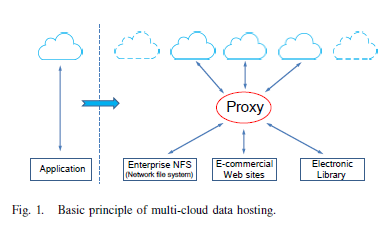
**INTRODUCTION**

Recent years have witnessed a “gold rush” of online data hosting services (or says cloud storage services) such as Amazon S3, Windows Azure, Google Cloud Storage, Aliyun OSS, and so forth. These services provide customers with reliable, scalable, and low-cost data hosting functionality. More and more enterprises and organizations are hosting all or part of their data into the cloud, in order to reduce the IT maintenance cost (including the hardware, software, and operational cost) and enhance the data reliability. For example, the United States Library of Congress had moved its digitized content to the cloud, followed by the New York Public Library and Biodiversity Heritage Library. Now they only have to pay for exactly how much they have used. Heterogenous clouds. Existing clouds exhibit great heterogeneities in terms of both working performances and pricing policies. Different cloud vendors build their respective infrastructures and keep upgrading them with newly emerging gears. They also design different system architectures and apply various techniques to make their services competitive. Such system diversity leads to observable performance variations across cloud vendors. Moreover, pricing policies of existing storage services provided by different cloud vendors are distinct in both pricing levels and charging items. For instance, Rackspace does not charge for Web operations (typically via a series of RESTful APIs), Google Cloud Storage charges more for bandwidth consumption, while Amazon S3 charges more for storage space (refer to x II-A). Vendor lock-in risk. Facing numerous cloud vendors as well as their heterogenous performances/policies, customers may be perplexed with which cloud(s) are suitable for storing their data and what hosting strategy is cheaper. The general status quo is that customers usually put their data into a single cloud and then simply trust to luck. This is subject to the so-called “vendor lock-in risk”, because customers would be confronted with a dilemma if they want to switch to other cloud venders. The vendor lock-in risk first lies in that data migration inevitably generates considerable expense. For example, moving 100 TB of data from Amazon S3 (California datacenter) to Aliyun OSS (Beijing datacenter) would consume as much as 12,300 (US) dollars. Besides, the vendor lock-in risk makes customers suffer from price adjustments of cloud vendors which are not uncommon. For example, the fluctuation of electricity bills in a region will affect the prices of cloud services in this region. We notice that giant cloud vendors like Windows Azure and Google Cloud Storage have been adjusting their pricing terms. Unexpected bankruptcy of cloud vendors further aggravates the situation. Nirvanix, which has thousands of customers including top 500 companies, suddenly shut down its cloud storage service in Sep. 2013. Ubuntu One, also a famous player in the market of cloud storage service, escaped in Apr. 2014. So clearly, it is unwise for an enterprise or an organization to host all data in a single cloud — “your best bet is probably not to put all your eggs in one basket.” Finally, uncontrolled data availability is (in a sense) another type of vendor lock-in risk. Though the service quality is formally guaranteed by service level agreements (SLA), failures and outages do occur. Almost all the major cloud vendors experienced service outages. Some outages even lasted for several hours. Multi-cloud data hosting. Recently, multi-cloud data hosting has received wide attention from researchers, customers, and startups. The basic principle of multi-cloud (data hosting) is to distribute data across multiple clouds to gain enhanced redundancy and prevent the vendor lock-in risk, as show in Fig. 1.



The “proxy” component plays a key role by redirecting requests from client applications and coordinating data distribution among multiple clouds. The potential prevalence of multi-cloud is illustrated in three folds. First, there have been a few researches conducted on multi-cloud. DepSky guarantees data availability and security based on multiple clouds, thus allowing critical data (e.g., edical and financial data) to be trustingly stored . RACS deploys erasure coding among different clouds in order to prevent vender lock-in risk and reduce monetary cost. Second, new types of cloud vendors have emerged and rapidly grown up to provide real services based on multiple clouds. Third, new development tools like Apache lib cloud provide a unified interface above different clouds, which facilitates migrating services among clouds. Nevertheless, as for multi-cloud people still encounter the two critical problems: (1) How to choose appropriate louds to minimize monetary cost in the presence of heterogenous pricing policies? (2) How to meet the different availability requirements of different services? As to monetary cost, it mainly depends on the data-level usage, particularly storage capacity consumption and network bandwidth consumption. As to availability requirement, the major concern lies in which redundancy mechanism (i.e., replication or erasure coding) is more economical based on specific data access patterns. In other words, here the fundamental challenge is: How to combine the two mechanisms elegantly so as to greatly reduce monetary cost and meanwhile guarantee required availability? The proposed CHARM scheme. In this paper, we propose a novel cost-efficient data hosting scheme with high availability in heterogenous multi-cloud, named “CHARM”. It intelligently uts data into multiple clouds with minimized monetary cost and guaranteed availability. Specifically, we combine the two widely used redundancy mechanisms, i.e., replication and erasure coding, into a uniform model to meet the required availability in the presence of different data access patterns. Next, we design an efficient heuristic-based algorithm to choose proper data storage modes (involving both clouds and redundancy mechanisms). Moreover, we implement the necessary procedure for storage mode transition (for efficiently re-distributing data) by monitoring the variations of data access patterns and pricing policies. We evaluate the performance of CHARM using both tracedriven simulations and prototype experiments. The traces are collected from two online storage systems: Amazing Store and orsair, both of which possess hundreds of thousands of users. In the prototype experiments, we replay samples from the two traces for a whole month on top of four mainstream commercial clouds: Amazon S3, Windows Azure, Google Cloud Storage, and Aliyun OSS. Evaluation results show that compared with the major existing schemes of monetary cost but also exhibits sound adaptability to data and ice adjustments. Summary of contribution. At last, our contributions in this paper can be briefly summarized as follows: 1) We propose and implement CHARM, a novel, efficient, and heuristic-based data hosting scheme for heterogenous multi-cloud environments. CHARM accommodates different pricing strategies, availability requirements, and data access patterns. It selects suitable clouds and an appropriate redundancy strategy to store data with minimized monetary cost and guaranteed availability. 2) We design and implement a flexible transition scheme for CHARM. It keeps monitoring the variations of pricing policies and data access patterns, and adaptively triggers the transition process between different data storage modes. It also starts a data migration process among different clouds if necessary. 3) We evaluate the performance of CHARM using two typical real-world traces and prototype experiments. Both trace-driven simulation and experiment results confirm the efficacy of CHARM. Roadmap. The remainder of this paper is organized as follows. First, we briefly introduce the pricing models of mainstream cloud vendors and the basic knowledge of erasure coding in x II. Then, we demonstrate the new opportunity of multi-cloud by combining replication and erasure coding in x III. In x IV and x V, we present the architecture and two key components of CHARM. x VI discusses the practical issues of CHARM. After that, we evaluate the performance of CHARM in x VII and review related work in x VIII. Finally, we conclude the paper in x IX.