UStore: A Low Cost Cold and Archival Data Storage System for Data Centers

Quanlu Zhang
Peking University
zql@net.pku.edu.cn

Yafei Dai
Peking University
dyf@pku.edu.cn

Fengqian Li
Shanghai Jiao Tong University
lfq0505@gmail.com

Lintao Zhang
Microsoft Research Asia
lintaoz@microsoft.com

***Abstract*—Recent trend in cloud computing demands vast and ever increasing storage capacity for data centers. For many cloud service providers, much of the storage capacity demand is driven by cold and archival data, such as user uploaded contents, system logs, and backups. In this paper, we describe UStore, a hard disk based storage system designed for such workloads. We make the assumption that most data centers are already populated with computer servers and networking gears, and propose a solution to attach additional disks to these servers reliably at extremely low cost. The main component of UStore is a novel fat tree interconnect fabric to connect hard disks to existing servers and network infrastructure. To reduce cost, UStore leverages the mature commodity USB 3.0 technology to build the fabric, which has extremely low amortized cost per disk while still providing sufficient throughput to satisfy cold and archival workload. The software of the UStore system abstracts the system's physical topology and provides a consistent view of the storage capacity to the upper layer services such as distributed file systems or backup services. In a sense, UStore can be regarded as external USB hard disks designed for data centers.**

I. Introduction

Cloud computing provides many cost saving opportunities for the end users, while opening new usage scenarios that were impossible before. Cloud services, such as web email, video hosting, cloud drive [1], and cloud archiving [2], provide large amount of storage space for user contents at very low cost, often free at the lowest service tier. To operate these cloud services, huge amount of storage capacity needs to be available in the data centers. Moreover, the cloud operators need to be able to keep up with the storage demand as business expands. Due to the scale of operation, infrastructure cost is often a significant part of the costs for cloud service providers [3]. Cost savings on storage capacity is a significant competitive advantage in this space.

Much of the data stored in the cloud are *cold* or *archival*. We distinguish cold data from archival data based on their access patterns. Cold data are for interactive usage scenarios. They are accessed rarely, but when accessed, a user would expect the response to come back after a short amount of time, usually in the range of seconds. Examples of cold data include older emails and shared photos of non-celebrity users. In contrast, archival data are usually accessed in large batches on a predictable schedule. Examples of archival data include file system backups and system logs. For archival data, the access latency requirement is usually less stringent, but they often demand high throughput.

Due to latency requirements, cold data usually has to be stored on spinning hard disks. As disk cost comes down, we are seeing many archival systems built on top of hard disks as well [4], [5], [6]. In this paper, we assume that the storage system is built with hard disks. Though other storage mediums, such as magnetic tapes, optical disks, have been used for archiving purposes, hard disk is still likely to be the preferred medium for cold and archival storage in data centers in the near future.

To address the demand for storage capacity in a data center setting, we describe a design of a storage system called UStore. We make the observation that most data centers are already populated with computer servers and networking gears. To add storage capacity, a cost effective solution would be to attach large amount of hard disks to existing servers. By separating storage from computation and networking demand, UStore provides more flexibility for cloud service providers. Of course, UStore can be co-deployed with additional servers, while still maintaining its cost advantage.

UStore is a combined hardware and software solution to cost effectively attach disks to servers with good performance and availability. A UStore system consists of one or more *deploy units*. A deploy unit is a hardware piece that contains many disks to be connected to multiple servers called UStore *hosts*. To connect the disks to servers, we leverage the widely adopted USB technology, whose recent iteration (USB 3.0) provides up to 5Gb/s full duplex throughput, competitive to other connection technologies at a much lower price point. We build a fat tree interconnect fabric with USB hubs and USB switches (*i.e.,* 2-1 multiplexer) to avoid single point of failure and provide higher throughput. Through reconfiguration, the fabric allows any of the disks to be connected to any hosts of the unit, thus providing fault tolerance. The traditional wisdom of multi-path attached storage being expensive is no longer true in our design. UStore builds the interconnect fabric at a much lower cost than traditional storage architectures. Based on the novel hardware platform, we design a software stack that provides storage capacity to upper layer applications in a flexible manner. The software hides the complexity of hardware failure detection and quick failover. It manages the huge storage pool (see § IV), and allows clients to access the allocated storage space as remotely attached disks.

The basic problem we want to solve with the UStore system is to find a method to connect hard disks to servers in a cost

effective manner and offer flexible access interface, while still providing high availability and good performance. We do not intend to build a fully functional storage system, instead, we set up a well-managed storage platform for upper layer usage. More specifically, traditional storage systems can be deployed above UStore with little modification, using UStore storage as raw disks, while employing their own specific techniques for data durability and availability.

To evaluate the UStore architecture, we built a proof-of-concept UStore prototype with a single deploy unit consisting of 16 hard disks. We evaluated the system with 4 hosts connected with the unit. Our evaluation shows that the system can sustain a total throughput of 2160MB/s and can recover from an arbitrary single host failure in 5.8 seconds.

The key contributions of this paper are:

1) We observe several desirable properties of a cloud based archival and cold data storage system, and argue that a cost effective way to add storage to data centers is to attach more disks to existing servers.
2) We describe the hardware design of UStore deploy unit, which employs a novel USB 3.0 based fat tree interconnect fabric to connect large numbers of hard disks to multiple existing hosts with no single point of failure. The hardware design offers extremely low amortized cost for each additional disk, while still providing reasonable performance.
3) We describe the UStore software system architecture, which leverages the UStore deploy unit hardware to provide a scalable and fault tolerant storage.
4) We built a prototype of UStore with 16 disks, and provide detailed measurement on performance and power consumption under different workloads.

II. Background

A. Storage Interconnect Technologies

We briefly review some popular interconnection technologies below.

SATA/SAS/FC: SATA (Serial ATA) [7] supports a native transfer rate of up to 6.0 Gb/s in SATA 3.0, and SAS(Serial Attached SCSI) [8] also allows transfer speed of 6 Gb/s. SATA multiplier and SAS expander can be used to deal with limited native ports on servers. However, SATA multiplier only supports up to 15 devices and does not support cascaded connections (*i.e.,* one multiplier plugged into another). SAS is often used in high-end storage systems, which is usually costly. On the other hand, SAS supports Dual-path and dual-domain architectures to avoid single point of failure, which trades off much more monetary cost for reliability and fast recovery. Another interconnection technology is fibre channel (FC) [9] which can reach 3200 MB/s (*i.e.,* 16 GFC). However, fibre channel is obviously not cost effective for archival and cold storage.

Ethernet: As a widely adopted and low-cost technology for connecting computers, Ethernet is also used for interconnecting disks [6]. The capital cost of building an Ethernet fabric is relatively low. However, in order to make a disk accessible on the network, a microprocessor (*e.g.,* low-power ARM) has to be attached to the disk, which inevitably increases capital expense (see § VI for detailed comparison). Moreover, low performance microprocessor may also be a bottleneck to support high network throughput and mechanisms such as erasure coding and encryption.

