properties of the data container. We use this fact to simplify the bandwidth transformation function ($B_{i,j}$) from being a function of both the replication technology R_j and data container D_i to being a function B_j of only the replication technology R_j.

We use the traffic independence of the bandwidth transformation function to simplify the optimization problem in the following manner. In the Cost function C_j of the replication technology R_j, we replace all instances of the write bandwidth w_j by the function $B_j(w_j)$ and replace the equation $w_j = \sum_{i=1}^{N} x_{i,j} B_{i,j}(w_i)$ by $w_j = \sum_{i=1}^{N} x_{i,j} w_i$ in Eqn. 3.

3.3 Independent Cost Functions

The framework of the *Plan Composition* problem in Eqn. 1 uses the simplifying assumption that costs of the various replication technologies are quasi-linear (additive) in nature. Hence, we compute the cost of the plan as a sum of the cost of all the replication technologies deployed. However, in real enterprises, cost of various technologies are not linear as they are bundled together. To take an example, buying the IBM Metro Mirror (MM) license or the IBM Flashcopy License requires one to purchase a DS6000 or a DS8000 storage controller first [7]. However, the same storage controller can be used by both the Global Mirror or the Flashcopy technology. Hence, in real practice, cost functions are not always additive in nature.

However, we will initially assume the additive cost constraint for ease of exposition of our algorithms. We will later show that our algorithms can take into account the non-additive nature of the cost functions as well.

4 Plan Composition Algorithms

We now present fast algorithms that solve the plan composition problem and prove approximation guarantees for the algorithms.

We first present algorithms for a simplified version of the problem, where cost is only a function of the size of data being protected, i.e., Cost of a replication technology is a function of only the total space taken by all the data containers protected by the technology. This one-dimensional cost problem, as noted earlier, is also NP-hard and is of independent interest, since it captures the bandwidth minimization variant of the plan composition problem. Moreover, we will later enhance the algorithm to work with multiple parameters or multi-dimensional cost functions.

4.1 Algorithms for the One-dimensional Cost Problem

In the One-dimensional variant of the problem, cost of a replication technology is dependent on only one parameter. Even though, this could be used to minimize any objective that depends on exactly one parameter (e.g., bandwidth can be minimized using write rate), we consider the case where cost is a function of space, i.e., $Cost_j = C(s_j)$, while noting that the same formulation and results hold for any other dimension as well.

A Plan Composition algorithm makes two decisions: (i) Pick a [cost,space] point for each selected replication technology and (ii) map data containers to a selected replication technology for DR protection. In order to solve the first problem, we use the greedy strategy of filling up as much space as possible at the least cost. Hence, we favour replication solution corner points $\min\{A_j\}$ that can provide protection to data at the least cost per unit amount of data protected (Fig. 1). (From now on, we use replication solution R_j to indicate R_j at the least slope corner point) This greedy strategy may lead to incompletely filled replication solutions R_j (or equivalently incompletely filled replication solution

```
function LeastSlopeFirst
 k = 0
 CurrDC = {D_1, ..., D_i, ..., D_N}
 For each R_j ∈ ∪(RS_1, ..., RS_i, ...)
    create an array A_j of corner points of the cost function
    Sort the array A_j by slope (C_j/s_j)
 End For
 Merge the M arrays (A_1, ..., A_M) in sorted order to get a merged array A
 While CurrDC ≠ φ
    Pick the least slope entry point in A as R_j
      (R_j denotes the selected corner point min{A_j} with cost C_j and space s_j)
    eligibleDC = selectMostConstrainedDC(currDC, R_j, s_j)
    If Σ_{i,D_i∈eligibleDC} s_i ≥ s_j
      ∀D_i ∈ eligibleDC   Match(D_i) = R_j
      CurrDC = CurrDC - eligibleDC
    Else
      ∀D_i ∈ eligibleDC   PartMatch_k(D_i) = R_j
      k = k + 1
      CurrDC = CurrDC - eligibleDC
    End If
 End While
 PickBest(Match, PartMatch, k)
end LeastSlopeFirst
```

