

User-Centric Data Migration in Networked Storage Systems

Sukwoo Kang
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120, USA
sukwkang@us.ibm.com

A. L. Narasimha Reddy
Texas A&M University
College Station, TX 77843, USA
reddy@ece.tamu.edu

Abstract

This paper considers the problem of balancing locality and load in networked storage systems with multiple storage devices (or bricks). Data distribution affects locality and load balance across the devices in a networked storage system. This paper proposes a user-centric data migration scheme which tries to balance locality and load in such networked storage systems. The presented approach automatically and transparently manages migration of data blocks among disks as data access patterns and loads change over time. We implemented a prototype system, embodying our ideas, on PCs running Linux. This paper presents the design of user-centric migration and an evaluation of it through realistic experiments.

1. Introduction

Storage "bricks" are being used to build cost-effective storage systems that can scale over a broad range of system sizes. A storage brick contains a microprocessor, a small number of disk drives and network communications hardware. Such an approach has been advocated by many researchers [3, 16, 17]. Some of the production systems have taken a similar approach [18].

Network attachment of storage systems has also resulted in the possibility of consolidating or pooling of storage systems over wider geographic areas, for example storage systems connected over Local/Metropolitan/Wide Area Networks (LAN/MAN/WAN). It is expected that such network connectivity of storage systems will (a) enable wider (geographically) distribution and access of storage and (b) enable new storage paradigms (such as "storage as a service" [1]).

It is possible to build a large-scale storage system based on the infrastructure mentioned above. For example, grid computing applications such as TerraGrid Cluster File System [4] or IBM General Parallel File System (GPFS) [5] can employ multiple storage bricks to provide high perfor-

mance and scalability. These storage bricks can be attached to clusters over LAN or over WAN [6].

Figure 1(a) shows the example usage of this infrastructure in a University campus. There are three Department file servers each of which has its "local" (which is attached directly to the server) disk. Each disk is also attached to network so that each server can access other server's disk if it is necessary. During normal times, each server uses its local disk, but there may be cases when one server needs more storage resources than it has. Such a case can occur, for example, when CS Department server temporarily needs lots of storage space due to large-scale computing. Rather than buying and attaching more physical disks to CS server, it would be better if we can use other server's available storage and network bandwidth. This application scenario of cooperative storage consolidation can be generalized to storage hosting service as shown in Figure 1(b). In Figure 1(b), storage devices are geographically distributed and attached to the IP network. Any file server can lease available storage resource as it needs. In this configuration, the increased flexibility of storage allocation poses challenges to data distribution among disks because data distribution of each server over the network impacts its performance directly. Similar applications of storage consolidation are possible in grid computing environments where large data sets are shared among multiple users at different supercomputing centers.

In a large-scale storage system, it is important to (a) allocate data efficiently because the cost of data reallocation is high due to the large scale and large network latencies and (b) redistribute data adaptively in an automated manner according to configuration changes or workload changes (after initial allocation) to improve performance.

The problem described in (a) is referred to in the literature as the file allocation problem (FAP). There is a significant body of research on this problem [7], but it is not our focus. In this paper, we focus on the problem of *data redistribution* (described in (b)) in storage systems built out of networked devices. This paper focuses on *data migration as a redistribution action*. An area of ongoing research, au-

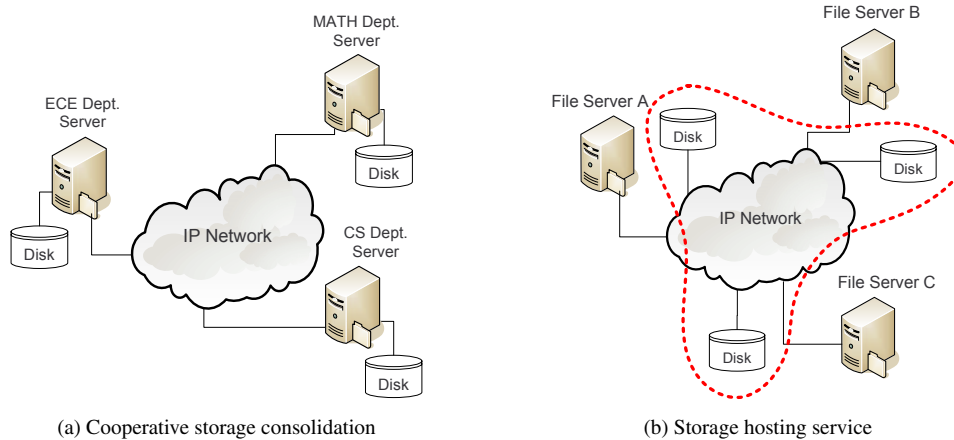


Figure 1. Application examples of networked storage systems.

tomated data migration requires deciding *when* to migrate, *where* to migrate, *what* data-set to migrate, and *how* to migrate (migration speed) [8]. In this paper, we consider these migration decision issues in terms of balancing locality and load balance as explained below.

When data is "local" or proximate to the user¹ over the network, user can observe lower network latencies and hence can observe better performance. However, if the user's data sets are larger than the capacity of the local devices, a tendency to keep data local may cause thrashing problems. Second, if the user's data set receives a significant number of I/O requests, the local devices may not be able to support the required I/O rates or data rates. On the other hand, if user's data is distributed widely over such a storage network, we may be able to exploit multiple devices to support the required I/O and data rates. Wider distribution of data also may result in better load balancing. However, accesses over the network to "remote" devices may incur extra network latency overheads and may tend to be slower than local accesses. Hence, there is a need to balance the locality of data accesses with load balancing in such a system. In this paper, we consider *longer-term locality* considerations at the I/O or storage system beyond what is already exploited in memory. Data locality and load balancing issues are studied earlier in memory systems, for example [9].

Multiple users, diverse workloads, and time varying I/O demands make this problem even more challenging. When a user accesses data from multiple locations on the network, data that was local during a previous access may become remote at a later time.

In networked storage systems, the performance difference arises due to network latencies and differences in load at the disks. When devices have equal response times, a lo-

cal disk will provide better performance than a remote disk, due to network latencies. However, the response times of devices may fluctuate over time due to differences in loads and workloads. A local disk may get slower than a remote disk due to the increased load as more and more data is migrated to a local disk. The migration of data has to take these issues into account. In networked storage systems, the direction of migration of heavily referenced data may not always be in a fixed direction, from a remote device to a local device. When multiple users are considered from multiple vantage points of the network, the data may have to migrate in different directions for different users at the same time.

