Minimizing Disaster Backup Window for Geo-Distributed Multi-Datacenter Cloud Systems

Jingjing Yao, Ping Lu, Zuqing Zhu†
School of Information Science and Technology
University of Science and Technology of China, Hefei, China
†Email: {zqzhu}@ieee.org

***Abstract*—We optimize the disaster backup in multi-datacenter (multi-DC) cloud systems and design disaster-aware algorithms to realize rapid backup with the objective of minimizing the backup window for all the DCs in the network. A mixed integer linear programming (MILP) model is first formulated to optimize the backup processes of all production DCs jointly. We then develop three heuristics that use the one-step or two-step approaches for the selection of backup DCs and the calculation of backup routing paths. Simulation results show that the *TwoStep* algorithm can achieve the shortest backup window with the lowest operation complexity.**

***Index Terms*—Multi-datacenter networks, Mutual backup model, Backup window.**

I. Introduction

Recently, with the rapid rise of internet-scale systems [1] and cloud services, datacenter (DC) networks have attracted intensive interests from both industry and academia. The cloud services running on DC networks provide a huge opportunity for interactive applications such as online games, interactive television, collaborative computing and etc. Meanwhile, user experience is vital for these interactive applications, since they usually have very small tolerance to service disruptions. Therefore, there is an increasing trend to deploy geographically distributed (geo-distributed) cloud computing systems. Large enterprises such as Google, Amazon and Microsoft, are building multiple DCs in different physical locations globally to provide geo-distributed cloud services. The rationale is to deploy DCs close to the end users for improving user experience by minimizing network latency [2].

It is known that DCs are vulnerable to nature disasters. Because a DC usually carries massive data and applications, disasters can bring it down and cause huge losses to both the infrastructure providers and the data owners. For instance, the 2008 Sichuan earthquake in China had destroyed more than 60 enterprise DCs [3], and the 2011 Tōhoku earthquake and tsunami had wiped off tens of DCs and even caused some companies to file bankruptcy due to losing key data [4]. Hence, disaster backup and recovery become not only important but also necessary for DCs. Multi-DC cloud systems can provide sufficient redundancy for disaster backup and recovery, several backup mechanisms were proposed for this type of systems in [5, 6]. Wood *et al.* discussed how to provide disaster recovery as a cloud service and summarized the economic benefits associated with this approach in [7]. A resource allocation scheme for disaster recovery was proposed in [8]. The authors of [9] investigated the routing and bandwidth allocation problem for bulk data transfers among multiple DCs. Note that the backup window, which is defined as the time period for backing up the new data among DCs, is a key parameter to evaluate DC backup plans, as a prolonged backup window can impact the normal operations of both production and backup DCs and may also introduce congestions on the network links between them [10]. Therefore, companies usually run DC backups during mid-nights and try to finish them within a shortest backup window. A few techniques have been developed for reducing the backup window, including data de-duplication [10] and snapshot [11]. However, to the best of our knowledge, there is no previous work that has considered to reduce the backup window with joint optimization of the selection of backup DCs and the corresponding routing paths.

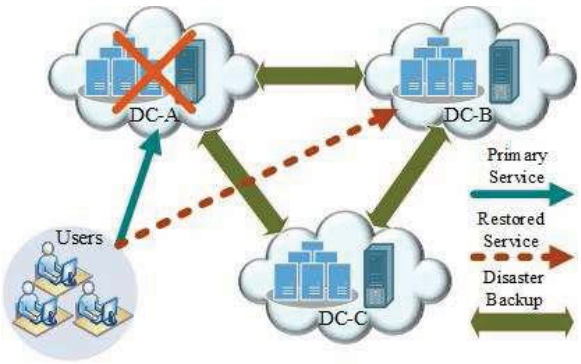


Fig. 1. Mutual backup in multi-DC cloud systems.

