# Towards Optimal Data Replication Across Data Centers

Fan Ping, Xiaohu Li, Christopher McConnell, Rohini Vabbalareddy, Jeong-Hyon Hwang
Department of Computer Science
University at Albany - SUNY
{apping, xiaohu, ctm, rohiniva, jhh}@cs.albany.edu

***Abstract*—There has been proliferation of data centers that provide both computation and storage resources at diverse geographic locations. In a variety of wide area applications, data can be replicated to serve users with lower latency. This paper presents a technique that can effectively reduce the overall data access delay through gradual migration of data replicas. In contrast to previous solutions that either randomly select replica locations or process a large log of past data accesses, this new technique maintains only a small, decentralized summary of recent data accesses while achieving near optimal performance. This paper also includes an evaluation study that substantiates the effectiveness of the developed technique, and plans for extending the current research outcomes.**

## I. Introduction

Recent increases in computational resources have driven the amount of digital data to expand at a nearly exponential rate. Major Internet companies such as Google, Yahoo, Amazon and Facebook now operate scores of data centers to meet their enormous data processing needs [1], [2]. Further, more than one thousand data centers are currently in operation, enabling elastic, pay-by-use provision of hardware and software at various geographic locations [3]. To manage large volumes of online data even across data centers, several storage systems have been developed [4], [5], [6]. These systems typically replicate data to offer fast and highly available services to users. In such a context, an important challenge is to determine the locations of data replicas such that the overall data access latency is minimized.

Determining replica locations as stated above enables prompt data access. In addition to this benefit, such replica placement may facilitate fast transfer of large data objects since low-latency network connections tend to have high bandwidth [7], [8]. As each data service can be completed quickly, the overall user requests may also be handled by a relatively small number of servers, thereby reducing the resource usage. In applications where users need to obtain data within a time limit (e.g., 300ms [4]), users may have time to access a second or more replicas if they cannot access the first replica. If such an approach is infeasible due to high network latency, each user must attempt to access multiple replicas in parallel, incurring a higher network cost.

Existing storage systems either ignore the aforementioned replica placement problem [4], [5], [6] or strive to solve the problem by analyzing the user locations with high computational or spatial overhead [9], [10], [11], [12], [13], [14], [15], [16] (Section V for details). In actual systems [4], [5], [6], a large number of users may access data replicas. Thus, the problem needs to be solved in an efficient and highly scalable manner.

In this paper, we propose a new replica placement technique which gradually migrates data replicas to reduce the overall data access delay. To identify more advantageous replica locations, this approach maintains, for each data replica, a small data structure that summarizes recent data accesses. These summaries are periodically collected at a node and then processed to identify the most beneficial replica locations (i.e., those that are expected to minimize the overall data access delay). If the benefit of the new locations is substantially larger than that of the previous locations, data replicas are created at the new locations and the replicas at the previous locations are discarded.

Our evaluation demonstrates that our replica placement technique can reduce the overall data access latency by more than 35% compared to random replica placement in various contexts. It also shows that the average access delay achieved by our technique is close to the lowest average access delay found by examining all possible replica placement scenarios.

The contribution of this paper can be summarized as follows:

- We developed an efficient, highly scalable replica placement solution that can achieve near optimal data access delay.
- We devised a technique that can summarize information about user populations with low spatial and computational overhead.
- We present evaluation results that demonstrate the utility of our replica placement technique.

The remainder of this paper is organized as follows. In Section II, we provide a formal definition of the data replica placement problem. Next, we present our solution to the problem in Section III. In Section IV, we present preliminary evaluation results while comparing our replica placement algorithm with other alternatives. We summarize related work in Section V and conclude in Section VI.

## II. Problem Statement

This section defines the problem of determining appropriate locations to store data replicas. Section II-A describes the

assumptions made in this paper. Section II-B provides a formal definition of the replica placement problem.

### A. Assumptions

We assume a wide area distributed system that needs to manage data objects [4], [5]. We also assume that multiple (e.g., 30) data centers can be used to run the system. To simplify exposition, we focus on the problem of replicating one data object at a small number (e.g., 3) of data centers such that users at diverse locations can obtain the closest data replica with low latency. Here, the closest data replica is defined as the data replica that can be accessed with the lowest latency. A solution to this problem can be applied to a group of data objects by treating accesses to any object of the group as accesses to a virtual object that represents all the objects of the group.

