Scheduling Memory Access on a Distributed Cloud Storage Network

Roberto Rojas-Cessa, Lin Cai, and Taweesak Kijkanjanarat

***Abstract*—Memory-access speed continues falling behind the growing speeds of network transmission links. High-speed network links provide a means to connect memory placed in hosts, located in different corners of the network. These hosts are called storage system units (SSUs), where data can be stored. Cloud storage provided with a single server can facilitate large amounts of storage to a user, however, at low access speeds. A distributed approach to cloud storage is an attractive solution. In a distributed cloud, small high-speed memories at SSUs can potentially increase the memory access speed for data processing and transmission. However, the latencies of each SSUs may be different. Therefore, the selection of SSUs impacts the overall memory access speed. This paper proposes a latency-aware scheduling scheme to access data from SSUs. This scheme determines the minimum latency requirement for a given dataset and selects available SSUs with the required latencies. Furthermore, because the latencies of some selected SSUs may be large, the proposed scheme notifies SSUs in advance of the expected time to perform data access. The simulation results show that the proposed scheme achieves faster access speeds than a scheme that randomly selects SSUs and another hat greedily selects SSUs with small latencies.**

I. Introduction

DATA link rates increase at rates with a steeper slope than that described by Moore's law. In contrast, memory access speed has been improving at a dismally small rate and has shown no signs of catching up any time soon. This paper refers to as memory to the general means of storing data, including hard drives (HD), flash memory, and random access memory (RAM) devices.

In contrast, the increase of bandwidth of optical network links has been staggering. Examples of these optical technologies are wavelength division multiplexing (WDM) and dense WDM (DWDM). The advances of interconnection technologies encourage the use of packet networks as the transport mechanism for data storage.

Storage access speed is mostly governed by the access time of the storage device. Therefore, network access speed may not be fully utilized when used to access storage. Furthermore, the speed gap may prevent the development of applications based on data handling and processing [1]. This bottleneck leads to the placement of high-density memory on devices that are as far as possible from the core of the network and as close as possible to links with the smallest speeds. Nevertheless, no matter how far storage is placed from high-speed links of a network, data must reach memory, sooner or later.

Datasets to be stored can be coarsely categorized as large or small. The content of a HD (e.g., all files or a large database in it), that is to be backed up, can be considered as an example of a large dataset that is desired to be stored quickly and reliably. Examples of small datasets, or metadata, are data of large search engines, such as Google's engines. Search engines use high-performance databases located in datacenters to search requests issued by Internet users and these search engines are required to respond to a very large number of queries each second. To keep up with such demand, Google's search engines are use more than twelve stealthy datacenters, strategically distributed [4], [5]. Databases in these datacenters store a large aggregated volume of data for long periods of time. Although these search engines use large and complex cache systems (e.g., including page ranking and other algorithms to sort information) to speedup responses, access time continues to be a bottleneck. Google has reported that memory access time is one of the most challenging issues [6].

Storage area networks (SANs) and network access storage (NAS) are two approaches to support large databases and fast data access. SAN [1] uses servers with large storage units, and clients access the servers to access storage. As a datacenter, SANs' addresses are known (through the domain name system, DNS, and IP address resolution) by users. NAS, on the other hand, uses (standalone) storage devices provisioned with their own network interfaces, and servers in the network are used to control and keep track of the clients and the stored files [7]. Combinations of SANs and NASs are commonly found. In both approaches, the storage systems may use HD arrays to achieve large storage capacities and high-speed access. Several examples of systems with high-performance goals have been proposed [9]-[11]. In these approaches, datasets from source hosts are stored in the disk arrays, one file per disk. Therefore, the storage and retrieval speeds are governed by the access speed of a single HD in the array.