In this paper, we consider a mutual backup model in multi-DC cloud systems. As shown in Fig. 1, the DCs in the network back up the generated data for each other, and the generated data are transferred to the backup DCs periodically. When a disaster happens, the backup DC will recover the data and restore the cloud service in the production DC. We design disaster-aware algorithms to realize rapid disaster backup and to minimize the backup window for all the DCs in a geo-distributed multi-DCs network. We first formulate a mixed integer linear programming (MILP) model to optimize the backup processes jointly, and then design three heuristics that use one-step or two-step approaches for the selection of backup DCs and the calculation of backup routing paths.

The rest of the paper is organized as follows. Section

978-1-4799-2003-7/14/$31.00 ©2014 IEEE

II formulates the problem and gives a mixed integer linear programming (MILP) model to optimize the solution. We then propose several heuristic algorithms in Section III to reduce the computational complexity. The performance evaluation with numerical simulations is discussed in Section IV. Finally, Section V summarizes the paper.

II. Problem Formulation

A. Network Model

We model the topology of the multi-DC network as a directed graph $G(V, E)$, where V denotes the set of network nodes, and E denotes the set of directed links. Within V, there are K DC nodes that each includes a DC and necessary interconnecting equipment. We denote the set of these DC nodes as V^{dc} and have $V^{dc} \subseteq V$. For a link $e = (v, u),\ u, v \in V$, its bandwidth capacity is defined as $B_{v,u}$. For a disaster backup, we assume that the amount of new data to be backed up in a DC node $v \in V^{dc}$ is A_v, and hence the total amount of data for backup is

$$M = \sum_{v \in V^{dc}} A_v. \tag{1}$$

In order to maintain DC survivability during disasters, we need to ensure that the production and backup DCs are located in different disaster zones [12] and a single disaster cannot destroy both of them. As the DC nodes are already geo-distributed, we assume a disaster zone only covers the set of nodes V_v^z that are directly connected to the production DC v, *i.e.*, $V_v^z = \{u : (v, u) \in E, u \in V^{dc}\}$. A DC can generate new data all the time, and these data will be transferred to its backup DC during the next backup window. Then, in order to achieve rapid disaster backup, we need to design the optimization mechanism that for a production DC, it can select the backup DC and the corresponding backup routing paths, with the objective to minimize the total backup time required for all DCs in the network (*i.e.*, the backup window). Note that for the backup process, the data transmission time is usually much longer than the signal propagation delay [13], and therefore we can ignore the propagation delay when calculating the backup time.

B. MILP Formulation

We assume that the network operates based on a discrete time interval, which is denoted as Δt. As the bandwidth capacity between any two nodes is not less than the smallest bandwidth on the links, we derive the upper bound of the backup window in number of the time intervals as

$$N = \lceil \frac{M}{min(B_{u,v}) \cdot \Delta t} \rceil, \quad \forall (u, v) \in E, \tag{2}$$

where M is the total amount of data to be backed up and can be calculated with Eq. (1). To maximize network bandwidth utilization during the backup window, we assume that the diaster backup adopts cloud service [14], which means that for a production DC, the backup DC location can vary in different time intervals. Therefore, for each production DC, we need to optimize the selection of backup DC and the corresponding routing paths for each time interval Δt, with the objective to minimize the backup window. Note that for each backup process, the production DC should set up flows on multiple routing paths simultaneously to fully exploit the network throughput. We formulate a mixed integer linear programming (MILP) model to solve this problem.

Notations:

- $G(V, E)$: the physical topology.
- V^{dc}: the set of DC nodes.
- A_v: the amount of data to be backed up in DC node v.
- V_v^z: the disaster zone of DC node v.
- M: the total amount of data to be backed up.
- N: the upper bound of the number of time intervals in backup window.
- $B_{w,z}^{(j)}$: the bandwidth capacity on link $(w, z) \in E$ for the j-th time interval.