Even when a single user is considered, migrating data to a local disk may not always be beneficial even when thrashing (due to large data sets) is not a problem. The local device may not be able to support the required I/O rates or data rates. It may be better to keep some data remote in order to exploit multiple devices to achieve higher I/O and data rates. Such considerations become more difficult when multiple users access storage from multiple vantage points of the network.

We propose *user-centric* migration to address these issues. In user-centric migration, each user or application makes migration decisions based on what is best for its application. This is contrasted with a centralized network-centric migration of data. We use I/O request response time as a measure of performance to decide on the suitability of a device. We explain user-centric migration with a single user and two devices (A and B), while the approach is more general. If device A is local to the user, initially the user may find A offers better performance than B. Data is then migrated to A. As more and more data is migrated to A, the load on device A may get so high that its performance may get worse than the performance of B. At that point, the

¹In this paper, *user* is file system or application of storage devices.

migration of data to A will stop. If the response time at A continues to remain high and is found to be worse than B, eventually some active data may be migrated to B to reduce the load on A in order to reduce the response times at A. This process results in a stable operating point when the response times to both devices A and B are nearly the same (including the effects of load and network latency). Locality is exploited as more data migrates to A, but load balancing will also be considered to make sure that disk A doesn't get too heavily loaded as data is migrated to A.

In general, when data can reside on multiple (more than two) devices, data currently on a device with higher response time will be migrated to the device with the lowest response time.

When multiple users share storage devices over the network, each user employs user-centric migration of his/her data sets to improve the performance or access times to his/her data. Data migration tries to balance locality and balance load of each user's data accesses in such an environment. Moreover, as the load from different users fluctuates over time, user-centric migration is expected to smooth out the loads at storage devices.

Our work makes the following contributions and differs from the previous work in the following ways: (a) proposes (and evaluates) user-centric migration policy to guide the migration process, (b) chooses *active data* (that is currently read or written) for migration to reduce migration cost, (c) considers the possibility of migrating active data to remote devices in order to improve load balancing, (d) uses longer-term performance metrics to guide migration decisions, and (e) considers data locality on networked devices.

The remainder of this paper is organized as follows. Section 2 describes the algorithm of the user-centric migration, followed by our prototype implementation and evaluation in Section 3. Section 4 points to related work and discussions. Section 5 concludes the paper.

2. User-Centric Migration

In this section, we first describe data placement and next describe user-centric migration policy. We present important design issues related to user-centric migration.

2.1. Data Placement

In large-scale storage systems, storage system's software may employ virtualization in order to allow data to be placed where they are needed. Such virtualization breaks direct coupling between physical and logical addresses and enables mapping of one logical set of data to one or many physical locations. Commercial products such as IBM Collective Intelligent Bricks (CIB) support this feature [3].

In such systems, an indirection map, containing mappings of physical to logical addresses, is maintained. Thus, every read and write request is processed after consulting the indirection map and determining the actual physical location of data. Similar structures have been used by others [2, 14, 15]. In our system, as the data migrates from one device to another device, there is a need to keep track of the remapping of the block addresses. When data is migrated, the indirection map needs to be updated. This is an additional cost of migration. We factor this cost into the design, implementation and evaluation of the proposed scheme.

Migrating data in units of file system blocks (usually 4KB) is easier, but requires a larger amount of information in the indirection map. In order to accommodate these two competing issues of flexibility and the need for smaller indirection maps, we allocate data in 128KB chunks. In order to allow migration of data in multiples of file system block sizes (typically, 4KB), we also maintain a block validity map. For this purpose, our indirection map contains a block validity bit of each file system block and two locations for each data chunk (e.g., part of a chunk may reside at device A and the other part may reside at device B). An entry of an indirection map, in our system, is a 5-tuple as shown below: (*logical-chunk-identifier*, *A*<*address*, *validity-map*>, *B*<*address*, *validity-map*>). We have considered multiple options for the indirection map, but the evaluation of these options is outside the scope of the current paper (primarily due to lack of space). Data caching in shared memory machines have explored similar issues of data structures, flexibility and overheads [32]. It is also noted that the indirection map itself can migrate over the network and can be cached within the memory of individual storage bricks for performance reasons.

Data is migrated to a device that has unused or unallocated storage. A background process merges inactive partial chunks between two devices to keep unallocated storage space available on each device for facilitating migration.

2.2. Design Issues

User-centric migration is based on virtualized storage architecture which is explained above. In this subsection, we explain the design rationale of user-centric migration in terms of migration decision issues.

When to Migrate: User-optimal routing has been proposed in [28] for adhoc wireless networks and has been applied in other contexts [27]. While switching paths over which packets are routed does not involve significant overheads, data migration in storage systems involves costs in reading (from old location), transferring (over the network) and writing data (into new location). These costs need to be taken into account in our approach of user-centric data migration.

We factor these costs by initiating migration only when the performance difference between devices exceeds by some amount, say δ (*migration threshold*). Considering two devices on the network, data present on one device A, is migrated to another device B, when response time $r_A - r_B > \delta$, where r_i includes both the network access times and device response times. When the device is locally-attached (directly connected to that server), r_i measures data access latency from the local disk (disk seek and rotation time plus data transfer time plus disk queuing time). For a network-attached disk, network latency and network storage protocol processing delay are additionally added. Our approach keeps track of response times of different devices over the network continuously (as explained below) to facilitate the migration. The δ parameter controls the onset of migration in the system. A small δ makes data migration more responsive to differences in response times and a very large δ leaves the data where it is allocated. Ideally δ should depend on the standard deviations of response times of both devices. We want to migrate when $\mu_A - \sigma_A > \mu_B + \sigma_B$ or $\mu_A > \mu_B + (\sigma_A + \sigma_B)$ where μ_i is the mean and σ_i is the standard deviation of response time (r_i). In this study, based on the observed statistics ($\sigma_A \approx \sigma_B$), we use $\delta = 2 * \sigma_A$.

Caching in memory can take advantage of short-term temporal locality of data access patterns. Data migration is intended to take advantage of longer-term temporal locality in data access patterns. Hence, we measure response times over longer periods of time to guide data migration decisions. To minimize the short-term variations while keeping track of longer-term trends in performance, we employed exponential averaging which is widely employed in network measurements, for example in estimating round trip times and queue lengths [31].

Where to Migrate: In general, when data can reside on multiple (more than two) devices, data currently on a device with higher response time will be migrated to the device with the lowest response time. It is possible to design several strategies that accomplishes this goal. In this study, we sort devices based on response times and prioritize migration from the device with the highest response time to the device with the lowest response time. If response times to be compared are the same or in similar range, we use randomization to break the tie.