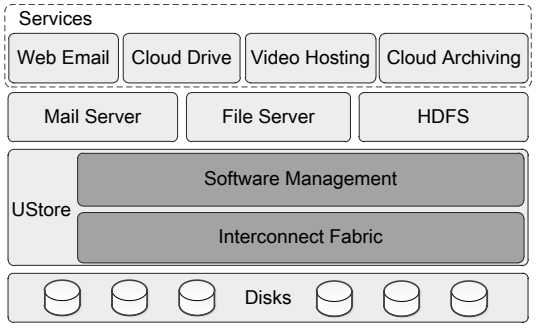

Fig. 1: Holistic system stack overview.

B. USB 3.0 Technology

USB is arguably the most widely adopted technology to connect peripheral devices to computer hosts. USB interconnects form a tiered tree structure, with hubs forming the internal nodes of the tree. Up to 5 levels of tiers are allowed in a USB tree. Each USB tree can contain a maximum of 127 devices including hubs (see § V-B for some discussions).

USB 3.0 [10] was introduced in 2008 as the second major revision of the USB standard. USB 3.0 specifies the SuperSpeed transfer mode, which supports 5.0Gb/s duplex transfer with 8b/10b encoding. Realistically, one can expect to achieve 300∼400MB/s transfer rate on a single USB 3.0 port. Very recently, a revision called USB 3.1 was announced, which provides a 10Gb/s transfer mode called SuperSpeed+. Though it will take some time before devices supporting 10 Gb/s appear on the market. There is no explicit length limit for USB 3.0 cable in the specification, but the electrical properties of the cable may limit the practical length to around 3 meters. This is sufficient to connect disks and servers in the same rack or even adjacent racks. Moreover, this length can be extended through the use of hubs or signal repeaters.

Since low cost hard disks usually come with native SATA interface, a bridge chip is required to connect disks to USB hosts. SATA to USB 3.0 bridges are widely used in commodity external USB hard drive enclosures. It is extremely cost effective since USB 3.0 technology is shipped in extremely high volumes with numerous vendors competing in the market. Moreover, USB 3.0 protocol is universally supported by all of the major operating systems in data centers, and almost all new chipsets and motherboards come with native USB 3.0 support.

III. Interconnect Fabric

We first give a high level overview from upper layer services down to the basic infrastructure, as shown in Figure 1. UStore

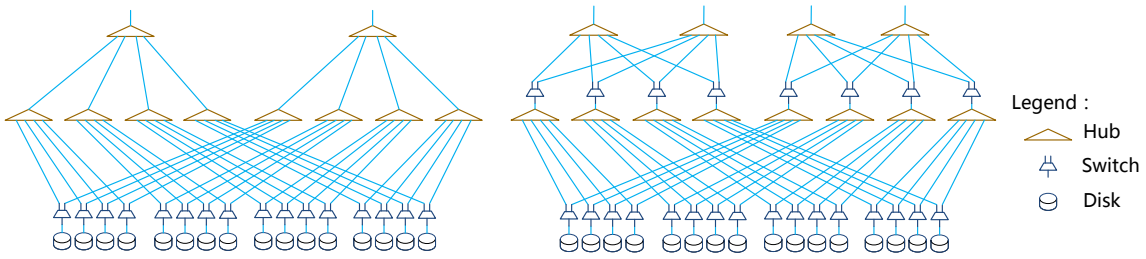

Fig. 2: Interconnect fabric. The left design uses hubs and switches to form two full trees to connect disks to two hosts, while the right design places switches at a higher level of the tree to allow better throughput.

manages large numbers of disks and supports the storage demand of various upper layer services. We elaborate the interconnect fabric in this section, and then demonstrate the software management in § IV.

As the core of the deploy unit, the interconnect fabric connects large numbers of disks in a UStore deploy unit to multiple computer servers called *hosts* of the unit. It is constructed with two primitives: *USB hubs* and *USB switches*. The hub is an aggregation device. It accumulates multiple downstream flows into a single upstream flow. We assume each hub can have at most k downstream flows, where k is called the fan-in factor of the hub. The switch is a multiplexer that multiplexes one downstream between two upstream flows. A control signal determines which of the two upstreams to connect with.

A. The Data Plane

In the scenario of cold data storage, we would like to connect a large number of disks (*e.g.,* 60 to 100 disks) to servers. The most simplistic way to connect disks to computers is through point-to-point interconnection, which requires an extra port on the server for each additional disk. Though providing good performance, such scheme is not scalable for cold data storage and very costly.

A common way to connect multiple disks to a single port is to use hubs. In this case, the interconnection forms a tree with disks at the leaf and hubs on internal nodes. Though such a solution is used widely in the industry, it is not ideal. There are two major drawbacks. The first is performance. Multiple disks share the same connection bandwidth at the root of the tree, which makes the host port a bottleneck: just 4 modern hard disks can saturate a SAS/SATA/USB port [1]. The second problem is the single point of failure at the root. If the server at the root of the tree is down, due to software or hardware failures, the entire tree of disks will be unavailable. Though modern data center storage systems can tolerate device failures through software means, reconstructing large amount of data still puts a heavy toll on the network and other non-faulty servers. SAS protocol supports dual-path capability, but it is traditionally used in demanding enterprise settings and is priced accordingly. The design requires special storage devices (*e.g.,* dual port SAS disks), which are not cost effective for providing cheap cold storage in data center environment.

We propose to use both hubs and switches to build the interconnection topology. To tolerate a single server failure, two independent hub trees can be constructed as shown in the left part of Figure 2. Each disk at the leaf is connected to both hub trees through a switch. By controlling the switch, the disk can be connected to either host at the root of the trees. Therefore, when a server fails, by reconfiguring the interconnection with the switches, the disks can be reconnected to a non-faulty server.

With the topology at the left of Figure 2, a disk can be independently connected to one of the two trees. This design can tolerate not only failures of a single host, but also any single failure of the hubs. In this design, disks are independent, each occupies a leaf node. The workload of the disks are distributed among the roots. To increase the aggregated bandwidth, it is possible to construct more than 2 full hub trees and switch the disks multiple ways at the leaf. This also provides tolerance for more than one failures. However, this is overkill, since single failure tolerance is most likely sufficient for the scale of a deploy unit. For a multi-level hub tree, it is possible to switch at a higher level of the tree, closer to the root as shown in the right part of Figure 2. By doing so, we can reduce the number of hubs and switches required to implement the fabric, thus reducing the overall cost.

The topology discussed is essentially the well known fat tree [11] or Clos network [12] topology, which has seen adoption in data center networking recently [13], [14]. A formal analysis of the fault tolerance property of the topology is out of the scope of this paper. In our setting, any switch configuration is a valid partition of the fabric into multiple non-overlapping trees, which connect each leaf node to one of the root ports.