Variables:

- $x_{v,u}^{(j)}$: Boolean variable that equals 1 if node u is selected as the backup DC of node v in the j-th time interval, and 0 otherwise. Also, $x_{v,u}^{(j)} = 0,\ \forall v, u \in V \setminus V^{dc}$.
- $b_{v,u,w,z}^{(j)}$: Non-negative variable that indicates the bandwidth used on link (w, z) to back up data from v to u in the j-th time interval, and $b_{v,u,w,z}^{(j)} = 0,\ \forall v, u \in V \setminus V^{dc}$.
- $d_{v,u}^{(j)}$: Non-negative variable that indicates the bandwidth used for the backup from v to u in the j-th time interval, and $d_{v,u}^{(j)} = 0,\ \forall v, u \in V \setminus V^{dc}$.
- N_v: Integer variable that indicates the number of the time intervals used for backing up data in DC v, and $N_v = 0,\ \forall v \in V \setminus V^{dc}$.

Objective:

$$Minimize \quad \max_{v \in V^{dc}} (N_v) \tag{3}$$

The objective of the MILP is to minimize the maximum number of time intervals used for backing up the data in any DC $v \in V^{dc}$, *i.e.*, to minimize the backup window. In order to make the objective function linear, we define another variable T and add a constraint to the MILP

$$T \geq N_v \cdot \Delta t, \quad \forall v \in V^{dc}. \tag{4}$$

Then, the objective function becomes

$$Minimize \quad T, \tag{5}$$

where T is the backup window of the multi-DC cloud systems.

Constraints:

1) Backup location constraints

$$\sum_{u \in V^{dc}} x_{v,u}^{(j)} \leq 1, \quad \forall v \in V^{dc}, j \in [1, N], \tag{6}$$

$$\sum_{v \in V^{dc}} x_{v,u}^{(j)} \leq 1, \quad \forall u \in V^{dc}, j \in [1, N]. \tag{7}$$

Eqs. (6)-(7) ensure that in each interval Δt, a production DC can only select one backup DC at most to send backup data, while a backup DC also can only receive backup data from one production DC at most. These constraints help to distribute the

backup-related traffic loads evenly in the multi-DC network and avoid generating "hot-spots" during the diaster backup, and they also assist the logging system to track the backup processes easily.

2) Disaster recovery constraints

$$x_{v,v}^{(j)} = 0, \quad \forall v \in V^{dc}, j \in [1, N], \tag{8}$$

$$x_{v,u}^{(j)} = 0 \quad \forall u \in V_v^z, j \in [1, N]. \tag{9}$$

Eq. (8) ensures that a production DC cannot select itself as its backup DC, while Eq. (9) makes sure that the production and backup DCs do not fall into the same disaster zone.

3) Flow conservation constraints

$$\sum_{w \in V} b_{v,u,w,z}^{(j)} - \sum_{w \in V} b_{v,u,z,w}^{(j)} = \begin{cases} -d_{v,u}^{(j)}, & z = v \\ 0, & Otherwise, \\ d_{v,u}^{(j)}, & z = u \end{cases} \tag{10}$$
$$\forall v \in V^{dc}, j \in [1, N].$$

$$d_{v,u}^{(j)} \leq x_{v,u}^{(j)} \cdot \sum_{(w,z) \in E} B_{w,z}^{(j)}, \quad \forall (w, z) \in E, j \in [1, N]. \tag{11}$$

Eq. (10) ensures that for each backup flow, the input bandwidth equals to the output bandwidth at any intermediate node on the routing paths. Eq. (11) ensures that no bandwidth is allocated from v to u, if u is not the backup DC of v in a time interval.

4) Bandwidth capacity constraint

$$\sum_{v \in V^{dc}} b_{v,u,w,z}^{(j)} \leq B_{w,z}^{(j)}, \quad \forall (w, z) \in E, j \in [1, N]. \tag{12}$$

Eq. (12) ensures that on each link, the bandwidth allocated for all the backup flows does not exceed its bandwidth capacity.

5) Backup window constraint

$$N_v \geq j \cdot x_{v,u}^{(j)}, \quad \forall v \in V^{dc}, j \in [1, N]. \tag{13}$$

Eq. (13) ensures that the backup window only ends when all the backup processes have been finished.