What Data to Migrate: In order to reduce the cost of migration, data that is currently read or written to device A is migrated to B. If migrated data is not accessed often, migration may not be helpful in improving performance. Therefore, we need to choose part of data from active data as migration candidates. For this purpose, we maintain *migration candidate list* employing probabilistic LRU algorithm [33]. In this scheme, an accessed item if not already on this list, is entered into the list with a small probability. Hence, the list contains mostly frequently accessed items. Since the list is

driven by the LRU policy, not recently accessed items fall off the list over time. As a result, the list mostly contains frequently and recently accessed data. When migration onset condition is met, we consult migration candidate list to check if current accessed data is in the list. Migration occurs only if data is in the list.

As a result, migration involves the steps of transferring the currently accessed data over the network and writing to the second device. This policy of migrating active data, more importantly, results in not causing any additional load at the device with higher response times.² We also considered other migration policies such as *migration on read* and *migration on write*. The former policy migrates data only when it is read and the latter one moves data only when it is written. We don't present these results here due to lack of space. The policy presented in this paper is *migration on access* where data migration occurs when data is read or written.

How to Migrate: When the response times satisfy the constraints mentioned above, active data migration is initiated. The migration rate, rate at which data is migrated, needs to be carefully chosen. High migration rate will impact normal read and write accesses by increasing the load on the network and the devices. High migration rate also may alter the data access patterns quickly and significantly to result in considerable changes to response times at different devices. This may result in oscillations if not carefully managed. Too small a migration rate may not adjust access time imbalances quickly enough to improve performance. We use *migration tokens* to control the speed of migration. The number of migration tokens controls the number of requests that are migrated at any given time. For data migration (read or write), it must get a token first. If it cannot get a token, read and write requests are processed normally; migration doesn't occur.

Allocation Policy: Data migration remaps initial allocation of blocks based on performance. Initial allocation influences the load balance and data locality and hence influences data migration. We consider two allocation strategies in our study: striping and sequential allocation. Striping initially allocates data evenly over the devices resulting in balanced load (in most cases). The sequential allocation method allocates data locally until storage space is exhausted on the local device, at which point data is allocated on a remote device. Sequential allocation favors local devices over load balancing.

We study user-centric migration with both allocation methods. When coupled with striping, hot data would be migrated to local device over time, improving locality and decreasing load balance. When coupled with sequential al-

²Migrating inactive data from heavily loaded device increases load further on that device because migration involves reading data from that device.

location, hot data may be migrated to remote device when the local device gets heavily loaded, thus improving load balance over time.

Multi-user Environment: User-centric migration migrates data in a user-selfish manner based on data access latency observed by each user. The migration actions initiated by one user may affect the performance of other users for some time. However, it is expected that the migration actions will tend to improve the performance of all users over longer periods of time. We will study user-centric migration in a multi-user environment to understand these issues.

3. Evaluation

For evaluation, we implemented the following systems as a Linux kernel driver. Each configuration name consists of $\langle allocation\ policy - migration\ policy \rangle$

- **STR-NOMIG:** Data is striped over available disks. No migration is used.
- **SEQ-NOMIG:** Data is sequentially allocated. No migration is used.
- **STR-MIG:** Data is allocated by striping and migration is performed using user-centric migration protocol.
- **SEQ-MIG:** Data is allocated sequentially and migration is performed using user-centric migration policy.

The next subsections present performance evaluation and comparison between each system.

3.1. Workload

We used SPECsfs benchmark as workload for our study. SPECsfs 3.0 is the latest version of the Standard Performance Evaluation Corp.’s benchmark that measures NFS file server’s performance [34]. It is a synthetic benchmark that generates an increasing load of NFS operations against the server and measures the response time and the server throughput as load increases. It defines both the operation mix and scaling rules. The operation mix of SPECsfs 3.0 is mostly small metadata operations and reads, followed by writes. SPECsfs creates files that will later be used for measurement of the system’s performance according to its scaling rules and performs transactions following the operation mix.

SPECsfs reports a curve of response time vs. *delivered* throughput (not offered load). The signature of the SPECsfs results’ curve contains three key features: the *base response time*, the *slope* of the curve in the primary operating regime, and the *throughput saturation point*. At low throughput, there is a base response time which is the best response time obtainable from the system. As load on the server increases, response time tends to increase (e.g., linear relationship). This slope indicates how well the server responds

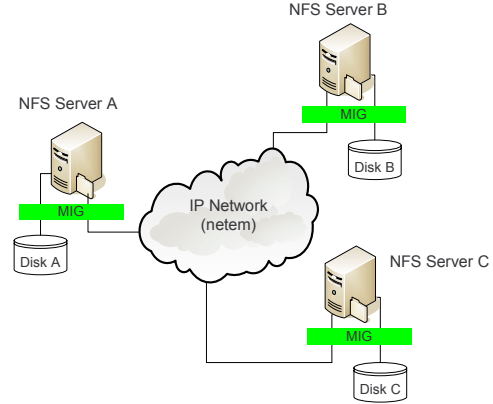


Figure 2. Experimental configuration. The filled box of MIG shows our implemented user-centric migration system. For single-user environment, NFS server A is used with Disk A and Disk B. In multi-user environment, multiple servers share multiple disks.

RTT			
avg.	min.	max.	mean dev.
9.0ms	4.3ms	18.5ms	2.3ms
17.0ms	12.4ms	24.3ms	2.2ms
25.0ms	20.5ms	32.2ms	2.2ms

Table 1. Three different network latencies used in this paper. Netem is used as a network emulator.

as load increases. As the load further increases, there will be a throughput saturation point at which a bottleneck in the system limits the throughput. Several factors such as network, the speed of the server and client processor, the size of file cache, and the speed of the server disks determine these features [35].

In this paper, we focus on data migration strategy on server disks. Thus, we configure the system to have a bottleneck in server disks to increase the likelihood of data migration. All the following results are for systems running SPECsfs 3.0, NFS version 3 using UDP.

SPECsfs is used for our evaluation because it reflects realistic workloads. It tries to recreate a typical workload based on characterization of real traces by deriving its operation mix from much observation of production systems [34]. At the same time, using SPECsfs workload is challenging. Its operation phase consists of file creation and transactions, each of which has different characteristics; the file creation phase is mostly write operations while the transaction phase involves random reads and writes. Thus, user-centric migration must adapt to these abrupt access pattern changes and load changes over time.

3.2. Experimental Setup

Figure 2 shows our experimental configuration. All of the machines in our experiments consist of a commodity PC system equipped with a 3GHz Pentium 4 processor, 1GB of main memory. Three disks are connected to each NFS server: one locally (directly) attached disk (a 10,000 RPM Seagate SCSI disk through Adaptec SCSI card 29160) and two network attached disks (same kind of disks connected through iSCSI³ protocol). We use two clients (load generators) for each NFS server.

