Zone-based data striping for cloud storage

L. Lu
D. Hildebrand
R. Tewari

Data centers in the "cloud" will need to support a wide range of applications, each with their own input/output (I/O) requirements. Some applications perform small and random I/O operations, whereas others demand high streaming bandwidth. In addition, in order to reduce costs, cloud data centers will contain thousands of commodity servers and network switches. Delivering high performance with unreliable commodity hardware for this range of workloads is a grand challenge. ZoneFS uses zone-based data striping in data center infrastructures built from commodity network switches. By striping data in zones, ZoneFS reduces network congestion and avoids one or more storage servers from becoming I/O bottlenecks. In this paper, we present the overall design and implementation of ZoneFS and evaluate its key features with several cloud-computing workloads.

Introduction

Cloud data centers will need to support thousands of simultaneous users, each running jobs that may generate and access petabytes of data with a variety of access patterns and then sharing this data in a secure manner with other data centers around the world [1]. Several specialized storage systems are emerging that would force administrators to divide the data center into several miniclusters [2–4]. In addition, while enterprise and web applications require standard file system interfaces, many of these specialized storage systems rely on specialized interfaces [5, 6]. With applications demanding increasingly more compute, bandwidth, and storage capacity, access to file data must scale with the available hardware. Dividing the data center into a series of specialized clusters limits scalability and can quickly become unmanageable. What is needed is a single storage solution that can offer scalability and performance for virtual and nonvirtual analytic, web server, and database applications.

Additionally, the use of commodity hardware suggests that any storage solution must be robust with respect to failures. Consider a typical compute-cloud architecture that presents a virtualized environment. Applications run inside a virtual machine (VM) and access data from a virtual logical unit number (LUN), which is typically stored as a file, e.g., a VMWare**.vmdk file, in the storage system. If the virtual LUN is stored on a single server, the VM must run on that same machine, creating possible "hotspots." Alternatively, if the virtual LUN is stored on a separate network-attached storage (NAS) server, then the VM can run on any server but must transfer all of its data (> 80 GB) over the oversubscribed data center network. A better solution would be to stripe all file data across the servers connected to a single switch, allowing VMs to run on any server without having to transfer data across higher level switches. This enormously reduces both the time and the complexity of configuring new VMs and dynamically migrating them from one server to another.

Supercomputers have been operating at the limits of modern compute and data-processing capabilities using a massive and expensive single network switch. By forcing all input/output (I/O) through a single switch, parallel file systems and other NAS solutions are severely bandwidth limited in cloud data centers that use inexpensive commodity-based compute clusters and networking hardware [7]. With each successive level in the data center switch hierarchy, increasingly more compute nodes must share a decreasing amount of available bandwidth to the parallel file system.

To take better advantage of commodity hardware and overcome switch limitations, Internet-scale file systems such as the Hadoop** distributed file system (HDFS) and the Google** file system have applications that use customized interfaces to access data locally by randomly assigning large

Digital Object Identifier: 10.1147/JRD.2011.2165681

0018-8646/11/$5.00 © 2011 IBM

data chunks directly on compute nodes. These file systems centralize large (128 MB) chunks of contiguous data on a single node. This can create data hotspots as compared to parallel file systems that tend to use a 64-KB to 1-MB stripe size. While triplication somewhat helps alleviate these hotspots, it provides little benefit if an entire rack of 20–40 servers needs to access the same piece of data. In addition, re-replicating the terabytes of data on a failed node may require hours, if not days, increasing the probability of data loss. Another challenge is that general applications have difficulty using these file systems since they do not support POSIX** (Portable Operating System Interface [for UNIX**]), and remote data access via Network File System (NFS) or Common Internet File System (CIFS) is slow. In addition, the tight coupling of data and compute processing ignores the required ratio of storage space to compute power in a data center. Administrators cannot just add storage or computing resources but must scale both in tandem, which is particularly problematic for large data sets that need to be accessible but are very infrequently accessed.

In this paper, we propose ZoneFS, which addresses the storage limitations imposed by commodity local disk servers and small commodity switches by providing the file system with an understanding of the network infrastructure of the data center. ZoneFS views the data center as a collection of miniclusters, or zones, each with their own high-bandwidth and fully connected switch. In this manner, ZoneFS uses parallel I/O to effectively load balance I/O requests across each zone, improving I/O performance over the entire range of I/O workloads. ZoneFS focuses on the storage architecture within a data center but can share data between data centers by using other existing data transfer and caching technologies [8].

ZoneFS is a novel POSIX-compliant parallel file system that retains the reliability, scalability, and performance benefits of standard parallel file systems while making use of the commodity switch architecture of Internet-scale file systems. ZoneFS distributes data across any number of nodes, eliminating data hotspots. As the number of zones and data sets increases, so does the aggregate I/O bandwidth. Numerous applications can be launched on any node or nodes in a zone, and all realize similar performance—even if they all access the same data sets. This paper uses benchmarks and applications to demonstrate that ZoneFS can exceed the performance of both parallel and Internet-scale file systems.

ZoneFS has a flexible data architecture that can support directly attached disks, storage area networks (SANs), or even SAS (serial-attached SCSI, or Small Computer System Interface) disk arrays, each offering different levels of cost, availability, and performance. For example, ZoneFS can use a client-driven redundant array of independent disks (RAID) across multiple storage nodes connected to SAS disk arrays. This allows clients to fail without affecting data availability and avoids the need to replicate data within a zone. In addition, by allowing a separation between compute and storage nodes, administrators can turn off underutilized compute nodes for power conservation. In this paper, we analyze and compare directly attached disk and fiber channel disk array architectures.