6) Backup data constraint

$$\sum_{j=0}^{N} d_{v,u}^{(j)} \geq \frac{A_v}{\Delta t}, \quad \forall v \in V^{dc}. \tag{14}$$

Eq. (14) ensures that all the data is backed up successfully.

This MILP model provides the optimal solution for the multi-DC network to finish backup in the smallest backup window. However, its computational complexity is also very high. Hence, we discuss a few heuristics in the next section.

III. Heuristic Algorithms

We still assume that the network operates based on the discrete interval Δt. At the beginning of each interval, based on the current network status, the heuristic algorithm should determine the backup DCs and the corresponding routing paths. We consider two types of heuristics, one-step and two-step approaches. In the one-step approach, the algorithm obtains the backup DC and routing paths for a production DC simultaneously, while the two-step approach performs backup DC selection first and then determines the corresponding routing paths. *Algorithm* 1 shows the overall procedures of the proposed disaster backup algorithms. Here, we define R^{dc} as the node set, which consists of the production DC nodes whose backup has yet been finished. Basically, the disaster backup executes one of the algorithms discussed below in each interval Δt until all the data has been backed up successfully.

Algorithm 1 Overall Disaster Backup Algorithm

Input:
$G(V, E)$, V^{dc}, V_v^z, A_v.

```
1: T = 0, R^dc = V^dc, j = 1;
2: while R^dc ≠ ∅ do
3:    invoke Algorithm 2 or 3 or 4;
4:    update all A_v;
5:    T = T + Δt, j = j + 1;
6:    for all v ∈ R^dc do
7:       if A_v = 0 then
8:          R^dc = R^dc \ v;
9:       end if
10:   end for
11: end while
```

A. One-Step Heuristic Algorithms

We first design a greedy one-step algorithm that tries to set up the backup process for each production DC with the global maximum flow in the network. *Algorithm* 2 shows the procedures. In the loop from *Line* 4 to 11, we calculate the maximum flows (MFs) for all the feasible DC backup pairs in the current network. Then, in *Lines* 12-17, among all the MFs, we choose the one that has the largest throughput (*i.e.*, the global MF (GMF)) to set up a backup process and then update the network status. The procedures in *Lines* 3-17 are repeated until all the production DCs are served or the network bandwidth is used up. We denote *Algorithm* 2 as the one step algorithm with global maximum flow (*OneStep-GMF*).

In *Algorithm* 2, we need to calculate all the feasible MFs in the current network to set up a backup process. However, this can cause high computational complexity. Note that in order to minimize the backup window, we should handle the production DC that has more data to be backed up earlier, as it usually consumes a longer backup time. Therefore, we develop another one-step heuristic as shown in *Algorithm* 3, which first sorts the production DCs based on the amount of data for backup and then set up the backup processes for them one by one. In *Lines* 3-5, we calculate all the feasible MFs for the production DC v, and then in *Lines* 6-11, among all these MFs, we select the one that has the largest throughput to set up the backup process for v. We denote *Algorithm* 3 as the one step algorithm with most data first (*OneStep-MDF*).

B. Two-Step Heuristic Algorithm

Algorithm 4 shows the proposed algorithm for the two-step approach (*TwoStep*). We still first sort the production DCs based on the amount of data for backup. Then, for each