The operating system was Red Hat Linux 9 with a 2.4.30 kernel and the exported file system (of each NFS server) was an Ext2 file system. To see the impact of network latency, we used *netem* [36] which provides network emulation functionality for testing protocols by emulating the properties of wide area networks. Netem is included, by default, in Linux kernel 2.6 distribution. Three subsystems are connected through a router with network emulator (*netem*) in 100Mb/s LAN. We used three different network latencies as shown in Table 1 (9ms RTT was used as a default, unless specifically noted). We employed $\delta = 2 * \text{standard deviation of device response times} + 2 * \text{standard deviation of RTT}$ to control the onset of migration. Inherent variance of netem added reality to the experiments in addition to access pattern changes of SPECsfs workload.

3.3. Single-User Environment

For single-user environment, we used NFS Server A with two load generators. NFS Server A employed Disk A (*local*) and Disk B (*remote*) each of which had a 16GB partition.⁴ It exports a 32GB file system which spans Disk A and Disk B to its load generators. First, we evaluate the impact of the design parameters in user-centric migration and then consider more complex scenarios.

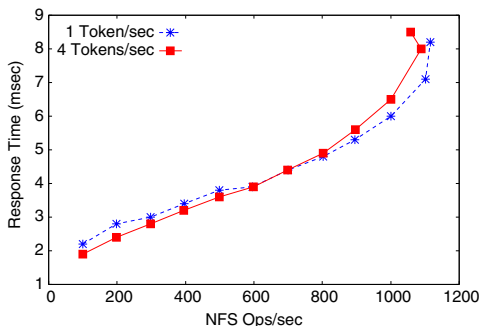


Figure 3. Impact of migration rate. It shows the sensitivity to the number of migration tokens.

³iSCSI stands for Internet SCSI, which enables to carry SCSI commands over IP networks.

⁴For intuitive explanation, we call Disk A as local disk and Disk B as remote disk in single-user environment.

3.3.1 Migration Rate

For this experiment, we used striping with the user-centric migration scheme (STR-MIG). Data is allocated by striping over local disk (Disk A) and remote disk (Disk B). If the request response time of remote disk is larger than that of local disk by δ , then data migration from remote disk to local disk is initiated and vice versa. We used 9ms of RTT. Figure 3 shows the results with different number of migration tokens.⁵ The base response time in the case of four tokens was better than that of one token case by 13.6% because a system with four tokens could reduce the number of remote disk accesses more due to a faster migration rate. However, faster migration caused earlier throughput saturation due to higher loads. In the system with one token, data migration occurred in one direction, from remote disk to local disk as the load is varied from 100 operations/sec to the saturation point. In the system with four tokens, the data was migrated from remote disk to local disk at lower NFS throughputs and from local disk to remote disk at higher NFS throughputs. The faster migration at lower throughputs to local disk resulted in higher loads at the local device at higher throughputs, which in turn, resulted in migrating data to remote disk.

Setting number of migration tokens dynamically based on workloads and network latencies seems feasible and is a subject of future study. In all the following experiments, we used the migration rate of four tokens per second.

3.3.2 Striping

Figure 4(a) shows the results of two configurations; normal striping (STR-NOMIG) and striping with user-centric migration (STR-MIG). STR-MIG showed better base response time by 24% and at higher loads, it improved performance from 6.7% to 20% when compared to STR-NOMIG. In addition, it could increase the saturation point by 9.6% over that of STR-NOMIG. This performance improvement is attributed to the fact that striping suffers from larger remote disk access latency, while user-centric migration could reduce the number of remote disk accesses by migrating data gradually from remote disk to local disk. In normal striping (STR-NOMIG), the remote response times limit the realized throughput. User-centric migration could increase the throughput saturation point by migrating data from remote disk.

Figure 4(b) shows the impact of network latency. As we expected, migration showed larger performance improvement as network latency got larger. STR-MIG with 17ms RTT improved the base response time by 24.3% and im-

⁵Since the SPECsfs benchmark reports the response time vs. delivered throughput, as opposed to offered load, attempts to exceed the saturation point can result in fewer operations per second than attempted. In Figure 3, the actual x value of data point indicates delivered throughput.

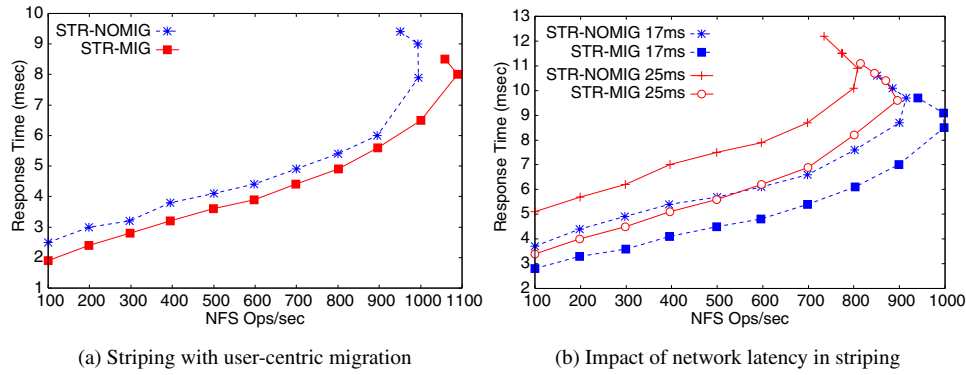


Figure 4. Single-user striping. Figure 4(a), 4(b) compares normal striping (STR-NOMIG) to striping with user-centric migration (STR-MIG).

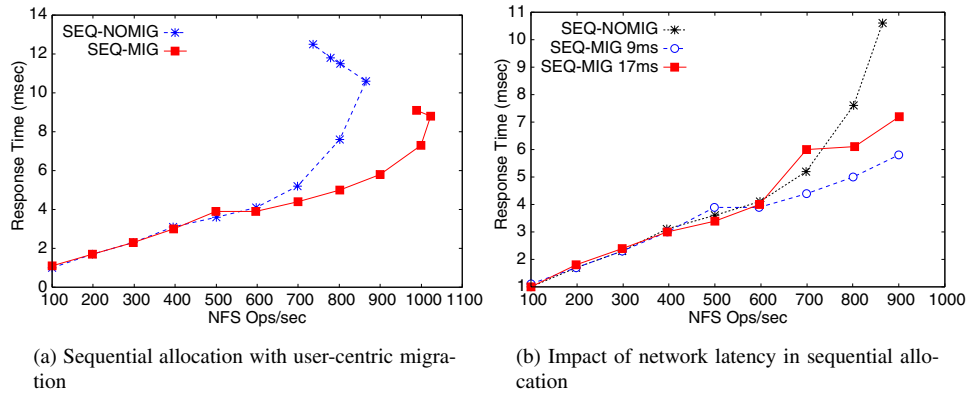


Figure 5. Single-user sequential allocation. Figure 5(a), 5(b) compares normal sequential allocation (SEQ-NOMIG) to sequential allocation with user-centric migration (SEQ-MIG).

proved the response time for higher loads from 8.5% to 26.5%. It increased the throughput saturation point by 9.1%. Similarly, STR-MIG with 25ms RTT improved the base response time by 33.3% and improved response times at higher loads from 7.0% to 29.8%. The throughput saturation point was increased by 10.8%. High network latency directly impacted performance of the normal striping system, while STR-MIG was less sensitive to network latency increase because data migration can effectively reduce the impact of higher network latency (by improving locality).