B. The Control Plane

In order to deal with device failures and possible load balance, we should make the interconnect fabric configurable. So the switches in the fabric must be controlled through a side channel signalling. In our implementation, we use a microcontroller connected through USB to one of the host machines to control the switches. The microcontroller receives commands from the Controller (elaborated in § IV) software

[1]SATA multiplier can only access one downstream disk at a time, which makes the situation even worse.

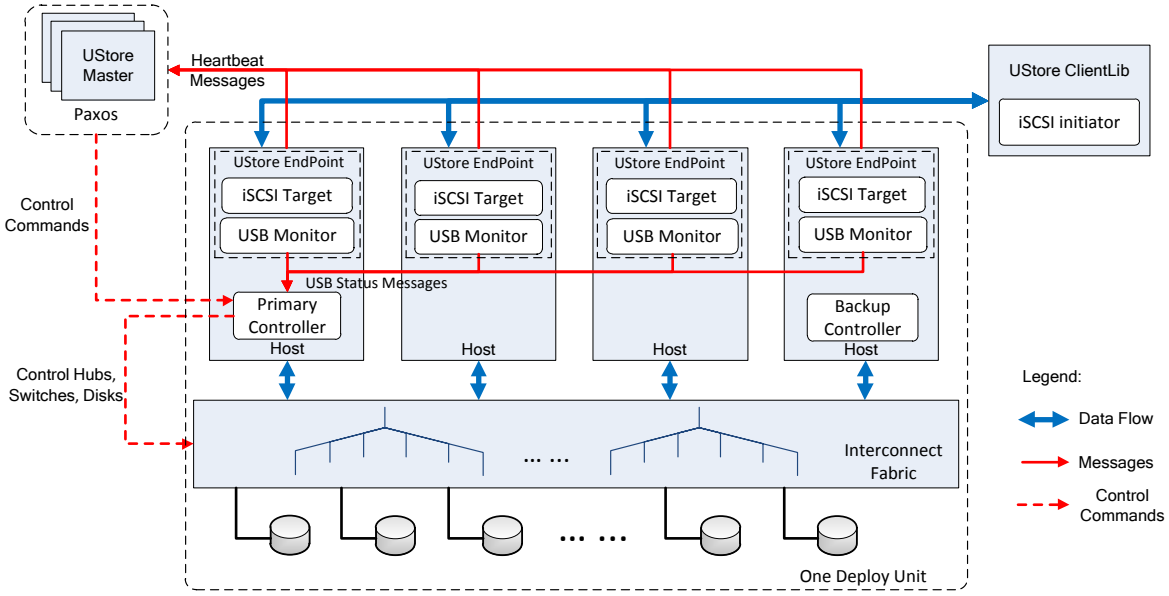

Fig. 3: The software architecture of UStore. The interconnect fabric uses the topology elaborated in § III, upper layer services interact with hosts directly for data flow after retrieving metadata from the Master.

in the connected host, and changes the signal to the switches accordingly.

Since the Controller runs on a host, if the host fails, we lose the ability to reconfigure the fabric. This would be unfortunate, because when the failure happens, we actually need to reconfigure the fabric, so that the disks originally connected to the failed host can be access through alternative paths. To avoid this failure scenario, we establish a secondary control path by adding a redundant copy of the microcontroller to a different host. The signals of the two microcontrollers are XOR-ed together to form the final controlling signal to the switches. During normal operation, only one of them is powered on. And when the control of this microcontroller is lost, we can switch to the other one, thus avoid the single point of failure in the control plane. When switch signals change, the configuration of the interconnect fabric is altered accordingly. And from a host's view, the USB devices are just inserted to or removed from the host.

Besides controlling the switches, it is also possible to control the power supplies to the individual disks and the hubs. Being able to control power supply enables us to perform rolling spin-up at the power-on time, thus avoiding a large number of disks spinning up at the same time and overwhelming the power supply. It is also possible to power down certain disks and part of the interconnection under predictable workloads to save energy.

IV. Software Architecture

Given that a large number of disks are attached to existing servers through the interconnect fabric elaborated above, the management of this huge storage pool is equally important. The software has to be carefully designed to meet three main objectives: (i) serving the storage allocation and access, (ii) monitoring failures and implementing quick failover, (iii) providing an appropriate interface for upper layer services and applications.

We design the software architecture of UStore, as illustrated in Figure 3. It consists of four components: *Master*, *EndPoint*, *Controller*, and *ClientLib*. A typical UStore deployment is composed of one Master and a number of deploy units, each of which is connected to multiple hosts that run the EndPoint and Controller. And the hosts are the servers that have already existed in data centers. We demonstrate the four components respectively, and then elaborate failure detection and power management.

A. UStore Master

The UStore system has a single Master that maintains the holistic view of the system to implement centralized controlling and scheduling. To make the Master fault tolerant, it is implemented as a replicated state machine using the Paxos consensus protocol [15], [16]. To control the whole system, the Master maintains three types of metadata: system configuration (SysConf), system status (SysStat), and storage allocation (StorAlloc).

SysConf includes basic system parameters, such as the number of deploy units in the system, the numbers of hosts and disks in each deploy unit, and the mappings from hosts to deploy units and from disks to deploy units.

SysStat is the real-time status of the system. It contains the status of the hosts and the disks (*i.e.,* online, spun down, or powered off), and the mapping from disks to hosts. The mapping reflects the current status of the interconnect fabric. SysStat is only kept in memory since the Master can always reconstruct it by interrogating the hosts.

StorAlloc is maintained to manage storage space (*e.g.,* allocating, reclaiming). We use a global namespace, *i.e.,*

$< /DeployUnitID/DiskID/SpaceID >$, to uniquely identify each allocated storage space. This metadata is stored persistently in the Master synchronously. We apply two rules for storage allocation. Firstly, a physical disk is preferred to be allocated to the same service, which facilitates power management since the service can control the disk without interfering with other services (see § IV-F). Secondly, a disk located near the client on the network is more likely to be allocated to this client, which improves locality and reduces networking overhead.

B. UStore EndPoint

An EndPoint runs in each host that is connected to a deploy unit. It has two main functions: monitoring the host's status, and exposing disk storage to clients. The EndPoint sends the healthiness and workload information of both the hosts and the disks back to the Master, through periodical heartbeat messages. USB Monitor monitors and sends the detailed status of the observed local USB tree (*e.g., lsusb -t* in Linux) to the Controller, which gives the Controller a integrated view of the interconnect fabric to implement failure detection and reconfiguration.

On the other hand, the EndPoint is responsible for exposing the disks onto the network, through a network storage protocol. It is potentially possible to use any SAN protocols to expose the raw disks or expose a networking file system interface. In our implementation, we choose iSCSI [17] as the protocol for exposing the storage (*i.e.,* iSCSI Target). When a client requests storage space, the Master allocates a piece of storage space (could be a disk, a disk partition or a big file in a disk) from UStore, and the corresponding host exposes it as a iSCSI target which can be mounted by the client.