In the lower layers, the well known redundant array of inexpensive disk (RAID) system is an approach to provide high-speed access to slow storage (i.e., HDs). RAID is basically built as a farm of disks interconnected in parallel and in arrays to increase access speeds (the reliability of RAID is rather a consequence of having several parallel HDs; this parallelism has also become a marketing feature). However, the parallelism of mechanical systems requires fault tolerance measures to provide reliability for devices with high possibility of failure (the larger the number of HDs, the higher the

R. Rojas-Cessa and L. Cai are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102. Emails: {rojas}@njit.edu.

T. Kijkanjanarat is with the ECE Department, Faculty of Engineering, Thammasat University (Rangsit Campus), Pathumtani, Thailand. Email: taweesak@engr.tu.ac.th.

possibility of a failure as a successful access to memory depends on a larger number of HDs). Still, a RAID system adopts a centralized implementation, where all HDs must be close to each other and internally interconnected to allow parallel access in a single place in the network. There are therefore disadvantages in this approach; one is that RAID may be prohibitively expensive for the common Internet user, and that the system would require a larger number of parallel (low speed) links to support all the aggregated users. In fact, by having a centralized approach, the number of users that can access the RAID system is limited. New approaches to substitute HDs by using flash memory have been attempted (FusionIO, Intel X25-E, Samsun PB22-J), but the fastest of them hardly sustains 0.4 Gbps, if at all available [15].

A datacenter used to provide cloud services, specifically cloud storage, may provide a very large storage. However, this storage may use the technologies discussed above. A scalable alternative is to use a *distributed cloud*, where the storage units can be placed in different places of the network. To make use of the network speed in a distributed cloud, the datasets are partitioned into blocks and blocks stored in different cloud nodes. In this way, several blocks can be accessed at the same time and from different hosts. This parallel access multiplies the memory access speed. However, the access to a storage unit depends on the part of the dataset required, as host may be reachable with different latencies (this includes the access time of storage device and the elapsed time to reach a storage unit).

Because of the location of different storage units, different latencies are also expected, and the set of storage units selected determines the data access speed. Therefore, the scheme used to select storage units becomes important, and scheduling of user access to the memory resources is then required to avoid increase of the memory access time added by the network properties.

This paper proposes a selection scheme based on estimation of latencies to access distributed storage units. The proposed scheme estimates the different set of latencies for a small set of storage units, with small latencies, to achieve high access speed. To make it effective, the scheme notifies the storage units about the access times in advance to avoid unnecessary delays. The performance of a proposed scheme is evaluated through computer simulation. The results of the proposed scheme is shown to provide faster access to data as compared to a scheme based on random selection of units and a scheme that greedily selects units with the smallest latencies. The results show that only a portion of small-latency storage units are needed to provide high storage and data access speeds.

The remaining of this paper is organized as follows. Section II describes the partitioning of datasets needed to make use of the high-speed network and memory at storage units. Section III describes different properties of a distributed cloud and introduces the proposed latency-aware access scheme. Section IV introduces the random-based access scheme and a greedy selection scheme, both used for comparison purposes. The section also presents a performance evaluation of these three schemes obtained through computer simulation. Section V presents the conclusions.

II. Partitioning of Data Sets and Storage Service Units

Datasets, in the form of files, are divided into small blocks and are distributed among several cooperative hosts, called here as storage system units (SSUs). Figure 1 shows an example of the partitioning of a dataset into small blocks of data. In the following description, SSUs perform either write or read functions, and each function is called a memory access. An SSU that issues a write process on another SSU(s) is called originating SSU. An SSU that provides data from its local memory system or hard drive (data retrieval) as a response to a request from an originating SSU is called a receiving SSU. Here, SSU can have several levels of memory, from

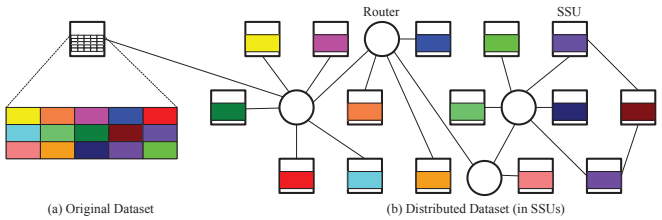

Fig. 1. A dataset partitioned in several blocks, where each block is stored in a different receiving SSU.

the fastest (but smallest, e.g., SRAM) to the largest memory (but slower, e.g., DRAM or HD) where data can reside for an arbitrary amount of time. The number of blocks that can be stored in a SSU depends on the size of the largest memory (memories of different speeds or capacity are considered). A block sent to a SSU is first received by the fast memory, and it is slowed down for long term storage. The latency of the transmissions of blocks from the originating SSU and each receiving SSU is considered to categorize SSUs in difference latency groups. The latencies from the originating to receiving SSU and those of the reverse path define the transmission rates of single SSU. For simplicity, it is considered that an originating SSU may store a data block per receiving SSU. Once blocks are assigned to different SSUs, which are selected by a management scheme according to their availability and other properties, they are transmitted by the originating SSU. Figure 2 shows an example of a local and extended network with SSUs. The receiving SSUs have similar delays to the originating SSUs, such that the access time is the same for each block and SSU.