In this paper, we assume that for each user, it suffices to access only one data replica. If data objects can be newly added or updated, accessing only one data replica leads to fast data acquisition at the expense of consistency. We plan to incorporate into our future study quorum-based approaches in which users need to access multiple data replicas to ensure stronger consistency [4], [6]. This paper also assumes that each user accesses the closest data replica. For example, a user may contact all the replica locations and use whichever replica it can obtain first. A user may also identify or estimate (Section III), before actual data transfer, a replica location that can transmit data with the lowest latency.

Another assumption made in the paper is that data objects are read much more frequently than updated. Thus, the cost of propagating updates among data replicas is ignored. We also assume that candidate replica locations (e.g., data centers) are considered only when they can handle the expected user requests. For this reason, load balancing is not studied in this paper.

### B. Formal Definition

Let $\mathcal{C}$ denote the set of available data centers and $\mathcal{U}$ denote the user clients that will access a data object $o$ within a period of time $P$ after replicas of $o$ are created. If $o$ is replicated at data centers $\mathcal{R} \subset \mathcal{C}$, the time it takes for user $u \in \mathcal{U}$ to read $o$ can be expressed as follows:

$$l(u, o) = \min_{c \in \mathcal{R}} l(u, c)$$

where $l(u, c)$ represents the latency between user $u$ and data center $c \in \mathcal{R}$. The above definition is based on the assumption that users access the closest replica (Section II-A).

Given a target degree of replication $k$, the goal of the replica placement problem is to find a set $\mathcal{R}$ of $k$ replica locations (i.e., $\mathcal{R} \subset \mathcal{C}$ and $|\mathcal{R}| = k$) that minimize the following objective function:

$$l(o) = \sum_{u \in \mathcal{U}} l(u, o) = \sum_{u \in \mathcal{U}} \min_{c \in \mathcal{R}} l(u, c).$$

Since the average access delay is $\frac{l(o)}{|\mathcal{U}|}$, $\mathcal{R}$ that minimizes $l(o)$ also achieves the minimum average access delay.

To solve the above problem in real systems, several challenges must be addressed. First, it is infeasible to know in advance the users that will access data after data replicas are created. If such future data usage is estimated based on past data accesses, we need to determine an appropriate time period for which the past data usage is analyzed. Further, the system may serve a large number (e.g., tens of millions) of users, thus we require a technique that can summarize data accesses with low spatial and computational overhead. Next, replicas may serve different user populations if each user contacts only a subset of replica locations (e.g., those that are close in the network). In such cases, information about data accesses needs to be processed efficiently even across data centers. Finally, the replica placement problem is known to be NP-hard [17], thus a low-cost solution to the problem is highly required.

## III. Replica Placement Algorithm

This paper considers the problem of determining replica locations that minimize the average data access delay. If $k$ replica locations need to be determined, a natural approach would be to group user locations into $k$ clusters and then deploy a data replica at (or near) the centroid of each cluster.

To enable the above clustering-based approach, we assign synthetic coordinates to each node in the system such that the round-trip time between two arbitrary nodes is close to the distance between the nodes in the coordinate space (Section III-A). With these coordinates, users can be treated as points in a Euclidean space and can be clustered according to proximity in the network.

As discussed in Section II-B, a large number of users may access a data replica. In this case, it would be impractical to collect information about all the users across data replicas and store it at a central server. Thus, our approach maintains for each replica, an efficient data structure summarizing the coordinates of the users that have recently accessed the replica (Section III-B). These summaries are periodically transferred from replica locations to a node and then processed to determine replica locations that would most reduce the average access delay (Section III-C). The overhead of the aforementioned approach is analyzed in Section III-D.

### A. Network Coordinate Systems

A network coordinate system embeds nodes (both servers and clients) into a virtual multi-dimensional space based on the network latencies between the nodes. This embedding is conducted such that the actual round-trip time between two arbitrary nodes is close to the distance between the nodes in the virtual space (Section V-A).

RNP [18] (Retrospective Network Positioning) is our prior work that improves both the network latency prediction accuracy and coordinate stability over Vivaldi [19], a representative network coordinate system. RNP achieves a prediction error typically lower than 10 ms for a majority of node pairs even if it runs on unstable platforms such as PlanetLab [20]. One immediate advantage of using RNP is that if a user node knows the coordinates of replica locations, it can predict the closest