3.3.3 Sequential Allocation

Figure 5(a) shows the results of sequential allocation (SEQ-NOMIG) and sequential allocation with user-centric migration (SEQ-MIG). When local disk is not heavily loaded (until load 400), there is no difference between the two schemes. At load 600, SEQ-NOMIG started suffering from heavy local disk load, while SEQ-MIG migrated active data

to remote disk to improve load balance.⁶ This caused performance improvement in response time up to 45.3% at higher loads. At the same time, it increased the throughput saturation point by 18.2%. In contrast to normal striping, in sequential allocation, the bottleneck of the system was local disk. Larger saturation point of user-centric migration is attributed to the fact that data migration to remote disk could reduce the load of local disk.

Figure 5(b) shows the impact of network latency. In SEQ-MIG with 17ms RTT, number of migrations were fewer than in SEQ-MIG with 9ms RTT. This resulted in worse response time than that of system with 9ms RTT at higher loads. In 25ms RTT, migration didn't happen; user-centric migration decided not to migrate data because of remote disk's higher network latency. The data set was small enough to fit on the local disk and hence the network latency has no impact on the performance on the SEQ-NOMIG system.

⁶When migration was initiated at load 500, it incurred a little overhead on response time.

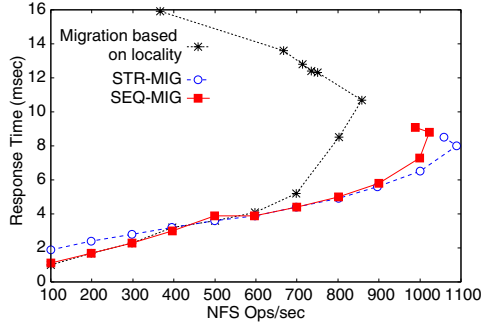


Figure 6. The comparison between different migration schemes.

3.3.4 Discussion of Results

From Figure 4(a) and Figure 5(a), we can observe that systems employing migration (STR-MIG and SEQ-MIG) exhibited better characteristics than systems without migration, with striping or sequential allocation. STR-MIG offered better response times than STR-NOMIG at most NFS throughputs. SEQ-MIG offered similar response times as SEQ-NOMIG at lower loads while improving the throughput saturation points and response times at higher loads. This adaptability comes from the flexibility of user-centric migration policy; the direction of migration changes based on changes in loads at different disks.

Figure 6 shows the performance comparison between two migration schemes; migration based on locality and user-centric migration (STR-MIG and SEQ-MIG). In migration based on the locality scheme, every accessed data on remote disk is migrated to local device while *inactive* data is migrated to remote device. The performance of this scheme became worse as load increased (due to the heavily loaded local disk). When the working set size exceeded the local device size, it could only deliver very low throughput due to thrashing. User-centric migration is observed to perform better than the locality based migration scheme and achieved higher throughput saturation points.

3.4. Multi-User Environment

For multi-user environment with striping, we used three NFS Servers A, B and C with two load generators each. NFS server A exports a 32GB file system which spans Disk A, Disk B and Disk C to its load generators. NFS server B exports a 32GB file system which spans Disk B, Disk C and Disk A to its load generators. NFS server C is similar. Thus, NFS server A, B and C share storage resources (Disk A, Disk B and Disk C), and data of three servers may be intermixed on the three disks. For multi-user environment with sequential allocation, we used two NFS Servers A and

B with Disk A and Disk B.

3.4.1 Striping

In this experiment, we used two configurations. In the first configuration, three NFS servers used striping simultaneously (STR-NOMIG) and in the second one, three NFS servers used striping with user-centric migration concurrently (STR-MIG). We compare results between the two configurations here.

We ran the three servers at different loads. We operated NFS server A at varying loads from 120 to 1200 and simultaneously operated NFS server B at varying loads from 80 to 800 and NFS server C from 40 to 400. Thus, the load distribution of server A, B and C was 3:2:1. This allowed us to study the impact of different loads at the different servers.⁷ We used 9ms of RTT. Figure 7(a) and 7(b) show the result of each NFS server’s performance and average performance of all three servers. In Figure 7(a), using user-centric migration, performance improvement in NFS server A was from 7.3% to 20.1% and performance improvement in NFS server B and in NFS server C were similar to server A. We could observe that each server tried to improve locality; each server migrated its remote data to its local disk. As a result, each server could improve response times significantly compared to normal striping case. Since the data was originally striped across all the disks, the loads at all the servers together impact the response times at the disks. Figure 7(b) compares the average performance of two configurations. Our system (STR-MIG) improved response time from 6.6% to 20.4% and increased the saturation point by 5.4% compared to STR-NOMIG.

Figure 8(a) shows the total number of requests served at Disk A, Disk B and Disk C as a function of time from the viewpoint of NFS server A. We can observe that migration from Disk B and Disk C to Disk A was continuously done during the benchmark to reduce the number of remote disk accesses (denoted as a Disk B request and Disk C request in Figure 8(a)). Without migration, nearly equal number of requests would have been observed at the three disks.

This migration helped the system to improve locality, but it may result in lower balance of load. Figure 8(b) presents overall request distribution among the three disks during the experiment. Each group consists of three bars: the left most bar denotes number of requests issued on Disk A and the next two bars represent the number of requests issued on Disk B and Disk C from the three NFS servers. In Figure 8(b), we can observe the following. First, the fraction of the server A requests served at disk A gradually increased indicating that server A increased its locality during the experiment. Similarly Server B increased its locality at Disk B.

⁷We also conducted experiments when each server has the same load. Similar results were observed and hence not reported here.