A dataset of size L bits is divided into L_i ($1 \leq i \leq n$) blocks of size of k bits, which is defined by size of the fastest memory size in the SSU. In cases where a larger data blocks are handled, several blocks can be concatenated to form a large segment. For simplicity, only blocks are considered in the remainder of this paper.

Figure 1 also shows the partitioned dataset distributed on SSUs in a network. With this distribution of blocks, new requirements are set:

1) blocks must be stored in known locations for scheduled and timely retrieval,
2) each SSU must be categorized by levels of reliability, accessibility (speed), and availability,

3) data must be replicated according to the required reliability level or use coding for data recovery [16], and
4) data must be secured.

Only item 1) is considered in this paper. SSUs are considered sufficiently reliable and available (Item 2) in the remaining of this paper. The discussion is then centered in the latency or response time of a SSU.

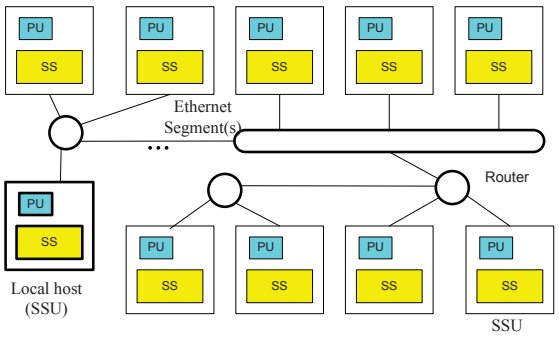

Fig. 2. Example of a network of SSUs with heterogeneous access links.

III. Segmentation and Localization of Datasets

Large datasets are partitioned into k-bit blocks, as described above. Each of the L_i blocks is sent to a receiving SSU. This network adopts a global management table that indicates the location of SSUs and their ranking features (e.g., access rate, level availability). The global management table is replicated and held by a set of selected SSUs distributed in the network to relax high traffic demand. In addition, each SSU has a local localization and evaluation table to keep information on where the blocks that belong to this SSU are kept. The table also includes information about the receiving SSUs who have interacted with the (originating) SSU. The local tables give SSUs and clients autonomy [20]. The localization table also keeps latency values to indicate the transmission delay that each block undergoes (and used for scheduling the retrieval of these blocks). The selection of SSUs may resemble the process to select hosts in a peer-to-peer (P2P) network, where each peer uses a location table to indicate the localization of files (blocks) and the SSU rating [17], [18].

To access receiving SSUs, the originating SSU sends a broadcast request to those SSUs with similar features obtained from the evaluation table. Available SSUs send an acknowledgement to the requesting SSU (otherwise, a negative acknowledgement is used), including an access number to either indicate the earliest time that the peer becomes available or the level of peer's load, and the originating SSU evaluates the receiving SSUs availability. Data blocks are dispatched afterwards, and the information of the localization and evaluation tables is updated. Although the broadcast request/reply process is simple, the database holding the information about the written blocks might have significant complexity. To avoid consuming extra resources for large datasets, information (i.e., addresses and other values) of the storing SSUs is compressed into a tree, where the root is the originating SSU. The access to the data blocks can use bandwidth reservation to reduce unexpected changes of access latencies. Figure 3 shows an example of originating SSUs interacting with receiving SSUs. The SSUs holding a Rx label are the receiving SSUs, and the SSUs holding a Tx label are the originating SSUs. The red line indicates the possible requests sent by originating SSUs and the blue line indicate the possible data transmission after receiving SSUs have been reserved.

The originating SSUs evaluate the properties of possible receiving SSUs. The parameters for evaluation are reliability (loss of data or access), availability (on-line time and storage availability), and data security (e.g., levels of security offered by the receiving SSU) are considered to rank SSUs. Information about participating SSUs is updated in the global management table and in localization and evaluation table of the originating SSU. Different from large datasets, small datasets can be stored in a one or several SSUs. However, the cost may be high as the utilization ratio is:

$$UR_j = \frac{D_i}{S} \tag{1}$$

where D_i is the size of the small dataset, with $D_i \leq k$, and S is the size of memory where the dataset is stored. This utilization ratio can be represented in terms of the link transmission capacity. However, the list of individual blocks may be large for some applications (e.g., a search engine application). In this case, the originating SSU keeps a large table of addresses of host storing several (small) data blocks.