C. UStore Controller

For each deploy unit, we have two Controllers running on two of the controlling hosts to avoid single point of failure. Though not strictly necessary, the Controllers usually run in a primary-backup manner. The Master sends commands to only one of the Controllers in normal operations. Only when the primary fails will the Master send commands to the backup Controller.

The Controller keeps track of the detailed interconnect fabric configuration of each deploy unit by collecting USB status from the EndPoints. The Master can change the configuration by sending explicit topology scheduling commands, such as "connect disk A to host H1 and disk C to host H2" to the Controller. After receiving the command, the Controller tries to execute it based on its knowledge of the current fabric. If the command cannot be accomplished, the Controller will report the error status back to the Master. For example, "connecting A to H1 will force disk E to be disconnected from host H3". In this case, the Master can either abort the command or issue another command that accommodates the conflicts.

We illustrate the execution of the command "connect disk A to host H1" in detail. The Controller takes the following three steps after receiving this command.

Algorithm 1 Lookup the switches to be turned to execute a command.

Parameters: Pairs of disk and host specified in the command.
Return Value: Switches to be turned or ErrInfo if conflict.

```
1:  function SwitchesToTurn(List disk_host_pairs)
2:      Set OccupiedSwitches
3:      Set SwitchesToTurn
4:      for all disk i ∈ DeployUnit
5:          if disk i ∉ disk_host_pairs
6:              host j ← GetAttachedHost(disk i)
7:              k switches ← GetSwitch(disk i,host j)
8:              add k switches to OccupiedSwitches
9:      for all disk_host pair ∈ disk_host_pairs
10:         k switches ← GetSwitch(disk_host pair)
11:         for all switch i ∈ k switches
12:             if switch i ∉ OccupiedSwitches
13:                 if desired_status ≠ current_status
14:                     add switch i to SwitchesToTurn
15:                     add switch i to OccupiedSwitches
16:             else if expected_status ≠ current_status
17:                 return ErrInfo
18:     return SwitchesToTurn
```

1. Lock the interconnect fabric to avoid another concurrent command causing inconsistency [16]. During the process of scheduling a command, no other commands will be accepted.

2. Determine which switch(es) should be turned to achieve the goal of this command. Firstly, the Controller traces from the disk up to the host, and finds out the k switches on the path from disk A to host H1 (*i.e.,* GetSwitch() in Algorithm 1). Some of them may already stay in the desired state, while others have to be turned. Before turning them, the Controller has to check the conflict with other disks, *i.e.,* comparing with the switch status of other disks which is maintained in the set $OccupiedSwitches$ in Algorithm 1. If there is no conflict or the conflict can be ignored, we collect the to-be-turned switches first, and then turn them one by one. Otherwise, the Controller will report error status with detailed information back to the Master. The detailed process is shown in Algorithm 1.

3. Send commands to the microcontroller to turn the switches collected above. The Controller then checks whether this scheduling is completed properly, through checking USB status reported by the involved EndPoints. If the expected connections cannot be detected in a pre-set time (*e.g.,* 30s), we roll back the command by turning the switches to original state, and report the situation back to the Master for further actions.

D. UStore ClientLib

The ClientLib abstracts away the details of the disk-host connection and exposes a consistent view of the storage capacity to the upper layer applications. Different users may want to provide different services based on UStore, we thus decide to provide the most basic storage interface, *i.e.,* the

block device interface in UStore. The interface is implemented through iSCSI protocol, so that the upper layer services can access UStore just like accessing local disks.

We design a simple UStore client library to facilitate building different access layers. The client library provides storage management APIs, such as applying for new storage space, mounting allocated storage. It provides simple directory lookup service to find a disk's host IP and provides notification call backs to notify the upper layer of disk status changes.

In the normal operation, the ClientLib keeps track of the locally mounted storage in UStore. If the storage becomes unaccessable due to the failover action of UStore, the ClientLib will retrieve the new host IP from the Master and remount the storage automatically. From the client's view, there is a temporary high latency accessing local disks, which usually does not have much impact. This is especially true for storage services that have their own redundancy mechanisms to deal with temporary unavailability (*e.g.,* HDFS).

E. Failure Detection

We design automatic failure detection in UStore, which can reduce failover time and make the system easy to operate. There are three main failure domains in a storage system: hosts, interconnect fabric and disks. In practice, hosts are more likely to fail due to software and network issues. According to [18], the MTTF of servers is 3.4 months while that of disks is 10-50 years, and physical interconnects have similar failure rate as disks [19]. We handle the host failures in a simple manner: if the Master does not hear the heartbeat message from a host for an extended period of time, the Master will treat the host as crashed and send command to the corresponding Controller to move the disks on this host to a non-faulty one.

For interconnect fabric, the Controller receives the detailed USB tree information from each host which obtains the USB tree from local operating system. So, it can detect the disappearance of hubs and disks by combining the non-overlapping USB trees. Since USB switches and bridges do not show up in the tree, we consider them to be part of the hubs or disks, viewing them as a single failure unit. For example in Figure 2, the switch with the hub connected to its downstream in the second layer is one failure unit. Similarly, in the last layer, each disk with its bridge and switch is one failure unit. If a device in the interconnect fabric fails, the Master switches away the paths going through this device, while reports the failure to system administrator for future replacement or repair.

UStore delegates data recovery of failed disks to the data redundancy mechanisms supported by upper layer services. It does not provide data redundancy by itself. On the other hand, the interconnect fabric of UStore can potentially facilitate disk data reconstruction. Since disks are not tightly coupled with servers, the involved disk can be switched to one or a small set of servers in order to reduce network load. We leave this for further work.

F. Power Management

Power consumption is one of the key metrics of modern storage systems. Spinning down or powering off disks is the most obvious way to reduce power consumption. However, the strategy to determine when to spin down disks greatly depends on the workloads. The upper layer services, which know the workload better, may have their own power saving mechanisms such as rearranged data placement [20] and powering off the disks that store redundant data [21]. We expose disk management interface (*e.g.,* spin up and down disks, change disk speed if supported) to the upper layer services by allowing them to change the state of the disks belonging to them. By default, UStore only provides a simple power saving mechanism: when a disk stays idle for a preconfigured period of time, it is spun down. But if it is detected that the disk is spun up and down too frequently, the host will increase the time interval. Furthermore, if the disks are spun down or powered off, the part of the interconnect fabric that connects these disks is powered off as well.

V. Physical Design and Prototype

In this section, we briefly discuss the physical design of a UStore deploy unit, and then present the prototype of UStore.

A. Physical Design

We envision that a deploy unit would be a rack-mountable enclosure. The physical volume, weight limitation, max power dissipation, cooling capability and vibration isolation of the enclosure dictate the number of disks that can be contained in a unit. In practice, a 4U rack unit can host around 40$\sim$70 3.5 inch hard disks comfortably [22], [23], [24], while leaving enough space for interconnect fabric, power supply and cooling. In this case, external connection to 4 hosts would be a reasonable configuration of the interconnect fabric. Such a unit would be able to provide around 200 terabytes of raw disk storage capacity using the available 4TB SATA disks, and has about 2$\sim$3 GB/s total aggregated throughput on all 4 ports.