The remainder of this paper is organized as follows: First, we review the data center networking architecture and discuss the limitations of using parallel and Internet scale file systems. We then describe the ZoneFS architecture, our Linux** prototype, and report the results of experiments with microbenchmarks, as well as analytic and general applications. Finally, we discuss related and future work, and then summarize and conclude.

Background

To better understand the motivation for ZoneFS, let us review current data center network topologies and how they influence modern file system design.

Data center network topologies

Many data centers use multitier trees of network switches or routers [9]. Compute servers are connected directly into the leaves at the bottom tier, which consist of smaller Gigabit Ethernet (GbE) switches, e.g., 48 ports. The upper tiers aggregate the leaves to create a fully connected system. While the upper tiers may use 10-GbE switches, the network infrastructure is oversubscribed. Typical data centers are oversubscribed by a factor of 8:1 or even larger. While it is possible to eliminate oversubscription and use a ratio of 1:1, it is cost prohibitive for most data centers. This oversubscription creates the interswitch bottleneck that constrains data access in data centers.

Parallel file systems

Over the last decade, parallel file systems such as GPFS* (General Parallel File System) [10], Panasas** [11], and Lustre** [12] have achieved unprecedented scalability in numbers of clients, servers, and disks. These achievements require that a system scale every aspect of its design—especially the network switching fabric. Parallel file systems increase aggregate I/O throughput by striping data across possibly hundreds of storage nodes. This technique can reduce the likelihood of any one storage node becoming a bottleneck and offers scalable access to a single file.

One example is the IBM Blue Gene*/P Intrepid supercomputer, which connects 40 racks of compute servers to 640 I/O servers, which, in turn, are connected over a Myri-10 GbE switch complex to 128 storage servers. While cloud data centers can require such bandwidth, they lack the budget for such switches, and so, it is doubtful that parallel file systems will emerge as a popular solution.

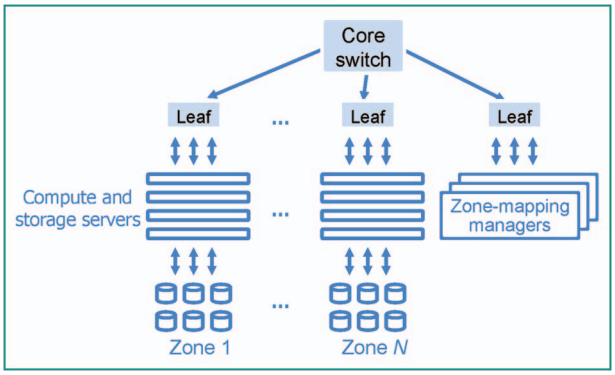

Figure 1

ZoneFS in the data center. ZoneFS creates zones from the leaf network switches in the data center. All written data is striped across the storage nodes in the same zone, avoiding transferring data across the upper level switch.

Scale-out network attached storage

Scalable NAS systems [13–15] use standard filing protocols such as NFS and CIFS to scale I/O throughput. Currently, many users of scalable NAS systems focus on I/O operations per second, e.g., creates per second, rather than sustained I/O throughput [16]. Consequently, administrators can oversubscribe their data center network infrastructure with little consequence since file create and POSIX `stat` (get file status) operations tend to require little raw bandwidth. If these users start executing "big data" applications, they would encounter the same network oversubscription problems as with using parallel file systems. (The term *big* data often refers to data sets so large that they become challenging to store and process.)

Internet-scale distributed file systems

Users of big-data applications have realized that POSIX-compliant file systems and their associated semantics can be excessive for their specific requirements. This has driven the creation of file systems that colocate compute processing and data on a single node with directly attached disk [5, 6]. However, while applications attempt to always access data locally, this is not always possible in a loaded system, and so, data continues to flow over the oversubscribed network. In addition, their use of replication and interrack access continue to place a load on the network infrastructure.

ZoneFS

In this section, we introduce and describe ZoneFS. As shown in **Figure 1**, ZoneFS is a novel parallel file system that uses parallel data access in zones that have the highest potential aggregate bandwidth.

Architecture

Figure 2 depicts the ZoneFS architecture, which organizes data center compute and storage resources into one or more zones, which consist of a leaf switch and its attached set of compute and storage servers. The number of zones can grow with the number of leaf switches in the data center, enabling vast and incremental expansion of the data center. In addition, ZoneFS uses a zone-mapping manager to maintain a mapping of the files and directories stored within each zone.

Limiting the striping of a file to a switch and its connected servers avoids many of the network oversubscription pitfalls discussed earlier. For example, standalone applications, Internet workloads, and moderate-sized parallel applications can all experience the performance of a parallel file system without the expensive switching hardware. It is also important to note that ZoneFS supports any number of switch ports and connected servers within a single zone. This gives administrators the flexibility to create zones of a standard size or create zones of varying sizes to handle different types of workloads.

Zones

A zone consists of a switch and its attached servers. Servers are logically divided into a set of compute servers and storage servers, with file data and metadata striped across the storage servers. Compute servers generate I/O requests to the storage servers on behalf of applications. Storage servers are attached to any number of storage mediums such as direct-attached disks, solid-state drives, or a SAN. This separation allows administrators to turn off underutilized compute nodes for power conservation.