Two different models for cooperative SSUs are considered, the server mode, where some SSUs only provide storage service but they do not demand storage from other SSUs, and the peer-to-peer mode, where SSUs can interact as storing and receiving SSUs. In both models, SSUs can be recruited from user's hosts that are considered cooperative hosts with a broadband interface. This paper considers the peer-to-peer model, where a SSU can either receive or store a data block at a time.

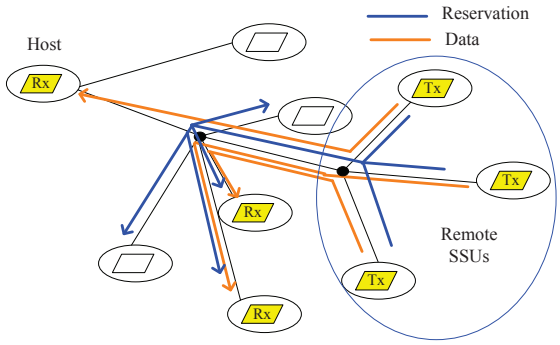

Fig. 3. Example of data access, where an originating SSUs request data blocks from different receiving SSUs.

A. Latency-Aware Scheme for Selection of Receiving SSUs

The localization and management table in an originating SSU has a list of possible receiving SSUs with the associated (average) latencies. This is, the originating SSU, denoted now as SSU_i, creates a dataset management table (DMT) to

indicate the blocks and assigned receiving SSU, or SSU_j. The DMT also indicates the average latency $l(j)$ for SSU_j. The number of SSU to be selected is determined by the number of blocks in a dataset. The average latencies of the receiving SSUs are sorted in ascending order, where the SSU with the smallest latency $l_s(j)$ is defined as $\forall j$, $l_s(j)$= $Min\{l(j)\}$, and the latency with the largest latency $l_l(j)$ is defined as $\forall j$, $l_l(j)$= $Max\{l(j)\}$. The latencies of SSU_j are then sorted as $\{l_s(j), \ldots, l_l(j)\}$, $\forall j$.

If a dataset of length L bits is to be accessed, and the access link is one of bandwidth B b/s, the time to download it is $\frac{L}{B}$ seconds. Then the blocks are scheduled to be transmitted (received) from time t_s, to time $t_s + \frac{L}{B}$. The first SSU that can be used is that with $l_l(j) \leq \frac{L}{B}$, and the second SSU is selected with $l_l(j) \leq \frac{L}{B} - \frac{k}{B}$, until the selection of the SSU to perform the access to the first block is assigned. In this way, the SSUs with different latencies are selected without recurring to only the SSUs with the smallest latencies.

After the set of SSUs are scheduled, they are assigned to blocks that are needed in that order by the originating SSU. Once the SSUs and blocks are cross assigned, the originating SSU can issue an access request (for either retrieval or storage) at least l_l seconds in advance to all receiving SSUs. The request indicates the required access time and the receiving SSUs calculate the start-of-transmission time. These time values are used to make SSUs with low latency hold their transmission until a time that after their latency would make a block arrive in the desired order.

Average latencies of the SSUs are monitored, and changes are registered in the latency table and in the DMT. Migration of memory blocks from SSUs that become ineligibly slow is performed only after a memory access is requested by the application using the dataset (the migration is not performed if access to the dataset is not required).

IV. Performance Evaluation

Performance evaluations of the latency-aware access scheme were performed to estimate the average access time of datasets with SSUs interconnected in a mesh network (i.e., grid topology). Figure 4 shows an example of the mesh network adopted. The SSUs in the mesh have been randomly selected as active or idle. Active nodes are active SSUs, indicated by a circle at an intersection. A single intersection (without a node) indicates an idle node and they are ignored. Path latencies are considered symmetrical for both directions.

