**Modules**

* **Multi-cloud**:

Lots of data centers are distributed around the world, and one region such as America, Asia, usually has several data centers belonging to the same or different cloud providers. So technically all the data centers can be access by a user in a certain region, but the user would experience different performance. The latency of some data centers is very low while that of some ones may be intolerable high. CHARM chooses clouds for storing data from all the available clouds which meet the performance requirement, that is, they can offer acceptable throughput and latency when they are not in outage. The storage mode transition does not impact the performance of the service. Since it is not a latency-sensitive process, we can decrease the priority of transition operations, and implement the transition in batch when the proxy has low workload.

* **Data hosting:**

In this section, we elaborate a cost-efficient data hosting model with high availability in heterogenous multi-cloud, named “CHARM”. The architecture of CHARM is shown in Figure 3. The whole model is located in the proxy in this system. There are four main components in CHARM: Data Hosting, Storage Mode Switching (SMS), Workload Statistic, and Predictor. Workload Statistic keeps collecting and tackling access logs to guide the placement of data. It also sends statistic information to Predictor which guides the action of SMS. Data Hosting stores data using replication or erasure coding, according to the size and access frequency of the data. SMS decides whether the storage mode of certain data should be changed from replication to erasure coding or in reverse, according to the output of Predictor. The implementation of changing storage mode runs in the background, in order not to impact online service. Predictor is used to predict the future access frequency of files. The time interval for prediction is one month, that is, we use the former months to predict access frequency of files in the next month. However, we do not put emphasis on the design of predictor, because there have been lots of good algorithms for prediction. Moreover, a very simple predictor, which uses the weighted moving average approach, works well in our data hosting model. Data Hosting and SMS are two important modules in CHARM. Data Hosting decides storage mode and the clouds that the data should be stored in. This is a complex integer programming problem demonstrated in the following subsections. Then we illustrate how SMS works in detail in x V, that is, when and how many times should the transition be implemented.

* **Cloud Storage:**

Cloud storage services have become increasingly popular. Because of the importance of privacy, many cloud storage encryption schemes have been proposed to protect data from those who do not have access. All such schemes assumed that cloud storage providers are safe and cannot be hacked; however, in practice, some authorities (i.e., coercers) may force cloud storage providers to reveal user secrets or confidential data on the cloud, thus altogether circumventing storage encryption schemes. In this paper, we present our design for a new cloud storage encryption scheme that enables cloud storage providers to create convincing fake user secrets to protect user privacy. Since coercers cannot tell if obtained secrets are true or not, the cloud storage providers ensure that user privacy is still securely protected. Most of the proposed schemes assume cloud storage service providers or trusted third parties handling key management are trusted and cannot be hacked; however, in practice, some entities may intercept communications between users and cloud storage providers and then compel storage providers to release user secrets by using government power or other means. In this case, encrypted data are assumed to be known and storage providers are requested to release user secrets. we aimed to build an encryption scheme that could help cloud storage providers avoid this predicament. In our approach, we offer cloud storage providers means to create fake user secrets. Given such fake user secrets, outside coercers can only obtained forged data from a user’s stored ciphertext. Once coercers think the received secrets are real, they will be satisfied and more importantly cloud storage providers will not have revealed any real secrets. Therefore, user privacy is still protected. This concept comes from a special kind of encryption scheme called deniable encryption.

* **Owner Module:**

Owner module is to upload their files using some access policy. First they get the public key for particular upload file after getting this public key owner request the secret key for particular upload file. Using that secret key owner upload their file.

* **User Module:**

This module is used to help the client to search the file using the file id and file name. If the file id and name is incorrect means the user does not get the file, otherwise server ask the secret key and get the encryption file. If the user wants the decryption file means user have the secret key.