The logical division of storage and compute servers allows significant flexibility in the composition of the zone. If every server has access to storage, then every server can serve data requests. On the other hand, all storage devices can be consolidated in a few robust storage servers that are outfitted with quad-GbE or 10-GbE cards, and connected to commodity disk arrays. This allows compute-intensive data centers to focus on a larger number of compute servers per zone and "storage-heavy" data centers to increase the number of storage servers per zone. Another option would be to set up a small SAN for each zone, allowing direct data access from every server.

Zone-mapping manager

For applications to take advantage of ZoneFS and avoid interswitch bottlenecks, it is important that applications be launched on one or more servers in the correct zone. Moving compute processing to a zone is 20–40 times more flexible (depending on how many servers are in one's zone) than the Internet-scale file system method of moving compute operations to a particular server.

A set of zone-mapping managers maintain a mapping of the files stored within each zone. A cluster job scheduler

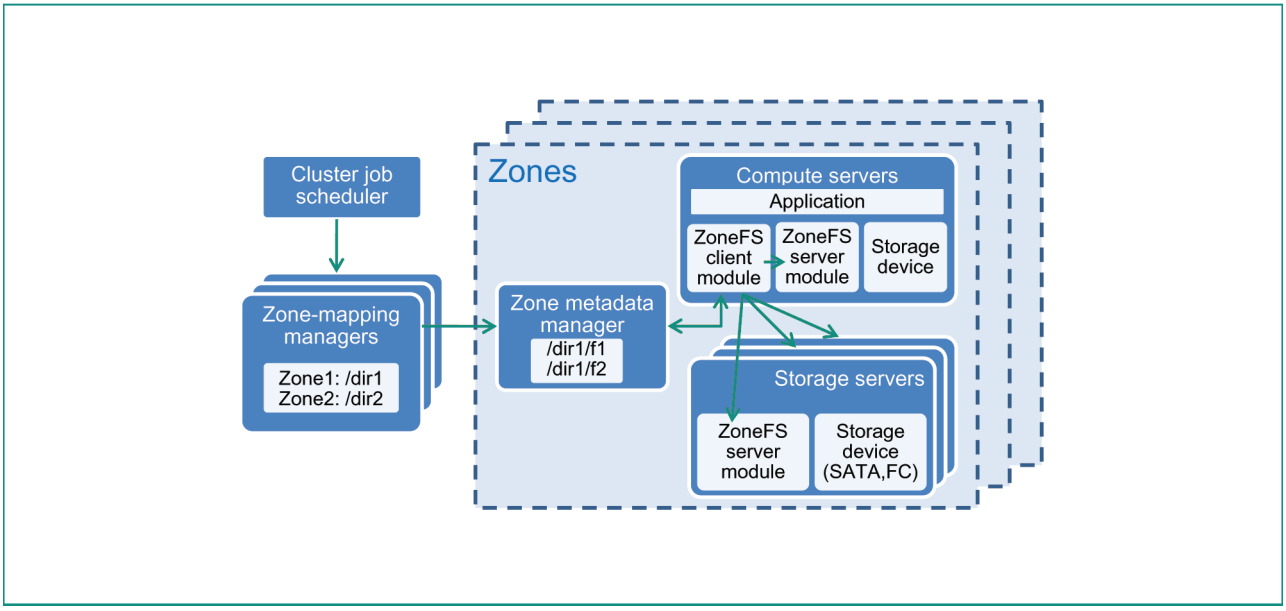

Figure 2

ZoneFS architecture. Each zone consists of a file-system metadata manager, a ZoneFS client on each compute server, and ZoneFS server modules, which can exist on compute servers or on separate storage servers. All data created within a zone is striped across the storage nodes in the same zone. Zone-mapping managers coordinate to track allocation of data to each individual zone. The cluster job scheduler launches applications in one or more zones, depending on the location of the required data set. Example storage devices include serial advanced technology attachment and fiber channel.

uses this service to launch applications on compute servers within the correct zone. Once an application is running in a zone, it accesses data on the local storage nodes.

ZoneFS scales metadata access by dividing the information into two separate groups: file metadata and data location information. File metadata is stored on storage servers within the zone in which the file is located. ZoneFS clients or a cluster job scheduler access and update data location information, i.e., the mapping of files to zones, by accessing a zone-mapping manager.

Note that applications running in one zone can access data in any zone in the data center but are subject to the available bandwidth between leaf switches. In addition, our current design places files within a single zone, but we are investigating whether we can stripe large chunks of a file in separate zones.

High availability

While this paper focuses on the performance benefits of ZoneFS, it is worth discussing how ZoneFS can handle disk, node, and rack failures. While Internet-scale file systems rely on triplication (mainly for cost reasons), the flexible architecture of ZoneFS allows for several different alternatives, depending on the storage subsystem architecture. For example, to recover from disk failures, each storage node can employ RAID across its local disks.