**Algorithm 1**: Macro-clustering($\mathcal{R}$)

1 obtain $m$ micro-clusters from each replica location in $\mathcal{R}$.
2 use weighted K-means to cluster the $m * k$ micro-clusters into $k$ macro-clusters.
3 **for** *each macro-cluster* $c$ *among the* $k$ *macro-clusters* **do**
4 find data center $d$ whose $d.coord$ is closest to $c.coord$.
5 create a data replica at data center $d$

| Symbol | Meaning |
|---|---|
| $\mathcal{R}$ | the set of data centers that currently hold data replicas |
| $m$ | the number of micro-clusters per data replica |
| $n$ | the number of all clients accessing the data |
| $k$ | the target degree of replication |
| $c.coord$ | the network coordinates of the centroid of a macro cluster $c$ |
| $d.coord$ | the network coordinates of the a data center $d$ |

TABLE I
SYMBOLS AND THEIR MEANINGS

| | online | offline |
|---|---|---|
| bandwidth overhead | $O(km)$ | $O(n)$ |
| computation overhead | $O((km)^k * \log(km))$ | $O(n^k * \log n)$ [23] |

TABLE II
COMPLEXITY COMPARISON BETWEEN ONLINE AND OFFLINE CLUSTERING

replica with a high accuracy although it has never accessed the replicas before.

In this paper, we use RNP to assign network coordinates to nodes with low overhead. A node that communicates with a remote node adjusts its coordinates based on the round-trip time to the remote node and the coordinates of the remote node. Based on such coordinates, nodes in the system can be treated as points and can be clustered based on the network latency (rather than the geographic distance) between them.

### *B. Per-Replica User Coordinate Clustering*

K-means clustering is a representative method that can be used to cluster data points. However, it can not be directly applied to a continuous stream of data points that represent the coordinates of the nodes that access data. Thus, we have devised a two-phase online clustering approach.

During the first phase, for each server that currently holds a data replica, the coordinates of the clients that access this server are classified into $m$ micro-clusters. For a micro-cluster $i$, only four variables are maintained: (1) $count_i$ for the number of data accesses by the clients whose coordinates belong to the cluster, (2) $weight_i$ for the overall amount of data exchanged with the users, (3) a multi-dimensional vector $sum_i$ for the sum of coordinate values for each dimension, and (4) another vector $sum2_i$ for the sum of squares of coordinate values for each dimension. It should be noticed that the centroid of the cluster can be expressed as $sum_i/count_i$ and the standard deviation of the coordinates in the cluster can also be computed from $count_i$, $sum_i$ and $sum2_i$.[1]

Whenever a client accesses the replica, a micro-cluster $i^*$ whose centroid is closest to the coordinates of the client can be easily determined since $i^* = \arg\min_i |sum_i/count_i - u|$, where $u$ is the coordinates of the user. If the distance between $u$ and $sum_{i^*}/count_{i^*}$ is within the standard deviation (i.e., $u$ is sufficiently close to the centroid of the cluster $i^*$) the variables of cluster $i^*$ are updated using $u$. Otherwise, a new cluster is created based on the coordinates of the new user and then two closest clusters are merged.

### *C. Determination of Replica Locations*

Whenever $k$ replica locations need to be determined, the micro-clusters described in Section III-B are sent to a central server. This server then merges the micro-clusters into $k$ macro-clusters each of which represents a major user population. As Algorithm 1 shows, the central server forms macro-clusters by running a weighted K-means algorithm [21]. This algorithm only differs with the regular K-means algorithm in that each micro-cluster is treated as a data pseudo-point and the distance between two pseudo-points is defined as the distance between the centroids of the corresponding micro-clusters.

When the weighted clustering is completed, the centroid of each macro cluster is conceptually the location that can serve the users within the cluster with the minimum average latency. Thus, for each macro-cluster, the closest data center is selected as a new location to host a data replica (lines 3-5).

The process described above is conducted periodically (e.g., everyday or every week). Since the cost of migrating data may not be ignored (e.g., $.1 per GB [22]), our approach carries out data migration only when the gain in the quality of service (e.g., reduction in latency) compared to the migration cost is higher than a certain threshold. This approach can also vary the number of replicas by setting the parameter $k$. Such adjustment is needed when it is desirable to create more replicas as the demand of an object increases or to discard replicas as the demand decreases.

### *D. Cost Analysis*

Table II compares our online clustering approach with an offline approach in terms of the network bandwidth and computation overhead. In our online clustering approach, only $k * m$ micro-clusters need to be sent to the central server. On the other hand, offline clustering requires transferring the coordinates of all the user clients that have accessed the replicas. In our current implementation, the size of each micro-cluster is less than 1KB. If 100 micro-clusters are maintained for each of three replicas, each replica placement involves transferring 300 micro-clusters (i.e., less than 300KB of data). If such micro-clusters were formed based 1 million user accesses, offline clustering would require transferring more than tens of megabytes of data.