Algorithm 2 One-Step Heuristic with Global Maximum Flow

Input:
$G(V, E),\ R^{dc},\ V_v^z,\ A_v,\ B_{w,z}^{(j)}$.
1: $R_j^{dc} = R^{dc}$, $V_j^{dc} = V^{dc}$, $flag = 1$;
2: **while** $R_j^{dc} \neq \emptyset$ AND $flag = 1$ **do**
3: $flag = 0$;
4: **for** each $v \in R_j^{dc}$ **do**
5: **for** each $u \in V_j^{dc} \setminus V_v^z$ **do**
6: get the maximum flow $\mathcal{F}_{v,u}$ for $v \rightarrow u$;
7: **if** the bandwidth of $\mathcal{F}_{v,u}$ is not 0 **then**
8: $flag = flag + 1$;
9: **end if**
10: **end for**
11: **end for**
12: **if** $flag > 0$ **then**
13: get $\mathcal{F}_{v,u}$ that has the largest bandwidth;
14: set up backup process with $\mathcal{F}_{v,u}$ for $v \rightarrow u$;
15: $R_j^{dc} = R_j^{dc} \setminus v$, $V_j^{dc} = V_j^{dc} \setminus u$;
16: update network status;
17: **end if**
18: **end while**
19: run all backup processes for Δt;

Algorithm 3 One-Step Heuristic with Most Data First

Input:
$G(V, E),\ R^{dc},\ V_v^z,\ A_v,\ B_{w,z}^{(j)}$.
1: $R_j^{dc} = R^{dc}$, $V_j^{dc} = V^{dc}$;
2: **for** each $v \in R_j^{dc}$ in descending order of A_v **do**
3: **for** each $u \in V_j^{dc} \setminus V_v^z$ **do**
4: get the maximum flow $\mathcal{F}_{v,u}$ for $v \rightarrow u$;
5: **end for**
6: get $\mathcal{F}_{v,u}$ that has the largest bandwidth;
7: **if** the bandwidth of $\mathcal{F}_{v,u}$ is not 0 **then**
8: set up backup process with $\mathcal{F}_{v,u}$ for $v \rightarrow u$;
9: $V_j^{dc} = V_j^{dc} \setminus u$;
10: update network status;
11: **end if**
12: **end for**
13: run all backup processes for Δt;

production DC v, we select the nearest feasible DC u as the backup DC. Here, the "nearest" feasible DC means that u is out of the disaster zone of v and the hop-count of the shortest path for $v \rightarrow u$ is the smallest. Then, in *Lines* 4-9, we calculate the MF for $v \rightarrow u$ and set up the backup process with it.

IV. Performance Evaluation

We evaluate the performance of the proposed algorithms with the US-Backbone topology shown in Fig. 2, and the DC nodes are located at *Nodes* 6, 8, 9, 15, 18 and 22, *i.e.*, $V^{dc} = \{6, 8, 9, 15, 18, 22\}$. The available bandwidth on the links is uniformly distributed within $[10, 30]$ units. In each simulation, the total data for backup is fixed and the actual data on each production DC is randomly chosen while satisfying Eq. (1).

Algorithm 4 Two-Step Heuristic

Input:
$G(V, E),\ R^{dc},\ V_v^z,\ A_v,\ B_{w,z}^{(j)}$.
1: $R_j^{dc} = R^{dc}$, $V_j^{dc} = V^{dc}$;
2: **for** each $v \in R_j^{dc}$ in descending order of A_v **do**
3: get the nearest node u from $V_j^{dc} \setminus V_v^z$;
4: get the maximum flow $\mathcal{F}_{v,u}$ for $v \rightarrow u$;
5: **if** the bandwidth of $\mathcal{F}_{v,u}$ is not 0 **then**
6: set up backup process with $\mathcal{F}_{v,u}$ for $v \rightarrow u$;
7: $V_j^{dc} = V_j^{dc} \setminus u$;
8: update network status;
9: **end if**
10: **end for**
11: run all backup processes for Δt;