To recover from storage server failures, for SAN architectures, any storage server can serve any piece of data. Compute nodes and file system clients can simply reroute their requests to other storage nodes within the zone. For local disk architectures, ZoneFS can guard against failures by using replication or client-driven (network) RAID 5/6 [11] across the storage servers in an individual zone. In either case, all storage nodes can participate in the reconstruction, greatly decreasing time to recovery. Another option is to not install any disks within a storage server at all but rather connect the storage servers to just a bunch of disks (JBODs) or SAS disk arrays. In this configuration, the disk-array interface cables can be connected to multiple storage servers, allowing failover from one storage server to another.

To recover from the failure or planned shutdown of a zone (possibly due to a top-of-rack switch failure), one option is for ZoneFS clients and storage servers to replicate data between zones and update the zone-mapping manager with new location information. This technique would force data to pass through the oversubscribed upper level switches (just like replication in HDFS). One way to avoid this would be to connect JBODs or SAS disk-array interface cables to storage servers in two zones. This allows a zone, with storage nodes that are inaccessible because its networking facilities are powered off, to continue serving I/O requests via the storage nodes of another zone.

ZoneFS implementation

We implemented a ZoneFS prototype that is layered upon the IBM GPFS parallel file system. To support data-intensive analytic workloads, we also modified Hadoop to support GPFS and ZoneFS as primary file systems.

ZoneFS prototype

We implemented a prototype of ZoneFS in a layer on top of a single running instance of GPFS for Linux. Storage nodes can support either local disk or a shared SAN per zone.

To perform I/O within each zone, compute nodes stripe data across only their local storage nodes. To perform direct and parallel I/O to the storage nodes in a different zone, compute nodes must "mount" the storage nodes from each remote zone. ZoneFS clients can access data in other zones by first contacting the remote zone metadata manager to determine the storage nodes of the zone. The ZoneFS client can then use parallel I/O to access the remote file. While these clients are subject to interswitch bottleneck, they prevent storage hotspots by balancing requests across all available storage nodes.

The zone-mapping manager maintains mappings of all files and directories stored in each zone, assigning a "local" or "remote" attribute to directories, depending on the zone in which they are located. To assign jobs to specific zones, we implemented a job scheduler that works with both message passing interface (MPI) and Hadoop. The job scheduler uses the zone-mapping manager to assign jobs to zones based on the pathname prefix of the required data for the job. Within each zone, the specific compute nodes that are utilized depend on many factors, e.g., the current number of running jobs and central processing unit (CPU) utilization. Currently, our prototype evenly load-balances jobs across all nodes and maps directory trees to zones, which drastically reduces the number of application mapping requests. Failure recovery relies on the fault tolerance mechanisms of GPFS and the storage controllers.

Our implementation of ZoneFS is separate from the file system and is compatible with other parallel file systems that can be run on commodity hardware and utilize direct-attached disks or a SAN. It is interesting to note that ZoneFS does not require all zones to use the same file system across its storage nodes. Heterogeneous data centers that use multiple file system across their compute clusters can use ZoneFS to build a single namespace and allocate jobs across all available file systems.

Hadoop-GPFS plug-in layer

In order for Hadoop to support a file system, it must be able to determine all available storage nodes and their associated data blocks. To support GPFS and ZoneFS, we made use of the Hadoop abstract file system application programming interface to implement Hadoop-GPFS and Hadoop-ZoneFS plug-ins. Hadoop uses these plug-ins to determine the optimal server to utilize for each data block. Our GPFS and ZoneFS plug-ins intercept file system data block requests and determine the node on which to launch a map/reduce job. For GPFS, the plug-in returns a random node from any zones, which works because data blocks are striped across all storage nodes in the data center. For ZoneFS, the plug-in uses the zone-mapping manager to return a random node in the zone that contains the requested data.

Evaluation

In this section, we evaluate ZoneFS, GPFS, and HDFS under typical data center workloads and cluster configurations. We first compare the I/O scalability and performance of GPFS and ZoneFS using the IOR microbenchmark. Next, we analyze file system performance with large I/O requests using three Hadoop applications. We then use both a microbenchmark and a Hadoop application to investigate the ability of these file systems to provide predictable performance, regardless of how data is striped across the storage nodes. Finally, we analyze small- and medium-sized I/O request performance with an image conversion cloud-computing application.

Experimental setup

In our experiments, we compare ZoneFS with GPFS and the Hadoop DFS. In addition, with GPFS and ZoneFS, we use both a SAN, which is used by many parallel file systems, and a local disk storage configuration, which is used by many data centers.

Our experiments compare the following configurations: 1) *HDFS* (Hadoop DFS with local disks); 2) *GPFS Local* (standard GPFS with local disks); 3) *GPFS SAN* (standard GPFS with SAN); 4) *ZoneFS Local* (switch-aware GPFS with local disks), and 5) *ZoneFS SAN* (switch-aware GPFS with SAN).

We use Hadoop 0.20.0 with a block size of 64 MB. Both GPFS and ZoneFS use a stripe size of 1 MB. Write experiments are complete when all data is on disk. Read experiments use an empty data cache.

All experiments are conducted on a 16-node cluster, with eight nodes on each of two racks. Three Netgear** gigabit switches are used to connect nodes within each rack and the two racks together. A GbE link connects the leaf switches to the parent switch. Each node is equipped with dual 3-GHz Intel Xeon** processors, 4 GB of memory, and a local disk that runs Red Hat** Enterprise Linux 5.2. All nodes have access to a shared fiber channel SAN, which comprises a 16-port fiber channel switch connected to an IBM DS4700 storage controller. Each node has five hard drives using RAID5. For local disk configurations, nodes can directly access data on their local disks but must communicate with other nodes for access to their data. For SAN configurations, each rack is allocated a set of devices and cannot directly access devices on the other rack, which must be done through the upper layer switch.