[1]The standard deviation of random variable $X$ can be calculated using $\sqrt{E[X^2] - (E[X]^2)}$

According to [23], the complexity of offline clustering can be expressed as $O(n^k * \log n)$ where $n$ denotes the number of data points (i.e., the number of users that have accessed the data replicas). In our online clustering, only $km$ pseudo node coordinates (i.e., micro-clusters) are clustered, leading to $O((km)^k * \log(km))$ computational overhead. The difference in the computational overhead between the offline and online clustering approaches becomes more significant, as the replicas are accessed more frequently (i.e., $n$ increases) and the degree of replication and the number of micro-clusters decreases (i.e., $k$ and $m$ decrease, respectively).

## IV. Evaluation

This section demonstrates preliminary evaluation results on the effectiveness of our replica deployment algorithm. The evaluation settings are described in Section IV-A. Results obtained by varying the number of data centers, the number of replicas, and the number of micro-clusters are discussed in Sections IV-B, IV-C and IV-D, respectively.

### A. Settings

To compare our replica placement algorithm with other alternatives, we have implemented an event-based simulator in Java. This simulator can emulate communications between nodes based on real network traffic data collected from 226 PlanetLab nodes [24]. Based on such emulated network communications, the simulator can assign synthetic coordinates to all the 226 nodes using RNP [18].

When the simulator begins, it selects a number (e.g., 20) of nodes as locations that may store data replicas. Since these nodes are dispersed at diverse geographic locations, each of them is assumed to represent a different data center. Next, the simulator determines, among the candidate replica locations (i.e., available data centers), those that will store data replicas using a certain replica deployment strategy (replica deployment strategies compared in the paper are explained in the next paragraph). Then, the simulator treats the remainder of the nodes as clients that want to access data and calculates the average of the data access delays that the clients perceive. As described in Section II, each client in the simulator accesses the closest replica. All the results presented in this section were averaged over 30 simulation runs each of which began with different candidate replica locations.

The replica placement approaches compared in the paper are as follows ($k$ denotes the target degree of replication):

1) **random** - This approach randomly selects $k$ data centers.
2) **offline k-means clustering** - In this approach, the coordinates of all the clients that access data are recorded at a server. When replica locations need to be determined, k-means clustering is applied to the recorded coordinates. Then, for each of the resulting coordinate clusters, a candidate replica location closest to the centroid of the cluster is chosen. This approach incurs high overhead and is not scalable since the coordinates of all the clients must be collected at a server.

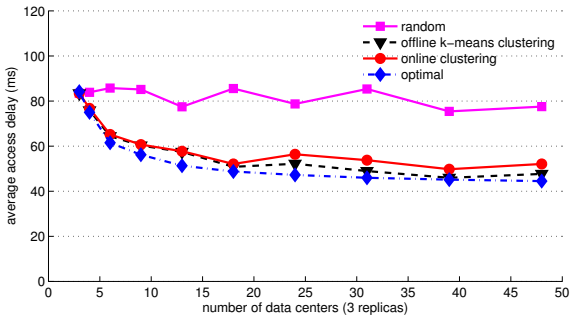


Fig. 1. Impact of the number of data centers

3) **online clustering** - This is the approach proposed in the paper (Section III).
4) **optimal** - This is an impractical approach in which for each possible replica deployment (i.e., each combination of replica locations), the true average access delay is calculated by considering all the clients and then the replica deployment that leads to the lowest access delay is identified. This exhaustive, impractical approach shows the true optimum thus included in this paper for comparison purposes.

### B. Impact of the Number of Available Data Centers

The first set of evaluations demonstrates the impact of the number of available data centers on the average access delay (Figure 1). In this study, the target degree of replica is set to 3. In all the replica deployment approaches except for "random", the average access delay decreases as more candidate replica locations are available. The reason behind this is that in contrast to "random" the other approaches tend to successfully find more beneficial replica locations as soon as they become available. Figure 1 clearly demonstrates the benefit of considering a large number of data centers as potential locations to store data.

Another crucial observation is that both our online clustering and the offline k-means clustering algorithms achieve near optimal performance. While the offline clustering algorithm incurs high storage and network costs, our online algorithm maintains (and transfers over the network) only a small amount of data that summarizes user locations, thereby achieving high scalability.