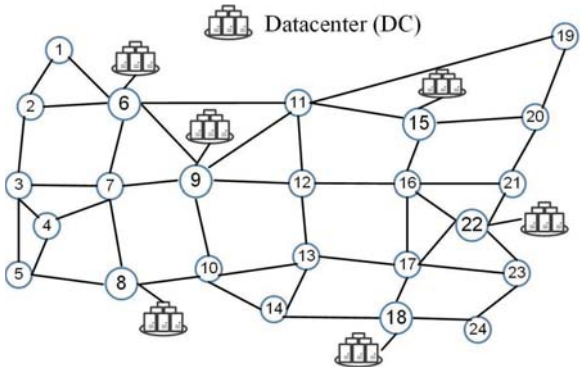
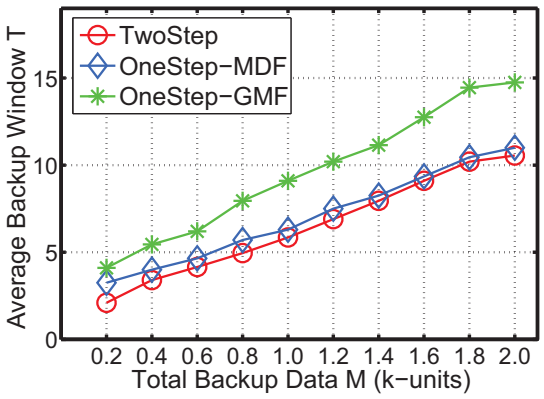
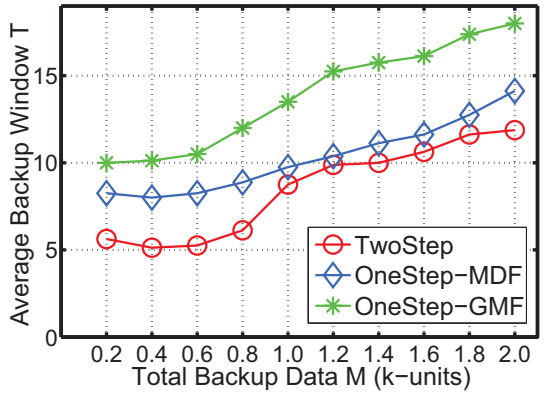


Fig. 2. US-Backbone multi-DC network topology.

Fig. 3 shows the simulation results on the average backup window obtained by the three heuristics, i.e., *OneStep-GMF*, *OneStep-MDF*, and *TwoStep*. To obtain each data point in Fig. 3, we fix M, simulate 40 different data distributions, *i.e.*, 40 different $\{A_v\}$, and then average the backup windows. Fig. 3(a) shows the simulation results when the time interval is set as $\Delta t = 2$, while Fig. 3(b) plots the results for $\Delta t = 5$. It can be seen that in both plots, *TwoStep* provides the shortest backup windows. Between the two one-step heuristics, *OneStep-MDF* provides smaller backup windows than *OneStep-GMF*. Since *OneStep-GMF* always tries to use the global MF to set up a backup process, it may only maximize the backup throughput for one DC pair. But if the geographic distance between this DC pair is relatively long, *OneStep-GMF* may also limit the throughput of other DC pairs and prolong the backup window. Meanwhile, the DC that has the most data for backup may not be handled first, which can cause dramatic increase on the backup window. As *OneStep-MDF* first sets up the backup process for the production DC that has the most data for backup, it allocates network bandwidth more wisely and therefore can provide a shorter backup window. The reason why *TwoStep* can provide the shortest backup window is that it considers the geographic distance between the DC pairs and can reduce the probability of using long routing paths with large hop-counts in the MF. Hence, the network



(a) Time interval $\Delta t = 2$



(b) Time interval $\Delta t = 5$

Fig. 3. Average backup window for different total amounts of backup data.

bandwidth can be allocated in the most efficient way. It is worth to noting that when the time interval is relatively small ($\Delta t = 2$), the average backup windows from *OneStep-MDF* are just slightly longer than those from *TwoStep*. This trend suggests that if we optimize the backup processes based on a relatively small time granularity, *OneStep-MDF* can provide comparable backup windows as *TwoStep*.

We then fix $M = 500$ units to investigate the impact of Δt on the backup window and Fig. 4 shows the results. We observe that for all the three algorithms, the backup window increases with Δt, but the backup window from *TwoStep* increases with the most moderate slope. Note that larger Δt means less frequent network optimizations and thus lower operation complexity. Hence, *TwoStep* is still the most preferred algorithm, as it achieves the shortest backup window with the lowest operation complexity.