Fig. 2. LeastSlopeFirst (LSF) Algorithm for Single Dimensional Costs

corner points $\min\{A_j\}$) and we bound this fragmentation cost by separately listing these partially filled replication solutions, to add to the least slope solutions later. For the second problem, we pick the data container D_i to be protected first that have the minimum number of eligible replication technologies (smallest $|RS_i|$). Fig. 2 details out the LeastSlopeFirst (LSF) algorithm that greedily picks replication technologies and bounds the cost due to fragmentation. The $selectMostConstrainedDC$ method captures the data container selection technology. The procedures used by LSF are detailed next.

Definition 7 *selectMostConstrainedDC Selection: Given a set of data containers D_i, a replication solution R_j, and a space constraint s_j, $selectMostConstrainedDC$ sorts the replication solutions by the arity of RS_i. It then keeps on adding data containers from the sorted list to its selection, till the accumulated space of the selected data containers equals s_j.*

Definition 8 *PickBest Procedure: Given a set of k partial matches PM_k and a match M, the PickBest procedure returns the minimum cost subset of PM_k and M that covers all the data containers.*

We start by noting that in its trivial version, $PickBest$ returns all the partial matches and the complete match as the output of LSF. Although, in actual practice, we use smarter implementations to cut down the cost, these optimizations in $PickBest$ do not change the approximation bounds. We first prove the optimality of the $selectMostConstrainedDC$ data container selection process and

then use it to prove approximation guarantees on LSF. The optimal selection of data containers is one that given (i) a set of data containers and eligible replication solutions for them and (ii) an ordered set of replication solution corner points, can protect the maximum amount of data using the selected replication solutions. We call this selection as the *Maximum Matching* Selection and show that our procedure is a *Maximum Matching* Selection.

Definition 9 *Maximum Matching Selection: Given a set of replication solutions R_j with space s_j, and a set of data containers D_i with space s_i and eligible replication solutions RS_j respectively, a selection is called a maximum matching selection if it can protect the maximum amount of data by the replication solutions R_j.*

Lemma 1. *$selectMostConstrainedDC$ is a Maximum Matching Selection.*

Proof. Let $Match_s$ be the match returned by *$selectMostConstrainedDS$* and let $Match_o$ be the optimal match. We first transform both of these matches to equivalent match that satisfies the following properties. (i)For any two data containers D_i, D_j protected by replication technologies R_i, R_j with corner points A_i, A_j, if the service class of D_i is greater than D_j, then the service class of R_i is higher than R_j. (ii) For any two replication solutions R_i, R_j s.t. the service class of R_j is higher than R_i, if R_i is used to protect any data from any data container, then R_j is completely filled up to its selected corner point $\min\{A_j\}$. Note that (i) can be trivially achieved by swapping any two data containers which violate this constraint and the swap is always admissible due to the *Pure Subset Replication Set Property*. Similarly, (ii) can be achieved by moving protected data containers (either fully or partially) to any partially-filled replication solution corner point of a higher class. Note that after the swaps and upward movement, both $Match_s$ and $Match_o$ would have unallocated space left only in the replication technology of the lowest class. Let D_k be highest service data container in $Match_o$ that is not in $Match_s$ and let it be protected by the l^{th} replication solution (ordered from highest to lowest service class). Note that D_k can be protected using any of the $l-1$ higher replication technologies as well since it is of the lowest DR Service Class amongst them. Hence, this was available for selection in invocations of *$selectMostConstrainedDC$* for any of those l replication solutions. Further, also note, that at least one such invocation of *$selectMostConstrainedDC$* returned *either* a set of data containers whose accumulated space was less than the space of the replication technology *or* selected a data container of lower service class. By the definition of *$selectMostConstrainedDC$* and the *Pure Subset Replication Set* property, either leads to a contradiction. Hence, such a D_k does not exist, i.e, $Match_o$ and $Match_s$ are identical. This completes the proof.