USB connectors are designed for frequent plugging and unplugging. It might be desirable to use a more robust connector design in the data center settings. The maximum cable length of USB 3.0 is sufficient to connect disks and servers within the same rack. Moreover, it would greatly simplify the cabling if we integrate the interconnect fabric into a printed circuit board (PCB).

B. Prototype

We implemented a minimal proof-of-concept UStore prototype, as shown in Figure 4, to demonstrate the feasibility of the UStore design. The prototype only contains a single UStore deploy unit with 16 disks connected to 4 hosts, using the interconnect fabric shown in the right part of Figure 2.

The hard disks we use are 3TB 7200 RPM 3.5" TOSHIBA DT01ACA300. To connect them to USB 3.0 ports, we use commodity external USB 3.0 HDD enclosures (*i.e.,* SSK HE-G130) as the USB bridge. For the hub we use commodity 4-port hubs of UNITEK Y-3044. The switch is also a commodity

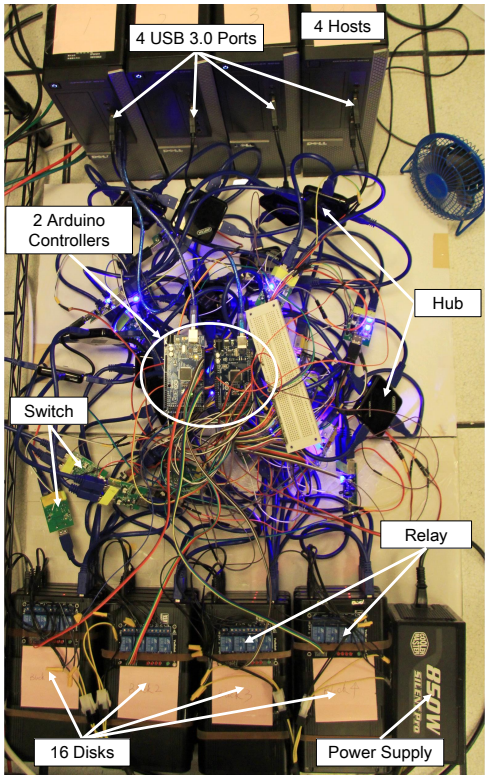

Fig. 4: UStore prototype.

USB 3.0 switch of SIIG JU-SW0012-S1. We modified the switch so that it can be controlled by electric signals instead of manual switches. The 12v power supply to each of the HDD enclosures pass through a relay, which can be turned on and off by the Controller. We implemented the control plane using commodity Arduino Mega 2560 boards to control the switches and the power relays.

USB 3.0 is still a relatively new technology. Therefore, some wrinkles still exist in current implementations. One major problem we found is that current Intel USB 3.0 root hub drive can only recognize less than 15 devices and we can only evaluate the throughput for up to 15 disks attached to a single server (we report 12 disk cases in evaluation). Another issue is that sometimes disk switching is not detected reliably by the hosts, forcing us to power cycle the devices. We found that USB 2.0 does not have these issues, which suggests that future iterations of the software driver and chipsets implementation might fix the problems.

For the software stack, we implement the UStore Master based on ZooKeeper [25], which acts as a fault tolerant metadata storage and execution coordinator. The Master and ZooKeeper are co-deployed in a small cluster (*e.g.,* 5 machines). The master processes are running in the active-standby mode. At any time, there is only one active master process and the others are standby. The active process is elected by ZooKeeper. The metadata of UStore is stored in ZooKeeper and is organized in a hierarchical tree structure. Each host creates an ephemeral znode in ZooKeeper to represent its liveness.

TABLE I: The comparison in price of different storage solutions. "AttEx" means the capital expense without disks. (unit: thousands of dollars)

System	Media	CapEx	AttEx
DELL PowerVault MD3260i	Near-line SAS	$3,340	$1,525
Sun StorageTek SL150	LTO6 Tape	$1,748	-
Pergamum	SATA HD	$756	$415
BACKBLAZE	SATA HD	$598	$257
UStore	SATA HD	$456	$115

VI. Cost Comparison

UStore is cost-efficient for two reasons: (a) leveraging the low cost USB technology for the interconnect fabric, and (b) utilizing existing infrastructure as much as possible. We compare UStore with other storage solutions in monetary cost. The cost of a storage system includes two aspects: the capital expense (CapEx) and the operational expense (OpEx).

The CapEx. CapEx of a storage system includes not only the cost for the medium (*e.g.,* disks), but also the cost of enclosure, power supply, cooling, and the cost of connecting them to the computers and network. Table I shows the estimated CapEx of a 10PB raw storage capacity. The first two systems are commercially supported products. The PowerVault MD3260i [26] enclosure is configured to contain 60 near-line SAS drives of 3TB each. It is intended for performance critical workloads and has much higher costs than other solutions. StorageTek SL150 is a tape storage system. It is much cheaper than MD3260i, but has poor performance for random access.

We use 3TB SATA HDDs, which cost about $100 each, as the storage medium for the last three solutions. BACKBLAZE [22] is a custom-made low-cost cloud storage solution. It attaches 45 disks to a low-end motherboard in a 4U enclosure. Though the cost is relatively low, it suffers from single point of failure that may render all 45 disks unavailable, and its performance is rather poor since all 45 disks share only a single GbE network interface. Pergamum [6] uses low-power ARM microprocessors to attach and expose disks on the network. The ARMs are interconnected through Ethernet. In order to make a fair side-by-side comparison, we remove the NVRAM in each tome of Pergamum and put 45 tomes into the same enclosure used by [22]. We use BACKBLAZE numbers to estimate the cost of the enclosure, power supply, fans and so forth. Cubieboard3 [27] is used to estimate the price of the ARM in the tome. We choose Cubieboard3 because it has native SATA and GbE support, and is one of the most widely used commodity ARM single board computers. Ethernet tree topology is used to interconnect the tomes [2].

At last, we estimate the CapEx of UStore. Due to huge amount of shipments and fierce competition among the vendors, USB 3.0 technology is extremely cost effective. All

[2]1Gb/s port is $4 and 10Gb/s port is $100. These prices are much lower than that in [28].

TABLE II: The performance of one disk for three connection types. H&S means hub and switch, *i.e.,* the third connection type. 100%, 50% and 0% represent the percentage of read operations.