### C. Impact of the Degree of Replication

In this evaluation, we fix the number of data centers to 20, and vary the degree of replication from 1 to 7 (Figure 2). In all replica deployment approaches, the average access delay decreases as the number of replicas increases. Such reduction in access delay was achieved since the distance from a client to the closest replica tends to decrease with more replicas. However, such reduction in access delay also decreases as the number of replicas increases (particularly after 4). Figure 2 shows that the performance of our online clustering algorithm is comparable to that of offline k-means clustering and slightly

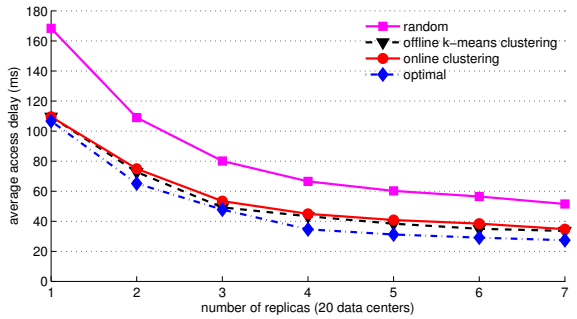


Fig. 2. Impact of the degree of replication

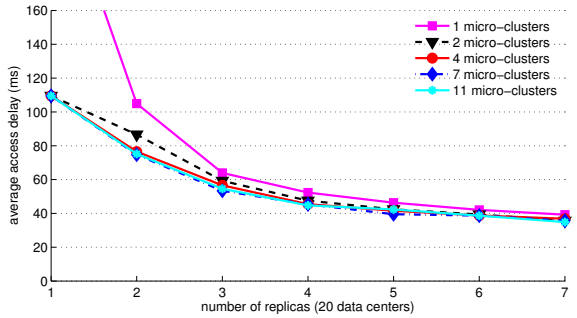


Fig. 3. performance vs. num micro-clusters

worse than that of the optimal deployment found through exhaustive search. It also shows that it consistently achieves at least 35% lower average access delay compared to random placement.

### D. Impact of the Number of Micro-Clusters

As illustrated in Section III, our online clustering algorithm maintains a number of micro-clusters per replica. Figure 3 shows the impact of the number of micro-clusters on the average access delay. As the number of micro-clusters for each replica increases, our online clustering algorithm can summarize the coordinates of the users with finger granularity, thus can more accurately estimate beneficial replica locations (those that lead to low data access latency). In our evaluation, the average access delay was nearly minimized when 4 micro-clusters are maintained for each replica. Based on this result obtained with 226 nodes, we anticipate that still a small number of micro-clusters would be needed even if a large number of clients are served. We intend to examine the impact of number of micro-clusters in a substantially larger setting.

## V. Related Work

This paper presents an approach that determines appropriate replica locations based on a summary of user locations. This section provides an overview of related work.

### A. Network Coordinate Systems

Several techniques have been developed to assign synthetic coordinates to nodes such that the round-trip time between arbitrary nodes can be estimated from the coordinates [18], [19], [25]. An early approach called GNP (Global Network Positioning) first embeds tens of landmark nodes into an Euclidean coordinate space based on the round-trip times between the nodes and then allows other nodes to determine their coordinates by communicating with the landmarks [25]. Vivaldi is a decentralized approach in which each node adjusts its coordinates using the round-trip time to a remote node and the remote node's coordinates [19]. The replica deployment technique presented in this paper assigns coordinates to nodes using our RNP (Retrospective Network Positioning) technique [18]. In contrast to GNP, RNP does not require pre-configured landmarks. RNP can also more accurately predict network latencies than Vivaldi by consuming information differently according to the reliability of the information.

### B. Data/Server Replication in Wide Area Applications

Data replication has been extensively studied in the context of web applications and content distribution networks. Pallis et al. design a greedy heuristic for content replication [26]. In their approach, only data movement cost (latency, bandwidth) between servers are considered and minimized through a gradual optimized greedy algorithm. Sivasubramanian et al. also develop a data placement approach that takes into account the system load as well as the read-write ratio of data [10]. This approach tries to place a master server to a position near users that intensively update data, and other replica servers based on the system load and the read-write ratio. Chandy et al. [11] solves the problem from a different perspective by splitting each data object and tries to place the pieces onto servers in a greedy way that minimizes data access latency. A recent study compares 10 representative data placement heuristics [12].