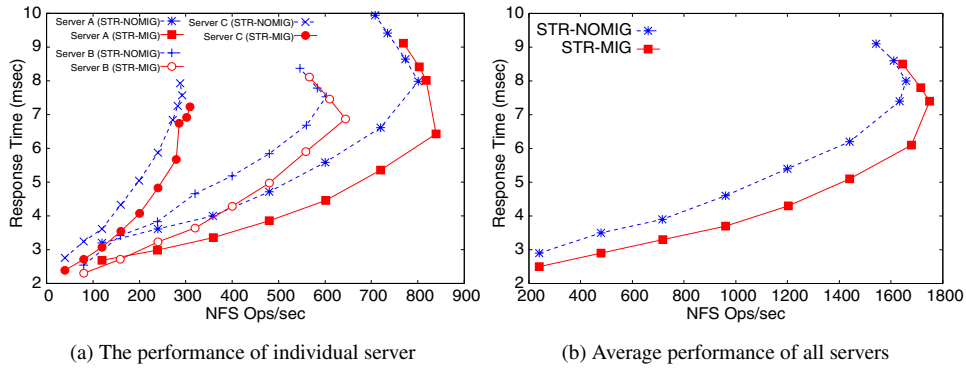


Figure 7. Multi-user striping with user-centric migration. Figure 7(a) shows each server’s result when each server uses striping without migration (STR-NOMIG) and when each server uses striping with user-centric migration (STR-MIG). Figure 7(b) shows the average performance of three servers for two configurations.

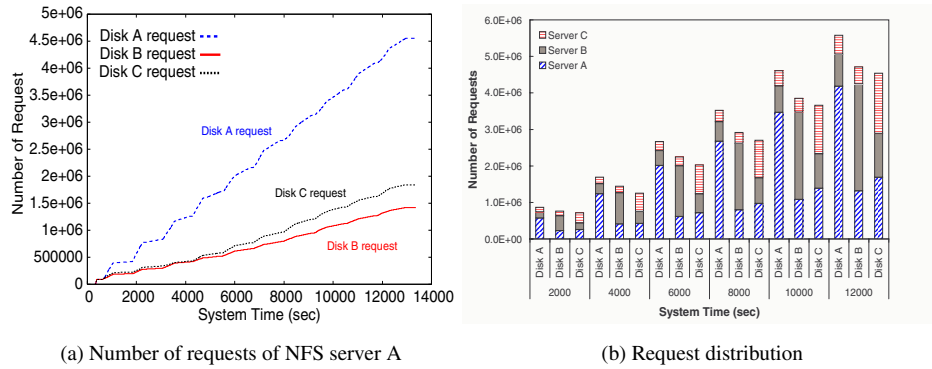


Figure 8. Request statistics. Figure 8(a) shows request statistics from the viewpoint of NFS server A during the experiment. In the viewpoint of NFS server A, Disk A is local disk and Disk B and Disk C are remote disks. Figure 8(b) shows distribution of requests issued on Disk A, Disk B and Disk C from the three NFS servers.

Disk C showed relatively balanced request distribution because server C was the most lightly loaded server. Second, in a normal striping system, the number of requests issued on Disk A, Disk B and Disk C would be the same (in most cases). It is observed that load balance was decreased by up to 12.8% in user-centric migration. This was a consequence of the improved locality and the imbalanced load across the three servers. It is also noted that the load imbalance at the disks is not at the same level as the load imbalance at the servers (3:2:1).

3.4.2 Sequential Allocation

Similar to the previous experiment, we configured two NFS servers to use sequential allocation without migration (SEQ-NOMIG) and with migration (SEQ-MIG). Server A allocates its data first on Disk A and then on Disk B in a concatenated fashion and similarly, server B allocates its data first on Disk B and then on Disk A, i.e., locality is maximized for each server’s data. We tested loads of 100 to 1100 at server A and tested loads of 25 to 400 at server

B. Server B’s load was kept at a quarter of server A’s load in this experiment. Different loads at the servers result in different loads at the devices when data migration is not employed. Figure 9(a) and 9(b) show the result of each server. At load 500, server A started data migration to Disk B due to the heavy load of Disk A. User-centric migration improved the performance of the heavily loaded server A. It improved throughput saturation point by 11.7% at server A. Data migration of server A impacted the performance of server B. However, response time improvement in server A (e.g., from 10.1ms to 6.5ms) was more significant than the response time increase in server B (e.g., from 2.2ms to 3.6ms).

This experiment also highlights the local good versus global good tension that is addressed by our approach. While server A’s performance improved in this experiment at the cost of server B’s performance, overall (global) average response time also improved at the same time (since there are more requests at server A). Over time, the response times of the two devices will be within δ (migration threshold) and migration will stop.

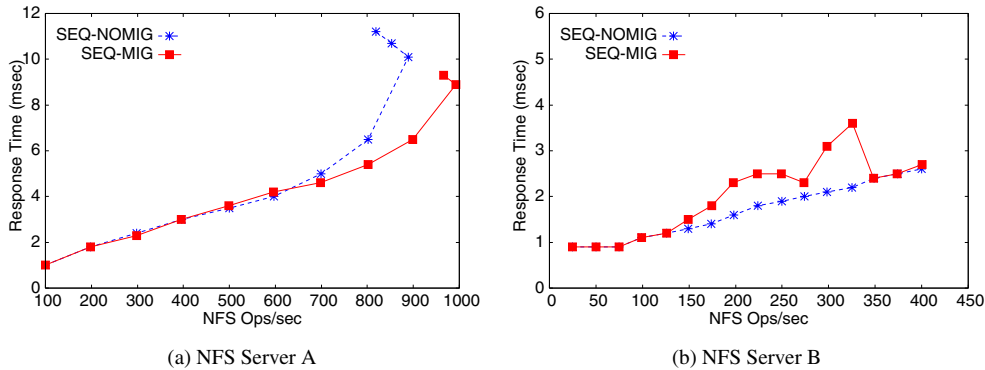


Figure 9. Multi-user sequential allocation with user-centric migration. In SEQ-NOMIG, each NFS server uses sequential allocation without migration and in SEQ-MIG, each server uses sequential allocation with user-centric migration.

3.4.3 Discussion of Results

When multiple users share storage resources by striping and those devices have performance differences (due to disk hardware or network latencies), user-centric migration could improve performance by improving locality. In multi-user workloads, the resulting improvement in locality did not decrease load balance much. If multiple users use sequential allocation and there are load differences, user-centric migration allows a heavily loaded server to reduce its load by migrating its data to a server with less load.

4. Related Work & Discussion

Data (file) migration has been studied extensively earlier in systems where disks and tapes are used [29, 7, 30, 11, 12]. As mentioned earlier, we are considering data migration among disks. Characteristics of data migration can be much different in two cases. First, the ratio of access time of local disk and remote disk is much smaller than that of disk and tape, the decision of data migration among disks can be different from disk and tape case. Second, the performance of disk is always better than that of tape, heavily referenced data always migrate from tape to disk. In contrast, in our case, heavily referenced data may migrate in different directions depending on workloads, making this problem more challenging.