Workloads	**4KB(IO/s) Seq**			**4KB(IO/s) Rand**			**4MB(MB/s) Seq**			**4MB(MB/s) Rand**		
Read Percentage	100%	50%	0%	100%	50%	0%	100%	50%	0%	100%	50%	0%
SATA	13378	8066	11211	191.9	105.4	86.9	184.8	105.7	180.2	129.1	78.7	57.5
USB	5380	4294	6166	189.0	105.2	85.2	185.8	119.7	184.0	147.9	95.5	79.3
H&S	5381	4595	6181	189.2	106.0	87.9	185.8	118.6	184.9	147.7	97.7	79.9

the IC components used in the interconnect fabric cost less than $1 each. We multiply bill of materials (BOM) cost by 2 [29] to estimate the cost of the interconnect fabric. We use deploy unit of 64 disks in one 4U enclosure to estimate the system cost. Similar to Pergamum, we estimate the cost of the enclosure and other components using the numbers from [22]. We justify UStore's larger number of disks in a single enclosure by noticing the large empty space occupied by motherboard in [22], and noticing that other systems can accommodate similar number of disks in a 4U enclosure [23], [24]. The result in Table I shows that UStore has the obvious advantage over other solutions in CapEx. For example, UStore costs 24% lower than BACKBLAZE, the next lowest cost solution, when media cost is included. Excluding the disk cost [3], UStore is 55% cheaper. Notice that UStore achieves the cost advantage while providing a much better throughput and fault tolerance.

The OpEx. OpEx usually includes power, cooling, data center floor space and maintenance cost. We are not able to compare different solutions quantitatively since it is very difficult to do such estimation without full deployment and years of usage data. Here we qualitatively compare the OpEx. UStore can pack more disks in an enclosure due to its simple construction and the power consumption is relatively low (see § VII-C), so it should be competitive in power consumption, cooling and space efficiency. Moreover, UStore system can detect failure through software (see § IV-E), implement fast failover automatically, and pinpoint the components that need repair, so the maintenance cost should also be competitive.

VII. Experiments

In this section, we evaluate the prototype described above to address the following questions: what throughput can UStore provide, how long does it take to switch disks and what the power consumption of UStore is.

A. Throughput

In order to show the throughput of UStore, we evaluate the prototype with different workloads by combining different values of three parameters: transfer size, read/write mix percentage and access patterns. We use Iometer [30] for this evaluation.

First, we evaluate the throughput of a single disk with three different connection configurations. The first configuration connects a disk to host directly with SATA, while the second one goes through an USB3.0 bridge. The third configuration is the full fabric as described in § V, except that only one disk is powered on and working. The disk goes through two hubs, two switches and a bridge. The results are shown in Table II. All three connection methods have similar throughput in large transfer workloads. In the 4KB sequential workloads, throughput of direct SATA connection is 2 times better than going through USB. We suspect this is due to the added latency introduced by the SATA to USB bridge. From this experiment we conclude that for large reads and writes, going through USB bridge, hub and switch almost have no impact on the performance of a single disk.

In the next experiment, we test the performance implications of sharing hubs in our prototype. Different numbers of disks are attached to a single host through the fabric of our prototype. We set the disk numbers to be 1, 2, 4, 8, 12 respectively. For the cases with 1, 2, 4, disks are connected to the same hub. For the cases with 8 and 12 disks, the disks are connected to 2 hubs and 3 hubs respectively. We use multiple workers in Iometer to test the system, each worker operates on one disk. We only show some workloads in Figure 5, other workloads have similar results. As shown in the left two figures, for small transfers the throughput increases with the number of disks. The sequential throughput of 8 disks can saturate the USB tree. For large transfers, two disks are enough to fill up the root hub's bandwidth, which is around 300MB/s. Though not shown in the graph, our experiments also find that when multiple disks are connected to a single host, the bandwidth is shared evenly among the disks.

Since USB3.0 is a duplex transfer protocol, the total throughput doubles when half of disks are read and the other half are written simultaneously. We perform the experiments for the 4MB workloads, the total throughput reaches 540MB/s which is the sum of the read throughput and the write throughput. Thus, with four root paths the prototype can sustain a total throughput of 2160MB/s.

B. Switching Time

We also measured the switching time which is the delay of switching disks from one host (*i.e.,* safely rejected) to another one. The delay is composed of three parts. The first part is between the disk being rejected from one host and being recognized by the USB driver of another host. The second part is the time between the disk being recognized to being exposed onto the network. The last part is from the time of

[3]Since the disk-based systems in Table I all use 3TB disks, "AttEx" can also represent the amortized cost per disk.

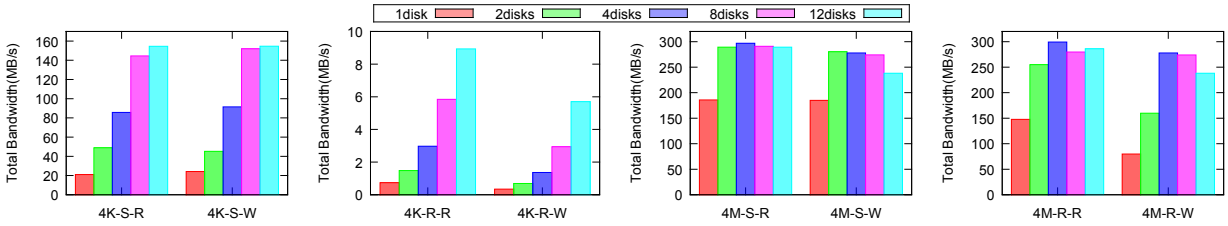

Fig. 5: Total throughput of multiple disks. The workload name in this figure is comprised of three parts. The first part is transfer request size. For the second part, S and R mean sequential and random respectively. For the third part, R means operations are all read, W means operations are all write.

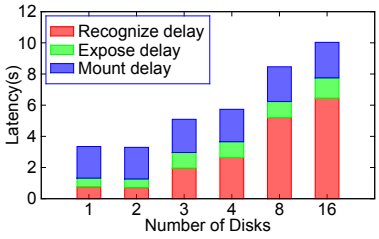

Fig. 6: Switching Time.

TABLE III: Power consumption of one disk. Specs means official disk specification [31]. (unit: watt)

Mode	Spin Down	Idle	Read/Write
Specs	1	5.2	6.4
SATA	0.05	4.71	6.66
USB bridge	1.56	5.76	7.56

the disk being exposed to being remotely mounted by the ClientLib. In the experiment, we switch different numbers of disks simultaneously and repeat each case 6 times to measure the delay. The result is shown in Figure 6, the first part delay increases with the number of switched disks while the second and third parts have little variation.

The delay is short enough for most services in data centers to be regarded as temporary failure and avoid a full rebuild of the disks. To verify this, we deployed Hadoop-1.2.1 on the four hosts of the prototype using disks in UStore as storage. We use one host for namenode and three hosts for datanodes. The system is configured with three replicas. When writing a file in HDFS, we switch one disk, the HDFS client encounters error only for several seconds, then it resumes the operation again. Read operation is not interrupted at all since there are three replicas. Due to the limited space, we do not show detailed performance numbers. It is sufficient to say that most services in data centers show similar behavior to Hadoop, which means switch operation has little impact on these services.