The proposed scheme is compared to the following two schemes:

Random Selection of SSUs (Random). This selection scheme selects SSUs randomly, without considering SSU's latency and availability. Each SSU has the same probability of being selected. Here, L_i depends on the size of the dataset. If SSUs reject the request (a SSU can decline to receive a block if the maximum number of sharing users per SSU is reached), the originating SSU can recur to several selective iterations. Until all blocks are assigned.

Smallest Latency First (SLF). This selection scheme aims to select the SSUs with the smallest latency needed for L_i blocks. If the SSUs with the latency are exhausted, the SSUs with the second smallest latency are then selected, and so on. In this way, it is expected that the SSUs with higher bandwidth or located close to originating SSU are selected.

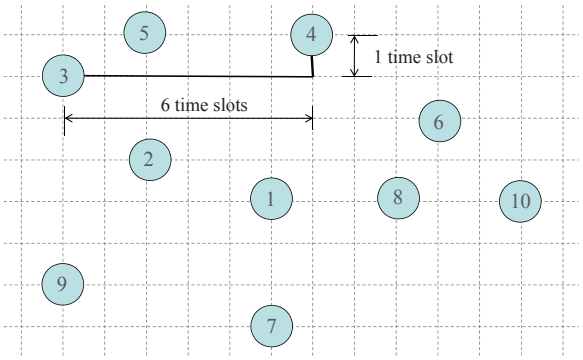

Fig. 4. Example of a mesh network. Inter-connectivity is shown with horizontal and vertical lines and active SSUs as circles.

Considering homogeneous SSUs, all blocks have fixed size and the time to process a retrieval/store takes one time slot. In this simulation, there are 50 SSUs and 10 files in the network, each file contains L_i blocks. An SSU requests access to a file with a frequency defined by a request probability (P_r). A file access for the functions of storing or retrieval are considered indistinct. When P_r=0, SSUs, and therefore blocks, remain without being accessed. When the P_r=1.0, files are selected in each time slot for access and SSUs with blocks are accessed. SSUs may receive multiple requests in a time slot and their response is bound by their processing and access capabilities. The latency between one SSU and another is set to 3, 6,..., 30 time slots, randomly assigned with a uniform distribution. Routing for the SSUs is indicated in the localization tables at SSUs and it is considered known during the simulation.

First, we investigate the average access time (for access to the complete dataset, L). Figure 5 shows the average access time (time slot) for datasets with sizes from 1 to 50 blocks. As the dataset size increases, the average access time increases quickly for the random selection scheme. From this figure, we can see that the average access time can be dramatically reduced by using the SLF scheme, as this scheme achieves lower access time (of several thousand time slots) than that of the random selection scheme. However, the proposed scheme achieves even lower average access times than the SLF scheme. This result indicates that there is a significant time saving during data access by using latency-aware selection schemes.

Second, we investigate how the average access time is affected by the network load. This network load is modelled by using the dataset request probability P_r from 0 to 1. Figure 6 shows the resulting average access time by the random selection scheme, the SLF scheme, and the proposed scheme. In this experiment, the bandwidth of the SSUs has been set to a limit: a receiving SSU can allow the access to one block each time slot. The average size of the dataset, in the average number of blocks, is set to 10 and 40, for separate tests. When

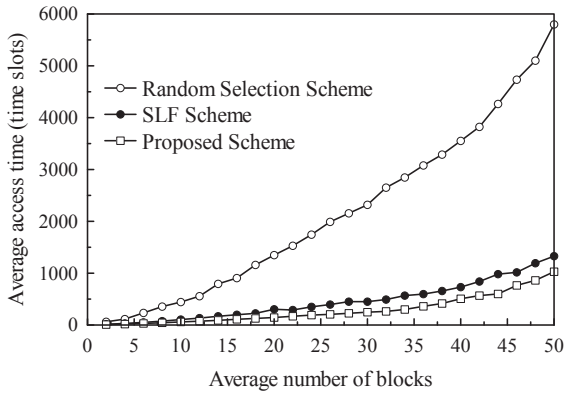

Fig. 5. Average access time of each file for different number of blocks (L_i).

the dataset size is small, 10 blocks, the SLF and the proposed schemes have similar average access times, while the average access time of the random selection scheme is larger. When the dataset size is larger, 40 blocks, the proposed scheme shows lower average access time than those of the random selection scheme and the SLF scheme. Still, the SLF scheme outperforms the random selection scheme, whose average access time increases greatly. Note that the average access time of the proposed scheme remains almost constant for different request probabilities while the SLF scheme, although with small average access time, it shows a slight dependency on the dataset size. The proposed scheme does not suffer from it because it schedules the transmission of the blocks from SSUs with different latencies, and in this way, the SSUs may be used more efficiently than by the other two schemes. This property of the proposed scheme is an indicator of its scalability as in some cases, we would expect to access very large datasets under large network loads.