Scalability: GPFS versus ZoneFS

This section compares scalability of standard GPFS, which stripes files across every node on both racks, and ZoneFS, which stripes files only within a zone. We use the IOR 2.10.1 benchmark [17] to read and write separate 2-GB files.

Local and cross-rack performance

We first motivate the design of ZoneFS by comparing the performance with respect to accessing files solely within a single rack versus across racks. As expected with GPFS, striping files across multiple racks bounds I/O performance by the upper level gigabit switch—increasing the number of nodes does not increase the read or write I/O throughput beyond 1 Gb/s (gigabits per second). When striping data within a single rack, GPFS saturates the available read bandwidth of 600 MB/s (megabytes per second) and the available write bandwidth of 275 MB/s.

Cluster performance

Now that we understand the baseline interrack and intrarack I/O performance of GPFS, we compare ZoneFS and GPFS on the whole cluster. With ZoneFS striping data within each individual rack, ZoneFS continues to saturate the available storage bandwidth. GPFS reads and writes data from both the local and remote rack. As such, it can perform fast I/O within the rack but is limited by the interswitch network bandwidth for data blocks on the remote rack. Consequently, it realizes a maximum read bandwidth of 425 MB/s and a maximum write bandwidth of 175 MB/s.

Analytics workloads

As data-intensive applications are growing in popularity, any file system for the data center must excel for these workloads. Our goal is to demonstrate that ZoneFS can outperform GPFS and match the runtime performance of HDFS, which is the file system built specifically for Hadoop. We choose the Hadoop workload Grep, with each active node accessing a separate 1-GB file.

For these experiments, we recreate the 10 : 1 network oversubscription bottlenecks in large data centers by scaling down the entire system, using only eight nodes per rack, a 1-GbE rack switch, and a 100-Mb/s (megabits per second) interswitch network link [9]. We find that ZoneFS can make use of switch awareness to achieve similar performance as HDFS, which uses node awareness. In addition, this experiment demonstrates that the specialization of HDFS, while possibly simplifying its implementation, seems to limit the range of applications HDFS can support and not increase its performance. On the other hand, GPFS, which stripes data across multiple racks, has poor performance due to the interswitch bottleneck.

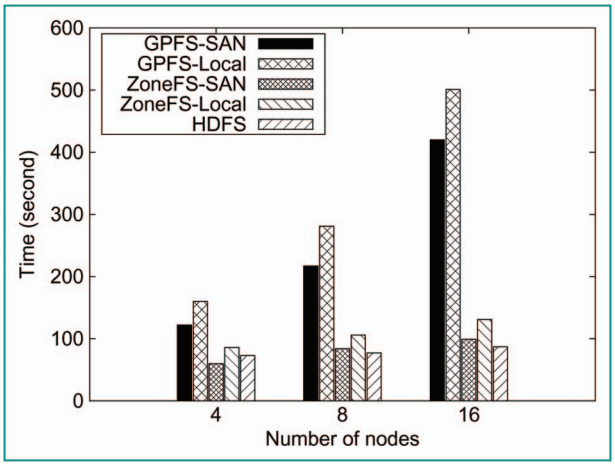

Figure 3

Hadoop Grep read-intensive application. ZoneFS and HDFS have similar runtimes, whereas GPFS continues to suffer from network switch oversubscription.

Grep

The Grep workload searches for an input pattern within a set of data files and is, therefore, CPU- and read-intensive. Grep results are shown in **Figure 3**. GPFS execution time increases linearly with the number of nodes since, as more nodes read data, the interswitch network traffic also increases. With four nodes, the execution time of GPFS SAN is twice that of ZoneFS SAN and 1.67 times that of HDFS. For 16 nodes, GPFS SAN execution times surge to 4.24 times that of ZoneFS and 4.83 times that of HDFS. This clearly shows that standard GPFS suffers from interswitch network bottleneck due to the wide stripe across racks. ZoneFS SAN and HDFS experience similar execution times, whereas ZoneFS Local runtime increases slightly as the number of nodes increases (due to increased packet loss in our Netgear switch).

Next, we execute a strong scaling Grep workload of eight 1-GB data files. All data is created on a single node, and all nodes are in a single rack. **Figure 4** demonstrates how ZoneFS can utilize the additional nodes to analyze the 8-GB data set, with execution time dropping almost linearly with the number of nodes. HDFS again suffers from the single node bottleneck and is unable to use additional compute resources to reduce execution time.