C. Power Consumption

In this section, we investigate the power consumption of two key components: USB bridge and hub. USB switch consumes very little power [32] (around 0.06W in our case). Then we measure the power consumption of the full UStore deploy unit and compare it with alternative solutions to demonstrate its power efficiency.

TABLE IV: The power consumption of one hub with different number of disks connected. (unit: watt)

Disk Count	0	1	2	3	4
Power	0.21	1.06	1.23	1.47	1.67

For the bridge, we evaluate one disk connected with SATA and with USB bridge respectively, in three different workloads including spin down, idle and normal read/write. The result shows that different read/write patterns (*e.g.,* different block size, sequential percentage) make little difference and the variation is within 1W. The power consumption of USB3.0 bridge is around 1W. The detailed result is shown in Table III.

For the hub, we evaluate one hub with 0 to 4 disks connected. The result is shown in Table IV, different modes of the disk, such as idle or busy, make no difference on hub's power consumption. We can see that the hub only consumes 0.21W if there is no disk device connected. Furthermore, with the increase of disks connected on the hub, the power increases in a linear manner except the first one.

We obtain the power consumption of the whole interconnect fabric by measuring the current in the root hub. With the measured power of disks and the estimated power consumption of fans (1W each x6), USB 3.0 host adaptor (2.5W each x4) and power supply (power factor 90plus), the total power consumption of UStore with 16 disks is estimated in Table V. In order to give a glimpse of the comparison with other solutions, we list the numbers quoted from other papers or estimated based on specifications and real measurement in Table V. We compare UStore with two other solutions: Pergamum (without NVRAM, using the same disks, power supplies and fans with UStore), EMC DD860/ES30 [33] which is an enterprise disk-based backup storage products. The numbers in Table V are the amortized power consumption of 16 disks for each solution (15 disks for DD860/ES30). There are two common states in archival storage system: disks serving read/write and disks spun-down/powered-off. For the first state, the ARM in Pergamum consumes around 2.5W and the amortized power consumption of one Ethernet port is 1.5W [34], so the power consumption of Pergamum is about 193.5W. UStore consumes less power in this state, because the power consumption of the interconnect fabric is relatively low at only 13.6W. For the purpose of comparison, the power consumption of DD860/ES30 in idle is much higher. When the workload of

TABLE V: Power comparison of different storage solutions (unit: watt). The first line is when disks are spinning, and the second line is when disks are powered off.

Solutions	DD860/ES30	Pergamum	UStore
Spinning	222.5	193.5	166.8
Powered off	83.5	28.9	22.1

the storage system becomes low, the disks can be spun down or powered off. In this state, DD860/ES30 still consumes much power. For Pergamum, the ARM staying in idle consumes around 0.8W and the power of each Ethernet port becomes 0.5W, making the total to be 28.9W. UStore still consumes much less power since there is no disk and bridge power consumption and the interconnect fabric consumes about 71% less power. Furthermore, UStore hosts can directly cut the power to the root hubs of the fabric to reduce power even further.

VIII. Related Work

Storage systems. Tape storage has been a low-price archival solution for a long time, with systems such as Oracle Tape Storage [35], Tape Cloud [36]. Archival systems using compact disc are also available [37]. Recently, due to the dramatic drop in price, hard disk is becoming an appealing storage medium for cold and archival data [38]. Venti [4] is an archival storage system which applies write-once model and uses secure hash for de-duplication. Other examples of disk-based secondary storage systems include Data Domain [5], HYDRAstor [39]. Pergamum [6] uses low-power CPUs (*e.g.,* ARM) to transform disks into self-contained network attached storage devices. However, it requires a dedicated network, and the performance of low-power CPUs are rather poor, which may limit throughput. Recently, Seagate introduced Kinetic drive [40] that provides Ethernet interface and key-value store API similar to Pergamum. This product is still in its early stage and is not yet widely available. Other large-scale mass storage systems such as Oceanstore [41], Glacier [42], FAB [43], implement decentralized design to provide persistent storage. Petal [44] focuses on providing virtual disks on a pool of physical disks. UStore also focuses on providing raw storage capacity, but unlike Petal, UStore does not natively support redundancy, load balancing, and shared access. It leaves the upper layer services to implement these features.

Commercial products. Commercial companies are providing cold and archival storage solution to the end users. Disk-based commercial products, such as Dell's MD3260i [24] and EMC NL400, provide mass storage capacity. These solutions are often pricey, and consume considerable amount of power [33]. In the cloud side, Amazon Glacier [2] provides cheap storage for archival purposes, but the internal technology is not publicly disclosed. From the physical perspective, BACKBLAZE [22] and Evtron [23] demonstrate how to pack disks densely into a 4U enclosure. A cold storage design [45] by Facebook packs SATA HDs densely in a rack and the disks are mounted by powerful servers interconnected through 10Gb network. This design is still relatively costly and it also suffers from single point of failure.

Reliability and availability. Data reliability is an important aspect of long-term storage systems. Many works have studied different factors that induce data loss, such as latent sector errors (LSEs) [46], disk failure [18] and failures from physical interconnects and protocol stacks [19]. Most of the systems forgo RAID and use software techniques, such as replication [47] and erasure coding [48], to provide fault tolerance under hardware failures. Even though failure of a few nodes can be tolerated in such systems, they still present challenges due to increased recovery traffic on the network and extended window of vulnerability with reduced replicas. Recent work shows that fast data recovery is possible provided that the networking infrastructure is well designed [49]. However, even with the throughput provided with these systems, it would still take many hours to recover all the data on the hard disks in a single deploy unit. UStore alleviate the burden with the reconfigurable interconnect fabric.

Power management. Power-saving is another main concern in data centers. Works, such as Rabbit [50], Sierra [21], rely on the arrangement of replicas. While PARAID [51] proposes power-saving mechanism for server-class RAIDs. Some works apply cache mechanism to prolong idle period of disk clusters [20]. Moreover, disk speed controlling is shown to be an effective approach of saving power [52]. These techniques can be applied on UStore with little modification.

IX. Conclusion

To gain competitive advantage, non-traditional hardware and software architecture is often required for cloud service operators. Vast demand for storage capacity in data centers motivates us to rethink current storage system design, and come up with more cost effective methods to connect disks to servers. UStore is designed with cost efficiency as the primary goal. It provides a substrate of reliable and high performance storage by attaching disks onto the existing servers and networks in a data center. The cost efficiency is mainly achieved by leveraging USB 3.0 technology, which is the cheapest way to connect peripherals. We designed a switching fabric to overcome the bandwidth and reliability limitations of the simple USB tree topology. UStore can be used as the storage capacity provider to a variety of upper layer services, such as distributed file systems and backup services.

X. Acknowledgments

The work was supported by the National High Technology Research and Development Program ("863" Program) of China under Grant 2013AA013203, National Basic Research Program of China ("973") under Grant 2011CB302305, and China NSF under Grant 61232004. We would like to thank our ICDCS reviewers for their valuable feedback.