We now use the above lemma to bound the cost of the solution returned by LSF.

Theorem 1. *The LeastSlopeFirst algorithm returns a plan P such that the cost C_P of the plan is no more than twice the cost C_O of the optimal solution O.*

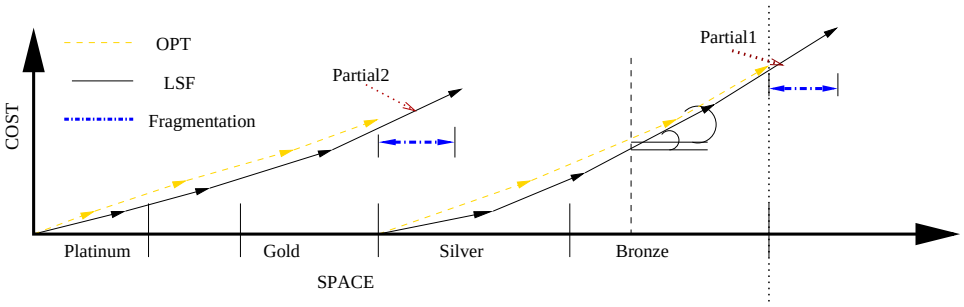


Fig. 3. Sample Run of LSF and Optimal. *Fragmentation* denotes the space wasted by LSF due to fragmentation and bounds the additional space for which LSF has incurred cost.

Proof. We first observe that as a result of Lemma 1, LSF matches the optimal data containers to any selected replication solution. Hence, any penalty that LSF pays is in selecting non-optimal replication technologies. Let $\mathbf{R^{lsf}}$ be the set of replication solutions picked by LSF and $\mathbf{R^O}$ be the set of replication solutions picked by the optimal solution. Note that LSF always gives higher priority to replication solutions with least slope (Fig. 3). Hence, for any point s_i in the space dimension, $\frac{\delta C_i^{lsf}}{\delta s_i^{lsf}} < \frac{\delta C_i^{O}}{\delta s_i^{O}}$, i.e. the slope of LSF is less than the slope of optimal. Hence, the only loss that LSF may sustain is because of fragmentation in the partial matches (Fig. 3). Multiplying both sides by $2 * \sum_{i=1}^{N} s_i$, and using the fact that the total fragmentation in LSF is bounded by $S = \sum_{i=1}^{N} s_i$ (the number of partial matches are bounded by the number of DRSC under protection, and each partial match is smaller than data protected by a DRSC), we obtain $C^{lsf} < 2 * C^{O}$. Hence, the result.

For an intuitive understanding of the result, observe that any selection process would pick replication solutions to fill data containers whose total space is given by $\sum_{i=1}^{N} s_i$. Since LSF always picks the least slope solutions, it incurs the least cost but it may pay by overshooting the space for each DRSC using partially filled replication solutions. Since the overshoot is bounded by the size of the replication solution, the total overshoot is bounded by $\sum_{i=1}^{N} s_i$.

4.2 Algorithms for General Cost Functions

We now consider the k-dimensional version of the Plan Composition problem, where the cost of a replication solution depends on k parameters/dimensions, from d^1 to d^k. Since any traffic transformation (e.g., bandwidth transformation) is already captured in the cost function (Sec. 3.2), the value of a replication solution R_j across any dimension d^l is summation of the value along the dimension d^l all the data containers protected by R_j.

$$\forall l \in [1, d], \forall j \in [1, m], \quad d_j^l = \sum_{i=1}^{N} x_{i,j} d_i^l \tag{5}$$

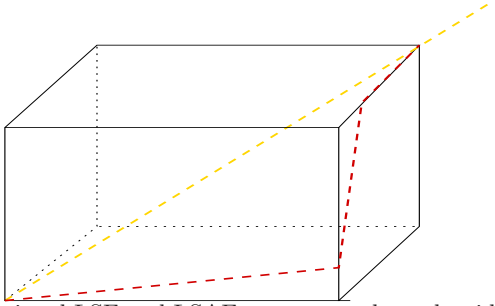


Fig. 4. Multi-dimensional LSF and LSAF may move along the sides and Optimal may follow the diagonal.