V. Conclusion

In this paper, we optimized the disaster backup in multi-DC networks with the objective to minimize the backup window for all the DCs in the network. We first formulated an MILP model to optimize the backup processes jointly, and then designed three heuristics that used either one-step or two-step approaches for the selection of backup DCs and the calculation

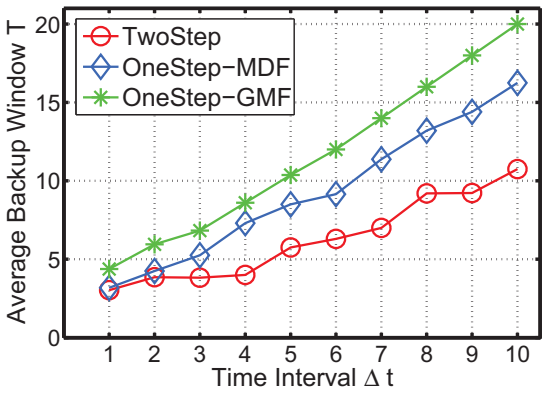


Fig. 4. Average backup window for different Δt when $M = 500$ units.

of backup routing paths. Simulation results showed that the *TwoStep* algorithm can achieve the shortest backup window with the lowest operation complexity.

Acknowledgments

This work was supported in part by the Program for New Century Excellent Talents in University (NCET) under Project NCET-11-0884, the National Natural Science Foundation of China (NSFC) under Project 61371117, the Fundamental Research Funds for the Central Universities (WK2100060010), and the Strategic Priority Research Program of the Chinese Academy of Sciences (XDA06010302).

References

[1] A. Qureshi *et al.*, "Cutting the electric bill for internet-scale systems," in *ACM SIGCOMM 2009*, pp. 1–12, Aug. 2009.
[2] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 68–73, Jan. 2009.
[3] http://en.wikipedia.org/wiki/2008_Sichuan_earthquake.
[4] http://en.wikipedia.org/wiki/2011_Tohoku_earthquake_and_tsunami.
[5] S. Frolund and F. Pedone, "Dealing efficiently with data-center disasters," *J. Parallel Distrib. Comput.*, vol. 63, pp. 1064–1081, Nov. 2003.
[6] J. Mehr *et al.*, "Hybrid distributed and cloud backup architecture," *US Patent 20100274982A1*, Oct. 2010.
[7] T. Wood, E. Cecchet, and K. Ramakrishnan, "Disaster recovery as a cloud serivce: Economic benefits & deployment challenges," in *Proc. of USENIX Workshop on Hot Topics in Cloud Computing 2010*, pp. 1–7, Jun. 2010.
[8] A. Bianco, L. Giraudo, and D. Hay, "Optimal resource allocation for disaster recovery," in *Proc. of GLOBECOM 2010*, pp. 1–5, Mar. 2010.
[9] Y. Wang *et al.*, "Optimal routing and bandwidth allocation for multiple inter-datacenter bulk data transfers," in *Proc. of ICC 2012*, pp. 5538–5542, Jun. 2012.
[10] N. Mandagere, P. Zhou, M. Smith, and S. Uttamchandani, "Demystifying data deduplication," in *Proc. of USENIX Middleware 2008*, pp. 7–12, Dec. 2008.
[11] H. Chan and T. Chieu, "An approach to high availability for cloud servers with snapshot mechanism," in *Proc. of USENIX Middleware 2012*, pp. 1–6, Dec. 2012.
[12] M. Habib *et al.*, "Design of disaster-resilient optical datacenter networks," *J. Lightw. Technol.*, vol. 30, pp. 2563–2573, Aug. 2012.
[13] L. Fleischer and M. Skutella, "Quickest flows over time," *SIAM J. Comput.*, vol. 36, pp. 1600–1630, Feb. 2007.
[14] C. Develder *et al.*, "Resilient network dimensioning for optical grid/clouds using relocation," in *Proc. of ICC 2012*, pp. 6262–6267, Jun. 2012.