References

[1] "DropBox," http://www.dropbox.com/.
[2] "Amazon Glacier," http://aws.amazon.com/glacier/.

[3] L. A. Barroso, J. Clidaras, and U. Hoelzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, 2nd ed. Morgan and Claypool Publishers, 2013.

[4] S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Storage," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST)*, 2002.

[5] B. Zhu, K. Li, and H. Patterson, "Avoiding the Disk Bottleneck in the Data Domain Deduplication File System," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST)*, 2008.

[6] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti, "Pergamum: Replacing Tape with Energy Efficient, Reliable, Disk-based Archival Storage," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST)*, 2008.

[7] "SATA Technical Overview," https://www.sata-io.org/technical-overview.

[8] "Redundancy in enterprise storage networks using dual-domain SAS configurations," http://h20565.www2.hp.com/hpsc/doc/public/display?docId=emr_na-c01451157-2&docLocale=.

[9] "Fibre Channel SAN Topologies," http://www.emc.com/collateral/hardware/technical-documentation/h8074-fibre-channel-san-tb.pdf.

[10] "SuperSpeed USB from the USB-IF," http://www.usb.org/developers/ssusb.

[11] C. E. Leiserson, "Fat-trees: Universal Networks for Hardware-efficient Supercomputing," *IEEE Transactions on Computers*, vol. 34, no. 10, pp. 892–901, 1985.

[12] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Elsevier, 2003.

[13] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009.

[14] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, 2008.

[15] L. Lamport, "Paxos Made Simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[16] M. Burrows, "The Chubby Lock Service for Loosely-coupled Distributed Systems," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.

[17] J. Satran and K. Meth, "Internet Small Computer Systems Interface (iSCSI)," 2004.

[18] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in Globally Distributed Storage Systems," in *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010.

[19] W. Jiang, C. Hu, Y. Zhou, and A. Kanevsky, "Are Disks the Dominant Contributor for Storage Failures?: A Comprehensive Study of Storage Subsystem Failure Characteristics," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST)*, 2008.

[20] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao, "Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management," in *Proceedings of the 10th International Symposium on High Performance Computer Architecture (HPCA)*, 2004.

[21] E. Thereska, A. Donnelly, and D. Narayanan, "Sierra: Practical Power-proportionality for Data Center Storage," in *Proceedings of the 6th Conference on Computer Systems (EuroSys)*, 2011.

[22] "Storage Pod 4.0: Direct Wire Drives C Faster, Simpler and Less Expensive," https://www.backblaze.com/blog/backblaze-storage-pod-4/.

[23] "Evtron," http://evtron.com/.

[24] "Dell PowerVault MD3260i," http://www.dell.com/support/Manuals/us/en/19/Product/powervault-md3260i.

[25] "Apache ZooKeeper," http://zookeeper.apache.org/.

[26] "PowerVault MD3 1GB iSCSI SAN storage array series," http://www.dell.com/us/business/p/powervault-md32x0i-series/fs.

[27] "Cubietruck Cubieboard3," http://cubieboard.org/model/.

[28] A. Hospodor and E. L. Miller, "Interconnection Architectures for Petabyte-Scale High-Performance Storage Systems," in *Proceedings of the 21st IEEE/12th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST)*, 2004.

[29] "Retail Pricing, Markup, and Margins," http://ceklog.kindel.com/2012/08/30/retail-pricing-markup-and-margins/.

[30] "Iometer," http://www.iometer.org/.

[31] "TOSHIBA DT01ACAxxx Disk Specs," https://storage.toshiba.eu/cms/en/hdd/computing/product_detail.jsp?productid=447.

[32] "USB 3.0 and USB 2.0 Differential Switch 2:1/1:2 MUX/DEMUX," http://www.ti.com/lit/ds/slas975/slas975.pdf.

[33] Z. Li, K. M. Greenan, A. W. Leung, and E. Zadok, "Power Consumption in Enterprise-scale Backup Storage Systems," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST)*, 2012.

[34] "Cisco 200 Series Smart Switches Cisco Small Business," http://www.cisco.com/c/en/us/products/collateral/switches/small-business-100-series-unmanaged-switches/data_sheet_c78-634369.pdf.

[35] "Oracle Tape Storage," http://www.oracle.com/us/products/servers-storage/storage/tape-storage/overview/index.html.

[36] V. S. Prakash, X. Zhao, Y. Wen, and W. Shi, "Back to the Future: Using Magnetic Tapes in Cloud Based Storage Infrastructures," in *Proceedings of the ACM/IFIP/USENIX International Middleware Conference (Middleware)*, 2013.

[37] T. Tanabe, M. Takayanagi, H. Tatemiti, T. Ura, and M. Yamamoto, "Redundant Optical Storage System Using DVD-RAM Library," in *Proceedings of the 16th IEEE Symposium on Mass Storage Systems*, 1999.

[38] K. Zhou, H. Wang, and C. Li, "Cloud Storage Technology and Its Applications," *ZTE Communications*, vol. 8, no. 4, pp. 27–30, 2010.

[39] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki, "HYDRAstor: A Scalable Secondary Storage," in *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST)*, 2009.

[40] "Seagate Kinetic Open Storage Platform," http://www.seagate.com/solutions/cloud/data-center-cloud/platforms/?cmpid=friendly-_-pr-kinetic-us.

[41] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer *et al.*, "Oceanstore: An Architecture for Global-Scale Persistent Storage," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[42] A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures," in *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[43] Y. Saito, S. Frølund, A. Veitch, A. Merchant, and S. Spence, "FAB: Building Distributed Enterprise Disk Arrays from Commodity Components," in *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2004.

[44] E. K. Lee and C. A. Thekkath, "Petal: Distributed Virtual Disks," in *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 1996.

[45] "OPEN Compute Project Cloud Storage Hardware v0.5 ST-draco-abraxas-0.5," http://www.opencompute.org/assets/download/Open-Compute-Project-Cold-Storage-Specification-v0.5.pdf.

[46] B. Schroeder, S. Damouras, and P. Gill, "Understanding Latent Sector Errors and How to Protect Against Them," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST)*, 2010.

[47] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[48] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure Coding in Windows Azure Storage," in *Proceedings of the 2012 USENIX Annual Technical Conference (USENIX ATC)*, 2012.

[49] E. B. Nightingale, J. Elson, J. Fan, O. Hofmann, J. Howell, and Y. Suzue, "Flat Datacenter Storage," in *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012.

[50] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, "Robust and Flexible Power-proportional Storage," in *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC)*, 2010.

[51] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning, "PARAID: A Gear-Shifting Power-Aware RAID," in *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST)*, 2007.

[52] E. V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving Disk Energy in Network Servers," in *Proceedings of the 17th Annual International Conference on Supercomputing (ICS)*, 2003.