Our strategy remains the same as in LSF. We pick replication solutions that can protect data containers at the least cost per unit amount of data protected. However, since data containers have more than one relevant dimension now, we need to make a slight modification. In our first version, we order dimensions by their relative cumulative sizes ($d_a^l = \sum_{i=1}^{N} d_i^l$). We then pick the dimension d^{max} with the greatest accumulated sum ($\max_{l=1}^{k} d_a^l$) and order replication solution corner points by $\frac{\delta C_j}{\delta d_j^{max}}$. We follow the LSF procedure with the dimension d^{max} replacing the space parameter s. Once, we use up the dimension d^{max}, we use the next largest dimension and continue till we have protected all data containers ($currDC = \phi$).

Fig. 4 illustrates the strategy, where we move along the dominant dimension until we hit the wall of the cuboid. We then pick another dimension to move along. Noting again that (a) we use the least per-unit dimension cost replication solutions, (b) optimal also needs to travel d_a^{max}, (c) we repeat the process for a maximum of k dimensions and (d) the cost of fragmentation is bounded by a factor of 2, we get a bound of $2k$, which leads to the following result.

Theorem 2. *Multi-dimensional LSF returns a plan that has a cost no more than $2k$ times the cost of the optimal solution.*

We now propose a variant of Multi-dimensional LSF called $LSAF$ that also has an approximation guarantee of $2D$. However, it satisfies additional properties which may lead to a better approximation guarantee. The LeastSolidAngleFirst ($LSAF$) algorithm differs from multi-dimensional LSF by ordering the replication solutions by $\frac{\delta C}{\delta \sqrt{\sum_{l=1}^{k} (d^l)^2}}$. Hence, $LSAF$ uses the derivative of the cost (or the solid angle) to pick the replication solutions. If at any given time, it consumes any particular dimension than it removes that dimension from the derivative calculation. Hence, the method has a list of active dimensions, which is initialized with all the dimensions and pruned as one or more dimensions get exhausted. To take a geometric view, if the process hits any walls of the hyper-cuboid (Fig. 4), it takes out that dimension from any future calculations (i.e. the set of active dimensions). Along the lines of theorem 2, one can show that $LSAF$ provides an approximation guarantee of $2k$. However, note that $LSAF$ always incurs less cost per unit distance traveled in the protected space of active dimensions than the optimal. On the other hand, the optimal can travel

along the diagonal whereas $LSAF$ can be forced to travel along the sides of the hypercuboid. We first prove some technical lemmas.

Lemma 2. *In a hypercuboid of dimension k, the ratio of the diagonal to the sum of the sides is maximized if all the sides are equal.*

Proof. The proof is by induction. It is easy to see that the above holds for $k = 2$. We make the inductive hypothesis that for a hypercuboid of dimension $k-1$, the ratio is maximized if all the $k-1$ sides are equal. We now use the following observation, which is easy to prove using any minima finding technique.

Observation 1 $\frac{(k-1)d^2+d_k^2}{((k-1)d+d_k)^2}$ *is minimized if $d = d_k$.*

The above is nothing else but square of the ratio of the diagonal to the sum of sides and proves the inductive hypothesis for k dimensions.

Lemma 3. *In a hypercube of dimension k, the ratio of the diagonal to the sum of the sides is given by $\sqrt{(}k)$.*

Lemma 3 is easy to verify and leads us to the following result.