Server replication has also been studied by various researchers [13], [14], [15], [16]. Qiu et al. present several server replica placement approaches and a naive greedy algorithm that effectively reduces latency at a high computation cost [13]. Szymaniak et al. present an offline replica placement approach that uses network coordinates [14], [15]. In this approach, the coordinate space is divided into many small cells and then a number of most crowded cells (i.e., those to which the largest number of clients belong) are chosen as locations to place servers. This approach has an inherent limitation of ignoring all the cells except for the most crowded cells, thus may not perform adequately [13].

A main difference of the above techniques compared to ours is that they record information about each individual client, thus are fundamentally limited in scalability. In contrast, our approach maintains only small amounts of data summarizing locations of users that have recently accessed data. Despite such summarization, our approach achieves near optimal performance (Section IV).

## VI. Conclusion

We have presented a data replica placement algorithm that can efficiently achieve near optimal data access delay. This

approach continually determines most advantageous replica locations by using a small, decentralized summary which is updated with low overhead for each data access. To the best of our knowledge, this is the first work that can effectively determine replica locations despite a large number of accesses to data replicas.

In this paper, we have focused on reducing the overall data access delay. We intend to extend this work by taking into account other aspects including load balancing and data availability. We also plan to carry out more realistic evaluation study based on data accesses in actual applications.

## REFERENCES


[1] L. Barroso and U. Holzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–108, 2009.

[2] CNET News, "Google spotlights data center inner workings," http://news.cnet.com/8301-10784_3-9955184-7.html.

[3] Data Center Map, "Data Center Statistics," http://www.datacentermap.com/datacenters.html.

[4] D. Hastorun, M. Jampani, G. Kakulapati, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazons highly available key-value store," in *Proceedings of 21th ACM SIGOPS symposium on Operating systems principles*. Citeseer, 2007.

[5] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[6] B. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H. Jacobsen, N. Puz, D. Weaver, and R. Yerneni, "PNUTS: Yahoo!'s hosted data serving platform," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1277–1288, 2008.

[7] S. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring bandwidth between planetlab nodes," *Passive and Active Network Measurement*, pp. 292–305, 2005.

[8] http://www.yuiblog.com/blog/2010/04/08/analyzing-bandwidth-and-latency/.

[9] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: automated data placement for geo-distributed cloud services," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*. USENIX Association, 2010, pp. 2–2.

[10] S. Sivasubramanian, G. Pierre, and M. van Steen, "Autonomic Data Placement Strategies for Update-intensiveWeb applications," in *Advanced Architectures and Algorithms for Internet Delivery and Applications, 2005. AAA-IDEA 2005. First International Workshop on*. IEEE, 2006, pp. 2–9.

[11] J. Chandy, "A generalized replica placement strategy to optimize latency in a wide area distributed storage system," in *Proceedings of the 2008 international workshop on Data-aware distributed computing*. ACM, 2008, pp. 49–54.

[12] S. Khan and I. Ahmad, "Comparison and analysis of ten static heuristics-based Internet data replication techniques," *Journal of Parallel and Distributed Computing*, vol. 68, no. 2, pp. 113–136, 2008.

[13] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1587–1596.

[14] M. Szymaniak, G. Pierre, and M. Steen, "Latency-Driven Replica Placement," in *Proceedings of the The 2005 Symposium on Applications and the Internet*. IEEE Computer Society, 2005, pp. 399–405.

[15] M. Szymaniak, "Latency-driven replication for globally distributed systems," 2007.

[16] Y. Chen, R. Katz, and J. Kubiatowicz, "Dynamic replica placement for scalable content delivery," *Peer-to-Peer Systems*, pp. 306–318, 2002.

[17] M. Karlsson and C. Karamanolis, "Choosing replica placement heuristics for wide-area systems," 2004.

[18] F. Ping, C. McConnell, and J.-H. Hwang, "A retrospective approach for accurate network latency prediction," in *Proceedings of the 2nd Workshop on Grid and P2P Systems and Applications (GridPeer)*, 2010.

[19] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *SIGCOMM*, 2004, pp. 15–26.

[20] http://www.planet-lab.org/.

[21] C. Aggarwal, J. Han, J. Wang, and P. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[22] Amazon EC2 Pricing, http://aws.amazon.com/ec2/pricing/.

[23] http://en.wikipedia.org/wiki/K-means_clustering.

[24] http://www.eecs.harvard.edu/~syrah/nc/.

[25] T. S. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *INFOCOM*, 2002, pp. 170–179.

[26] G. Pallis, A. Vakali, K. Stamos, A. Sidiropoulos, D. Katsaros, and Y. Manolopoulos, "A latency-based object placement approach in content distribution networks," 2005.