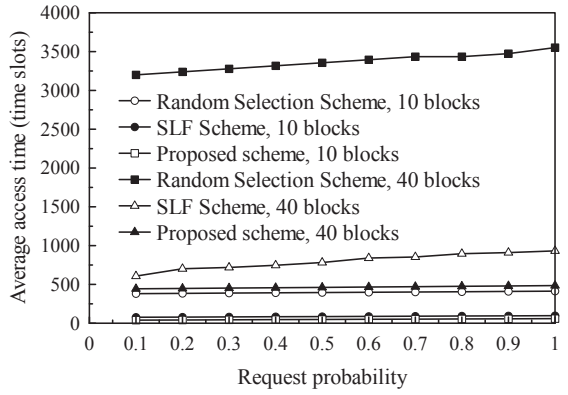

Fig. 6. Average access time of random and latency-aware schemes under different request probabilities.

We also investigated the average access time for SSUs for different levels of sharing in SSUs. In the previous experiments, SSUs were not limited to a number of originating SSUs. For example, a receiving SSU could share its resources with the other 99 SSUs. In this experiment, an SSU is limited to share its resources from 1 to 99 SSUs. As discussed before, the largest the sharing in a receiving SSU, the larger the probability of having contention for resources by originating SSUs, and therefore, larger average access time. Figure 7 shows the average access time in function of the level of sharing, which is expressed in the number of originating SSUs sharing the resources of a receiving SSU. We observe average access time increases as more originating SSUs share a receiving SSU. Again, the random selection scheme shows the largest average access time, and the SLF and proposed schemes show significantly lower average access times. From these two schemes, the proposed scheme shows the lowest average access time for different numbers of SSUs sharing a receiving SSU.

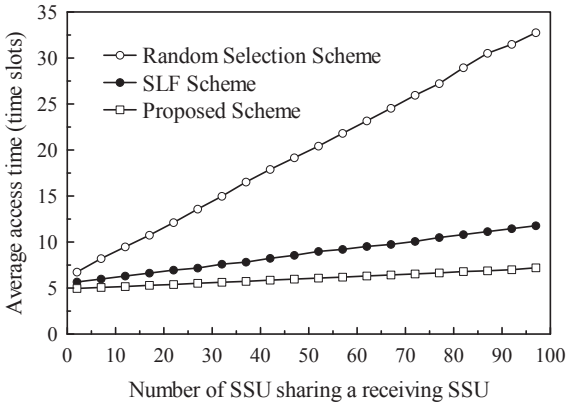

Fig. 7. Average access time for different number of SSUs sharing a receiving SSU.

V. Conclusions

This paper proposed a scheme to select distributed storage units to obtain small access times to data. These distributed storage units, also called SSUs, avail large storage for a large number of users. The different geographical location and network conditions of the SSUs may be represented with different latencies, and the selection of these SSUs determines the access times.

The proposed scheme selects SSUs with a wide range of latencies in order to use participating storage units and to not overuse SSUs with lower latencies. This makes the proposed scheme latency aware. It access SSUs with small latencies in the beginning of the access to a dataset, and those hosts with larger latencies for the remaining parts of the dataset. The performance of the proposed scheme is compared to that of a random selection scheme, which is a latency unaware scheme, and to the smallest latency first (SLF) scheme, which is also a latency aware scheme. The simulation results show that the proposed scheme achieves lower average access times than those of the random selection scheme and SLF scheme. Furthermore, the proposed scheme shows more resilience under a network with high load and under a network with than high level of sharing, than the other two schemes.

References

[1] X. Molero, F. Silla, V. Santonja, and J. Duato, “Performance analysis of storage area networks using high-speed LAN interconnects,” (ICON 2000), IEEE International Conference, pp. 474 - 478, September 2000.