Data migration has also been studied in HP’s AutoRAID system where data can migrate between a mirrored device and a RAID device [2]. In these systems, hot⁸ data is migrated to faster devices and cold data is migrated to slower devices to improve the access times of hot data by keeping it local to faster devices. When data sets are larger than the capacity of faster devices in such systems, thrashing may oc-

⁸Data that is currently being accessed or active is called “hot” data and data that is not currently being accessed or inactive is called “cold” data.

cur. Some of the systems detect thrashing and may preclude migration during such times [2]. Our data migration scheme is more general and considers migration among multiple disks over a network with dynamically changing characteristics. Data migration direction is decided by workload changes - a faster device at some time can become a slower device later and vice versa.

Aqueduct [19] deals with data migration among storage devices without application down-time. It takes a control-theoretical approach to provide QoS guarantees to client applications during a data migration, while still accomplishing the data migration in as short a time as possible. Similar to our system, Aqueduct uses I/O request response time as a performance measure, but there are significant differences from our work. First, Aqueduct is only focused on how to migrate data without significantly impacting foreground activities. Other migration decision issues mentioned in the earlier section, i.e., migration initiation, destination, data set to migrate were statically decided (e.g., by system administrator). Only migration speed is determined dynamically according to the impact on foreground performance. In contrast, in our system, all of the migration issues are decided dynamically based on system characteristics such as workload, disk access locality and disk load. Second, we focus on migrating active data (that is currently being accessed), which results in reduced migration cost.

The problems of data distribution in large-scale storage systems are addressed in [20, 21, 22]. This body of work is focused on data migration as a result of configuration changes such as insertions or removals of disks. Handling heterogeneity in shared-disk file systems is addressed in [23], where observed request latencies were used for adaptive file set migration among servers to balance the meta-data workload on servers. Our work is different from this work in two points: (a) Our system is dealing with migrating block-level data in contrast to file-level data (e.g., file

sets) and (b) we consider a multi-user environment where several applications compete for shared resources.

Request distribution, load balancing and memory cache hits are considered in LARD [10] for improving the performance of clustered file servers. D-SPTF [13] additionally considered disk head scheduling when redirecting requests in brick-based storage systems where multiple copies of a data item may be present. In this paper, we are considering the case where there is single copy of a data item, i.e., no replication.

A number of studies have pointed to the need for improving the locality of accesses when devices are networked [24, 25, 26]. These studies have pointed to the importance of reducing the impact of network access latencies on applications' performance. File systems and other applications already employ caching to exploit the locality characteristics of data accesses. As mentioned earlier, we consider *longer-term locality* considerations at the I/O or storage system beyond what is already exploited in memory.

In this paper, we address data distribution among networked disks. Caching or migration can be employed to improve performance in this case. Caching normally relates to bringing data items closer to the requesting user to improve locality. Migration, considered in this paper, allows data to be migrated both close to or away from the user depending on the load and performance dynamics. That is, while caching can be used to improve locality, migration (as considered in this paper) allows both improvement of locality and load balance with one mechanism.

Consider the case when a new and a faster disk is added close to a user. In this case, we can employ caching or migration to improve data distribution. In the case of caching, we use the faster disk as a cache to slower disks. However, it may result in wastage of resources in disk space and network bandwidth (used when data is written back). Alternately, the entire data can be copied over from the slower device to the faster device. However this does not guarantee that the new device won't become a bottleneck as the older disk sees lower load. We employed user-centric migration to consider both load balance and locality and to reduce possible resource waste.

File-level and user-centric (block-level) migration have different characteristics. In some cases, block-level migration can be more efficient than file-level migration because (a) it can migrate only part of the file as needed and (b) it can increase the available parallelism by using all available devices. In other cases, as in Information Lifecycle Management (ILM) [42], file-level migration would be a better fit because we usually apply policies per file or per fileset. Since our system migrates frequently accessed data to better performing disk and only a part of a file can be accessed often, we employed block-level migration. However, we believe our design rationale of monitoring disk request re-

sponse time and deciding various data migration issues can be applied to file-level migration as well.

In our approach, data is migrated to other locations only when there is a benefit in terms of performance. Ideally, when all the devices have nearly the same response times, data is not migrated from one device to another. To avoid oscillations and to take costs of migration into account, we set thresholds (the δ parameter) to dampen migration around this ideal operating point. Our system adapts to workload and data distribution changes to reach this operating point.

As an alternative approach to ours, we could think of a policy which would implicitly create a storage hierarchy, where heavily referenced data is migrated in the local storage, and less referenced data is migrated in the more distant networked storage. This idea is tested against our system and the results are presented in Figure 6. The primary difference from hierarchical memory systems is that the loads at the devices can exceed disk throughput capacities and queuing delays can build up. Hence, a local disk can be slower if all the needed data is on the local disk (and hence a higher load) than a remote disk (because of lower load).

Disks have been traditionally attached to hosts or servers. Recently, they are being attached to networks, either Storage Area Networks or IP networks. Most of current storage systems do static allocation of storage at the time of file system allocation and do not provide the flexibility that is being pursued here.

When block network storage protocol devices (like iSCSI) are used, data cannot be simultaneously shared by multiple file systems without additional locking mechanisms. In the work here, we assume a single file system or user is accessing data at a time. In such scenarios, migration will provide similar benefits as caching [38]. Andrew File System [37] employed disk caching to reduce the load on the file server and to exploit the disk access locality at each client.

In this paper, we considered two allocation policies; striping and sequential allocation. Random allocation has been proposed by others [3, 18] to achieve load balancing with heterogeneous devices and for minimizing disruptions as devices are added to the system. The remapping required for migration would have to be suitably modified to work with random allocation. We plan to pursue this in the future.

We also plan to evaluate our migration policy in more diverse workloads and test it in a wide area testbed such as Planetlab [41].

5. Conclusion

In this paper, we have proposed user-centric migration for balancing load and locality in a networked storage sys-

tem. Through realistic experiments on an experimental testbed, user-centric migration is shown to automatically and transparently balance disk access locality and load balance through migration of active data blocks. Migration is decided by longer-term performance metrics and it is shown to adapt to changes of workloads and loads in each storage device in a networked environment. User-centric migration was shown to improve locality with an allocation policy of striping and to improve load balance with an allocation policy of sequential allocation. We have shown that user-centric migration is effective in multi-user environments and can effectively share common storage resources under different user loads. We have also shown that the user-centric migration policy can outperform policies based on maximizing locality or load balance alone.