General applications

With the increasing popularity of cloud-computing services such as Amazon EC2, successfully handling of traditional applications is critical. A well-known example is the *New York Times*, which used 100 EC2 virtual machine instances to convert 11 million articles to portable document format [18]. In this section, we analyze the I/O performance

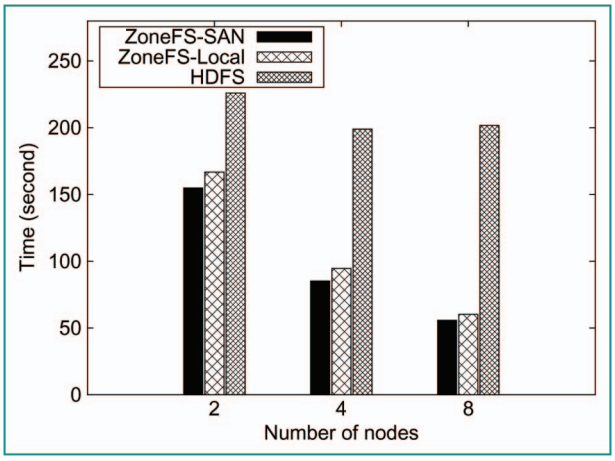

Figure 4

Hotspot Grep workload of an 8-GB data set. By storing data on a single node, HDFS cannot scale with increasing number of compute nodes, whereas ZoneFS stripes data across all nodes in the zone and effectively utilizes additional compute resources.

of applications that generate small and random I/O requests across the file system. This I/O workload is typical of many different applications including application programming interface (OLTP), VM accesses to virtual LUNs, e.g., a VMWare.vmdk file, and many other applications.

To represent this workload, we use an image-conversion benchmark to convert the encoding of 200 images (per node) from a 1-MB JPG image to a 9-MB BMP image using the Python Imaging Library. Image conversion, which consists of sequential and random reads and writes to many small to medium files, is similar to the workload exhibited by OLTP and mail server applications. In addition, beyond the well-known *New York Times* example, image conversion is a very common operation for data centers that allow users to upload images and movies, e.g., Facebook** and Google Picasa**, since they are not stored in the format in which they were uploaded.

We vary the number of nodes from 4 to 16 on two racks and use a GbE intraswitch and interswitch network. To conduct a fair comparison, the benchmark performs the same three-step process for all three file systems. With HDFS, we also ensure that the file conversion application executes on the same node as its input images. HDFS has a specialized local-disk architecture designed for Hadoop applications and was not designed to support large numbers of files with a size less than 64 or 128 MB. The Hadoop experiments showed that the general local disk (and SAN) architecture of ZoneFS could match the specialized architecture of HDFS. These cloud workload experiments demonstrate the degree to which this specialization has hindered HDFS for general applications.

The single HDFS NameNode is overwhelmed with so many small files and requires more than 300 seconds to complete with four, eight, and sixteen nodes. Since each node is reading data locally, HDFS can scale linearly with each node added. GPFS requires 75, 100, and 190 seconds for 4, 8, and 16 nodes, respectively. While the fully distributed architecture increases performance, the runtime increases with the number of nodes (and images) since increasingly more data must travel over the interswitch link bottleneck. ZoneFS runtimes are 50, 72, and 120 seconds for 4, 8, and 16 nodes, respectively. While ZoneFS does not scale linearly due to the GbE network, ZoneFS outperforms HDFS because of its distributed architecture and it outperforms GPFS by avoiding the interswitch bottleneck.

Discussions

Scalability

ZoneFS does not limit the number of racks and switches that it can support. As the number of nodes per rack increases, the available disk and network bandwidth will increase proportionately. The ability to stripe data across all nodes within a rack enables it to scale with available storage and network bandwidth. This allows ZoneFS to avoid single nodes and their associated disks from becoming bottlenecks as Internet-scale file systems.

Efficiency

Our experiments show that the local disk architecture performed only slightly worse than the SAN-per-rack architecture for Hadoop workloads, because the efficiency of Hadoop is relatively low [19], which means that Hadoop cannot drive the I/O subsystem of each individual node. For Grep, the maximal I/O throughput per node during execution is 40 MB/s, which is much less than the peak IO throughput of a storage node. As the efficiency of Hadoop improves over time, ZoneFS will be able to keep pace with performance improvements. In addition, with commodity 10-GbE switches emerging on the market, the performance of the local disk architecture will improve and may exceed that of a low-end SAN such as the one we used for our experiments.

Flexibility

We used a SAN architecture to determine the potential benefits of using a more expensive solution than local disk. Interestingly, beyond the standard benefits of a SAN, e.g., storage virtualization, we found little performance benefits. In addition, although all of our directly attached disk experiments used the same nodes for both compute and storage, we believe that separating compute and storage nodes can bring similar high-availability benefits as a more expensive SAN solution. ZoneFS clients can stripe across robust storage nodes that are outfitted with more disk

and network bandwidth (possibly using client-driven or network RAID). In addition, the decoupling of storage and compute nodes would allow better flexibility, turning off underutilized compute nodes for power conservation.

Related work

Parallel file systems have been widely deployed in high-performance computing (HPC) environments, including IBM GPFS [10], Lustre [12], Parallel Virtual File System (PVFS) [20], and Panasas [11], all of which rely on massive aggregate bandwidth between separate compute and storage nodes. Several of these file systems have specialized interfaces that allow applications to specify per-file striping parameters, e.g., stripe size and RAID number, but they continue to lack knowledge of the data center network topology.

On the other end of the spectrum, Internet-scale file systems [5–21] and even some HPC targeted file systems [22] all try to avoid the network altogether by attempting to colocate compute processing with large chunks of data on individual nodes. This movement away from parallel I/O has many roots, including the existence of incast, the lack of an affordable 10-GbE switching fabric, and an understanding of how to limit striping to avoid oversubscribed switches. Today, incast is an understood phenomenon that can be avoided (to some extent) [23]. In addition, 10-GbE switches are 50% cheaper than GbE switches when considering per-megabyte-per-second throughput cost, and file systems such as ZoneFS provide insights into how to control file striping.