[2] T.C. Jepson, "The basics of reliable distributed storage networks," IT Professional, Vol. 6, Issue 3, pp. 18-24, May-June 2004.

[3] Q. Xin, E.L. Miller, S.J.T.J.E. Schwarz, and D.D.E. Long, "Impact of failure on interconnection networks for large storage systems," Mass Storage Systems and Technologies Proceedings, 22nd IEEE / 13th NASA Goddard Conference, pp. 189 - 196, April 2005.

[4] "Map of All Google Datacenter Locations," [Available Online]. http://royal.pingdom.com/2008/04/11/map-of-all-google-data-center-locations/

[5] R. Miller, "Google Data Center FAQs," [Available Online]. http://www.datacenterknowledge.com/archives/2008/03/27/google-data-center-faq/, March 2008.

[6] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," Computer Networks and ISDN Systems, vol. 30, Issues 1-7, pp. 107-117.

[7] N. Ansari and S. Yin, "Storage Area Networks Architectures and Protocols," The Handbook Of Computer Networks Volume III: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications (Hossein Bidgoli, ed.), pp. 217-234, John Wiley & Son, ISBN 978-0-471-78458-9, 2008.

[8] S. Yin, Y. Luo, L. Zong, S. Rago, J. Yu, N. Ansari, and T. Wang, "Storage Area Network Extension over Passive Optical Networks (S-PONs)," IEEE Communications, Vol. 46, No.1, pp. 44-52, January 2008.

[9] H. De-zhi and F. Feng, "Research on Theory and Experiment of a High Performance Storage Network," ChinaGrid Annual Conference, pp. 153-157, August 2008.

[10] T. Kanada, T. Onada, I. Yamashita, and T. Aoyama, "Netwarp: an ultra-high-throughput computer communications service," IEEE Network, vol. 11, issue 1, January-February, pp. 44-50, 1997.

[11] V. Vishwanath, R. Burns, J. Leigh, and M. Seablom, "Accelerating Tropical Cyclone Analysis using LambdaRAM, a Distributed Data Cache over Wide-Area Ultra-Fast Networks," Future Generation Computer System, vol. 25, Issue 2, February 2009 (in press), pp. 184-191.

[12] L. Zhu and M. Zhong, "Research on Data Security of the Storage Network," Communications, Circuits and Systems Proceedings, 2006 International Conference, Vol. 3, pp. 1596 - 1601, June 2006.

[13] E. Burns, and R. Russell, "Implementation and Evaluation of iSCSI over RDMA," Storage Network Architecture and Parallel I/Os, Fifth IEEE International Workshop, pp. 3 - 10, Sept. 2008.

[14] I. Fumito, O. Hiroyuki, N. Yoshihiro, and I. Makoto, "On Maximizing iSCSI Throughput Using Multiple Connections with Automatic Parallelism Tuning," Proc. *IEEE International Worskshiop on Storage Network Architecture and Parallel I/Os*, pp. 11-16, September 2008.

[15] D. A. Patterson, P. Chen, G. Gibson, and R.H. Katz, "Introduction to Redundant Arrays of Inexpensive Disks (RAID)," Proc. IEEE Computer Societey International Conference 1989, pp. 112-117, Feb 27-Mar 3, 1989.

[16] D. Bauer, P. Hurley, and M. Waldvogel "Replica Placement and Location using Distributed Hash Tables," Proc. IEEE LCN, pp. 316-324, Oct. 2007.

[17] M. Macedonian, "Distributed File Sharing: Barbarians at the Gate?," IEEE Computer, Vol. 33, Issue 8, pp. 99-101, Aug. 2000.

[18] Y. Wang, X. Yun, Y. Li, "Analyzing the Characteristics of Gnutella Overlays," Proc. IEEE IV International Conference in Information Technology, 2007, pp. 1095-1100, 2-4 April, 2007.

[19] E. He, J. Leigh, O. Yu, and T.A. Defanti, "Reliable Blast UDP: predictable high performance bulk data transfer," Proc. *IEEE International Conference on Cluster Computing (CLUSTER'02)*, 23-26 Sept. 2002, pp. 317-324.

[20] H.-C. Hsiao and C.-T. King, "Tornado: a capability-aware peer-to-peer storage network," Parallel and Distributed Processing Symposium, pp. 22 - 26, April 2003.