Lemma 4. *The total distance traveled by $LSAF$ is no more than $\sqrt{(}k)$ times the distance traveled by optimal.*

We also observe that along the number of active dimensions, $LSAF$ has the least cost per unit distance traveled amongst all possible algorithms (including optimal). Hence, $LSAF$ has a lower bound on the approximation guarantee of of $2\sqrt{(}k)$, whereas it is easy to see that the multi-dimensional LSF has a lower bound of $2k$. This leads us to the conjecture that a tighter analysis of $LSAF$ may leads to an approximation guarantee of $2\sqrt{(}k)$, which if true, would be included in the final version.

$LSAF$ also runs in time that is almost linear (linear in the number of data containers but not in number of replication solutions). Further, even though it has an approximation guarantee of $2k$, $LSAF$ is expected to perform very well in practice because the approximation guarantee is given by the ratio of the length of the sides of the hypercuboid to the length of the longest side. Another way to look at it is that the approximation bound depends on the number of dominant dimensions. (A dominant dimension in a hyper-cuboid has length along that side (dimension) greater than the average length of all sides (dimensions) Also, the number of such dimensions is small if values are power law distributed. Based on our modeling of the various replication technologies, we found both the observations to be true making $LSAF$ a promising plan composition algorithm.

4.3 Handling Non-independent Cost Functions

We now relax our last assumption on the independence of cost functions. As stated earlier, cost functions may not always be dependent. To take an example, a disk controller with Metro Mirror (MM) replication and the same controller

with Flashcopy are two different solutions. However, their costs are dependent as controller cost is present in both the replication solutions since the same controller can be shared.

In order to take into account this dependence, we combine dependent solutions into one solution with the union of dimensions as the dimension of the combined entity. In the earlier example, we replace the two dependent replication solutions by a single replication solution. The solution is disk controller with Flashcopy and Long Distance replication with two separate space (data capacity) dimensions: one for data protected by Flashcopy and the other for data protected by Long Distance Remote Copy. We repeat the same process for all dependent replication solutions until we have a replication solution space with all replication solutions having independent cost functions. The above process leads to a new problem with potentially much larger number of dimensions where the cost functions of the technologies are independent. Hence, we can use $LSAF$ to solve the problem. We also note that since the approximation bound of $LSAF$ depends on the number of dominant dimensions only, an increase in the number of dimensions may not impact the performance of $LSAF$ significantly. We show in the next section that the number of dominant dimensions is typically a small constant in practise.

5 Experimental Evaluation

5.1 Implementation and Setup

We implemented the plan composition logic as a Java-based library that exports a planning interface. The library has a configuration file to capture (i) the supported DR SLA levels (ii) the data containers discoverd in the cloud with their SLA levels and (iii) the replication technologies available in the cloud. The library returns an XML that captures a mapping for each data container to an instance of a replication solution, along with associated configuration parameters. The XML can be used by the resiliency management layer in the cloud to deploy the plan.

Stack	Type	Min Cost (x1000$)	Incr. Cost	Min. Capacity	RTO	RPO	Distance(Km)
Block	MGM	400	80$/GB	100GB	15	0	1500
Block	Sync	250	55$/GB	100GB	15	0	300
FS	Sync	60	100$/GB	25GB	15	0	300
DB	Sync	20	1600$/user	25	15	0	300
Block	Async	150	25$/GB	100GB	15	3	1500
DB	Async	20	800$/user	25	15	3	1500
Block	FC	100	5$/GB	100GB	60	900	300
FS	FC	25	10$/GB	25GB	60	900	300
DB	FC	5	500$/user	10	60	900	300

Table 1. Replication Technologies Used

We created 4 SLA classes to capture a cloud with diverse data governance requirements. The SLA classes are named as Platinum, Gold, Silver, and Bronze to capture the entire spectrum from low RTO, RPO and high distance protection to high RPO, low distance protection. We also created a knowledge base of popular replication technologies, which were used to protect the data in the cloud. We considered Metro Global Mirror, which provides long distance protection using 2 sites, for the most critical requirements. We also considered synchronous (Sync), asynchronous (Async), and flashcopy (FC) based technologies at block-level, filesystem (FS) level and database (DB) level replication. Table. 1 captures all the replication technologies considered and Table. 2 captures the mapping of replication technologies to SLA classes.