Investigation into understanding Hadoop-specific enhancements to the PVFS [24] and GPFS [25] parallel file systems has begun. These efforts focus on colocating compute processing and data on a single node, i.e., making parallel file systems act more like Internet-scale file systems. By writing large chunks of data to the local disk, these enhancements reduce the key load-distribution benefits of parallel file systems.

Recent work in data center network design strives to eliminate oversubscription to the core switching fabric [26–28]. This work focuses on removing the interswitch bottleneck and is therefore complementary to ZoneFS, which focuses on improving I/O performance within a single switch. Some of this research might enable wide striping of data across all racks in a data center, but this approach continues to have several drawbacks. First, while incast [23, 29] can be mitigated, every additional switch between the source and target presents an opportunity for buffer overflow. Second, wide striping of data means that performance will be limited to the slowest node on which the data resides, which means that striping too wide may start to reduce performance.

Job placement tools such as Tashi [30] and Dynamo [31] seek to optimize the colocation of compute processing and data through an understanding of the data placement within the data center. To assign jobs across the data center, Tashi uses a fine-grained data location service to track data blocks and their associated server. ZoneFS is a candidate file system for use with Tashi. In fact, ZoneFS simplifies and possibly even increases the scalability of Tashi's data location service by mapping data to an entire zone.

Porter explores the advantages and limitations of decoupling storage from computation in Hadoop [32]. Each SuperDataNode (SDN) contains an order of magnitude more disks than traditional Hadoop nodes. This proposal does not support parallel file access across multiple SDNs; thus, each SDN must be managed in isolation without a single complete file system management framework. Consequently, each SDN can become an I/O bottleneck for frequently accessed data. ZoneFS avoids these bottlenecks by using parallel I/O across all storage nodes within a zone.

Future work

ZoneFS targets oversubscribed hierarchical data center network architectures and not supercomputer architectures. As such, ZoneFS does not target HPC workloads since MPI applications that share ghost cells on every iteration would perform unacceptably slow in an oversubscribed data center. However, ZoneFS does support efficient execution of HPC workloads within a single zone, and it would be a very interesting to investigate whether such HPC applications could be modified to account for interswitch bottlenecks.

We are currently implementing ZoneFS in a layer above the file system. This allows our prototype to work with any parallel file system but at the cost of additional management and setup overhead. Integrating directly with GPFS (or any other parallel file system) would not only remove this overhead but would also provide easier management for administrators.

Exploring local-disk and SAN storage systems with a wider variety of applications workloads such as web servers and databases would allow us to provide more insight into their true costs and benefits. For example, with local-disk architectures, nodes have higher load since they must both issue and serve data requests. This was not a factor in our experiments but may become a challenge with other workloads.

ZoneFS provides flexibility in recovering from failures. To avoid replication within a zone, we would like to explore the use of client-driven (network) RAID5/6 or declustered RAID across the storage servers [11]. This could especially provide a significant benefit to local-disk architectures in which it is difficult to add additional disks without adding additional servers.

Conclusion

This paper addresses the varied storage requirements of applications running in cloud data centers. The inexpensive

multitier network infrastructures that utilize small switches at the leaves create I/O bottlenecks that pose problems for today's file systems. ZoneFS address these bottlenecks by combining the scalability, performance, ease of management, and standard semantics of a parallel file system with the cost effectiveness of an Internet-scale file system. Striping data in zones with the highest bandwidth balances I/O requests across all the disks on all servers within a zone. For applications that operate on large data sets, we found that ZoneFS could outperform GPFS and match or outperform HDFS with identical hardware and storage configurations.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of VMWare, Inc., Apache Software Foundation, Google, Inc., Institute of Electrical and Electronics Engineers, The Open Group, Linus Torvalds, Panasas, Inc., Cluster File Systems, Inc., Netgear, Inc., Intel, Inc., Red Hat, Inc., Facebook, Inc., or Google, Inc., in the United States, other countries, or both.

