RSCMap: Resiliency Planning in Storage Clouds

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

[1] IBM Research, India
[2] JIMS, India

Abstract. 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 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 variety 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. 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. Further, enterprises must comply with many regulations that require data governance. By moving the data into the cloud, end users 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 [4,5]; 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

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 505–512, 2011.

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 [1,2,11]. 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. Further, 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. 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.

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. 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 [8,3,10,12].

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. (e.g., 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. *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 [10]. 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 [10].

2.1 The DR Cost Minimization Framework

Consider the problem of designing a storage provisioning 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 Disaster Recovery Service Class(DRSC) with it. We focus only on the Disaster Recovery Service Class in this work. 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). 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 $B_{i,j}$. 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. A formal mathematical description of the problem is available in an extended report [6].

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 is defined as the maximum time that will be taken to recover data after a disaster.

- Recovery Point Objective (RPO): The Recovery Point objective 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), which is captured by resource impact.
- Distance:This represents the maximum supported distance between the copies of a replication solution.

2.3 Replication Solution Model

A Disaster Recovery requirement is met by deploying a replication technology. A replication technology is characterized by the RTO, RPO, impact and distance parameters provided by the technology. In addition, deploying a replication technology leads to license cost, labour cost, and network costs. The sum of all these costs is captured as total cost, which we aim to minimize in this work. The network costs of replication are determined by the amount of network traffic generated due to the technology. A synchronous replication solution generates network traffic equal to the total amount of writes on the data container. Hence, the network bandwidth provisioned equals the peak write rate (or burst rate) to the data container. On the other hand, an asynchronous replication technology only needs network bandwidth equal to the average write rate. Further, some technologies like IBM Global Mirror use techniques like write-coalescing to reduce the number of writes and need to provision only for unique writes. We use the bandwidth transformation function $B_{i,j}$ is to capture the real network bandwidth required by the replication solution.

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 public as well as private clouds that protect a large amount of data. 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

We observe that a replication solution that can meet the *Gold* service class would also meet *Silver* or *Bronze* service class. 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* defined next.

Definition 4. *Pure Subset Replication Set: A Plan Composition problem is said to satisfy the Pure Subset Replication Set Property if*

$$\forall D_i, D_j \quad RS_i \cap RS_j \neq \phi \quad \Rightarrow \quad RS_i \subseteq RS_j \quad or \quad RS_j \subseteq RS_i \tag{1}$$

3.2 Traffic Independent Bandwidth Transformation

The replication bandwidth generated by a replication solution typically 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. This greatly simplifies the problem since the the cost of a replication solution only depends on additive properties of data containers (space, burst rate, write rate etc for multiple data containers can be added to obtain the parameters for the replication solution whereas $B_{i,j}$ can not be added).

4 RSCMap 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. We assume that 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. (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 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 $|RS_i|$). The LeastSlopeFirst (LSF) algorithm 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 5. *selectMostConstrainedDC Selection: Given a set of data containers D_i, a replication solution R_j, and a space constraint s_j^*, selectMostConstrainedDC sorts the data containers by the arity of their list of feasible replication solutions. 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 6. *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 proved the optimality of the *selectMostConstrainedDC* data container selection process and then use it to prove approximation guarantees on *LSF*. For lack of space, all proofs have been omitted and are available in [6].

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

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.*

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.

$$\forall l \in [1, d], \forall j \in [1, m], \quad d_j^l = \sum_{i=1}^{N} x_{i,j} d_i^l \tag{2}$$

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 ($currDC = \phi$).

We also make the following enhancement to *selectMostConstrainedDC* to capture the affinity of data containers with specific dimensions. When two or more data containers have the same arity, we use a tie-breaker method to choose an ordering among them. For tie breaking, we consider the solid angle the data container's requirement vector makes with the dominant dimension axis. We first select containers whose requirement runs only along the dominant dimension, in decreasing order of magnitude. We next select containers in decreasing order of their solid angle computed in the previous step. We add data containers from this sorted list to its selection, till the accumulated value in any dimension of the selected data containers equals the capacity of the replication solution in that dimension. To illustrate the strategy, consider a hyper-cuboid, where we move along the dominant dimension until we hit its wall. We then pick another dimension to move along. We have proved the following result for the algorithm.

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 $2k$. However, it satisfies additional theoretical 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, 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 have proved the following results.

Lemma 2. *The total distance traveled by LSAF is no more than* $\sqrt(k)$ *times the distance traveled by optimal.*

5 Related Work and Conclusion

Disaster Recovery in data centers has attracted a lot of attention in recent times with work on improved replication mechanisms [7], modeling dependability of a system [9], planning papers [8,3,10], plan deployment papers [10] and recovery papers [12]. 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 [10]; our work is geared towards providing the optimization engine for

the *plan composition* phase. Keeton *et al.* [8] tackle the problem of creating a cost-effective disaster recovery plan for a single application site and extend it in [3] to include shared applications. They use a linear cost model, which does not take into account price-bundling, which makes the cost functions incapable of capturing many typical planning scenarios. Further, they do not propose any optimization algorithms and suggests using off-the-shelf algorithms. The optimization algorithms used (including the new one in [3]) are based on mixed-integer programming and expensive, making them unsuitable for use in highly interactive disaster recovery planning tools [10]. Hence, earlier planning work [8,10,3] 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 [10] and the cost-functions proposed in [8,3] for the optimization framework. Our proposed algorithms provide the last missing piece required for integrated DR Planning.

References

1. Datamonitor Computer Wire Article, `http://www.computerwire.com/industries/research/?pid=1CEC81FD-5FDA-41D8-8FFC-79A959A87FD7`
2. Synchronous Optical Network, `http://www.iec.org/online/tutorials/acrobat/sonet.pdf`
3. Gaonkar, S., Keeton, K., Merchant, A., Sanders, W.H.: Designing Dependable Storage Solutions for Shared Application Environments. In: Proc. DSN (2006)
4. IBM TotalStorage Solutions for Disaster Recovery. In: IBM Redbook, `http://www.redbooks.ibm.com`
5. IBM TotalStorage Business Continuity Solutions Overview. In: IBM Redbook, `http://www.redbooks.ibm.com`
6. Jaiswal, V., Sen, A., Verma, A.: RSCMap: Resiliency Planning in Storage Clouds. In: IBM Technical Report RI11012 (2011)
7. Ji, M., Veitch, A., Wilkes, J.: Seneca: remote mirroring done write. In: Proc. USENIX Annual Technical Conference (2003)
8. Keeton, K., Santos, C., Beyer, D., Chase, J., Wilkes, J.: Designing for Disasters. In: Proc. USENIX FAST (March 2004)
9. Keeton, K., Merchant, A.: A framework for evaluating storage system dependability. In: Proc. DSN (2004)
10. Nayak, T., Routray, R., Singh, A., Uttamchandani, S., Verma, A.: End-to-end Disaster Recovery Planning: From Art to Science. In: IEEE NOMS (2010)
11. Oracle License Prices, `http://www.pro-dba.com/pricing.html`
12. Verma, A., Voruganti, K., Routray, R., Jain, R.: SWEEPER: An Efficient Disaster Recovery Point Identification Mechanism. In: Usenix FAST (2008)