SLA Class	Feasible Replication Solutions
Platinum	Block MGM
Gold	All Platinum solutions + Block Sync, FS Sync, DB Sync
Silver	All Gold solutions + Block Async, DB Async
Bronze	All Silver solutions + Block FC, FS FC, DB FC

Table 2. Replication Technologies available to each SLA class

The third input to *RSCMap* is the set of data containers hosted on the cloud. In order to create the data containers for our evaluation, we took a configuration snapshot of the production data center of a Fortune 500 company. We selected 100 virtual machines and used their datastore for protection using our *RSCMap* framework. These VMs can be classified as those that use raw volumes, use filesystems, or use databases. The distinction is important from the DR perspective since a DB-based datastore hosted on filesystem can be protected using block replication, filesystem replication or database replication. On the other hand, a raw volume can only be protected using block replication technologies. Finally, we also capture the number of users for databases, since DB replication licenses are driven by the number of users.

In order to evaluate the value addition of *RSCMap*, we also implemented 2 algorithms that capture the state-of-the-art in DR planning. The first algorithm *Least Cost Global* (*LCGlobal*) picks a common replication technology to protect all data containers in the cloud. It uses the cheapest technology that provides the required protection level for the most constrained data container in the cloud and captures the conservative approach followed in data centers. Our second algorithm *LCLocal* is a finer-grained approach, which picks the cheapest replication technology for each service class separately. Hence, this algorithm is aware of multiple service classes and uses it to reduce costs.

In our evaluation, we created a baseline setting using six instances of each replication technology (which is enough to protect all the data containers), and use the various algorithms to come up with a disaster recovery plan. We present the results of this baseline evaluation next. We also vary the properties of the

replication technologies and the data containers to capture the performance of the algorithms under diverse settings.

5.2 Baseline Results

	MGM	Sync	Async	Flashcopy	Total
LCGlobal	2832000	1952000	0	0	4784000
LCLocal	1344000	577000	680000	623000	3224000
RSCMap	944000	339500	557500	458000	2299000

Table 3. Cost Breakup (in $) using replication technologies of various types

We report the cost achieved by all the algorithms in Table. 3. We also report the type of technologies used to protect the data containers. The *LCGlobal* algorithm picks the least cost technology that can protect the most constrained data container. Hence, it first tries to protect all the data containers using MGM. When it runs out of the technology, it picks the least cost solution that can protect all the remaining data containers. In order to pick a common replication solution for as many data containers as possible, LCGlobal selects expensive replication solution and incurrs a very high cost. *LCLocal* improves on *LCGlobal* by selecting the most cost-efficient technology for data containers belonging to each service class. However, this selection does not take into account the disparity between data containers belonging to the same service class. *RSCMap* takes all aspects of replication solutions and the requirements of individual data containers to protect the data at the minimum cost.

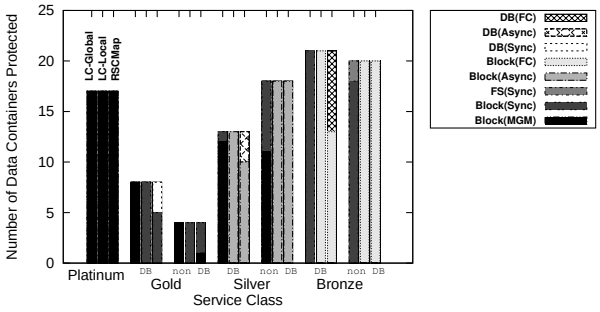


Fig. 5. Replication Solution selected by different algorithms

In order to understand the performance of the algorithms, we observe the actual placement in Fig. 5. The most striking thing about the placement is that *RSCMap* uses a wide variety of replication technologies for protecting