References

1. "Big data," *Nature (entire issue)*, vol. 455, no. 7209, pp. 1–136, 2008.
2. Apache Hadoop. [Online]. Available: http://hadoop.apache.org
3. J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. USENIX Symp. Oper. Syst. Design Implementation*, San Francisco, CA, Dec. 2004, pp. 1–13.
4. D. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, "FAWN: A fast array of wimpy nodes," in *Proc. 22nd ACM Symp. Oper. Syst. Principles*, Big Sky, MT, 2009, pp. 1–17.
5. S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. ACM Symp. Oper. Syst. Principles*, New York, 2003, pp. 137–150.
6. HDFS: Architecture and Design. [Online]. Available: http://hadoop.apache.org/hdfs
7. L. A. Barroso and U. Holzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. San Rafael, CA: Morgan and Claypool, 2009.
8. M. Eshel, R. Haskin, D. Hildebrand, M. Naik, F. Schmuck, and R. Tewari, "Panache: A parallel file system cache for global file access," in *Proc. USENIX Conf. File Storage Technol.*, San Jose, CA, 2010, pp. 155–168.
9. Cisco Systems, Inc., Cisco Data Center Infrastructure 2.5 Design Guide. [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/solution/dcidg21.pdf
10. F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proc. USENIX Conf. File Storage Technol.*, Monterey, CA, 2002, pp. 231–244.
11. B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou, "Scalable performance of the Panasas parallel file system," in *Proc. USENIX Conf. File Storage Technol.*, San Jose, CA, 2008, pp. 17–33.
12. Lustre File System. [Online]. Available: wiki.lustre.org
13. S. Oehme, J. Deicke, J.-P. Akelbein, R. Sahlberg, A. Tridgell, and R. L. Haskin, "IBM scale out file services: Reinventing network-attached storage," *IBM J. Res. & Dev.*, vol. 52, no. 4/5, pp. 319–328, Jul. 2008.
14. M. Eisler, P. Corbett, M. Kazar, D. Nydick, and C. Wagner, "Data ONTAP GX: A scalable storage cluster," in *Proc. USENIX Conf. File Storage Technol.*, San Jose, CA, 2007, pp. 139–152.
15. Bluearc. [Online]. Available: http://www.bluearc.com
16. Specsfs2008. [Online]. Available: http://www.spec.org/sfs2008
17. IOR HPC Benchmark. [Online]. Available: http://sourceforge.net/projects/ior-sio
18. D. Gottfrid, "Self-service, prorated super computing fun!" in *New York Times Blog*, Nov. 1, 2007. [Online]. Available: http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/
19. E. Anderson and J. Tucek, "Efficiency matters!" *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 1, pp. 40–45, Jan. 2010.
20. W. B. Ligon, III, and R. B. Ross, "PVFS: Parallel virtual file system," in *Beowulf Cluster Computing With Linux*. Cambridge, MA: MIT Press, 2001, pp. 391–430.
21. CloudStore. [Online]. Available: http://kosmosfs.sourceforge.net
22. O. Tatebe, Y. Morita, S. Matsuoka, N. Soda, and S. Sekiguchi, "Grid datafarm architecture for petascale data intensive computing," in *Proc. IEEE/ACM Int. Symp. Cluster Comput. Grid*, Berlin, Germany, 2002, pp. 102–110.
23. V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained TCP retransmissions for datacenter communication," in *Proc. ACM SIGCOMM*, Barcelona, Spain, 2009, pp. 303–314.
24. W. Tantisiriroj, S. Patil, and G. Gibson, "Crossing the chasm: Sneaking a parallel file system into Hadoop," in *Proc. Petascale Data Storage Workshop (poster)*, Austin, TX, 2008.
25. R. Ananthanarayanan, K. Gupta, P. Pandey, H. Pucha, P. Sarkar, M. Shah, and R. Tewari, "Cloud analytics: Do we really need to reinvent the storage stack?" in *Proc. USENIX Workshop Hot Topics Cloud Comput.*, San Diego, CA, 2009, pp. 1–5.
26. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, Seattle, WA, 2008, pp. 63–74.
27. 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 *Proc. ACM SIGCOMM*, Barcelona, Spain, 2009, pp. 51–62.
28. Ethernet in the data center. [Online]. Available: http://www.ethernetalliance.org
29. D. Nagle, D. Serenyi, and A. Matthews, "The Panasas ActiveScale storage cluster: Delivering scalable high bandwidth storage," in *Proc. ACM/IEEE Conf. Supercomput.*, Pittsburgh, PA, 2004, p. 53.
30. M. A. Kozuch, M. P. Ryan, R. Gass, S. W. Schlosser, D. O'Hallaron, J. Cipar, E. Krevat, J. López, M. Stroucken, and G. R. Ganger, "Tashi: Location-aware cluster management," in *Proc. Workshop Automated Control Datacenters Clouds*, Barcelona, Spain, 2009, pp. 1–6.
31. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *Proc. ACM Symp. Oper. Syst. Principles*, Stevenson, WA, 2007, pp. 205–220.
32. G. Porter, "Decoupling storage and computation in Hadoop with SuperDataNodes," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 41–46, Apr. 2010.

Received February 1, 2011; accepted for publication March 13, 2011

Lanyue Lu *University of Wisconsin-Madison, Department of Computer Sciences, Madison, SI 53706 USA (ll@cs.wisc.edu).*
Dr. Lu is a Ph.D. student in computer sciences at the University of Wisconsin-Madison. He received a B.E. degree from the University of Science and Technology of China in 2006 and an M.S. degree from Rice University in 2009. His research interests include storage and file systems.

Dean Hildebrand *IBM Research Division, Almaden Research Center, San Jose, CA 95120 USA (dhildeb@us.ibm.com).*
Dr. Hildebrand is a Research Staff Member in the Distributed Storage Systems department at the Almaden Research Center. He received a B.Sc. degree in computer science from the University of British Columbia in 1998 and M.S. and Ph.D. degrees in computer science from the University of Michigan in 2003 and 2007, respectively. He is the primary researcher of parallel network file system (pNFS), a standard extension that provides direct storage access to a diversity of parallel file systems while preserving operating system and hardware platform independence.

Renu Tewari *IBM Research Division, Almaden Research Center, San Jose, CA 95120 USA (tewarir@us.ibm.com).* In 1998, Dr. Tewari became an IBM Research Sta Member at the IBM Thomas J. Watson Research Center, soon after completing her Ph.D. at the University of Texas at Austin. In 2002, she moved to the IBM Almaden Research Center. Her areas of interest include all aspects of caching, edge servers, and networked and parallel file systems.