References

- [1] Storage Networks. Managed services for the enterprise. <http://www.storagenetworks.com>, 2002.
- [2] J. Wilkes, R. A. Golding, C. Staelin, T. Sullivan. The HP AutoRAID Hierarchical Storage System. *ACM Trans. Comput. Syst.* 14(1): 108-136 (1996).
- [3] IBM Almaden Research Center. Collective Intelligent Bricks (CIB). http://www.almaden.ibm.com/StorageSystems/autonomic_storage/CIB/index.shtml, 2001.
- [4] Terrascale Technologies, Inc. TerraGrid Cluster File System. http://www.terrascale.com/prod_over_e.html, 2005.
- [5] F. Schmuck and R. Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. In *Proc. of USENIX FAST* (2002).
- [6] IBM High Performance Computing. An Introduction to GPFS. http://www-03.ibm.com/systems/clusters/software/whitepapers/gpfs_intro.pdf, 2006.
- [7] B. Gavish and O. Sheng. Dynamic File Migration in Distributed Computer Systems. *Commun. ACM* 33(2): 177-189 (1990).
- [8] L. Yin, S. Uttamchandani and R. Katz. SmartMig: Risk-modulated Proactive Data Migration for Maximizing Storage System Utility. In *Proc. of IEEE MSST* (2006).
- [9] E. P. Markatos and T. J. LeBlanc. Load Balancing vs. Locality Management in Shared-Memory Multiprocessors. *Tech. Report: TR399, University of Rochester*, 1991.
- [10] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. M. Nahum. Locality-Aware Request Distribution in Cluster-based Network Servers. In *Proc. of ASPLOS* (1998).
- [11] E. Miller and R. Katz. An Analysis of File Migration in a UNIX Supercomputing Environment. In *Proc. of USENIX* (1993).
- [12] M. Lubeck, D. Geppert, K. Nienartowicz, M. Nowak and A. Valassi. An Overview of a Large-Scale Data Migration. In *Proc. of IEEE MSST* (2003).
- [13] C. R. Lumb, R. A. Golding and G. R. Ganger. D-SPTF: Decentralized Request Distribution in Brick-based Storage Systems. In *Proc. of ASPLOS* (2004).
- [14] R. M. English and A. A. Stepanov. Loge: A Self-Organizing Disk Controller. In *Proc. of USENIX Winter* (1992).
- [15] R. Wang, T. E. Anderson, and D. A. Patterson. Virtual Log-Based File Systems for a Programmable Disk. In *Proc. of OSDI* (1999).
- [16] J. Gray. What Happens When Processing, Storage Bandwidth are Free and Infinite? Keynote speech at IOPADS '97, 1997.
- [17] G. A. Gibson and et al. File Server Scaling with Network-Attached Secure Disks. In *Proc. of SIGMETRICS* (1997).
- [18] S. Ghemawat, H. Gobioff and S-T. Leung. The Google File system. In *Proc. of SOSP* (2003).
- [19] C. Lu, G. A. Alvarez and J. Wilkes. Aqueduct: online data migration with performance guarantees. In *Proc. of FAST* (2002).
- [20] A. Brinkmann, K. Salzwedel and C. Scheideler. Efficient, distributed data placement strategies for storage area networks (extended abstract). In *Proc. of SPAA* (2000).
- [21] Y. Saito, S. Frølund, A. Veitch, A. Merchant and S. Spence. FAB: building distributed enterprise disk arrays from commodity components. In *Proc. of ASPLOS* (2004).
- [22] R. Honicky and E. Miller. A Fast Algorithm for Online Placement and Reorganization of Replicated Data. In *Proc. of IPDPS* (2003).
- [23] C. Wu and R. Burns. Handling Heterogeneity in Shared-Disk File Systems. In *Proc. of SC* (2003).
- [24] W. T. NG and B. Hillyer. Obtaining High Performance for Storage Outsourcing. In *Proc. of USENIX FAST* (2002).
- [25] Y. Lu, F. Noman and D. H.C. Du. Simulation Study of iSCSI-based Storage System. In *Proc. of IEEE MSST* (2004).
- [26] P. Radkov and et al. A Performance Comparison of NFS and iSCSI for IP-connected Networked Storage. In *Proc. of USENIX FAST* (2004).
- [27] L. Qiu, R. Yang, Y. Zhang and S. Shenker. On Selfish Routing in Internet-like Environments. In *Proc. of ACM SIGCOMM* (2003).
- [28] V. Borkar and P. R. Kumar. Dynamic Cesaro-Wardrop Equilibration in Networks. *IEEE Trans. on Automatic Control* 48(3): 382-396 (2003).
- [29] A. J. Smith. Long Term File Migration: Development and Evaluation of Algorithms. *Communications of ACM* 24(8): 521-532 (1981).
- [30] V. Cate and T. Gross. Combining the Concepts of Compression and Caching for a Two-level File System. In *Proc. of ASPLOS* (1991).
- [31] S. Floyd and V. Jacobsen. Random Early Detection Gateways for Congestion Avoidance. *ACM/IEEE Trans. on Networking* 1(4): 397-413 (1993).
- [32] A. Agarwal, R. Simoni, J. Hennessy and M. Horowitz. An Evaluation of Directory Schemes for Cache Coherence. In *Proc. of ISCA* (1988).
- [33] Smitha, A. L. N. Reddy. LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers. In *Proc. of Globecom* (2001).
- [34] Standard Performance Evaluation Corporation. SPEC SFS97.R1 V3.0 <http://www.spec.org/osg/sfs97r1>.
- [35] R. P. Martin and D. E. Culler. NFS Sensitivity to High Performance Networks. In *Proc. of SIGMETRICS* (1999).
- [36] Linux Network Developers. Netem. <http://linux-net.osdl.org/index.php/Netem>.
- [37] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, and R. N. Sidebotham. Scale and Performance in a Distributed File System. *ACM Trans. Comput. Syst.* 6(1): 51-81 (1988).
- [38] Y. Hu and Q. Yang. DCD — Disk Caching Disk: A New Approach for Boosting I/O Performance. In *Proc. of ISCA* (1996).
- [39] D. He, and D. H.C. Du. An Efficient Data Sharing Scheme for iSCSI-Based File Systems. In *Proc. of IEEE MSST* (2004).
- [40] L. Peterson and B. Davies. Computer Networks: A Systems Approach. Morgan Kaufman Publishers (2000).
- [41] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3-12 (2003).
- [42] IBM. Information Lifecycle Management Solutions. <http://www-03.ibm.com/systems/storage/solutions/ilm/index.html>