the data containers whereas *LCGlobal* uses very few technologies. *LCLocal* lies between these 2 extremes. Hence, *LCLocal* is able to do a more informed mapping between data requirements and replication technologies than *LCGlobal* and *RSCMap* improves the solution even further. The strength of *RSCMap* is highlighted by the selection for the database (DB) containers. *RSCMap* uses both block-level and DB-level replication for these data containers. We looked at the placement closely and observed that the databases with a large number of users were protected using block-level replication whereas the databases with small number of users were protected using DB-level replication. This distinction between data containers within the same SLA class and type helps *RSCMap* map data containers to the replication solutions, which optimize their cost. The performance of *RSCMap* also highlights the importance of incorporating multiple dimensions (e.g., space, number of users) in the overall optimization.

5.3 Sensitivity Analysis

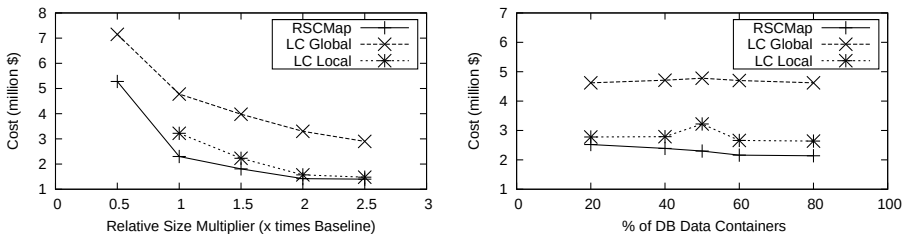
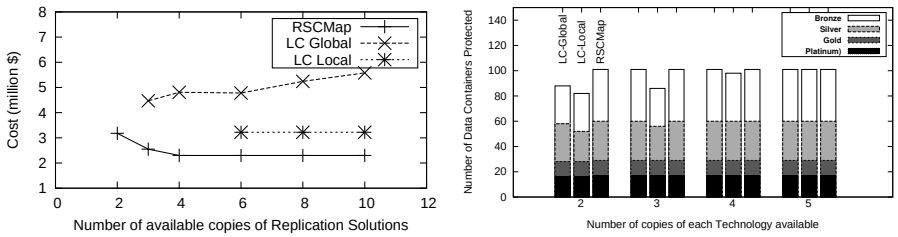


Fig. 6. Impact of (a)relative size of replication solutions and (b) heterogeneity in data containers

We next investigate the impact of fragmentation by increasing the relative size of various replication solutions. We observe (Fig. 6(a)) that an increase in the relative size of replication solutions leads to a decrease in the cost incurred to protect a given set of data containers. As the relative size of the technology increases, we reduce the number of copies of a particular solution needed, which in turn reduces the total cost. At very high sizes, all data containers in a class can be protected using a single instance of a DR technology. Hence, *RSCMap* protects all data containers belonging to a class using the least expensive replication solution for the class. The above experiment highlights the ability of *RSCMap* to adapt to various workload settings and tradeoff between fragmentation and differntiated technology selection.

In our next experiment, we investigate the impact of heterogeneity. In our baseline setting, we had data containers equally divided between DB-based and non-DB based. In this experiment, we vary the percentage of DB-based data containers. We observe that *RSCMap* outperforms *LCLocal* more significantly

when the degree of heterogeneity is high (e.g., improvement of 50% when the mix is 50 : 50). However, when all data containers are of only 1 type, the performance improvement due to $RSCMap$ reduces (15% improvement when the mix is 20 : 80). This can be explained by the fact that $RSCMap$ uses diversity between data protection requirements to find the optimal solution for each data container. When most data containers have similar requirements, the impact of intelligent technology selection is not as high.



(a) (b)

Fig. 7. (a)Impact of number of copies of replication solutions available and (b) Extent of protection

In all our earlier experiments, we assume that replication technologies are not constrained. We now investigate the performance of various algorithms when the number of available replication technologies of each type are varied. As the number of instances decrease, the impact of selecting a sub-optimal replication technology for a data container does not only lead to increased cost, but also in lack of protection for some data containers. The impact is highest for $LCLocal$, which uses the least cost solution to protect data containers of a class. However, it protectsh DB-based, File-Based and raw data containers using the least cost block-based replication technology. When the block-based replication technology instances run out, it protects DB-based data containers using appropriate DB-based replication. However, the raw volumes can not be protected using DB-based or FS-based replication. Hence, the algorithm finds no eligible replication technologies to protect these data containers (Fig. 7(b). Our final experiment highlights the importance of incorporating all attributes of data containers and replication technologies in designing the DR plan for a cloud. A holistic approach like $RSCMap$ not only ensures that resiliency requirements are met at the lowest cost but also ensures that all data containers in a cloud are protected.

References

1. N. R. Alur et al. System and method for automatically and dynamically optimizing application data resources to meet business objectives. In *US Patent Application No 20050015641*, 2005.
2. A. Azagury, M. E. Factor, and J. Satran. Point-in-Time copy: Yesterday, today and tomorrow. In *Proc. IEEE/NASA Conf. Mass Storage Systems (MSST)*, 2002.

3. Datamonitor ComputerWire Article. Available at http://www.computerwire.com/industries/research/?pid=1CEC81FD-5FDA-41D8-8FFC-79A959A87FD7
4. Synchronous Optical Network. Available at http://www.iec.org/online/tutorials/acrobat/sonet.pdf
5. Eagle Rock Alliance Ltd. Online survey results: 2001 cost of downtime. Available at http://contingencyplanningresearch.com/2001Survey.pdf, Aug. 2001.
6. S. Gaonkar, K. Keeton, A. Merchant, W. H. Sanders. Designing Dependable Storage Solutions for Shared Application Environments, In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2006.
7. IBM TotalStorage Solutions for Disaster Recovery. In *IBM Redbook* Available at http://www.redbooks.ibm.com
8. IBM TotalStorage Business Continuity Solutions Overview. In *IBM Redbook*, Available at http://www.redbooks.ibm.com
9. M. Ji, A. Veitch, and J. Wilkes. Seneca: remote mirroring done write. In *Proc. USENIX Technical Conf. (USENIX03)*, June 2003.
10. K. Keeton, C. Santos, D. Beyer, J. Chase, J. Wilkes. Designing for Disasters. In *Proc. USENIX File and Storage Technologies (FAST)*, March 2004
11. K. Keeton, D. Beyer, E. Brau, A. Merchant. On the Road to recovery: Restoring Data After Disasters. In *Proc. European Systems Conference (EuroSys)*, April 2006.
12. K. Keeton and A. Merchant. A framework for evaluating storage system dependability. In *Proc. Conf. on Dependable Systems and Networks (DSN)*, 2004.
13. T. Nayak, R. Routray, A. Singh, S. Uttamchandani, A. Verma. End-to-end Disaster Recovery Planning: From Art to Science. In *IEEE NOMS*, 2010.
14. Oracle License Prices. Available at http://www.pro-dba.com/pricing.html.
15. Sprint Communications - Internet Services E-RATE MASTER CONTRACT NUMBER: SRC26115. Available at http://www.state.sc.us/oir/rates/docs/sprint-internet-rates.htm.
16. A. Verma and A. Anand. On Store Placement for Response Time Minimization in Parallel Disks. In *Int'l Conf on Distributed Computing Systems (ICDCS)*, 2006.
17. A. Verma, K. Voruganti, R. Routray, and R. Jain. SWEEPER: An Efficient Disaster Recovery Point Identification Mechanism. In *Usenix Conf on File and Storage Technologies(FAST)*, 2008.